MULTI-SENSOR DEEP LEARNING FOR AUTONOMOUS POPULATION MONITORING OF MARINE SPECIES

by

Jiayi Zhao

A proposal submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Winter 2022

MULTI-SENSOR DEEP LEARNING FOR AUTONOMOUS POPULATION MONITORING OF MARINE SPECIES

by

Jiayi Zhao

Approved: ______________________________
Rudolf Eigenmann, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: ______________________________
Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: ______________________________
Louis F. Rossi, Ph.D.
Dean of the Graduate College and Vice Provost for Graduate and Professional Education

I certify that I have read this proposal and that in my opinion it meets the academic and professional standard required by the University as a proposal for the degree of Doctor of Philosophy.

Signed: ____
Christopher E. Rasmussen, Ph.D.
Professor in charge of proposal

I certify that I have read this proposal and that in my opinion it meets the academic and professional standard required by the University as a proposal for the degree of Doctor of Philosophy.

Signed: ____
Chandra Kambhamettu, Ph.D.
Member of proposal committee

I certify that I have read this proposal and that in my opinion it meets the academic and professional standard required by the University as a proposal for the degree of Doctor of Philosophy.

Signed: ____
Xi Peng, Ph.D.
Member of proposal committee

I certify that I have read this proposal and that in my opinion it meets the academic and professional standard required by the University as a proposal for the degree of Doctor of Philosophy.

Signed: ____
Arthur Trembanis, Ph.D.
Member of proposal committee

ACKNOWLEDGEMENTS

My deep gratitude goes first to my advisor, Professor Christopher Rasmussen, who expertly guided me through all stages of my graduate study. His insight launched the greater part of this dissertation. His mentoring with kindness, patience and enthusiasm have been inspired and encouraged me to a great extent to accomplish this work.

My appreciation also extend to Professor Arthur Trembanis who provided me the opportunity to work on this exciting project and invaluable guidance throughout this research. I would also like to acknowledge and give sincere thanks to the rest of my committee members, Professor Chandra Kambhamettu and Professor Xi Peng, for their brilliant comments and suggestions.

My completion of this project could not have been accomplished without all the people who helped me in the past. I am extremely grateful to my family and friends for their continuously support and love that carried me through all difficulties.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Monitoring marine species, both at an individual and population level, is a critical part of protecting the marine ecosystem and managing modern fisheries. The development and utilization of Autonomous Underwater Vehicles (AUV) provides a more efficient and safer way for the task of population assessment than traditional dredging methods, as well as yielding greater data collection for biological and oceanographic surveys. However current settings of the AUV still have limitations such that it has to follow preset searching patterns for data collection and often requires offline processing with manual analysis. Therefore we are motivated to develop a vision-based deep learning approach to bring on-board intelligence to the vehicle for autonomous population monitoring of marine species.

Our contributions in this work are the following: we first implemented deep learning based frameworks on the task of image-based scallop detection and further analyzed the performance of different architecture settings to demonstrate their capability of detection on low-contrast images in real-time. We also explored several ways to automatically upgrade the groundtruth annotation process. With our preliminary results, we extended our work on multi-class classification for scallop mortality rate estimation as well as analyzing the dynamics of predation. Next we experimented with optical flow for temporal analysis on sequential data. Furthermore we investigated deep learning based image registration and mosaicing methods to remove overlapping areas of successive images and therefore achieved a more precise scallop population census. Finally we proposed multi-sensor terrain analysis that combined information from optical images and side-scan sonar imagery in order to gain detailed representations of various substrate types. We also established the scallop-habitat relationship utilizing the results from our terrain classifier associated with scallop density distribution.

Chapter 1

INTRODUCTION

1.1 Overview

The amount of visual data in our world has been exploded to a massive degree in the last decades and large data is being produced by seconds every single day. Our job as computer vision scientists is to develop effective algorithms that are capable of understanding the content from the visual data and utilizing them on solving real-world complex problems.

Around the time of late 90s, statistical machine learning techniques, such as support-vector machine (SVM), boosting, graphical models, developed expediently for image segmentation. In 2001, [138] proposed an algorithm for face detection based on Adaboost and five years later the first digital camera with a real-time detector was announced. There is no doubt that it was a rapid transition from basic science research to real world applications. Later on, the idea of feature-based object recognition became influential as some features of the object tend to remain invariant and diagnostic to changes. Instead of matching patterns of an entire object, [75, 76] identify and match critical features only between objects. [66] stacks features extracted at different resolution levels from sub-regions of an image to create a spatial feature pyramid for recognizing holistic scenes.

As more and more sophisticated image understanding algorithms emerged, benchmark datasets were created for end-to-end deep learning training and performance measurement. The PASCAL Visual Object Challenge (VOC) [34] contains 20 object categories and over 11K images in total. For the sake of having adequate training data to fit the large amount of parameters within neural networks, ImageNet [115] with a hierarchical structure was introduced. This much larger dataset is composed of 20K categories and a total of 14M images. In 2012 the error rate of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) dropped approximately 10% from the previous year and the winner [63] beat all other algorithms participated in the challenge using a convolutional neural network (CNN) architecture. Over the years deep learning has shown its tremendous capacity and ability in making good progress in the fields of Computer Vision, Natural Language Processing, Bioinformatics and Robotics. Today, modern computer vision systems are powered primarily by convolutional neural networks, and our goal is to solve real-world complex problems among other scientific fields by the adoption of deep learning frameworks.

Monitoring marine species, both at an individual and population level, is a critical part of protecting the marine ecosystem and managing modern fisheries. Nowadays, there is a growing trend towards using autonomous underwater vehicles (AUV) for oceanography study. It allows collecting data under the sea on a larger scale than the use of ships and takes the place of sending human down to the deep sea for ocean exploration.

Our preliminary work focused on vision-based counting of wild scallops for population health measuring. Sequential images collected by the AUV are independently analyzed by convolutional neural networks which offers state-of-the-art object detection accuracy at real-time speeds. To augment the training dataset, a denoising-auto-encoder network is used to automatically upgrade manually-annotated approximate object positions to full bounding boxes, increasing the detection network's performance. The system can act as a tool to improve or even replace an existing offline manual annotation workflow, and is fast enough to function " in the loop " for AUV control. A web application is developed as a visualization tool to plot the quantitative distribution of scallops in different regions of each mission.

Next we extended our one-class detection experiments to multi-class classification for scallop mortality rate estimation as well as analyzing the dynamics of predation. The newly added categories are other relevant creatures or predators of sea scallops.

As available annotations are limited among some of these categories, we proposed data augmentation strategies in order to increase the diversity as well as ameliorate the issue of data scarcity.

Furthermore we conducted experiments on optical flow and recurrent neural networks (RNNs) for temporal analysis on sequential data. Since we observed that there were often large overlapping areas between successive images resulting in inaccurate scallop counts, we developed deep learning based image registration techniques and generated image mosaics from the keypoint matches to avoid double-counting the same scallop.

Since terrain types is one of the most important characteristics in habitat structures that determines the abundance of sea scallops and from our previous observation there are certain types of benthic substrate where the scallops often aggregates. Hence for this reason we built terrain classifiers associated with scallop density distributions to establish the scallop-habitat relationship. Meanwhile as side-scan sonars are commonly carried on AUVs for detection in deep water, we propose multi-sensor terrain analysis that combines information from optical images and side-scan sonar imagery in order to gain detailed representations of various substrate types.

1.2 Autonomous Underwater Vehicles (AUV) and Data Collection

In previous years population assessment of marine species were completed in a dredge-sample fashion. During each survey, a fishnet alike dredge would be towed along the seabed collecting target species at the standard commercial fishing speed. All collected samples were then counted manually to estimate the overall population within fishing areas before being released back to the water. This traditional dredging approach is most likely to cause incidental mortality to the animals. For example, physical damage due to contacting with the gears, being exposed to improper temperature or environment would all result in accidental fatality as well as inaccurate stock assessment.

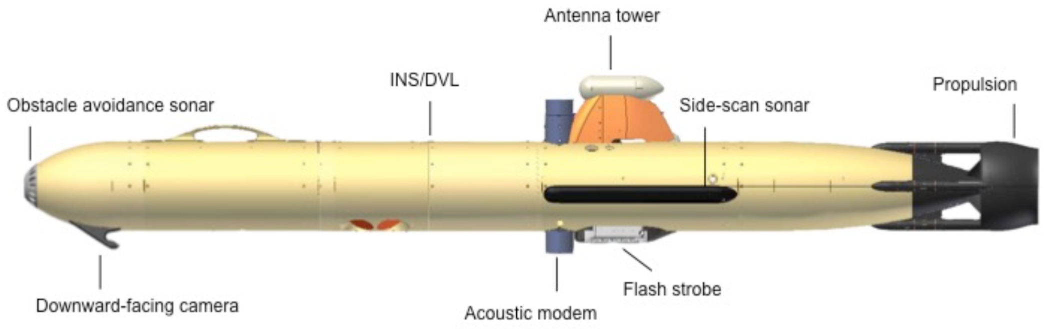
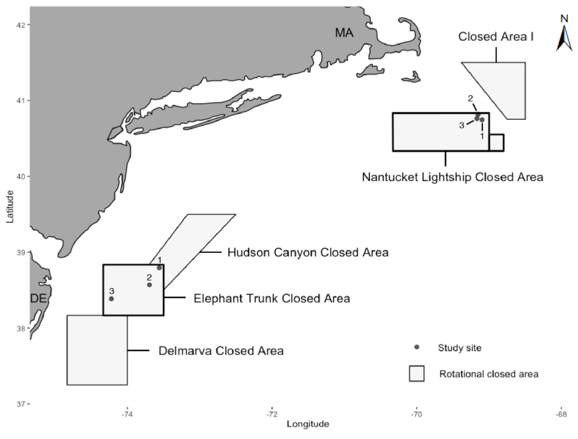


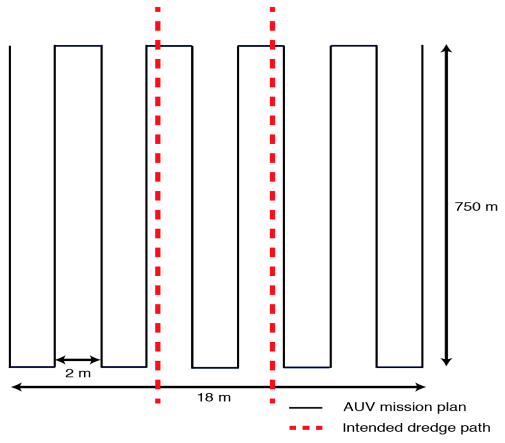
Figure 1.1: Gavia AUV

Most recently, autonomous underwater vehicles (AUVs) have been developed for scientific, military and commercial applications that provides us an efficient solution for conducting oceanographic surveys without undermining natural habitats. For our study the AUV manufactured by Teledyne-Gavia is shown in Figure 1.1. It is a type of self-propulsion underwater robots that are equipped with different kinds of sensors, an inertial navigation system and control modules allowing the AUVs to travel in the water with low deployment cost little or no human efforts. Configurations of the AUV include an inertial navigation system, Doppler velocity log(INS/DVL), control module, a downward-facing digital color camera, side-scan sonars, propulsion, nose cone with a forward-looking obstacle avoidance sonar and battery. Movements including forward, pitch, roll, heading can be controlled through the Gavia mission control program and the radius is about 8-10 m radius depending on speed settings and length of the vehicle. The inertial navigation system (INS) is incorporated with a Doppler velocity log (DVL) for navigation and during each mission the bottom lock of the DVL module keeps the vehicle remain a constant altitude of 2.25m from the seabed. The depth that is continuously calculated from the internal pressure sensor in the control module can also be adjusted. Positions of the AUV is recorded through a GPS in the control module. The integrated downward-facing camera paired with a flash strobe for illumination collects geo-reference photos of the seabed at a capture rate of 3.75 Hz (Figure 1.2c. Each photo is at a resolution of 1280 x 960 pixels covering a 1.88 m x 1.45 m area. All photos are stored in portable pixmap (ppm) format with image metadata of latitude, longitude, altitude, capture time, etc. The side-scan sonars are well-suited for depicting texture of the seafloor, measuring height distributions and differentiate substrate types. While in motion the side-scan sonars keep sweeping along the seabed from both sides at 1800 kHz high-frequency.

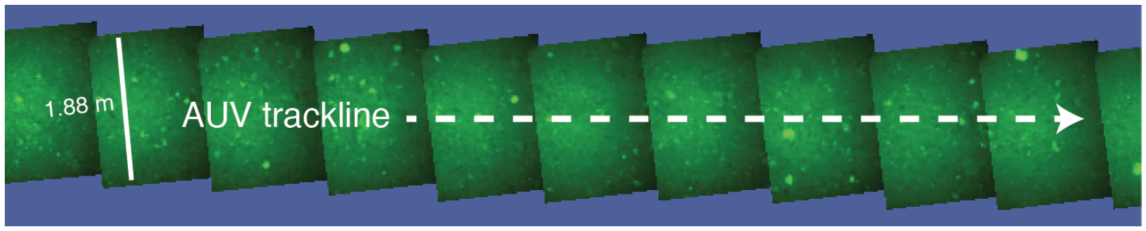
Since 2014 a Before-After Control-Impact (BACI) study [37] of estimating sea scallop incidental mortality has been conducted off the east coast of the United States at two study areas: the Elephant Trunk Closed Area (ETCA) and the Nantucket Lightship Closed Area (NLCA). As shown in Figure 1.2a, the ETCA study area is located in the Mid-Atlantic Bight 65 km east of Fenwick Island, Delaware, with a depth range of 50-60 m. Regions in ETCA is dominated by sandy substrate. The NLCA study area is 60 km off of Nantucket Island,Massachusetts with a depth range of 60-70 m. Substrate of NLCA is a mix of sand, gravel and rocks. 5 Cruises of 95 missions has completed and collected over 1 million camera images and sonar data.



(a)



(b)



(c)

Figure 1.2: (a) Map of scallop dredging study areas; (b) Lawn-mowing track of typical AUV mission; (c) AUV trackline for image collection.

1.3 Background and Related Works

1.3.1 Introduction on Convolutional Neural Networks

Multilayer Perceptron (MLP) is a supervised learning algorithm and a type of feedfoward artificial neural network (ANN). A basic MLP network is consist of an input layer, a hidden layer and an output layer. Every neuron in the current layer is connected to all the other neurons of the next layer and each neuron in the hidden layer is associated with a weight parameter that multiplies the input and a bias parameter adding the result of multiplication. The weights and bias are commonly initialized randomly and updated during the training process depending upon the error rates after each iteration.

While MLP is suitable for both classification and regression problems, it is insufficient of dealing with high-dimensional inputs such as images or videos. Therefore, Convolutional Neural Networks (CNNs) are designed to map high-dimensional representations to output variables. The benefit of CNNs is that they are capable of learning position and scale in variant structures of the input data which is important when working on images.

A common CNN is composed of convolutional layers, pooling layers, normalization layers and fully-connected layers. Convolutional layers serve the purpose of feature extraction. In order to preserve the spatial structures without connecting all neurons together, filters which can be seen as weight matrices of a certain size with the same depth as the input are used in convolutional layers. A filter is slide over the image in a grid fashion and computer dot product at every spatial location to generate activation/feature maps. The small region in the input image that is connected to a hidden neuron is called local receptive field. Stride is the number of pixels the filters shift over the input matrix each time. Weights and bias are shared among all hidden neurons and thus it allows the CNNs to detect same features at different locations making the network invariant to image translation. Filters extract low-level features such as edges at earlier layers and high-level ones at later layers.

Nonlinear activation functions added after every convolutional layer help the model to adapt with the variance of input data. Table 1.1 lists out the most commonly seen and popularly used activation functions. The sigmoid function generates a smooth range of values between 0 and 1 therefore is suitable for models that output probabilities. However the sigmoid function may cause an inefficient weights update because it is not zero-centered and the small derivative values would converge to zero result in slow learning during training time. The tangent function ($tanh(x)$), on the other hand, is a zero-centered activation function that generates a range of values between -1 and 1. The advantage over the sigmoid function is that it has wider range for faster learning. Again, it would kill off gradients when saturated at the ends of the function. In practice the Rectified Linear Unit (ReLU) function is the most frequently used in CNNs. It is very computationally efficient and converges much faster than either sigmoid or tangent functions. It is also possible to converge the combinations of ReLU with other functions. By initializing the ReLU neurons with small positive values it allows us to increase the likehood of the neurons being active at initialization for an efficient computational load. One problem of ReLU, called the ”dying ReLU problem”, is that it kills the gradients in the negative region. Thus variations of ReLU are designed targeting at relieving this problem, such as the Leaky ReLU function and the Parametric ReLU function shown in Table 1.1. Both of them have a parameter added to determine the slope in the negative region with more flexibility. The Exponential Linear Unit (ELU) function is similar to ReLU except it has a separate function defined for negative inputs to ensure noise-robust deactivation. ELU tends to converge towards zero faster leading to a faster training time. The Maxout function is an another nonsaturable function that generalizes ReLU and Leaky ReLU by taking the max over two linear functions. Because the parameters are doubled for each neuron it is computationally expensive and increases the overall model size.

Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$
Tanh(x)	$f(x) = tanh(x)$
ReLu	$f(x) = max(0, x)$
Leaky ReLu	$f(x) = max(0.01x, x)$
Parametric ReLu	$f(x) = max(\alpha x, x)$
ELU	$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha(e^x - 1), & x < 0 \end{cases}$
Maxout	$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$

Table 1.1: Activation Functions

Pooling layers always follow Convolutional layers if used for down-sampling the feature maps in order to reduce the number of parameters. Similarly to Convolutional layers, filters and stride are needed for each pooling layer. Max-pooling is one of the most commonly used pooling operation that calculates the maximum values of receptive fields. The last few layers in a network usually are fully-connected layers that assemble features extracted from previous layers for a final output. Fully-connected layers have the same characteristic as MLP. The softmax activation function at the final layer is used to transform the logits into probabilities from which classify objects of interest into corresponding categories.

In CNNs, backpropagation algorithm is used as training the other feed-forward neural networks. It computes the gradient of the loss function with respect to the neural network's weights. Backpropagation is essential for model training of efficiently updating the weights backwards using derivatives. Gradient optimization methods are used to adjust the parameters of the model in order to reduce the error from the loss function. First-order optimization methods include Batch Gradient Descent (BGD) which updates weights after calculating gradient on the entire training set; Stochastic Gradient Descent (SGD) which updates weight more frequently using one training example each step; Mini-batch Gradient Descent, a combination of SGD and BGD. SGD with Momentum [63] is used in practice as well. Other optimization methods such as AdaGrad, RMSProp, Adam, etc are frequently adopted for training neural networks.

Two common reason causing poor performance of the model are overfitting and underfitting. Overfitting occurs when a model is too closely fit to a limited set of data, even possibly picking up noise as learning data. As a result the model is not generalized enough to new unseen data. Underfitting, on the other hand, means that the model is neither fit the training set nor generalize well to new data. Underfitting is usually easier to detect than overfitting during training by monitoring the training loss. In order to improve single-model performance, regularization techniques are often used to slightly adjusting the learning algorithm making the model generalized better. Dropout prevents co-adaption of features by randomly setting some neurons to zero in each forward pass during training. Data augmentation techniques such as flipping, scaling, filtering, etc. are advantageous in increasing the diversity of data without collecting new ones. Early stopping is that we stop the training process the moment the validation loss stops decreasing thereby preventing the model from overfitted. Batch normalization is another technique to improve the gradient flow through the network and stabilize the learning process by standardizing input data.

1.3.2 Classic CNN Architectures

LeNet-5

LeNet-5 [67] is a pioneering CNN developed in 1998 for handwritten character recognition. It consists of 7 layers as shown in Figure 1.3a : 3 convolutional layers with

2 subsampling pooling layers in between, 2 fully-connected layer at the end. Every convolutional layer uses a 5×5 filter with 1 stride. The network was designed to recognize handwritten digits in the MNIST dataset where the input were 32 x 32 grey scale images. LeNet is a straightforward and relatively shallow architecture comparing to modern CNNs.

AlexNet

AlexNet [63] outperformed all the other non-deep-learning-based models on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. The error rate dropped significantly by 10% comparing to the best model of the previous year. The network structure (Figure 1.3b) is similar to LeNet but has two more convolutional layers and two more fully-connected layers which makes it more capable of processing higher resolution images. It uses max pooling instead of average pooling. Dropout, ReLU activations and SGD with momentum were also introduced in this work. The entire network was split into two pipelines and trained across two GPUs by putting half the neurons on each GPU. One drawback of AlexNet is that the model contains a large number of parameters and hyper-parameters making It prone to be overfitted.

ZFNet

Following AlexNet, ZFNet [151], the winner of ILSVRC in 2013, was designed to visualize intermediate features and operations of the model. ZFNet has a similar structure (Figure 1.3c) compared to AlexNet with the same number of layers, ReLU activations and SGD. The major difference is that ZFNet reduced the filter size to 7×7 filter instead of a 11×11 in AlexNet. The intuition behind this is that using smaller filters help us to preserve more pixel information thus increasing the accuracy.

GoogleNet

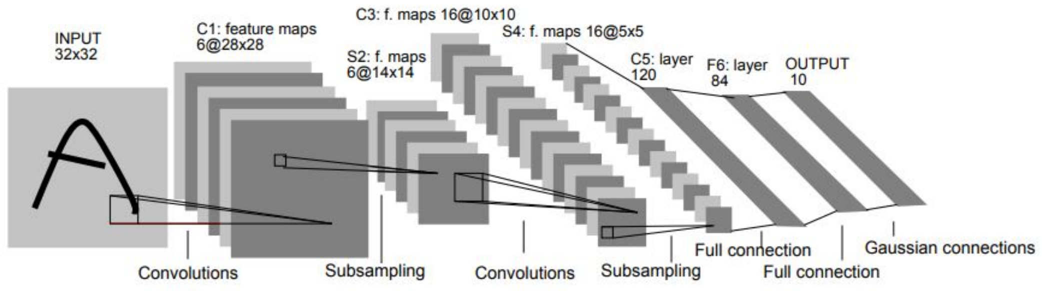
GoogleNet (Inception) is the winner of the ILSVRC in 2014. It achieved a top-5 error rate of 6.67% which was very close to human-level performance on the task. GoogleNet proposed a new concept of inception module shown in Figure 1.4b which had filters in multiple sizes operating in parallel. Having different sizes of filters at the same level would help the model to capture both global and local features. Outputs from each filter are combined before feeding to the next layer. In addition, GoogleNet replaced fully-connected layers with average pooling layers to largely reduce the number of parameters but preserved a deeper structure. The 22-layer CNN has 4 million parameters comparing to 60 million in AlexNet.

VGGNet

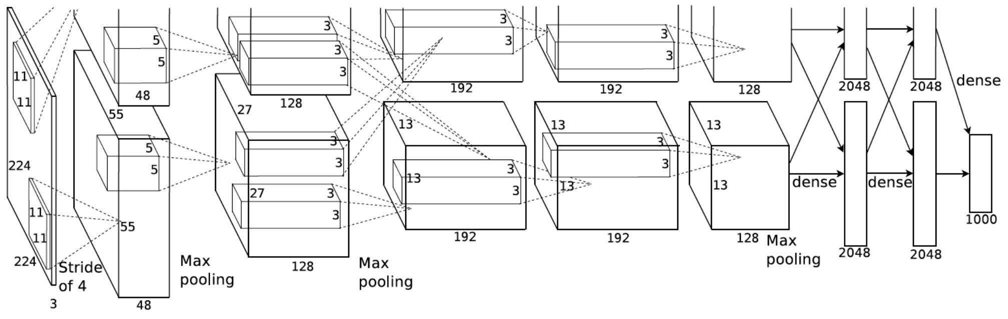
Another outstanding work from the ILSVRC 2014 competition is VGGNet [125], which is another deep CNN that has a uniform architecture (Figure 1.4a). VGGNet used stacked smaller-sized filters instead of larger kernel size filters used in AlexNet. The intuition behind this is that multiple small-sized filters give the same effective receptive field of one larger-sized filter but add depth to the model to learn more representative and complex features at lower cost. Until today VGGNet is widely used as a baseline feature extractor in many other applications. However given the size of the network it can be challenging to train the model due to the computational costs.

ResNet

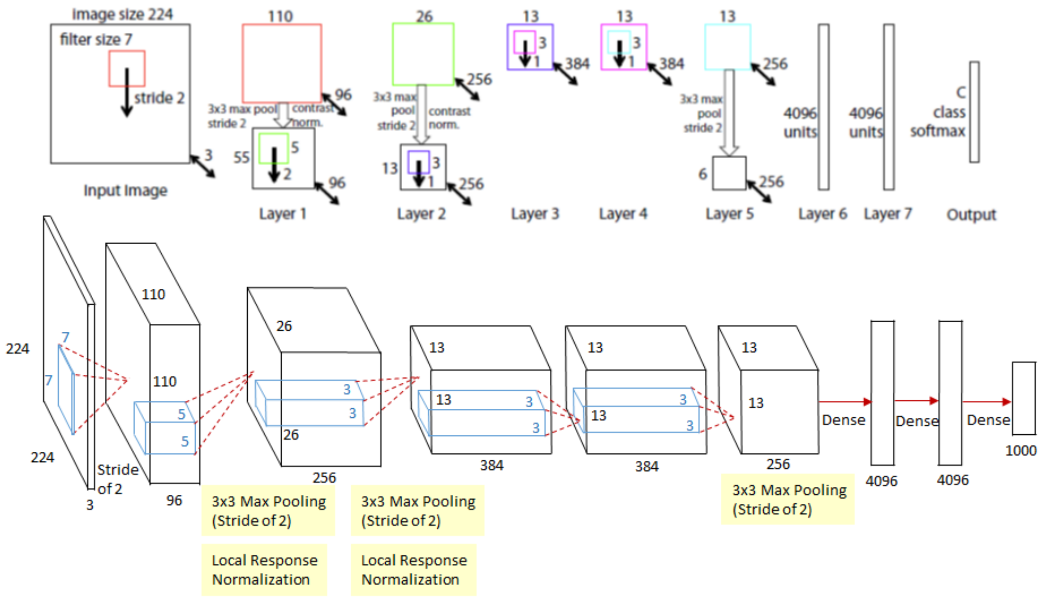
ResNet, the winner of the ILSVRC competition in 2015, is a much deeper network than the others we discussed above. It consists of 152 layers claiming that adding more layers should keep decreasing the error rate. However going deep makes the model hard to train at a low computational cost and inefficient learning due to the vanishing gradients problem. To avoid such issues, ResNet proposed residual blocks (shown in Figure 1.5b) and “shortcut connection” so that the gradients computed can directly be used to update weights in the first layer. It is assumed that optimizing the residual mapping function would be easier than optimizing the original unreferenced mapping. The added weight layers learn a residual mapping and even if vanishing gradient problem occurs the identity function can still transfer back to earlier layers preventing any loss of performance. The complete network architecture is shown in Figure 1.5a.



(a) LeNet-5 Architecture [67]

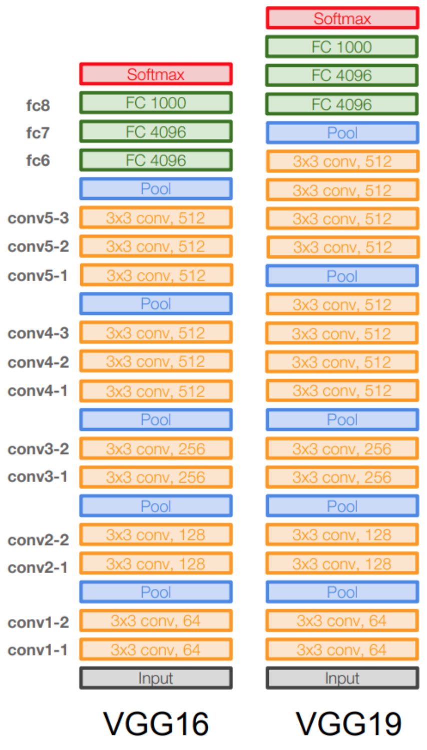


(b) AlexNet Architecture [63]

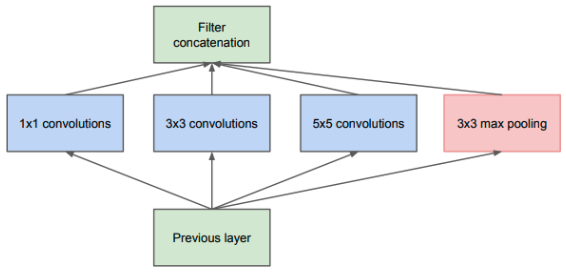


(c) ZFNet Architecture [151]

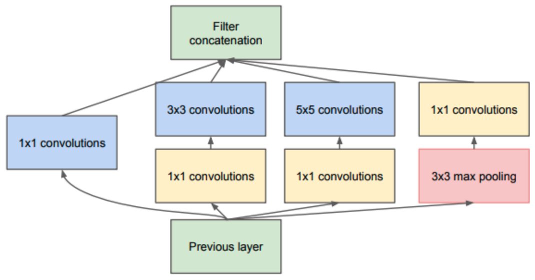
Figure 1.3: Basic CNN architectures



(a)

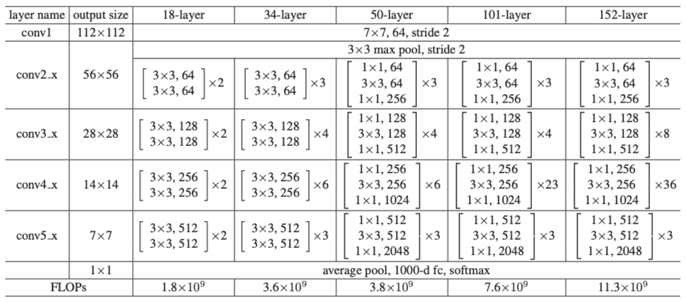


(b)



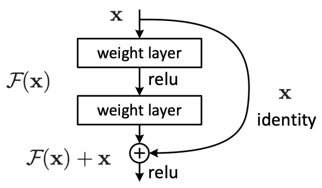
(c)

Figure 1.4: (a) Configurations of VGG16 (left) and VGG19 [125]; (b) Naive Inception module [128]; (c) Inception module with dimensionality reduction [128]



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

(a)



(b) Picture 2

Figure 1.5: (a) Overall Resnet architecture [46]; (b) Residual Learning Block [46]

1.3.3 Deep Learning for Object Detection

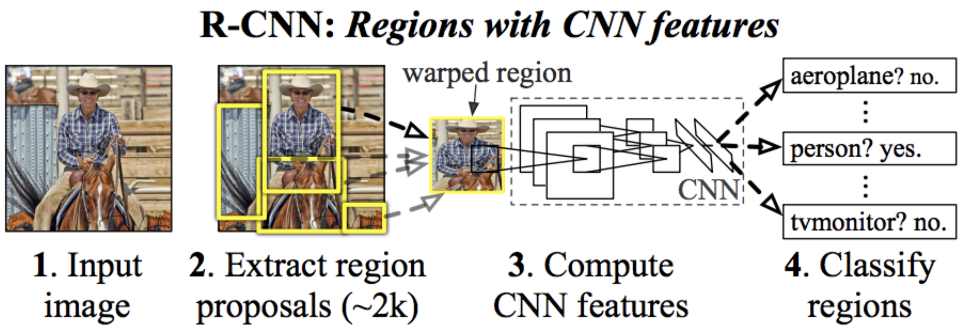
Early CNN-based object detectors operated either in a sliding window fashion or generating object proposals separately [133, 157] and then evaluating these hypotheses. These approaches achieved good results on difficult detection benchmarks, but were fairly slow and not well-suited to real-time deployment. More recently, it has been shown that CNNs have sufficient power to represent geometric information for localizing objects, opening the possibility of building state-of-the-art object detectors that rely exclusively on CNNs free of proposal generation schemes [108, 109, 110]. In such approaches, the network is trained end-to-end to predict both the appearance and geometric information of an object. At test time, given an input image, the entire network is only evaluated once instead of at different locations and scales of the image, enabling a large speed-up.

Two-stage detectors

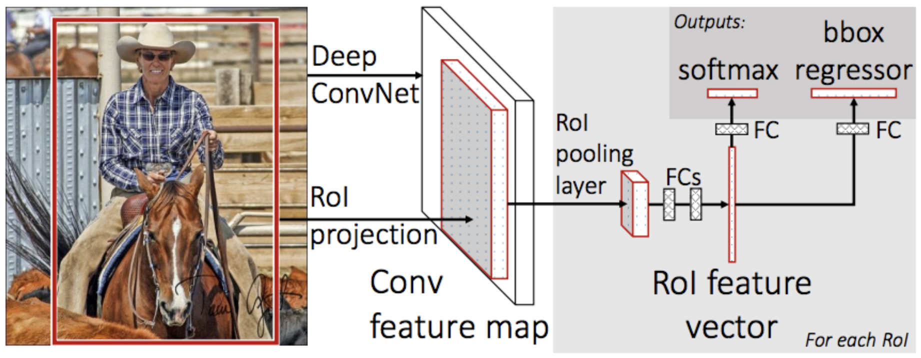
Two-stage detectors use selective search or region proposal networks to generate regions of interest in the first stage. Assuming there is infinite numbers of candidate bounding boxes the proposed regions can be very sparse. During the second stage the regions proposed are processed for bounding box regression and object classification.

R. Girshick et al. proposed a method R-CNN [42] in 2014 using selective search [133] to extract region proposals from images and feeding the regions into a CNN to generate feature vectors. All features extracted are then passed to a SVM classifier for final predictions. Overview of the R-CNN detection system is shown in Figure 1.6a. Additionally the algorithm also predicts four offset values for bounding box adjustment in order to increase the precision of localization. R-CNN achieves 0.537 mAP on VOC 2007 dataset and 0.585 mAP on VOC 2010 dataset. With deep networks as feature extractor, training R-CNN can be quite expensive in space and time, and it is hard to be implemented in real-time, i.e. 47s per test image. Meanwhile, there is no training during the selective search phase causing bad candidate region proposals generated which leads to poor overall performance. Aiming at solving the drawbacks of R-CNN, the same authors proposed Fast R-CNN [43] in 2015 which feed the input images instead of region proposals to the CNN for feature map generation. In this case the convolution operation is applied only once per image. The feature maps are then warped and reshaped into fixed size by a RoI pooling later so that they can be connected to FC layers (Shown in Figure 1.6b. Finally predictions and offset values are computed through a Softmax layer. Fast R-CNN significantly reduced the training/test costs over R-CNN and achieved higher mAP (0.70 on VOC 2007 and 0.688 on VOC 2010) in the meantime.

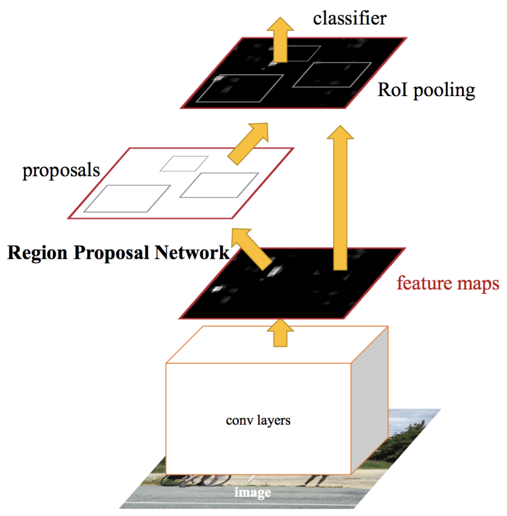
S. Ren et al. proposed another region-based detection network, called Faster R-CNN [111]. The novelty of their work is that they replaced the Selective Search algorithm with a Region Proposal Network (RPN) in order to largely reduce the costs of producing region proposals. The RPN can be merged with the detection network into a unified architecture (as in Figure 1.6c) by sharing the convolutional features. It reached a frame rate of 5fps on one GPU for a deep VGG-16 [125] model. Using a neural network for region proposal generation also improve the quality of proposals and hence increase the overall detection accuracy.



(a) R-CNN object detection system overview [42]



(b) Fast R-CNN architecture [43]



(c) Faster R-CNN architecture [111]

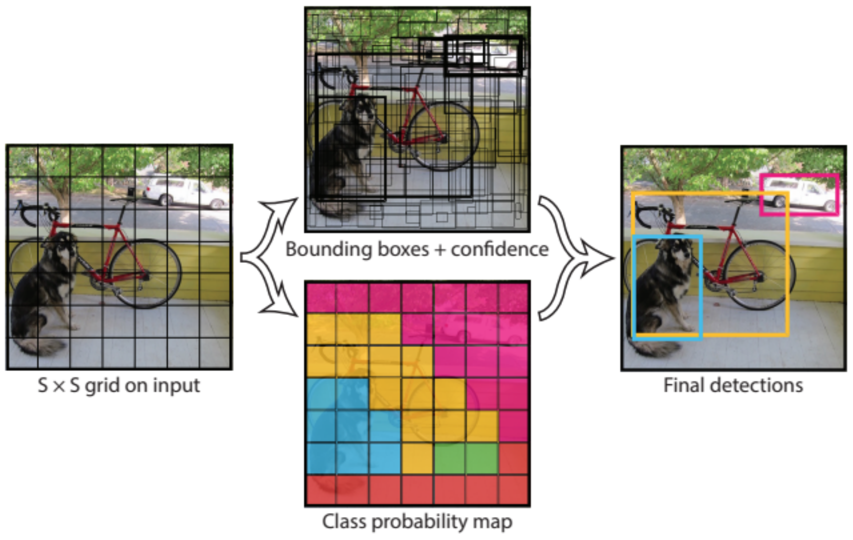
Figure 1.6: Overview of exemplary 2-stage detector architectures

Two-stage detectors have been improved over the years in terms of speed and accuracy. Because that all objects should be included in the region proposals generated during the first stage and negatives are then filtered out before further being classified into corresponding categories, two-stage detections can often achieve top accuracy in localization and detection.

One-stage detectors

One-stage detectors using a single feed-forward convolutional network to sample across an input image densely predicting the bounding boxes and class probabilities in one stage. The Single Shot MultiBox Detector (SSD) [73] is one of the first proposing to use an unified architecture for object detection. SSD is built on VGG-16 [125] for feature extraction and uses multi-scale feature maps to detect object independently. Given the confidence scores, Non-maximum Suppression (NMS) technique is adopted to discard duplicate predictions of which scores are lower than the threshold. SSD achieves 0.721 mAP on VOC2007 [34] dataset at 58 FPS for 300×300 input. One of the drawbacks of SSD is that it struggles to differentiate objects in similar categories and recognize small objects.

Redmon et al. developed one such unified CNN for realtime object detection which they called "You Only Look Once" (YOLO) [108] The core idea of YOLO (as shown in Figure 1.7a) is that it considers the detection task as a regression problem. Specifically, the input image is divided into an $S \times S$ grid of cells, each of which contains information on B hypothetical object bounding boxes. Each such bounding box is parametrized by a 5-D vector $[x, y, w, h, P(Obj)]$, where $P(Obj) = 1$ if the center of any ground-truth object bounding box is inside the cell and $P(Obj) = 0$ otherwise. Each grid cell also includes a conditional class probability: $Pr(c \mid Obj)$, where $c \in \{C\}$. Accordingly, the class-specific confidence is given by: $P(c) = Pr(c \in Obj)P(Obj)$. For each presented image, the output layer of the network is an $S \times S \times (B * 5 + C)$ tensor. Non-maximal suppression is used to remove duplicate detections, followed by thresholding on $P(c)$.



(a)

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

(b)

	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	64	$3 \times 3 / 2$	128×128
1×	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	$3 \times 3 / 2$	64×64
2×	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	$3 \times 3 / 2$	32×32
8×	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	$3 \times 3 / 2$	16×16
8×	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	$3 \times 3 / 2$	8×8
4×	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8
	Avgpool		Global	
	Connected		1000	
	Softmax			

(c)

Figure 1.7: (a) The regression process of YOLO detector [108]; (b) Darknet-19 [109] classification model from YOLOv2; (c) Darknet-53 [110] classification model from YOLOv3.

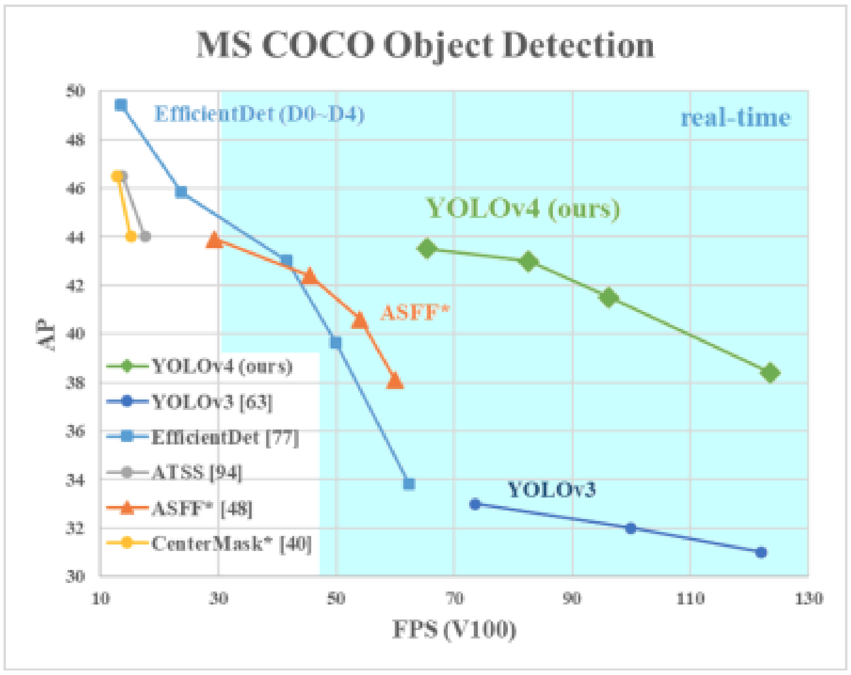
Using values of $S = 7$ and $B = 2$, YOLO achieved a very high mAP (mean average precision) on the 20-class PASCAL VOC 2007 [34] while running at a speed of 45 fps, faster than any comparable system. One shortcoming of YOLO is its difficulty at detecting small objects that are very close to each other since it only predicts two bounding boxes within one class per grid cell. And the maximum number of objects the model can detect is limited by the grid cell.

In 2017, Redmon and Farhadi proposed another realtime detection system, YOLOv2 [109], that aims to improve on the localization and recall performance of YOLO. Several key modifications in YOLOv2 include batch normalization on every convolutional layer to enhance convergence and regularize the model, a higher input image resolution of 448×448 and finer gird with $S = 13$, and the placement of YOLO's fully connected layers which directly regress bounding box coordinates with and "anchor box" concept adapted from [111] which allows multi-object prediction per grid cell. YOLOv2 uses a new classification model as the front end which is somewhat similar to VGG models [125], but with global average pooling to make predictions and 1×1 filters to compress features representations between 3×3 convolutions. The classification model is called "Darknet-19" and has 19 convolution layers and 5 max-pooling layers. Details of the configurations are shown in Figure 1.7b. YOLOv2's mAP on PASCAL VOC 2012 is significantly improved over YOLO an comparable to the current leaders like Faster R-CNN with ResNet [46] and SSD512 [73] while running from 2 to 10 times faster.

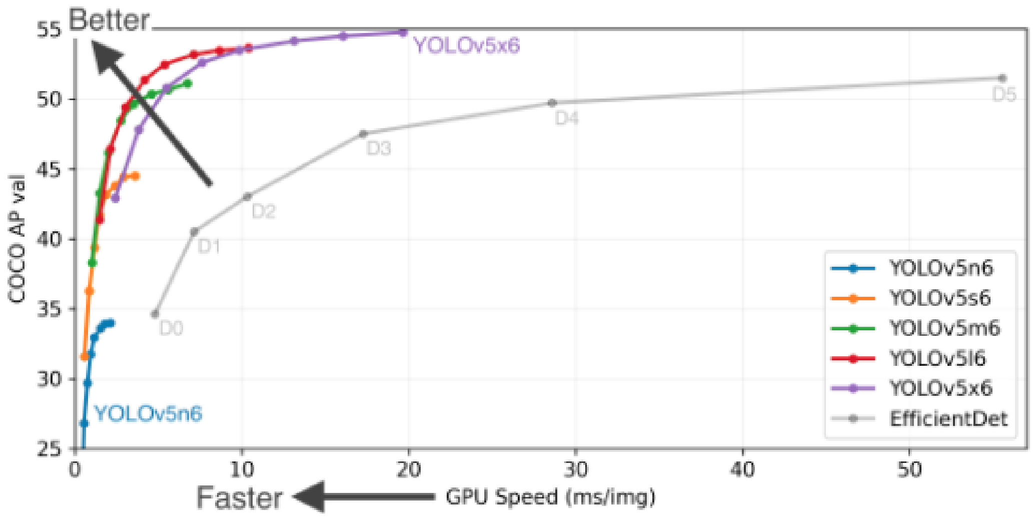
YOLOv3 [110] was proposed to further improve the detection accuracy with a more complex underlying architecture than YOLOv2. A new network, Darknet-53 was designed for feature extraction. For the task of detection, it stacks another 53 convolutional layers giving a 106-layer architecture. A full description of the Darknet-53 model is in Figure 1.7c. YOLOv3 was added several popular components appeared in most of state-of-art detectors, such as skip connections, residual blocks, and upsampling. Following [111], for each bounding box YOLOv3 predicts an objectiveness score using logistic regression but only assigns the box with score = 1 prior for each groundtruth object.

The major modification in YOLOv3 is that it predicts boxes at three different scales. The feature map that is taken from 2 layers previous is then upsampled twice in the network generating three different sizes of feature maps at different places. The 3 features maps are concatenated together and processed through the following convolutional layers. The same design repeats one more time to predict boxes at final scale. This procedure allows the network to gain more meaningful semantic information and finer-grained features which makes YOLOv3 better at detecting small objects. YOLOv3 uses 9 anchor boxes and since it predicts boxes at 3 different scales, results in 10x more number of predictions than YOLOv2.

The next generation of YOLO, YOLOv4 [11], adopts Bag-of-Freebies and Bag-of-Specials methods to improve detection accuracy without increasing inference time. Universal features that are well-known of improving CNN accuracy were also included in YOLOv4: Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT), Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss. YOLOv4 improved the mean average precision (mAP) by 10% and the number of frames per second by 12% on MS COCO dataset compared to YOLOV3. Comparison of YOLOv4 and other advanced object detectors is show in Figure 1.8a . The YOLOv4 architecture is composed of 4 blocks: CSPDarknet53 as the backbone, the neck, dense prediction (head) and sparse prediction. Shortly after, the YOLOv5 implementation has been released publicly [58]. YOLOv5 has similar structure (i.e. CSP backbone and PA-NET neck) as YOLOv4 but has improvements in mosaic data augmentation and auto learning bounding box anchors. The author of YOLOv5 claims that YOLOv5 achieved comparative accuracy with YOLOv4 meanwhile is faster and more lightweight (Figure 1.8b). However there was not a paper published along with the implementation releasing, and some questions have been raised among the community on whether the results are reproducible. There it is hard for us to conclude the performance of YOLOv5 without in-depth investigation.



(a)



(b)

Figure 1.8: (a) Comparison of YOLOv4 and other object detectors on MS COCO [11]; (b) Performance of 4 available models of YOLOv5 on MS COCO[58]

One-stage detectors have high inference speeds as they contain a single feed-forward convolutional network which skip the region proposal stage and perform object detection directly over a dense set of candidate locations. Because of this property, one-stage detectors are often favored for real-time applications. However the training procedure for one-stage models is easily to be dominated by the large number of background samples result in increasing number of false positives. This foreground-background imbalance has become the bottleneck for one-stage detectors achieving higher accuracy. RetinaNet [72] specifically addressed this issue using a newly proposed Focal Loss (FL) function. The traditional Cross Entropy (CE) loss is prone to overlook rare classes when summed over a large number of easy examples and thus a weighting factor $\alpha \in [0, 1]$ is often introduced to address the class imbalance. The α factor can be set by cross validation as a hyperparameter or set by inverse class frequency. In RetinaNet, the balanced CE loss is considered as a baseline for comparison with proposed loss function. The Focal Loss is defined as: $\mathrm{FL}(p_t) = (1 - p_t)^\gamma \log(p_t)$. The modulating factor $(1-p_t)^\gamma$ with a tunable focusing parameter $\gamma >= 0$ enforces the model to down-weight easy examples and focus training on hard negatives. For an example that has a p_t approaching to 1, the modulating factor is closed to 0 and thus the loss for this well-classified example is down-weighted. On the other hand, a hard example with a small p_t approaching to 0 would have a higher loss as the modulating factor closing to 1. Additionally, a α balanced FL function ($\mathrm{FL}(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t)$) is also implemented in their work to boost accuracy.

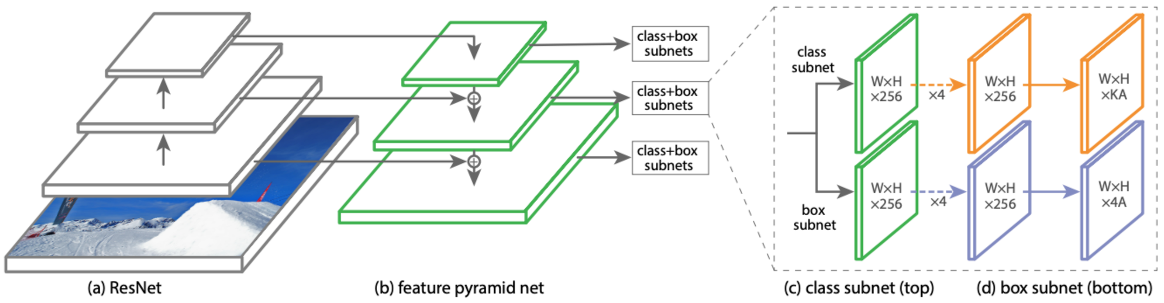


Figure 1.9: The RetinaNet network architecture [72].

The complete architecture of RetinaNet is shown in Figure 1.9. The backbone (Figure 1.9 (a) and (b) is consist of ResNet and Feature Pyramid Network (FPN) on the top of it to generate rich, multi-scale convolutional feature pyramid from input images. Two subnets are connected to the back for one classifying anchor boxes and regressing from anchor boxes to ground-truth object boxes, respectively. In Table 1.2 we show the object detection performance comparison of RetinaNet among other one-stage and two-stage detectors on on COCO *test-dev* [70] from [72] paper. RetinaNet with ResNet-101-FPN exceeded the average precision of Faster R-CNN [111] while running 50ms faster per image.

1.3.4 Deep Learning for Semantic and Instance Segmentation

Over the years, we have witnessed the accomplishment of deep learning on the task of semantic [19, 23, 74] and instance [17, 20, 18, 142] segmentation cross various application domains. Semantic segmentation requires each pixel to be associated with a class label. Pixels of objects in the images are grouped together based on pre-defined categories. Multiple objects of the same class are treated as a single entity in semantic segmentation problems. Instance segmentation, on the other hand, separates each object in the same class and treats them as distinct individual instances. It not only needs to detect objects in an image but also requires to determine which category each pixel belongs to.

	AP	AP_{50}	AP_{75}	AP_s	AP_M	AP_L
Two-stage methods						
Faster R-CNN+++ [46]	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [71]	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [50]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [123]	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods						
SSD513 [73]	31.2	50.4	33.3	10.2	34.5	49.8
YOLOv2 [109]	21.6	44.0	19.2	5.0	22.4	35.5
YOLOv3 [110]	33.0	57.9	34.4	18.3	35.4	41.9
RetinaNet w ResNet[72]	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet w ResNeXt[72]	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2

Table 1.2: One-state and two-stage methods performance comparison of object detection results on COCO *test-dev* [70]

The task of segmentation is to assign class labels to each pixels across an image. CNN models can be adjusted to complete this task by removing the FC layers at the end to produce dense predictions. The challenge is that the process of labeling each pixel in order to train deep neural networks for segmentation is often tedious and expensive. Weekly supervised segmentation techniques have been developed over the past decades as they only require partial annotations making them more practical than fully-supervised learning for training segmentation models. [25] used bounding box annotations as an alternative or extra source to generate segmentation masks. [94] developed Expectation-Maximization (EM) methods to train image-level and bounding-box annotated data for image segmentation. [104] extended the famous GrabCut [114] method and defined an energy minimisation problem over a densely-connected conditional random field in order to collect pixel-wise object segmentations by updating training targets iteratively. [129] proposed a partial cross entropy (pCE) loss function to improve the performance of deep CNNs on generating full masks from partial scribble groundtruth. [102] developed a Point-based Distance Metric Learning method that does not demand dense annotations but uses a sparse number of labeled points to guide the training process. [9] also introduced point-level supervision combined with Objectness Prior in the loss function to further substantiate whether each pixel belongs to any object class. Other types of labels for supervising segmentation labels, such as patches [10, 99], image tags [57, 7], have been applied as guidance as well.

Semi-automatic object segmentation is an another direction to alleviate the heavy workload to obtain dense annotations which includes human in the loop to provide class information. GrabCut [114] is one of the pioneering works that estimates color distribution between the target object and background using a Gaussian mixture model based on user-defined bounding boxes of the object. The estimated segmentation regions are gradually updated from a graph cut based optimization. Users can be further included in the process by correcting edges or misclassified regions. [68] aimed to solve the ambiguity between the user's input clicks and the intended segmentation regions. Their proposed architecture consists of two CNNs. One is trained for synthesizing a diverse set of solutions that it produces a set of possible segmentation areas of an image based on the user's input. The second is a context aggregation network that evaluates the predicted masks.

[93] first developed a groundtruth annotating system using 4 extreme points, i.e. the top, bottom, left- and right-most points of an object. The 4 points are on the boundary of a object where can be easily used to obtain a groundtruth bounding

box. The authors argue that traditional annotating systems of dragging rectangles or clicking the opposite corners of objects always require refinement in order to acquire tight bounding boxes. Figure 1.10-a gives an example of the conventional way of annotating an object comparing to [93] ‘s method in Figure 1.10-b. [93] crowd-sourced extreme point annotations for PASCAL VOC 2007 and 2012 datasets showing that extreme clicking led to high quality bounding boxes with a 5x faster speed.

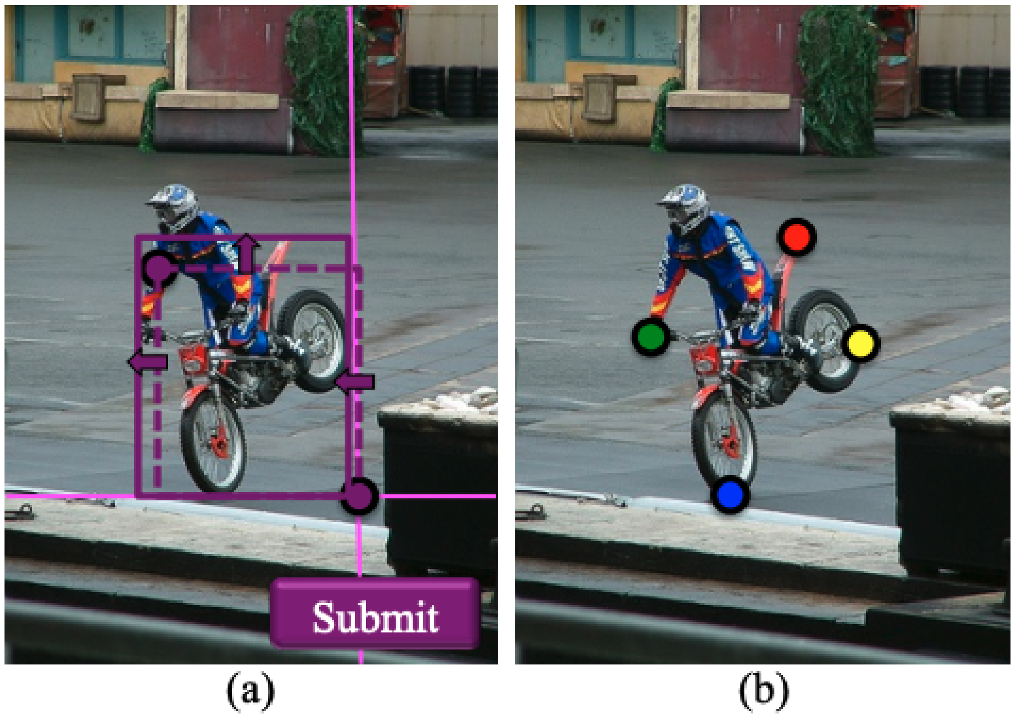


Figure 1.10: Comparison of conventional method of annotating bounding box (a) with [93]'s extreme point method (b).

Deep Extreme Cut (DEXTR) [81] further explored transforming extreme points as input for instance segmentation. Figure 1.11 shows the overview architecture of DEXTR. The extreme point annotations are used as guidance to the input of the network. Each of the points is centered by a 2D Gaussian to generated a heatmap with activations which is then concatenation with the RGB channels to create a 4-channel input. It introduced atrous convolutions to replace the fully connected layers at the end of the ResNet-101 backbone in order to preserve the same receptive field and output resolution for dense prediction.

DEXTR were then implemented on various applications to test its generalization capabilities and applicability. For class-agnostic instance segmentation, mask predictions that can be of any class including unseen categories were acquired from extreme points. The method was validated on PASCAL [34], COCO [70], GrabCut [114] public datasets and it surpassed the state-of-the-art techniques such as Sharpmask [100], DeepGC [149], MILCut [147], etc. For video object segmentation, it followed a semi-supervised framework but used the masks generated from DEXTR on the DAVIS datasets [96, 101]. It showed that using DEXTR reduced the annotating time meanwhile maintaining the same performance as the state-of-the-art trained on the groundtruth masks. In other words DEXTR would be helpful to created more masks in the same time period producing better segmentation results. The authors also provided annotation and interactive object segmentation pipelines assisted by DEXTR to alleviate the workload of groundtruth mask generation.

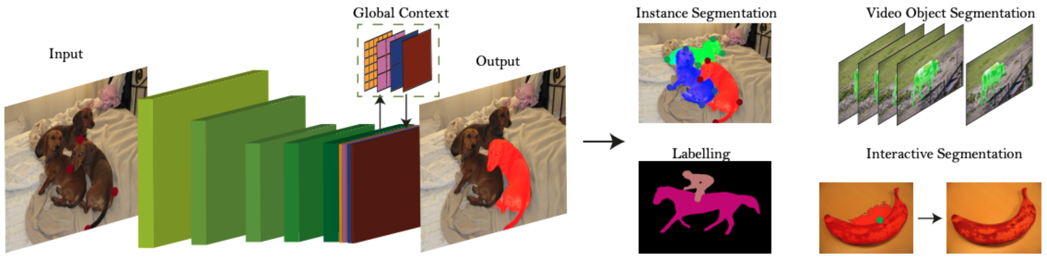


Figure 1.11: Architecture of Deep Extreme Cut (DEXTR) [81]

Inspired by DEXTR, Zhang (et, al.) [153] developed an Inside-Outside Guidance (IOG) approach for interactive segmentation. Given the fact that DEXTR requires users to carefully click all extreme points on the object boundaries, the authors of [153] proposed extreme clicking at the interior and exterior of object regions. The inside point should be clicked near the center of object while two outside points are at the symmetrical corner location. The authors also claimed that the DEXTR annotation process might cause confusions when the points were at similar spatial locations whereas the IOG approach could provide indications of foreground and background regions. Similar to DEXTR, each point is centered by a 2D Gaussian generating two heat maps, i.e. foreground and background relatively. For the segmentation network, [153] adopted a coarse-to-fine designed architecture [22] in order to refine the inaccurate segmentation along the object boundaries. The CoarseNet is FPN-like with lateral connections that fuses deep-layer information with low-level details. The pyramid scene parsing module at the end helps with enhancing the representation with global contextual information. FineNet is designed to recover the missing boundary through upsampling and concatenating information across every layer in CoarseNet. Additionally the IOG approach supports interactive segmentation that allows adding extra clicks of foreground or background from users for region correction and refinement.

1.3.5 Image Registration

Image registration is a method of mapping two or more images into a particular coordinate system in order to integrate the information. Depending on the manner of data acquisition, different registration techniques may be applied for various application domains. Images of the same scene or object that are captured at different viewpoints can be registered together to gain a better representation of the scene, as known as the multi-view analysis. Example applications include image mosaicing, shape recovery from stereo, etc. For multi-modal analysis, data is acquired from various sensors. By merging information obtained from different sources, it is able to provide greater details or spatial and spectral characteristics than individual images. This type of registration has been widely used in medical imaging, remote sensing, 3D reconstruction, etc. Images taken during a period of time can be registered together to monitor or track changes for a multi-temporal analysis.

The main steps of an image registration procedure typically include feature detection and matching, model transformation and resampling. In the step of feature detection, salient structures (such as edges, corners, distinctive regions, etc) shared among the images are extracted as control points. These features are useful to find correspondence and establish spatial relations between sensed images and the reference. A mapping function is then designed to map the control points through a transformation model. Types of transformation model include affine or linear transformation (scaling, rotation, reflection, translation, shearing) and nonrigid transformation (thin-plate, multiquadrics, deformation, etc). Affine transformation methods are typically global whereas nonrigid transformation models can locally warp target images to be aligned with the reference image.

In general, there are two types of alignment algorithms: The intensity-based methods register entire images or sub-images by computing intensity patterns for similarity measurement; The feature-based methods utilize distinct points in images to obtain point-by-point correspondence. [51] uses shape context as a feature-based method for remote sensing image matching. The shape context exploits feature similarity between circular regions of the two images to find corresponding control points on the sensed image. [148] combines intensity-based and feature-based methods proposing a hybrid approach that matches interior intensities of scale-invariant salient region features using robust similarity measures. Wavelet-based techniques are commonly used for feature extraction [82, 117]. [77, 2, 1, 3, 137] propose adjustments and improvements based on the Scale Invariant Feature Transform (SIFT) algorithm[75] to detect features to identify similar objects in two images. [86] applies automatic Harris corner detection and a Random Sample Consensus (RANSAC) transformation model on the task of automatic satellite image registration. [62] proposes a fast affine template matching algorithm and a branch-and-bound scheme to accelerate the algorithm.

The multi-modal registration can be considered as a special case of image registration that data to be registered does not belong to the same modality. Nowadays, data is often collected from various sources (e.g. X-ray/MRI/CT volumes, camera/sonar/LiDAR) in many application domains. Such data may have different dimension, structure or density which introduces challenges that it is not straightforward to come up with a general framework to relate multiple modalities. [139] introduce a information theoretic approach to analyze similarity across modalities using maximization of mutual information. Without prior knowledge of the correspondences across different modalities, [87] projects multi-modal problem into uni-modal registration.

Plenty of research has been conducted on optimization-based algorithms [95, 140] that optimize the transformation parameters iteratively over the quality of registration process. [121] presents a Particle Swarm Optimization (PSO) technique to register 3D MRI and 3D CT medical data. [21] proposes a hybrid approach that comprises genetic algorithms and conventional PSO. Expectation-Maximization (EM)-based registration is an iterative method to find local maximum likelihood of the best alignment.

Deep learning based approaches have been increasingly implemented in solving image registration problems. D2-Net [31] finds reliable pixel-level correspondences using a single convolutional neural network that works as a dense feature descriptor and a feature detector simultaneously. [84] proposes a unified deep network pipeline of learning detection, orientation estimation, and feature description while preserving end-to-end differentiability. [79] augments local feature descriptors using cross-modality contextual information.

DeTone *et al.* developed deep image homography estimation architectures (i.e. HomographyNet) on image pairs in 2016 [27]. The regression network directly produces homography parameters that maps one image to another. Additionally the architecture can be converted with minor changes to a classification model that delivers distributions over quantized homographies indicating the confidence level of the estimations. Their work casts the homography estimation into a learning problem by using VGG-like deep neural networks without extra efforts on local feature detection and transformation estimation. The same authors proposed a point tracking system powered by two deep convolutional neural networks for SLAM tasks [29]. Given the fact that the 2d point location detection stage in most sparse SLAM pipelines often requires hand engineering and expert knowledge, the first network the authors designed, MagicPoint, takes an input image and extracts salient mapping points that are SLAM-ready to the other image. The second network, MagicWarp, generates homography from the point image pairs produced from MagicPoint. Unlike traditional approaches, the transformation engine finds the correspondences in image pairs using point location information instead of descriptors. Experiments in [29] shows that the MagicPoint detector is more robust to lighting variation than traditional corner detection baselines and MagicWarp exceeds the Nearest Neighbor matching approach with higher pose estimation accuracy. The authors came up with the idea of SuperPoint [28] in the same year that is a self-supervised learning approach for interest point detection and description. The architecture is a fully-connected neural network that consists of a shared encoder operating a full-sized images, a interest point decoder for detection and a descriptor decoder. Locations of pixel-level interest point and associated descriptors are learned jointly in one forward pass. In order to increase the robustness of the network, a multi-scale Homographic Adaptation was introduced as well that warps the input image multiple times to understand interest points from different viewpoints and scales. The SuperPoint network was trained in conjunction with Homographic Adaptation on synthetic dataset to generate pseudo-groundtruth interest points on real images.

From SuperPoint, SuperGlue [116] was introduced for learning feature matching with Graph Neural Networks (GNNs) that jointly found correspondences and rejecting non-matchable points. It adopts an attention-based context aggregation mechanism to learn geometric transformation and underlying 3D scene through end-to-end training. SuperGlue is designed as solving a optimization problem of finding the matches as partial assignments between two sets of local features. The architecture is made of an

Attentional Graph Neural Network and an Optimal Matching Layer. Given initial local features, the Attentional Graph Neural Network computes matching descriptors with long-range feature aggregation across images. The keypoint encoder is formulated such that it allows the network to collectively reason about positions and appearances. The multiplex graph contains intra-image edges that connect keypoints within the same image and inter-image edges that connect keypoints in one image to all keypoints in the other image. Representations of the nodes are updated at each layer of the network and simultaneously aggregating with all edges. Attentional Aggregation includes self-attention based for intra-image edges and cross-attention based for inter-image edges. Furthermore the aggregation mechanism is formulated with maximum flexibility so that the network is able to focus training on selected subsets of keypoints. The purpose of the Optimal Matching Layer is to produce a partial assignment matrix. All possible matches are included to compute the score matrices and pairwise scores are used to represent the similarity of matching descriptors. Similar to [28], the authors adopted the common technique in graph matching of dustbins to augment assignments by suppressing unmatched keypoints. Finally the Sinkhorn algorithm [118] was adopted to normalize the score matrices.

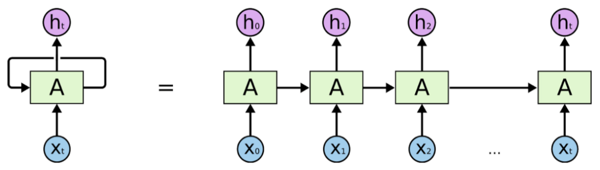
Compared to related works such as Instance Normalization [134], Transformer [136] and ContextDesc [80], SuperGlue has advantages of processing both appearance and position together as well as a more flexible context aggregation mechanism. For homography estimation, Superglue outperformed PointCN [85] and OANet [152] with both RANSAC and DLT estimators on the Oxford and Paris dataset [103], achieving 98 % recall and high precision. For indoor and outdoor pose estimation, Superglue was evaluated with baseline matching approaches using root-normalized SIFT [75] and SuperPoint [28] features on the indoor scenes dataset from [24] and the PhotoTourism dataset from the CVPR'19 Image Matching Challenge, respectively. SuperGlue surpassed all baselines with higher pose accuracy and matching scores.

1.3.6 Temporal Data Processing

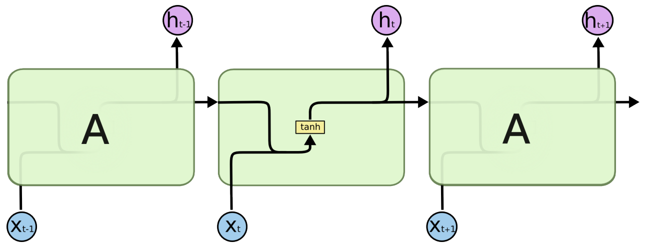
Sequence Learning

Feed-forward networks as we discussed in previous sections, do not have notion of order in time that only the current input is considered and transformed to an output. However under the circumstances where the current output does not only depend the current input but also on the previous input, it would be necessary to make our model be flexible in order to process sequential information and data with dynamics. Recurrent neural networks (RNNs) are a type neural network that addresses this issue with a so-called internal memory, that is, they are able to memorize the input they received. Typically in RNNs, there is a looped recurrent cell that keeps the input repeated for t time steps and pass the output of the recurrent neurons to the next layer only after completing all the time steps. A recurrent neural network can be unfolded into multiple copies of the same network (shown in Figure 1.12a), each passing information to its successor. Advantages of RNNs includes having less limitation of the input length, taking historical context into account and sharing weights across time. However when the gap between the past relevant information and the current task grows, RNN becomes incapable of learning to connect the information.

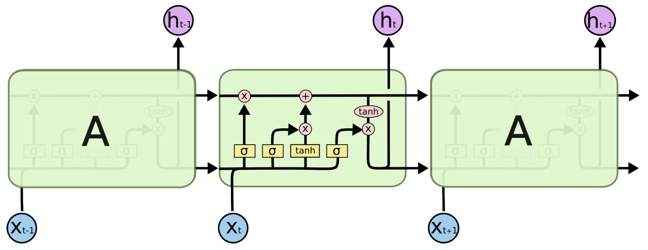
Long Short-Term Memory network (LSTM) [48] is developed from tradition RNNs that is able to learn long-term dependencies. From Figure 1.12b and 1.12c we show the differences of the recurrent block between a standard RNN and the LSTM network, such that a standard RNN has a simple one-layer ($tanh$) structure inside the recurrent block whereas LSTM adds 3 sigmoid layers inside the block having a more complex structure. The sigmoid layers help to decide whether the input should be pass on to the next step or not. Within LSTM, the information go through a cell states mechanism and there are several different types of gates to help the network selectively deciding what inputs to remember or forget. LSTM is capable of reserving data for a longer period of time as well as balancing between important information and the ones that are not considering relevant.



(a) An unfold recurrent neural network



(b) Inside of the recurrent block of standard RNN



(c) Inside of the recurrent block of LSTM

Figure 1.12: Illustration of RNN structures

CNN-based Video Classification Models

Videos are another interesting data type from the perspective of dimensionality in modern deep learning researches. A video is usually treated as a stack of image sequences with temporal component that can be fed into CNN models. There are two general concerns with video datasets, one is that loading the entire dataset into local memory is certainly impractical and second the length of each video within one dataset normally varies.

Since a video clip can be seen as a sequence of images then a decision on how should we select the most informative frames as input to the CNN is needed to be made. [60] proposed four different strategies of combining frames. First one is *Single*

Frame that simply takes one frame as input and aggregates predictions across all the single frames. The *Late Fusion* concatenates the first and last frames and *Early Fusion* takes a contiguous segment from the clip. The last one is *Slow Fusion* that takes four partially overlapping contiguous segments and then progressively combines them in the network. Detailed diagrams of these four approaches can found in Figure 1.13. [60] also proposed a multi-resolution CNN model for image processing. Two sets of downsampled inputs, one is half of the original spatial resolution and the other one is the center region, are fed respectively into two series of convolutional layers and the activations from both streams are concatenated before pass to the fully-connected layer. Additionally [60] published an annotated dataset called Sports-1M as an benchmark for model evaluation.

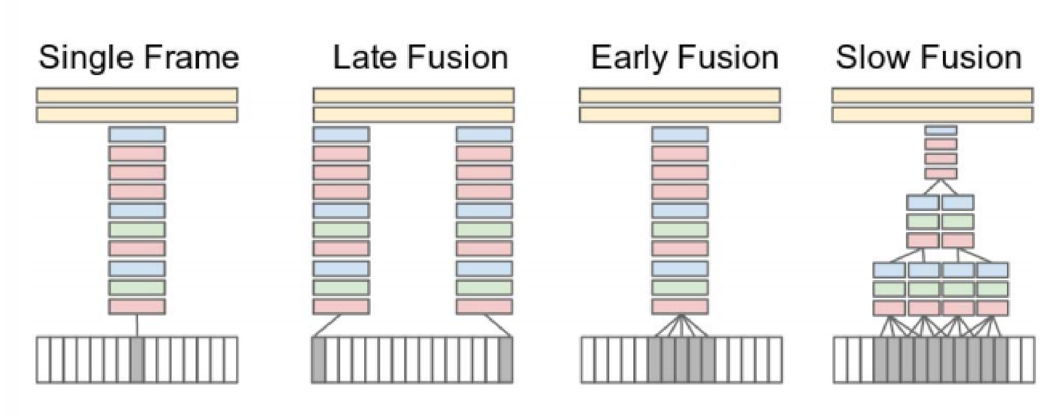
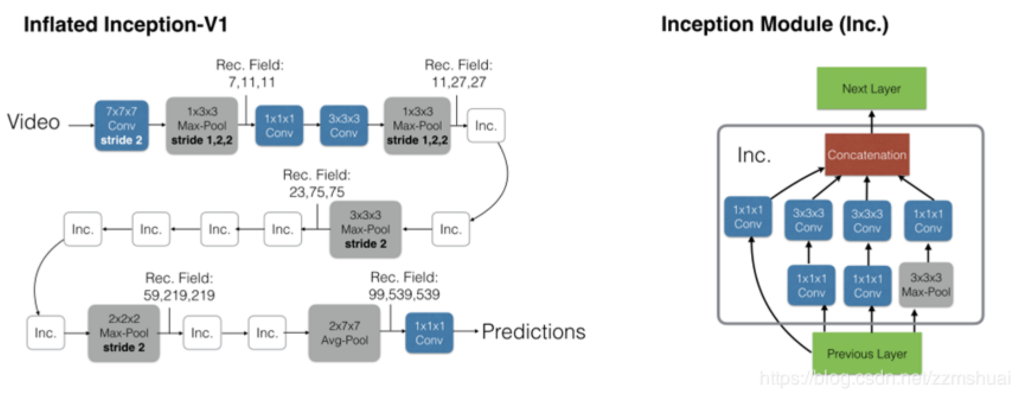
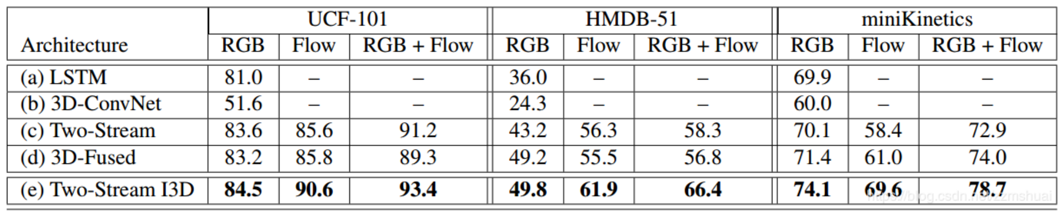


Figure 1.13: Four strategies for combining information over temporal dimension through the network [60]. Red, green and blue boxes indicate convolutional, normalization and pooling layers respectively.

[60] is a single stream network using 2D pre-trained convolutions. [124] on the other hand is a two-stream network consisting of a pre-trained spatial network and a temporal network for motion context. Motion features are formalized as stacked optical flow vectors. The input to the spatial net is a single frame of the video and for the temporal net the input is the optical flow across successive frames. The two streams are trained separately and then outputs are combined using SVM.

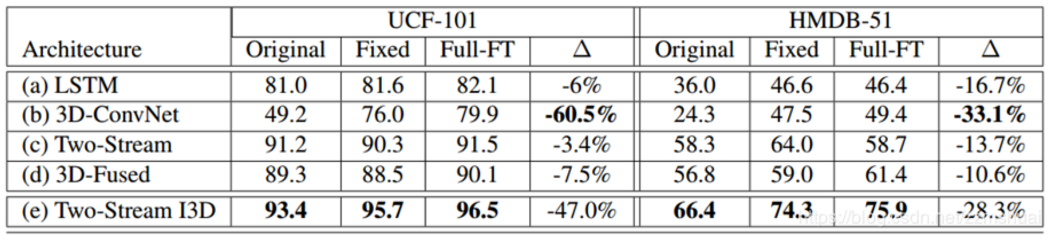


(a) The inflated Inception-v1 (left) of I3D [16] and the inside of inception module (right)



Architecture	UCF-101 RGB	UCF-101 Flow	UCF-101 RGB + Flow	HMDB-51 RGB	HMDB-51 Flow	HMDB-51 RGB + Flow	miniKinetics RGB	miniKinetics Flow	miniKinetics RGB + Flow
(a) LSTM	81.0	–	–	36.0	–	–	69.9	–	–
(b) 3D-ConvNet	51.6	–	–	24.3	–	–	60.0	–	–
(c) Two-Stream	83.6	85.6	91.2	43.2	56.3	58.3	70.1	58.4	72.9
(d) 3D-Fused	83.2	85.8	89.3	49.2	55.5	56.8	71.4	61.0	74.0
(e) Two-Stream I3D	**84.5**	**90.6**	**93.4**	**49.8**	**61.9**	**66.4**	**74.1**	**69.6**	**78.7**

(b) Architecture comparison on UCF-101, HMDB-51 and Kinetics



Architecture	UCF-101 Original	UCF-101 Fixed	UCF-101 Full-FT	UCF-101 Δ	HMDB-51 Original	HMDB-51 Fixed	HMDB-51 Full-FT	HMDB-51 Δ
(a) LSTM	81.0	81.6	82.1	-6%	36.0	46.6	46.4	-16.7%
(b) 3D-ConvNet	49.2	76.0	79.9	**-60.5%**	24.3	47.5	49.4	**-33.1%**
(c) Two-Stream	91.2	90.3	91.5	-3.4%	58.3	64.0	58.7	-13.7%
(d) 3D-Fused	89.3	88.5	90.1	-7.5%	56.8	59.0	61.4	-10.6%
(e) Two-Stream I3D	**93.4**	**95.7**	**96.5**	-47.0%	**66.4**	**74.3**	**75.9**	-28.3%

(c) Performance improved using pre-training on Kinetics. Original: No pre-training implemented; Fixed: features from Kinetics, fine-tune and train the last layer; Full-FT: pre-training with end-to-end fine-tuning; Δ shows the difference in misclassification as percentage between Original and the best of Full-FT and Fixed.

Figure 1.14

As previously discussed LSTM is capable of sequence processing and capturing temporal information. [30] was built on the idea of using LSTM decoder after the convolutional operation for video representations. The architecture is trained end-to-end with two types of inputs: RGB and optical flow. Results showed that by averaging predictions over both inputs made it achieved the best performance among the others. [132] used assembled 3D convolutional networks that was pre-trained on Sports-1M benchmark dataset as feature extractors for other dataset. During training 3D convolutional operations were applied as a spatiotemporal cube on the input. The deconvolutional layers were adopted in order to understand what the network was learning internally. By conducting methodical research on proper temporal filter length it demonstrated that 3D ConvNets outperformed 2D ConvNets on various video analysis tasks. [150] is another work utilizing 3D ConvNets but incorporated with LSTM as an encoder-decoder architecture for video description tasks. Additionally their work first introduced attention mechanism for video representations.

Other two-stream networks evolved from [124]. [36] focused on multi-level fusion of spatial and temporal streams. [141] suggested sampling clips sparsely through the video based on the idea of long-range temporal structure modeling. They also experimented with typical CNN regularization techniques such as batch normalization, dropout, etc, to improve the performance of ConvNets on video data. Two new input modalities were introduced as alternates to optical flow: warped optical flow and RGB difference. [155] recommended to use an unsupervised architecture to capture motion information between adjacent frames without explicitly computing optical flow.

As a considerable number of successful architectures for image classification has been developed over the decades, intuitively we should be able to adapt robust 2D CNN architectures for temporal data processing. In [16], a Two-Stream Inflated 3D ConvNet (I3D) is proposed which is built upon on 2D ConveNet inflation. I3D is expanded from a 2D inception-v1 network. We show the inflated Inception-V1 architecture in Figure 1.14a. [16] conducted several arrangements to complete the expansion. First all the filters and pooling kernels were boosted by adding an additional temporal dimension (i.e., a $N \times N$ 2D filter became $N \times N \times N$). Next a boring-video (simply copying single frames repeatedly) fixed point was introduced to bootstrap parameters from pre-trained ImageNet models by repeating the weights of 2D filters N times along the time dimension and then divided by N for re-scaling. The last step was to pace the receptive fields' growth in space, time and network depth. RGB inputs and optical flow were trained separately and predictions were averaged during testing.

A new human action video dataset (Kinetics) that has two orders of magnitude more data than HMDB-51 [64] and UCF-101 [124] is provided for re-evaluation on state-of-the-art video recognition models. Kinetics contains 400 human action classes and over 400 examples per class. [16] compared the performance of several well-known temporal analysis architectures (i.e., LSTM, 3D-Convnet, Two-Stream, 3D-Fused Two-Stream) with I3D on Kinectics as well as the other two datasets. Details are shown in Figure 1.14b and results in Figure 1.14c validate that using weights pre-trained on Kinectics benefits all the models in the comparison scheme.

Deep Learning for Optical Flow Estimation

Optical flow is defined as the instantaneous velocity of the pixel movement on the image plane. It calculates the change of pixels caused by the movement of foreground objects or the camera in image sequences to capture motion information. Since it is able to characterize and quantity apparent motion, optical flow estimation has been studied and widely applied in computer vision tasks, such as object detection and tracking, video compression, motion-based segmentation. Figure 1.15 shows an example of optical flow applications on recognizing human actions based on pose relationships between frames.

The concept of optical flow was first proposed by Gibson in the 1940s. Later on Horn and Schunck introduced the optical flow constraint equation in 1981 as the basic algorithm for optical flow calculation. It assumes constant brightness that the pixel intensity of object does not change between frames. The equation has two unknowns and thus requires additional constraints in order to find a unique solution. This is known as the aperture problem in optical flow algorithms. Horn and Schunck then suggested a global smoothness constraint [49] claiming that the motion speed of pixels are the same or close to their neighbor pixels in an image.

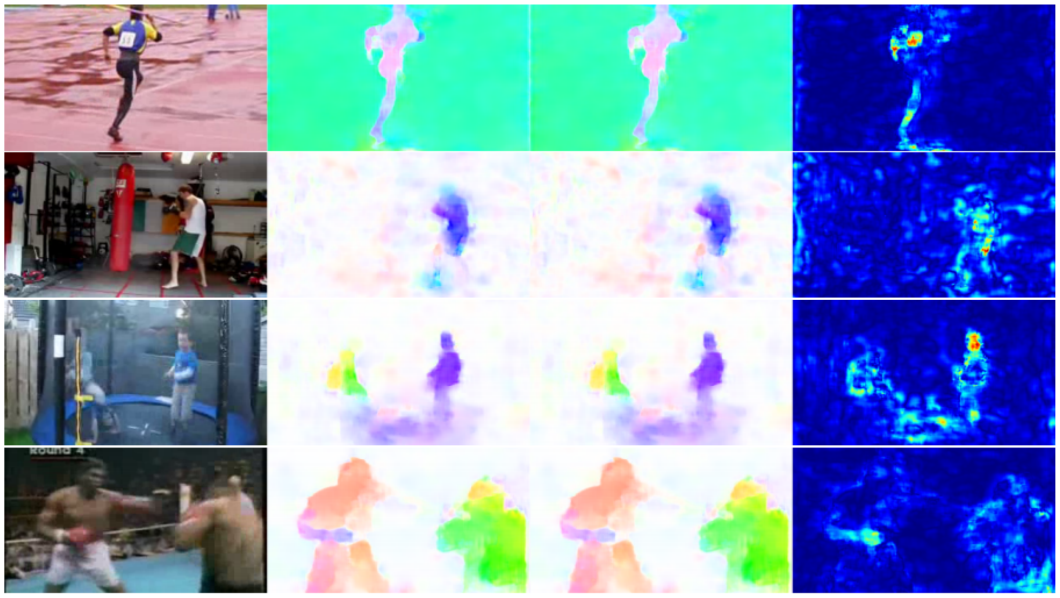


Figure 1.15: Examples of classifying actions with optical flow [32]

Lucas and Kanade [78] proposed to calculate dense optical flow using local windows around points of interest based on the assumption that the flow is essentially constant in a local pixel neighbourhood. The Lucas-Kanade method deals with the inherent ambiguity of the optical flow equation and is less sensitive to noise. However it has trouble detecting large motion that is exceeding the local window. [12] improved the Lucas-Kanade method by using image pyramid hierarchy on tracking large moving targets.

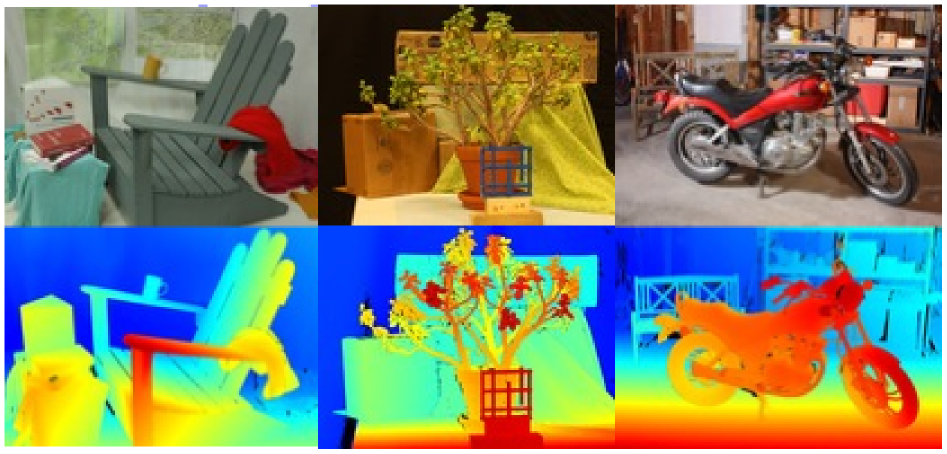
Since the work of [49], variational approaches have been leading the development of optical flow estimation [83, 13, 14]. DeepFlow [143] combined a descriptor matching algorithm with a variational method to handle large displacements. EpicFlow [112] generated dense matches to initialize variational energy minimization. Traditional methods of flow estimation often require large and complicated computation and hence are not suitable for real-time applications. The evolution of CNN-based flow estimation approaches, on the other hand, detects regions of motion in a more effective and accurate fashion.

We first introduce the most commonly used benchmarks on evaluating performance of optical flow estimation models. The Middlebury [120] is the first dataset of real image sequences with independent motions and optical flow groundtruth. However it only consists of 10-15 image pairs for training and testing which is too small and insufficient to evaluate CNN-based optical flow models. Examples are shown in Figure 1.16a. The KITTI [41] dataset is recorded by a moving autonomous driving platform including cameras, laser scanners and other sensors for measurements. The full benchmark contains optical flow, stereo, video odometry, depth, etc, with groundtruth annotations. The latest KITTI dataset for optical flow has 200 training scenes and 200 test scenes. As the purpose of KITTI is to support the development of computer vision algorithms on the task of autonomous driving, motion of distant objects are not captured (Figure 1.16b. The MPI Sintel benchmark [15] is composed of synthetic data that derived from a 3D animated short film. It contains long sequences, large and varied motion, different scene structures and illumination, etc. Motion blur and atmospheric effects are applied on the images rendered from artificial scenes (Figure 1.16c). The dataset contains over 1000 training image pairs with dense groundtruth.

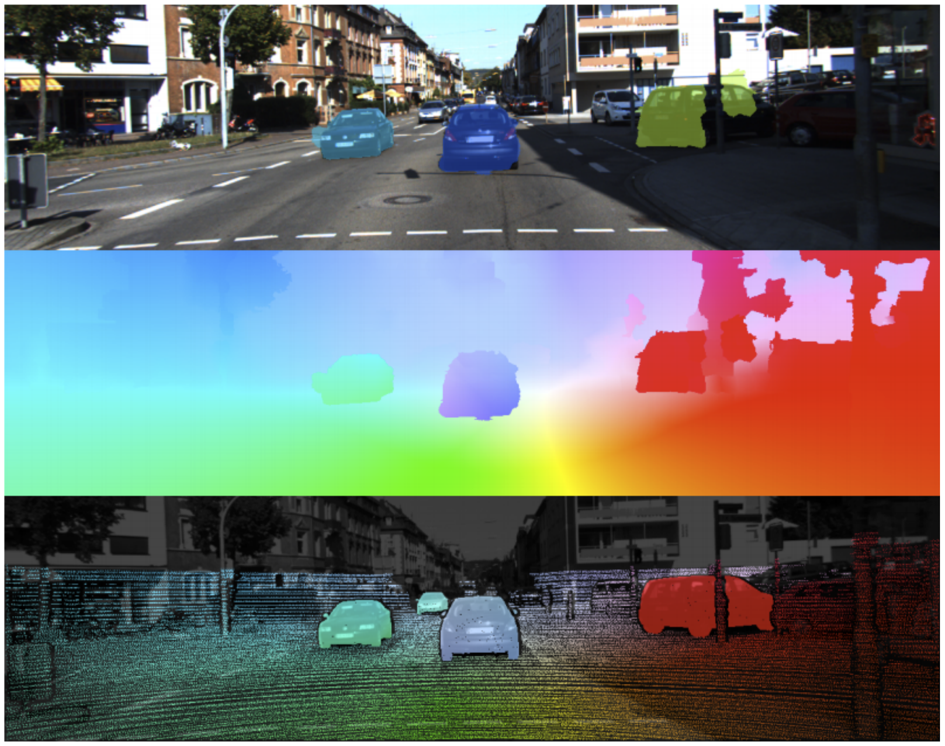
Fischer *et al.* [38] first proposed the idea of using a CNN architecture, i.e. FlowNet, to learn and predict optical flow directly from image pairs. The authors experimented with two types of implementations to evaluate the learning ability of the networks. One was to train a single network on stacked input images, named FlowNetS. The other one, FlowNetC used two subnetworks to process images separately and then combined their representations at later stage. The two network architectures were trained and tested on the well-known existing benchmarks (Middlebury [120], KITTI [41], MPI Sintel [15]) of optical flow estimation as well as a synthetic dataset created by the authors. Results show that FlowNetC suffered from overfitting and had difficulties dealing with large displacements. However the two network structures proposed achieved close error rates comparing to traditional well-performed methods on optical flow estimation and most importantly using CNNs made it be capable of transferring the entire learning process onto GPUs which largely reduced the runtime.

Instead of image warping, LiteFlowNet[53] proposed to use feature warping to reduce the feature-space distance and thus cut down computational complexity. We show their network architecture in Figure 1.17. It splits feature extraction and flow estimation into two sub-networks, i.e. NetC and NetE. NetC serves as a feature descriptor that transfers an input image pair into two pyramids of multi-scale high-dimensional features. NetE consists of a cascaded flow inference that generates flow fields and flow regularization modules to correct vague flow boundaries. Figure 1.18 presents example flow fields generated from different methods on the KITTI[41] benchmark dataset. LiteFlowNet [53] generalizes better to real-world data than SPyNet [105] and FlowNet2 [55] as it is able to preserve fine details with less artifacts and clearer flow boundaries. LiteFlowNet [53] also has less training parameters and faster runtime comparing to the other two CNN-based flow estimation approaches discussed above.

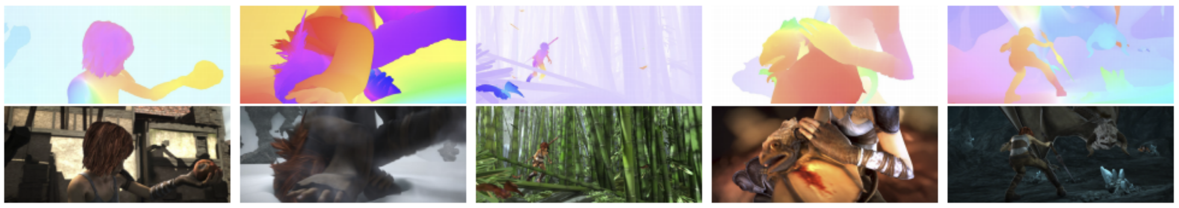
FlowNet [38] has proved the capability of neural networks for predicting optical flow directly from images but it is still having limitations in real-world applications. FlowNet2 [55] proposed to improve the performance by stacking multiple networks with image warping on the second images. They also explored the idea that the order of data presented during training played an important role in supervised learning and helped to optimize the network on small motions with small displacements. FlowNet2 [55] outperformed FlowNet [38] in accuracy while maintaining a fast runtime on GPUs however it introduced a large amount (about 160M) of parameters and hence increased the computational complexity. Ranjan *et al.*[105] proposed to use a spatial pyramid network architecture that a deep neural network was trained at each level of the pyramid to estimate a flow instead of solely training one deep network. Comparing to FlowNet2 [55], SPyNet [105] drastically reduced the numbers of parameters in the network but its accuracy was only comparable with FlowNet [38].



(a) Middlebury [120]



(b) KITTI [41]



(c) MPI Sintel [15]

Figure 1.16: Example data and groundtruth of the optical flow benchmarks

LiteFlowNet2 [52] optimized the network structure of LiteFlowNet [53] in the following aspects: First, it has a reduced numbers of pyramid level in NetE to decrease computation time spent on the flow decoder; Second, the depth of NetE is limited with two additional convolutional layers to compensate the loss; Third, a simplified pseudo flow inference is introduced to refine the estimates. LiteFlowNet2 [52] runs 2.2 times faster and achieves higher flow accuracy on the KITTI [41] benchmark dataset than LiteFlowNet [53].

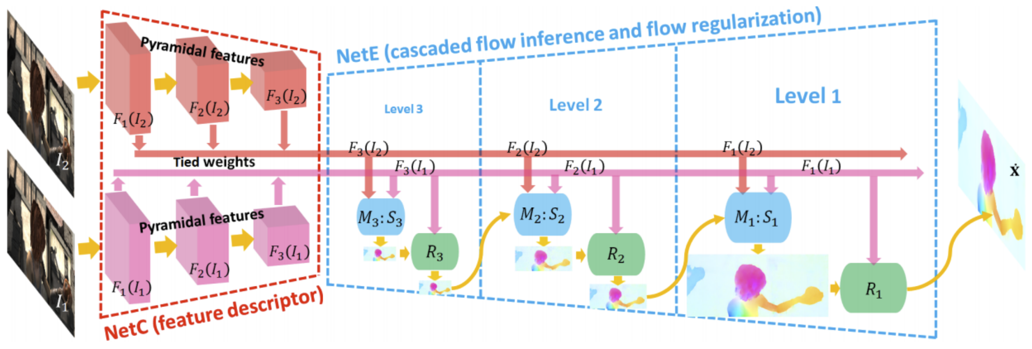
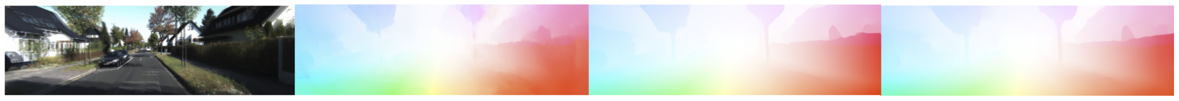
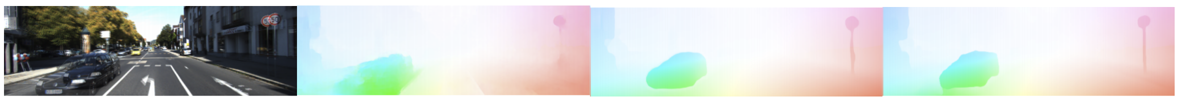


Figure 1.17: The network structure of LiteFlowNet [53]



(a) From left to right:KITTI image overlay, SPyNet[105] FlowNet2[55] LiteFlowNet[53]



(b) From left to right:KITTI image overlay, SPyNet[105] FlowNet2[55] LiteFlowNet[53]

Figure 1.18: Examples of flow fields from different methods on the KITTI benchmarks

Chapter 2
IMAGE-BASED MARINE SPECIES POPULATION MONITORING

The Atlantic sea scallop (Placopecten magellanicus), pictured in Figure 2.1a, is highly important economically. Although mobile, it is not migratory, and it is commonly found on the sea floor in the mid-Atlantic at a 35 - 100 m depth on sand and gravel sediments. Wild scallops are typically caught using a dredge dragged along the seabed, accurate and timely assessments of local population numbers and size/age distributions are important for setting sustainable quotas. Historically, censuses have been conducted by systematically sampling different locations by dredging, but recently there has been great interest in increasing the efficiency and coverage of this process (including for other species) through analysis of images obtained by fixed, towed, and AUV-borne cameras. Thus far, such analysis has been either manual [113, 37] or primarily based on hand-selected features [6, 26, 59]. In this chapter, we demonstrate the efficacy of applying deep learning methods to achieve fast and accurate scallop detections over a range of substrates, despite suboptimal image quality. In the first section, We are concerned primarily with the problem of detection, or placing bounding boxes around an unknown number of scallops in each image as illustrated in Figure 2.1c. We studied the behavior and performance of different CNN-based visual detectors on the scallop dataset. In the second section, we extend our work to multi-class classification tasks. Because dredging can cause habitat damage and mortality among uncaught scallops, detected scallops are further categorized as *healthy* vs. *compromised*. Also, other creatures such as starfish, monkfish, crab are of scientific interest as well in order to investigate the predator-prey relationships.

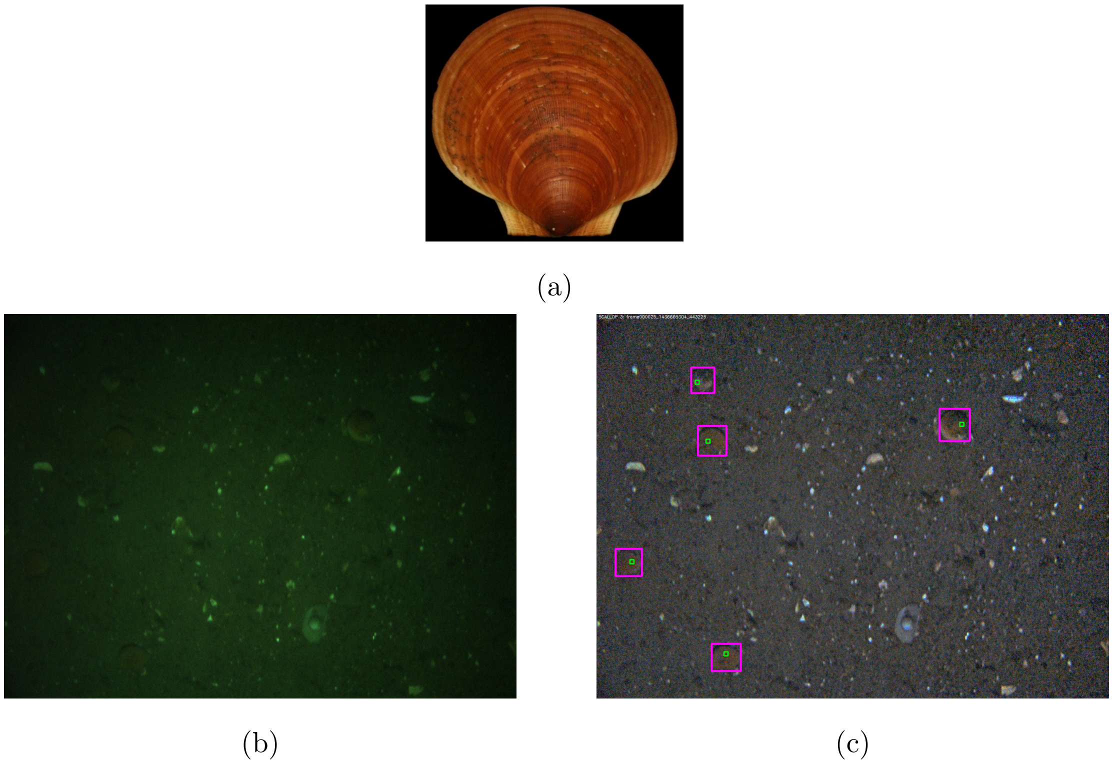


Figure 2.1: (a) P. magellanicus; (b) Raw seabed image taken from AUV; (c) Corresponding image after contrast enhancement, with scallops manually annotated

2.1 CNN-based Sea Scallop Detection

2.1.1 Background on Image-Based Benthic Creature Detection

A fixed underwater camera was used for decapod (e.g., lobster) detection in [6]. A top-hat transform was applied to images acquired at hourly intervals, followed by thresholding of the distance between the red and green color channels to obtain a binary saliency image. Fourier descriptors and SIFT features were extracted on large connected components, which were then classified using partial least squares discriminant analysis.

An early approach to scallop detection relied on their regular shape, employing a Hough transform to look for circles [33]. Kannappan *et al.* [59] presented a layered saliency-based approach to identifying scallops in AUV images in which top-down visual attention was followed by segmentation, shape extraction, and classification. The image analysis pipeline was hand-tuned, and their results showed a high number of false positives associated with acceptable recall levels.

Another computer vision system for scallop detection, based on the object identification paradigm, was presented in [26]. Images collected from the HabCam towed camera system [130] were subjected to illumination and color correction, then likelihood images based on grayscale and color histograms were formed, and finally four hand-selected operators were applied to identify candidate regions. For each candidate region, a variety of color, texture, and edge features were extracted and fed to a series of cascaded AdaBoost classifiers [39, 119] for the final detection decision. The multiple classifiers were tuned to different substrates and scallop types, but the results were generally quite good.

2.1.2 Problem Statement

We started by focusing on simply identifying healthy scallops, making the task a 1-category detection problem.

Our strategy is to leverage access to an existing dataset of 190, 000+ manually-annotated images captured in scallop-rich waters in order to learn a sufficiently robust visual representation of *P. magellanicus* that the system can recognize individual scallops at different stages of their life cycle and on a wide variety of substrates. The images in the dataset were collected by a downward-pointing digital camera in the nose of an AUV (shown in Figure 1.1) moving at an altitude of a few meters above the seabed. Despite illumination with a flash strobe to compensate for low ambient light at the operating depth of 50+ m, the raw images are quite dim (a sample is shown in 2.1b) and the scallops are often difficult to discern even to human eyes. Other confounding issues include the small size of the scallops in the image; non-uniform illumination or vignetting from the flash; a wide range of similar-looking features such as clams, small rocks, shell hash and sediment textures. partial burial of some scallops in sand or gravel; and occlusions by swimming creatures. Despite these difficulties, we demonstrate that by training deep learning detection networks with minimal modifications on just a fraction of the annotated dataset, the system can achieve superior performance on the task. We evaluate and compare the performance of YOLOv2 [109], YOLOv3 [110] and RetinaNet [72] detectors on the scallop dataset. All the three networks mentioned have been proved that they have the ability to work extremely well and at high speed on benchmark datasets such as PASCAL VOC [34], MS COCO [70] crossing classes of objects as diverse as people, bicycles, chairs and birds. Training on scallops is not completely straightforward, however, as the available annotations contain approximate position information only and must be converted to precise bounding boxes. One novelty of this work is that we proposed a method for automatically upgrading these annotations by training a separate denoising auto-encoder network, thereby enabling efficient augmentation of the training data.

2.1.3 Scallop Dredging Dataset

Since 2014, 5 Cruises of 95 missions were conducted off the east coast of the United States at bottom depths of 50-80 m in two areas (shown in Figure 1.2a): the Elephant Trunk Closed Area (ETCA) and Nantucket Lightship Closed Area (NLCA), for a Before-After Control-Impact (BACI) [126] study of the effects of dredging on incidental scallop mortality [37].

During the study each area was divided into 3 sites, and at each site 3 types of experiments were run, denoted by A, B, and C. A experiments consisted of a pre-dredging AUV *mission*, followed by 1 dredging tow by a scallop fishing vessel, and then a post-dredging AUV mission hours later to assess the effects of the dredging. B experiments were the same, except that the dredge was towed through the site 5 times for a heavier impact. C experiments were controls where no dredging was done, but the AUV went on tow missions. Over the course of the study, ETCA was surveyed twice and NLCA once, resulting in a total of 54 AUV missions which we refer to as `M1`, `M2`, ..., `M54`.

In order to photographically cover the area where the dredging occurred, each AUV mission followed a preset "lawn-mowing" pattern as depicted in Figure 1.2b.

Using its inertial navigation system (INS) and Doppler velocity log (DVL) for state estimation, the AUV followed 10 parallel lines 2 m apart and 750 m in length (ETCA), or 14550 m long lines (NLCA), at an altitude of 2.5 m above the sea floor. Given the AUV camera's intrinsic parameters and altitude, each raw 1280×960-pixel image such as that shown in Figure 2.1b represents a seabed area of roughly 1.88 m $\times$ 1.45 m (an area of 2.73 m^2). The images were captured at 3.75 Hz with the AUV traveling at a speed which resulted in an average overlap of 45% between consecutive images in a line (Figure 1.2c). Images were not captured while the AUV was turning to begin the next line, so there is a discontinuity between each line of image.

2.1.4 Post-processing and Manual Annotation

Due to the depth of the AUV and the limitations of the strobe used to illuminate the seabed, raw captured images exhibit low contrast that makes scallop detection difficult (Figure 2.1b). Accordingly, brightness and color contrast were enhanced on every image using one of two post-processing procedures: either a multiscale retinex algorithm or a stretch contrast function which works on each color channel independently, as detailed in [37]. A sample result of contrast enhancement is shown in Figure 2.1c.

In order to obtain ground-truth information for the dataset, a team of 15 student annotators were trained to identify scallops (as well as other creatures) and mark them as healthy or compromised using an online annotation system [37]. Each annotator was directed to click *somewhere* inside the shell perimeter, but not necessarily the center. We call this a *rough position*. The green dots (inside the purple boxes) in Figure 2.1c are a sample of these original scallop position annotations. At ETCA sites 1 and 2, images were downsampled by 2 before annotation - every other image was skipped. However, annotating a single image took an average of almost 30 seconds, so to speed the process the images were downsampled by 8 before annotation at ETCA site 3 and all NLCA sites. In total, 171,860 images were annotated in this fashion during the study at a cost of almost 1,150 person hours [37].

In order to train the detector networks, each scallop needs to have a tight bounding box. For this work, we create a tool to manually *upgrade* the existing scallop *rough position* data to *precise position and scale*. Bounding boxes were constrained to contain an existing rough position, and allowed to be non-square. Where scallops appeared to be partially buried or not completely inside the image, the annotator only indicated the visible part.

We applied this tool to `M49`, a post-dredging mission from NLCA containing 2,430 images with 4.267 original rough position annotations of all types of creatures. 1,736 of these images contained at least one healthy scallop, of which there were 3,863 total. Precise bounding boxes were added for these; sample for one image are shown as purple rectangles in Figure 2.1c. This entire process took roughly 20 hours.

Other missions which we will reference from the original study dataset:

- `M46`: `M49`'s pre-dredging twin with 1,857 annotated images that contained at least one healthy scallop; there were 4,835 healthy scallops in all.
- `M6`: An ETCA mission with 5,204 healthy scallop containing images (because of less downsampling); 8,663 healthy scallops total.

2.1.5 Automatically Upgrading Rough Positions to Bounding Boxes

Many studies have supported the rule of thumb that machine learning performance is improved through a larger and more diverse training set, both in general [8, 45] and most recently for CNN-based object detection/recognition [127]. One can also observe on the leaderboard for the widely-used PASCAL VOC detection challenge that mAP scores increase dramatically for algorithms which use COCO + PASCAL data (100K examples) vs. PASCAL alone (10K examples) [34, 70].

In our dataset, the rough positions in the entire original set of annotations represent one to two orders of magnitude more training examples than `M49` alone, yet the detection networks training requires that each of these be converted to a bounding box. Based on our experience with applying the software tool described above to `M49`, doing this manually for the entire dataset could take up to 1000 hours, almost as much as time as the original annotation project.

Instead, we propose to use a neural network to automatically upgrade the original rough positions to bounding boxes. We found that a 160×160 subimage centered at a rough position is sufficient to completely contain any size scallop in our dataset (see the first row of Figure 2.2 for several sample subimages), so the task of outputting exactly one bounding box for each subimage known to contain a scallop is equivalent to pure *localization* [122]. The difficult work of deciding *whether* there is an object at a particular AUV image location, and *what kind* of object it is, has already been done by human annotators, so we want to exploit this.

Directly regressing a bounding box (i.e., outputting coordinates) from an image is a highly non-linear and difficult-to-learn mapping [131, 98]. We were unsuccessful in our attempts to do it with a variety of convolutional front-ends, including Darknet-19, attached through several fully-connected layer to an (x,y,w,h) output layer. Rather, we borrow an idea from [98], which labels joint locations in images of people, and train the network to output a *scallop likelihood image*, or "heatmap", where high-value pixels belong to the scallop and low-value pixels to the background. In [98], at training time each ground truth joint position is indicated by a fixed-variance Gaussian (each joint has a separate heatmap) and the loss is the sum of squared differences. At test time, each joint's predicted heatmap is post-processed to obtain the most likely joint location as the brightest pixel location. We want to generalize this to learn *masks* of scallop pixels, such as shown in row 2 of Figure 2.2.

One way to think of the heatmap network is as a kind of *denoising auto-encoder* [44]. Input images can be regarded as clean scallop masks transformed and corrupted by appearance + lighting + background variation, plus artifacts introduced by the contrast enhancement stage. The network must learn to extract the essential information-the geometric attributes of the scallop-form which it can reconstruct the original mask minus any image "noise". Here we use a simple auto-encoder architecture consisting of an AlexNet CNN [63] encoder (5 convolutional and 3 max-pooling layers with ReLU activations), 2 fully-connected layers of 4096 units each, and a decoder network that inverts the AlexNet structure through a series of upsampling "deconvolutions".

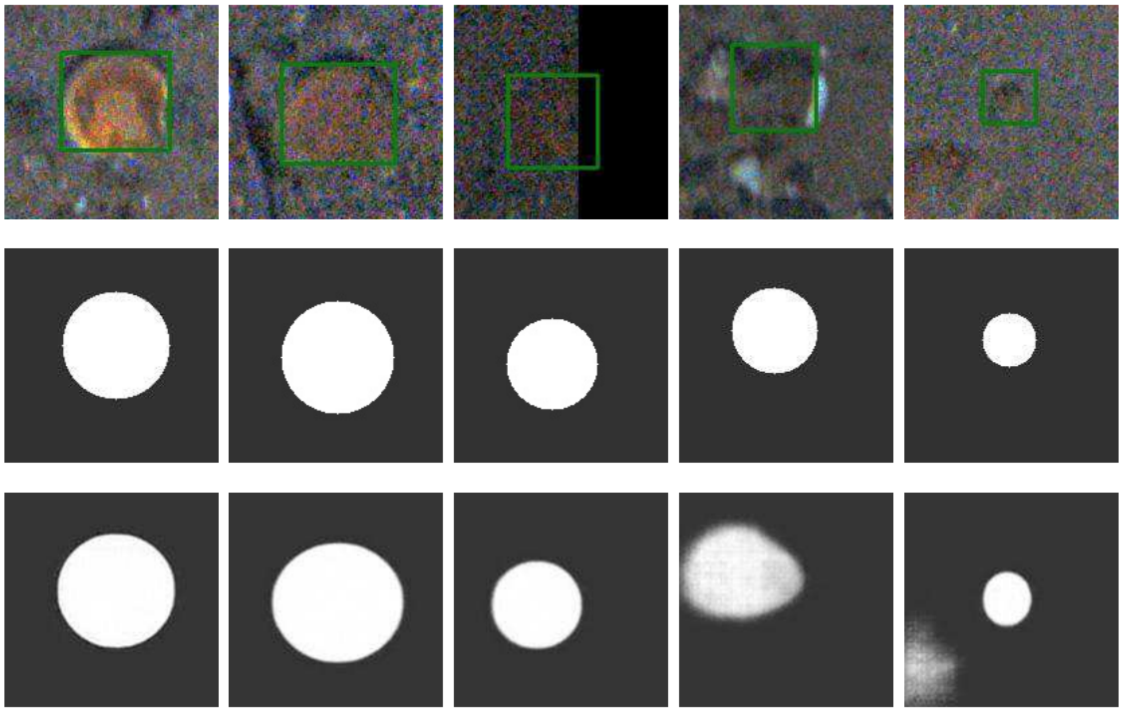


Figure 2.2: (1st row) Sample scallop subimages from `M49` test set centered on rough position annotations with ground truth bounding boxes overlaid; (2nd row) Corresponding circular masks used to train heatmap neural network; (3rd row) Corresponding outputs of heatmap network

2.1.6 Training Procedures

Both of the YOLO networks and heatmap networks were trained and tested using the `Darknet` open source neural network framework [107]. RetinaNet framework was cloned from [40]. We modified the configuration files where the network architectures are defined to change the number of filters of the last convolutional layer because the number of object categories differs from the 20 classes of the original PASCAL

VOC task. Unless noted, all other parameters such as learning rate, batch size, etc. were not changed, and training started from random weights.

2.1.6.1 Heatmap Network

Following [98], we first train on fixed-radius circular masks($r = 10$) to learn position only. Example subimages were taken from the **HighRes** training set (see section 3.5.2 below), with each one randomly distorted geometrically and photometrically. This phase lasted 50K iterations with a learning rate of 10^{-4}. Then, because of the need for a scale estimate, there is a second phase of fine-tuning in which the targets are circles scaled to match the maximum dimension of each scallop, as shown in row 2 of Figure 2.2. When a scallop is only partially visible at the edge of the image, such as in column 3 of Figure 2.2, we draw the entire inferred scallop mask and background. This phase lasted 475K iterations with the learning rate dropped to 10^{-5}.

At test time, the scallop heatmap (examples of which are shown in row 3 of Figure 2.2) is threshold to make a binary mask and the bounding box of the largest connected component is output. The threshold was chosen to maximize the median overlap between the predicted and ground truth bounding boxes as calculated by the intersection over union (IoU) formula over the training set.

2.1.6.2 YOLOv2 Detector Network

We conducted four major experiments varying the training and structure of the YOLOv2 detector network. The first three experiments below used only the `M49` annotations, while the fourth incorporated data from `M6` and `M46`.

NoEdges

In this experiment any AUV image with a scallop too close to an edge (i.e, the distance from any part of the bounding box to any edge is ≤ 10 pixels) was discarded. This was under the assumption that cropped bounding boxes would undermine the learning process by suggesting incorrect dimensions for partially visible scallops. After filtering, the dataset was made up of 1,664 images containing 3,436 scallops, or 81.6% of the original `M49` data. The images were split 80/20 for training/testing: 1,331 images (and all of their scallops) in the training set, and 333 images in the test set. The YOLOv2 network was trained for 10K iterations with a batch size of 64. The entire training process took 12.5 hours.

Edges

Here all of the original scallops in the `M49` dataset were used without regard for image edge proximity. Again using an 80/20 split, the training set has 1,389 images with 3,121 scallops and the test set had 347 images with 742 scallops. We trained YOLOv2 with this approach for 17K iterations.

HighRes

In order to better resolve small scallops, we doubled the width and height of the network's input layer to 832×832 to preserve image detail. This allows the network to "see" approximately 61% of the original AUV image information instead of 14% as in the first two experiments. The total number of images and train/test split was the same as for the **Edges** network. The network was trained for 20K iterations.

AugmentedHighRes

The `M49` manual bounding box annotations used to train **HighRes** were augmented with bounding boxes generated automatically by the heatmap network by upgrading all rough position annotations from `M46` and `M6`. This amounted to a total of 13,498 scallops from 7,601 images. YOLOv2 training was performed as fine-tuning from the **HighRes** 4K weights, continuing for another 10K iterations. This additional data constitutes more than $4\times$ the size of the **HighRes** training set alone.

2.1.6.3 YOLOv3 Detector Network

We trained YOLOv3 detection network on a larger dataset consisting of 67 missions from cruise 1, 3 and 5. There are 97,344 images containing 373,806 scallops and each image has at least 1 healthy scallop. We mixed raw images (shown in Figure 2.6g - 2.6i) from cruise 5 and Retinex-enhanced images (shown in Figure 2.6a - 2.6c) from cruise 1 and 3 for this dataset to verify the robustness of the network across both types of image. The images were split 80/20 for training/testing: 77,875 images with 300,927 scallops in the training set and 19,469 images with 72,879 scallops in the test set. The YOLOv3 network was trained for 20K iterations.

2.1.6.4 RetinaNet Detector Network

In order to perform a fair comparison of RetinaNet and YOLO detectors, we trained the RetinaNet detector on the same datasets as mentioned above, including validation on a separate mission. Additionally, we added background images as hard negative examples as suggested in the [72] paper to evaluate the detector's effectiveness on addressing the issue of foreground-background class imbalance encountered during training.

We also evaluated RetinaNet on a 2-class classification problem. Two categories are *Alive Scallop* and *Sea Star*. We consider this as a preliminary experimentation for our mortality rate estimation in Section 2.2.1.

Primitive groundtruth information indicated objects of interest using line segments. Noted that each line segment is not necessarily across the center of object. Thus we first programmed to convert the original scaled annotations to groundtruth bounding box annotations and translate YOLO-formatted coordinates into required format for training the RetinaNet detector.

Datasets used for the RetinaNet detector experiments:

- YOLOv2 *Edges* dataset: 1,736 images in total from the `M49` mission, 1,389 images for training and 347 images for testing.

- YOLOv3 dataset: 97,344 images from 67 missions, 77,857 images for training and 19,946 images for testing.
- 1075 background images, i.e.images do not contain any object of interest, from the `M49` mission.
- *Scallop-Star* dataset: 81,363 images from 67 missions of cruise 1,3,and 5 containing 223,409 scallops and 7,452 stars.

We used a train/test split ratio of 80/20 for all datasets. Each model was trained using ImageNet pre-trained weights and ResNet50 as backbone. All training processes lasted for 50 epochs.

2.1.7 Results

2.1.7.1 Heatmap

We evaluate the heatmap network's ability to accurately upgrade rough positions in a few ways. At the end of training, the mean IoU between the predicted and ground truth bounding boxes on the `M49` test set is 0.80, the median is 0.82, the min is 0.35, the max is 0.99, and 97.4% of upgrades are "correct" using the standard threshold of IoU ≥ 0.5. For comparison, on the training set itself the mean IoU is 0.90, median is 0.92, min is 0.36, max is 1.0, and 99.6% of upgrades are correct.

The predicted bounding boxes are robust to different backgrounds from other missions not seen during training. In particular, `M6` was run in a completely different area of the Atlantic, and many of the images have features not seen in `M46/49`. A sample of the bounding boxes created for this mission are shown in Figure 2.3. Another piece of evidence is that when the YOLOv2 detector is trained with these automatically-created bounding boxes in **AugmentedHighRes**, performance as measured by average precision improves modestly (see below).

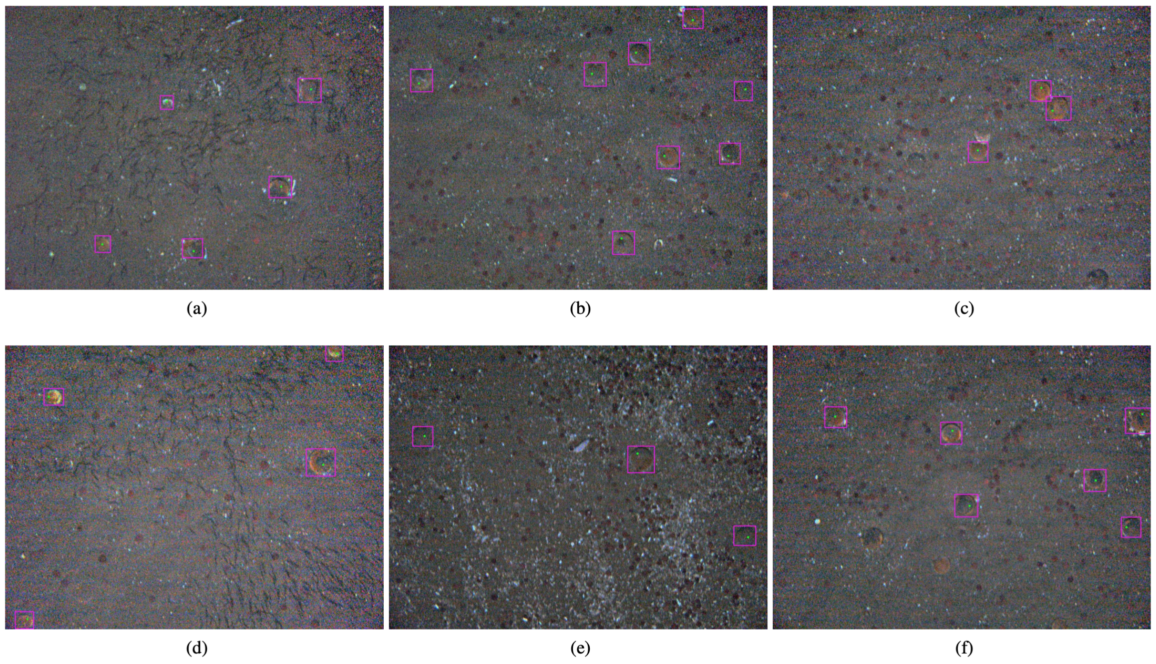


Figure 2.3: Example annotation upgrade results from the heatmap network on `M6` images. Green dots indicate the rough position annotations, pink boxes are the predicted bounding boxes. Because the original annotators did not mark them, the seemingly missed scallops in (f) are not upgraded.

2.1.7.2 YOLOv2

Compiled with GPU acceleration and running on a machine with an NVIDIA Titan X GPU, each AUV image took from 70 to 100 ms to process for detections by the YOLOv2 network. Precision and recall for all detection experiments are plotted in Figure 2.5a. The **NoEdges** network exhibits an average precision (AP) of **0.782**. To help assess how "good" this is, for comparison the easiest category for PASCAL VOC 2012 [34] according to the public leaderboard is *airplane*, for which YOLOv2 demonstrates the highest AP of 0.695. This makes our result quite competitive. However, it is harder to train on multiple classes rather than only one class as we do. Furthermore, when allowed to train with PASCAL as well as COCO [70] data, the best algorithm (not YOLOv2) reaches 0.95 on *airplane*. Also, it is clear that even though scallops near the image edges were not included in **NoEdges** s training, the network is still capable of finding them. However, they are not in the test set and thus are counted as false positives. Including these objects during training and testing lifts the results for the **Edges** network, resulting in its AP reaching a maximum of **0.818** after 10K iterations.

For the **HighRes** network, the AP peaks after 4K iterations at **0.836**, a notable improvement over the lower-resolution versions of YOLOv2. The measured AP drops later in the training cycle, possibly due to overfitting. **AugmentedHighRes** lifts this score slightly, achieving an AP of **0.847** after 9K training iterations. We believe that there is considerable headroom to boost this number by including substantially more missions in the training set.

Figure 2.4 shows some example detection results on the test set of the **HighRes** network. For these images, a confidence threshold of 0.2 was chosen, corresponding to a precision of 86% and recall of 54%. Green boxes are the ground truth, pink boxes denote correct detections by the network, and purple boxes are false positives. Numbers written next to each pink or purple box indicate how confident the network is that the object inside the box is indeed a healthy scallop.

After reviewing many of the detection images such as those shown in Figure 2.4, we believe that the PR curves in Figure 2.5 (a) may be under-reporting performance due to some erroneous annotations. Some of the false positives may actually be healthy scallops that were overlooked in the initial annotation process–for example, the objects with a confidence score of 0.78 in Figure 2.4 (a) and a confidence score of 0.34 in Figure 2.4 (b). Conversely, in Figure 2.4 (c) it is hard to argue that the small green box which counts as a false negative actually contains a scallop. Also causing false negatives, some scallops may be falsely marked as healthy when they are in fact compromised, as the green boxes in Figure 2.4 (e) and (f) would seem to show. Errors in the training set are not so critical, but certainly the test set should be cleaned carefully to better assess algorithms.

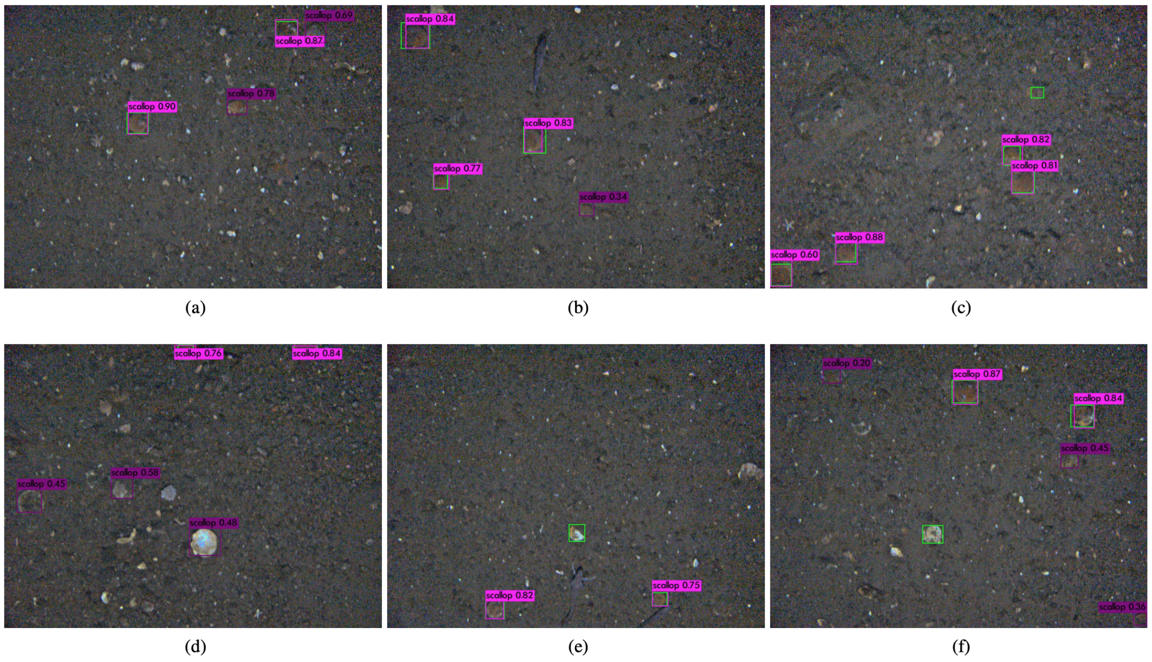
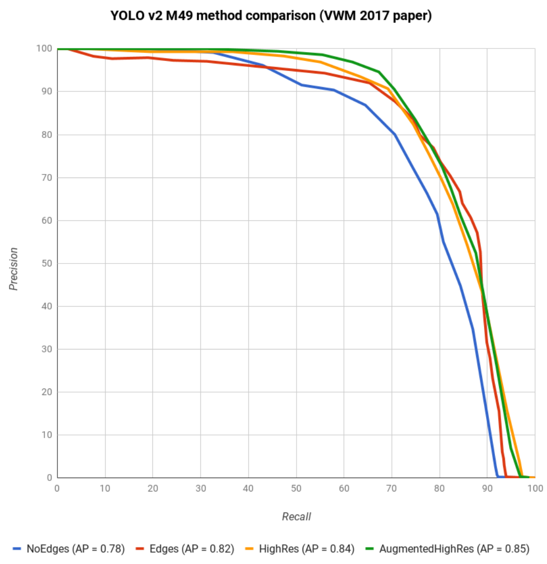


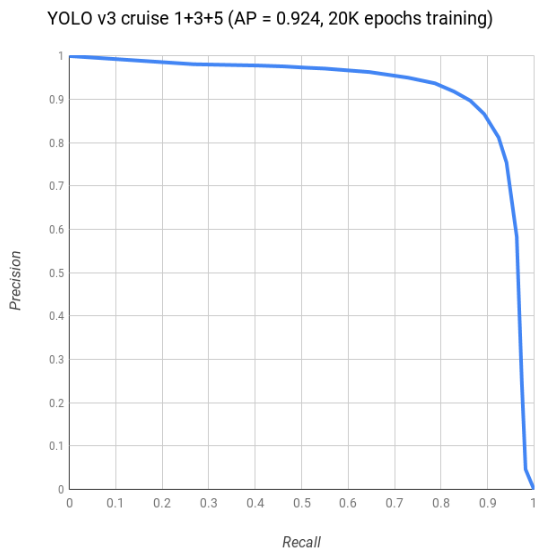
Figure 2.4: Example detection results from the HighRes network on the `M49` test set with detection confidence indicated (IoU threshold = 0.2).Green dots indicate the rough position annotations, pink boxes are the predicted bounding boxes. Because the original annotators did not mark them, the seemingly missed scallops in (f) are not upgraded.

Validation on a separate mission

We implemented a separate testing experiment for **HighRes**, which was trained on `M49`, tested on `M46` data. Images in this dataset are only annotated with rough position information, so we cannot compute IoU's in the standard manner. Instead, we see if the annotated rough position is simply *inside* a predicted scallop bounding box as the new criterion of correct detection. To do this, we changed the IoU threshold to 0: if the rough position is inside the prediction box, the IoU would be greater then zero. Otherwise, the IoU would equal to zero. Following this protocol, the network achieves an AP of 0.926, which is encouraging, but really just an upper bound on true performance.



(a)



(b)

Figure 2.5: (a) Precision-Recall (PR) curves of different YOLOv2 detection approaches, blue is *NoEdges* with average precision(AP) = 0.78, red is *Edges* with AP =0.82, yellow is *HighRes* with AP = 0.84, green is *AugmentedHighRes* with AP = 0.85; (b) PR curve of YOLOv3 detection approach with AP = 0.824.

Comparison to detectors from the literature

[26, 59] used different datasets, so it is difficult to make a fair comparison with their results. [59] only gives two precision and recall data points which indicate a high number of false positives for their chosen parameters: on their “Dataset 1”, they report 1% precision with a recall of 73%, and on “Dataset2” they report 3% precision with a recall of 63%. From a glance at Figure 2.5 (a), our **HighRes** network reaches 85% and 93% precision at those recall levels, respectively.

[26] reports considerably more success, getting precisions ranging from 28% to 99% and recalls ranging from 69% to 94% with their “general” detector, depending on which subsets of data they tested on. Their results were improved slightly when first categorizing the substrate of the entire image and then using a specialized detector,

which we do not do. Using a multicore processor, they are able to process the images at a rate of 10 Hz, just a little slower than this system. Despite the smaller size of scallops (due to the camera intrinsics) and lower contrast of our images, our results are still competitive.

2.1.7.3 YOLOv3

The YOLOv3 network achieves an average precision (AP) of **0.927** which outperforms the best result among the YOLOv2 networks (**AugmentedHighRes** network that has the highest AP of 0.847). Precision and recall for this network is plotted in Figure 2.5 (b). We figure that such a significant improvement happened mainly for two reasons: One is that YOLOv3 is intrinsically a better detection network with a deeper architecture than YOLOv2 in the respect of accuracy. As YOLOv3 makes predictions at 3 different scales throughout the network and concatenates feature maps by upsampling, it is able to capture more meaningful semantic information that helps with detecting small objects. On the other hand, the dataset we used for this experiment is 50x larger and has 100x more scallops than what we used for YOLOv2. Having more data for training would clearly improve the overall performance.

Figure 2.6 shows some example detection results of the YOLOv3 network. The confidence threshold is set to 0.1 to filter the predicted detections. Groundtruth is labelled with blue boxes and predictions output from the network are marked in purple boxes with detection confidence scores on top. The first row in Figure 2.6 are the examples of Retinex-enhanced images where the third row shows the example of original raw images. The second and fourth rows are the corresponding prediction images which show that the network is capable of maintaining stable performance on both types of input resolutions.

In Table 2.1 we show the detection results by scallop density. The formulas and variables are set as following:

- N = the number of groundtruth annotations in an image

- D = the number of scallops predictions in an image
- TP = the number of predictions that has a confidence $\geq$ 0.5
- Precision = TP / D
- Recall = TP / N

Scallops/Image	Precision = TP/D	Recall = TP/N	Num at this density
≥ 1	0.812	0.924	19469
≥ 5	0.861	0.940	3551
≥ 10	0.850	0.953	801
≥ 20	0.828	0.957	269
≥ 50	0.790	0.962	120
≥ 100	0.809	0.957	68

Table 2.1: Precision and recall values of YOLOv3 detection by scallop density

To clarify we use an IoU threshold of 0.5 for analyzing detection results by scallop density and a prediction is considered as correct if it has a confidence score large than or equal to this threshold (0.5). Precision measures the fraction of correct predictions over all predictions and recall indicates how much of the groundtruth were found by the network. Both of these numbers were computed individually for each image, then averaged over all images. Any images with 0 detections have undefined precision, so these were only counted in the recall computation. From table 2.1 (and Figure 2.6 (l)) it appears that the network works well over a range of densities, finding almost 96% of the scallops in images with over 100 scallops while maintaining 81% accuracy.

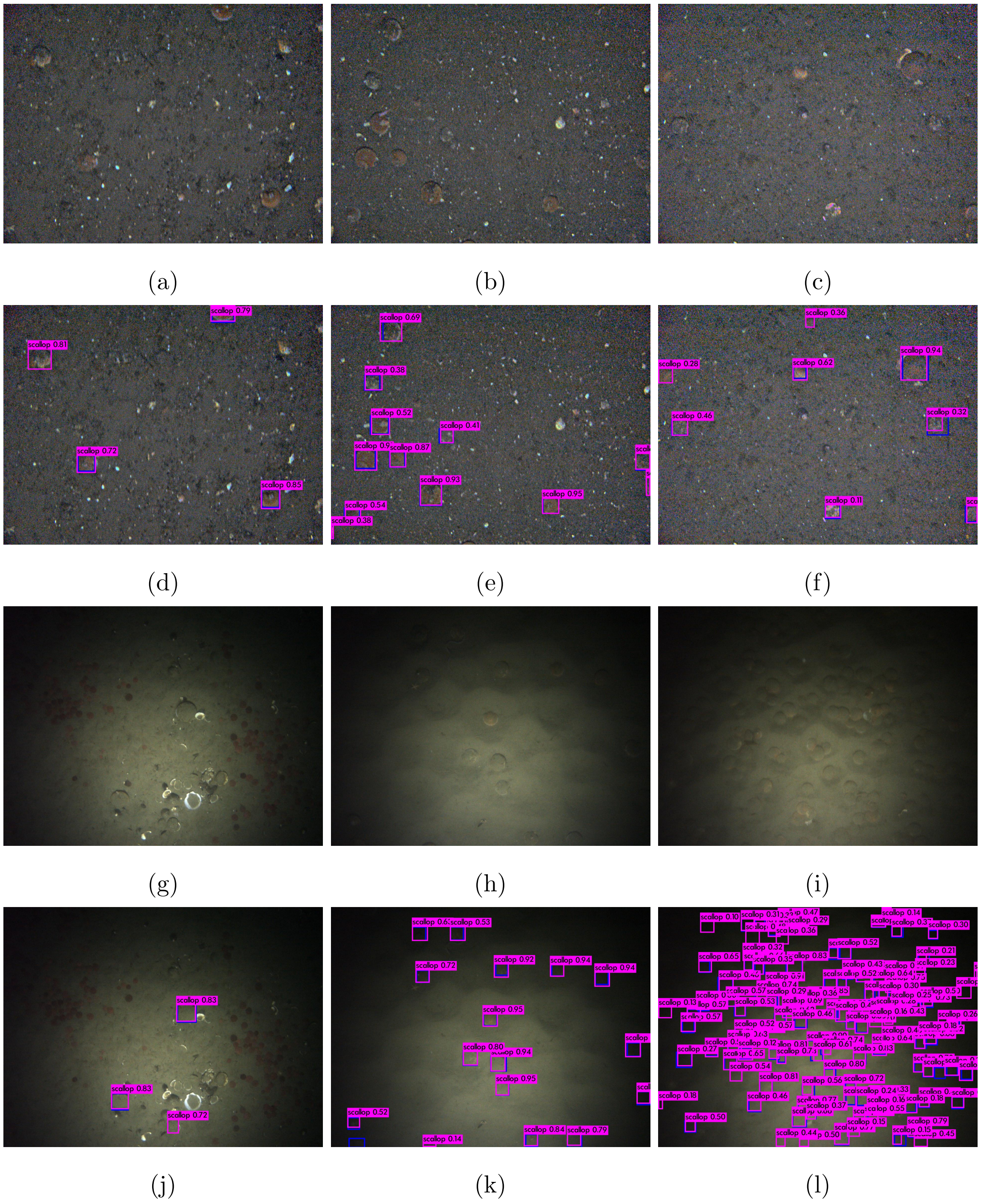


Figure 2.6: Example detection results of YOLOv3 (IoU threshold = 0.1); (a)-(c) Retinex-enhanced images; (g)-(i) Raw images; (d)-(f) and (j)-(l) are corresponding detection output images. Blue boxes are the groudtruth, pink boxes are the prediction with detection confidence on top.

2.1.7.4 RetinaNet

We summarize the performance of the detector on single-class detection problems in Table 2.2 where we can find that by adding background images for training increases the average precision (AP) from 0.807 to 0.814, closing to the AP (0.818) achieved by YOLOv2 training on *Edges* dataset only. However, when validated on a separate mission, RetinaNet detector achieves an AP of 0.747 which is approximately 18 percentage points lower than YOLOv2. Objectively, we used a different protocol of computing the Intersection over Union (IoU) for YOLOv2 since we did not obtain the appropriate groundtruth information for this mission at the time we conducting the experiment. This may slightly increase the number of true positives while evaluating the performance of YOLOv2 on the task. A fair comparison can be done by re-evaluating YOLOv2 using the standard manner of IoU computation. Nonetheless, YOLO detectors outperformed RetinaNet detector in all 4 experiments.

Detector	*Edges* dataset	*Edges* dataset + background	Validation on `M46`	YOLOv3 dataset
RetinaNet	0.807	**0.814**	0.747	0.889
YOLOv2	**0.818**	–	**0.926**	–
YOLOv3	–	–	–	**0.927**

Table 2.2: Summary of model performance on test sets

Precision-Recall (PR) curves of RetinaNet training on YOLOv2 *Edges* and YOLOv3 datasets are shown in Figure 2.8a and Figure 2.8b respectively. Examples of detection results on the test sets can be found in Figure 2.7a - 2.7f with IoU threshold of 0.2. We show that RetinaNet is confused alive scallops with other objects that have similar appearances or features, such as small orangy rocks(Figure

2.7b, 2.7d, 2.7e), sand dollars (Figure 2.7i), therefore misclassified them as positives. Meanwhile, further improvement on extracting objects of interest from low-contrast background is needed. Figure 2.7j - Figure 2.7l indicate that RetinaNet is not as capable as YOLOv3 of detecting objects in high density. Even on a lower IoU threshold (0.2 for RetinaNet, 0.5 for YOLOv3), RetinaNet detector still missed a considerable amount of true positives. Again this can be caused by the struggle of subtracting enough accurate background information. On the basis of performance comparison between RetinaNet and YOLO detectors we can conclude that YOLO intrinsically outperforms RetinaNet on single-class detection problem.

For the 2-class classification problem, We monitored the losses as shown in Figure 2.9(d). To clarify, regression loss implies the difference between predictions and actual observations; Classification loss measures how accurate the network is of classifying objects into the correct categories. Figure 2.9 (a) - (c) shows the example predictions: the white lines and rectangles indicate the original annotations; Green boxes are the groundtruth bounding boxes; Red and Yellow boxes are the predicted bounding boxes for *Alive Scallop* and *Sea Star* respectively.

Using a IoU threshold of 0.2, the mean average precision (mAP) achieved **0.732** where the average precision (AP) of *Alive Scallop* is 0.903 and of *Sea Star* is 0.561. It is not surprising that the classifier performs notably better on one category than the other since the number of *Alive Scallop* in this dataset is approximately 30x more than the *Sea Star*. In Figure 2.9 (a) we can see that the network neglected 2 true positives in the *Sea Star* class (showing in white boxes on the left of the image). Apparently having a balanced dataset (adding more examples of *Sea Star*, etc.) is the key factor to improve the classification accuracy. This issue is further addressed in Section 2.2.

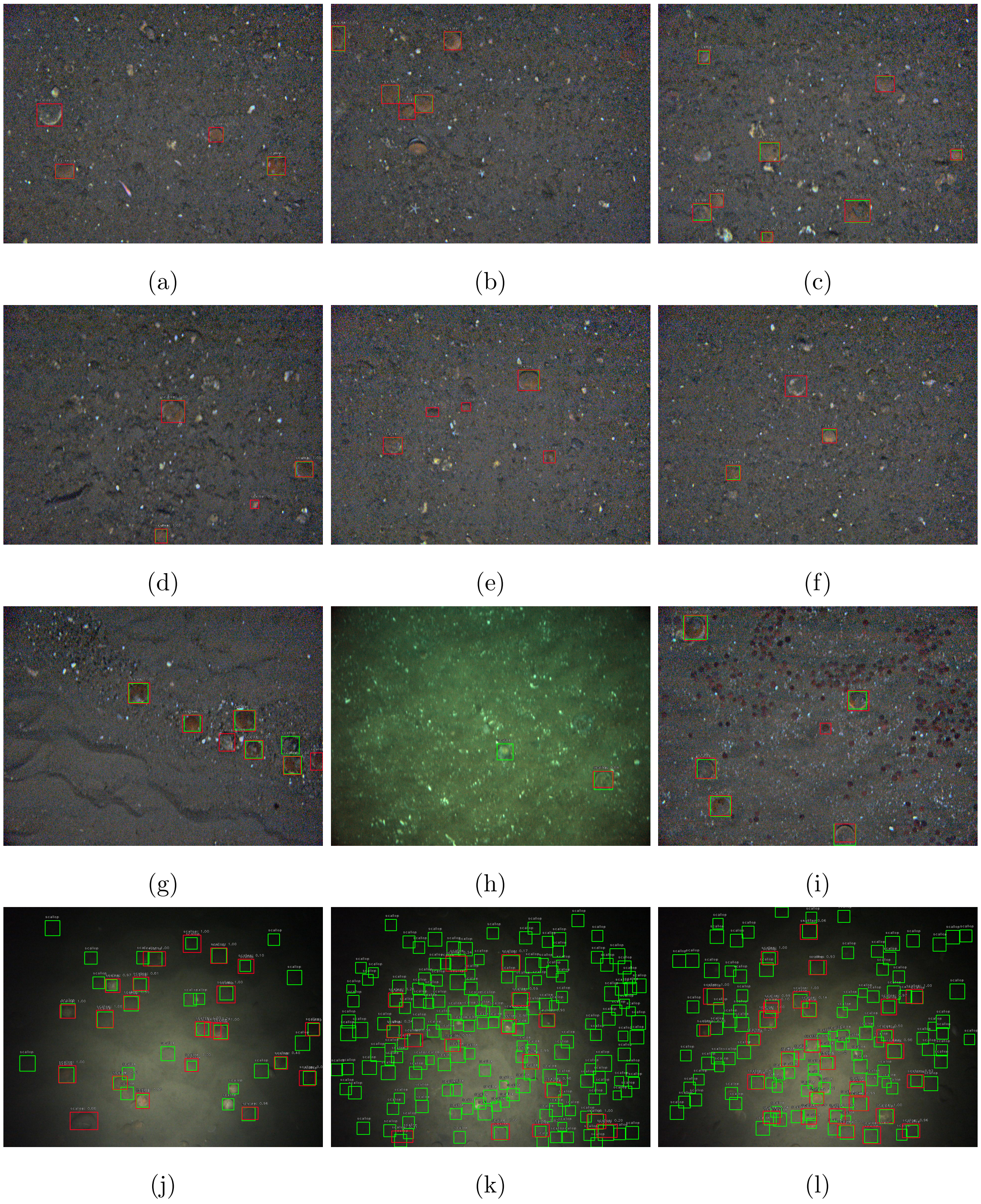
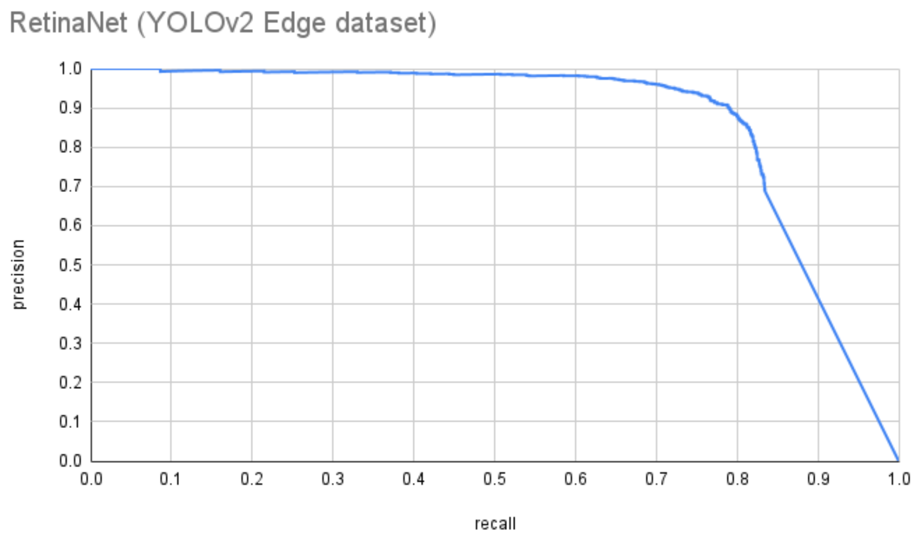
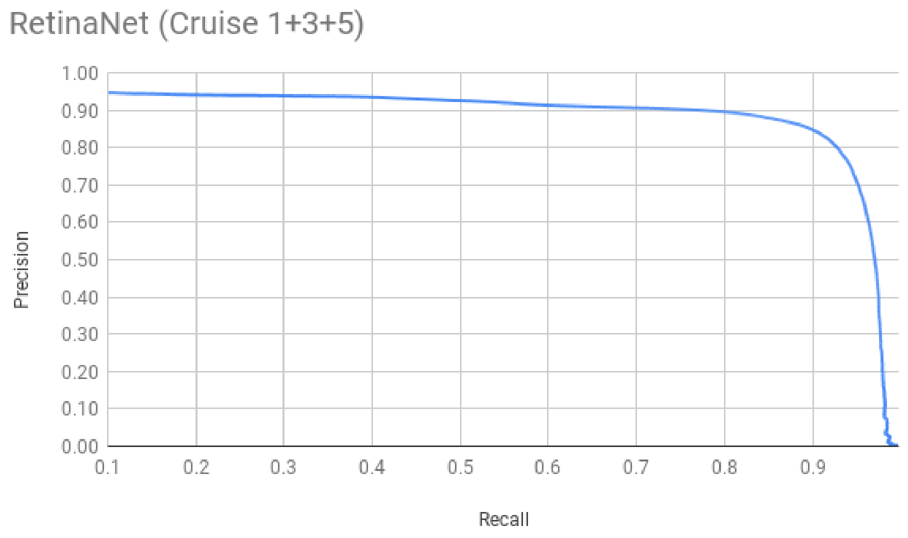


Figure 2.7: Example detection results of RetinaNet (IoU threshold = 0.2): (a)-(f) Predictions on YOLOv2 *Edges* test set; (g)-(l) Predictions on YOLOv3 test set; Green boxes indicate the groundtruth and red boxes are the predictions with detection confidence on top.

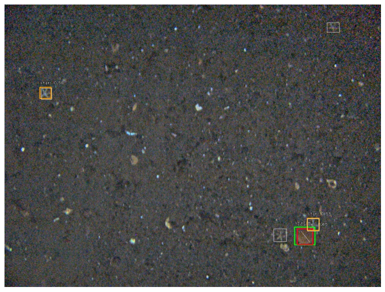


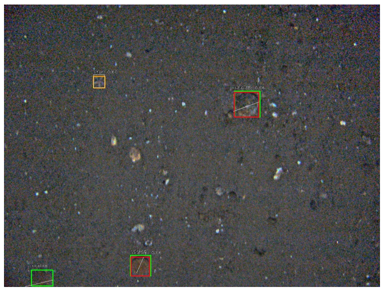
(a) PR curve of RetinaNet on the YOLOv2 *edge* dataset

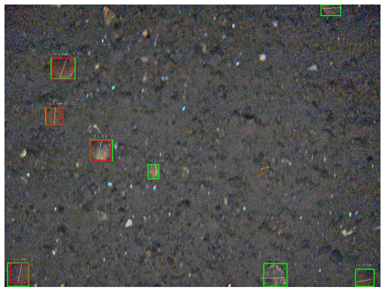


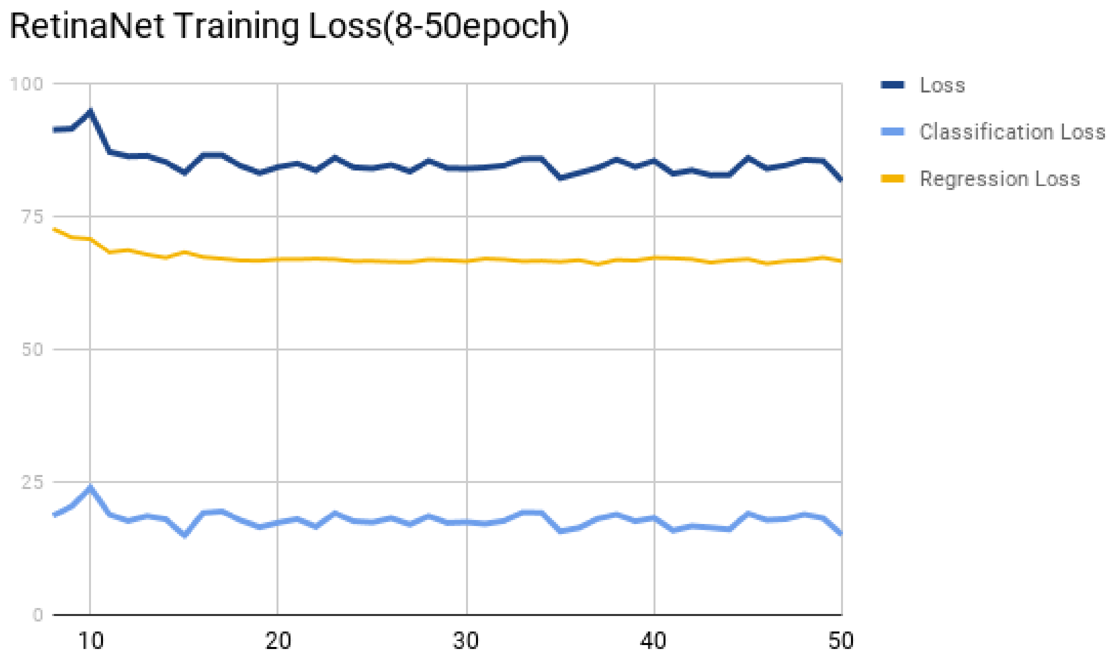
(b) PR curve of RetinaNet on Cruise 1+3+5 dataset

Figure 2.8


(a)


(b)


(c)



(d)

Figure 2.9: (a)-(c): Example detection results of RetinaNet on 2-class classification: white lines and rectangles indicate the original annotations, green boxes are groundtruth, red and yellow boxes shows the predictions for *Scallop* and *Star* respectively; (d):Training losses.

2.2 Multi-class classification

From our previous experiments, we see that CNN-based visual detectors have the ability to identify small objects that only occupy a fraction of an entire image with high accuracy as well as at real-time speed. To extend our work on population monitoring, we developed multi-class classification models to estimate mortality rate of sea scallops as well as including other relevant creatures for a pray-predation study.

2.2.1 Mortality Rate Estimation

In our previous discussion, the dredging approach can result in increasing scallop incidental mortality when the scallop shells are fatally damaged during the process or being exposed to non-suitable environments that surpass the lethal limit for scallops. In order to further improve the BACI study, compromised (*dead*) scallop count is indispensable to be considered in our detection schema. Additionally the results from our single-object detection experiments in Section 2.1 showed that the false positives contained a certain amount of compromised scallops that were misclassfied as *alive*. Therefore adjusting our detection algorithm to a classification problem including the *dead* category would support reducing the number of false positives in the *alive* class from our previous experiments. Provided with adequate groundtruth labels in the *dead* category, we then developed a 2-class classifier on both of the categories. The motivation behind this experiment can be summarised as: a). Analysis on compromised scallops count can further provide critical statistic information for dredge-included incidental mortality rate estimation; b). Allowing the network to learn on features of compromised scallops would help with improving the detection performance by better distinguishing between healthy and unhealthy scallop.

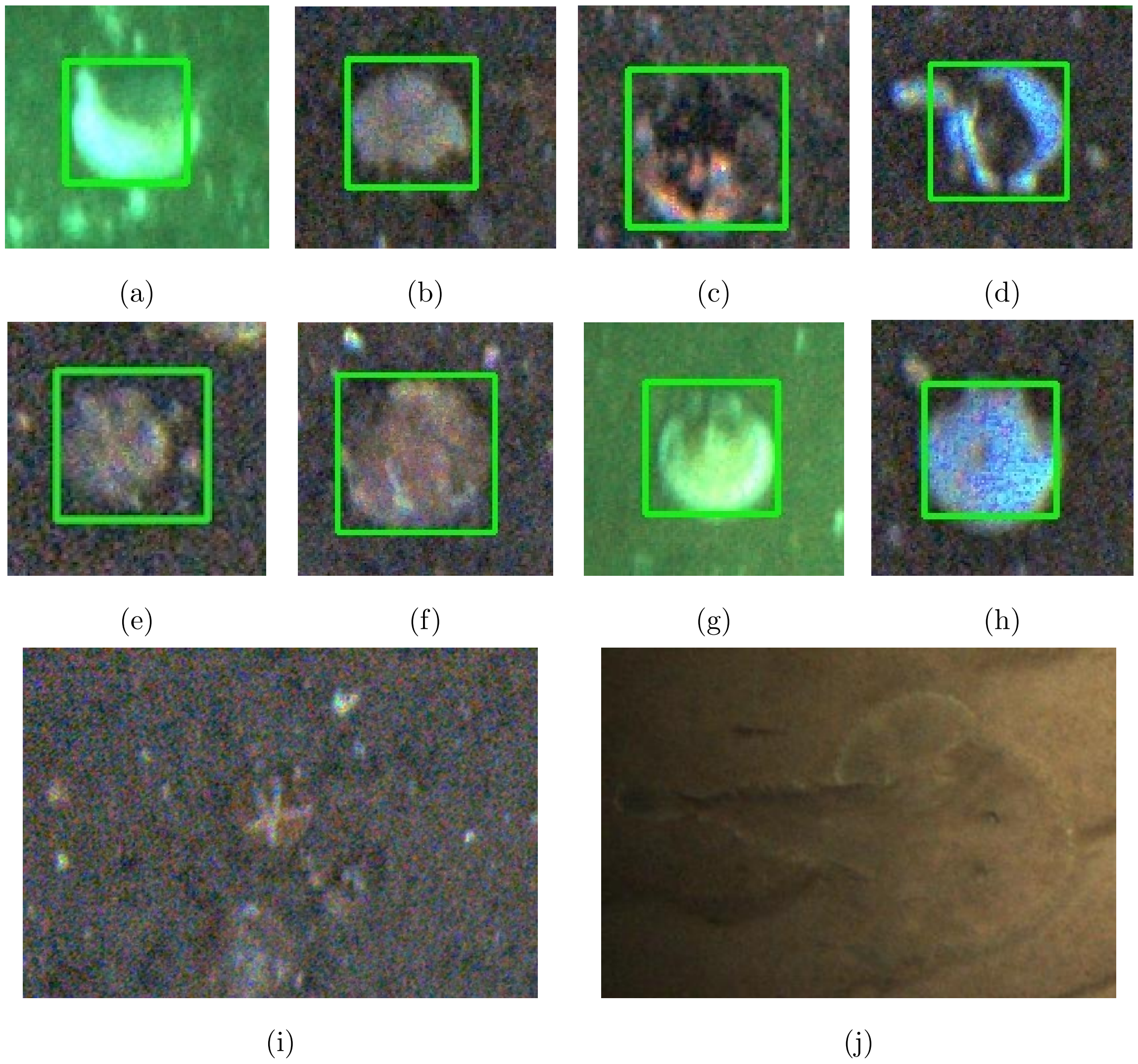


Figure 2.10: (a)-(h): Example of compromised scallops; (i): Star preying on scallop; (j): Monkfish.

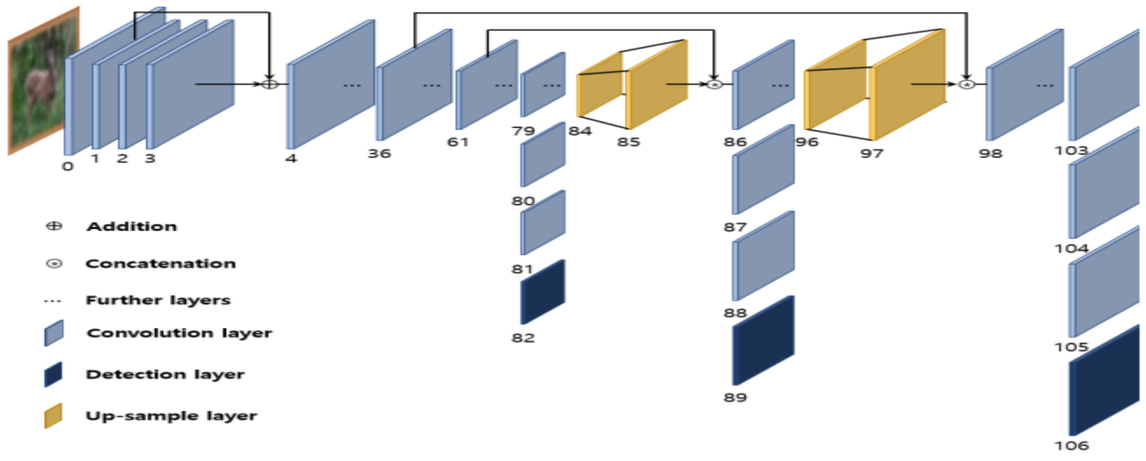
Dataset Description

Examples of compromised scallops are shown in Figure 2.10. Scallops that have broken shells or were crushed were labeled as *dead* (Figure 2.10f, 2.10c). Inverted scallops (Figure 2.10h, 2.10b) were also considered as compromised since healthy scallops would always have their right (darker) valve facing upward and left (white-ish) valve facing down to the seafloor. And healthy scallops are always able to quickly flip back to normal when inverted. Dredging can often cause the inversion leaving the scallops damaged or being vulnerable to predation. Single shells (Figure 2.10d) were noted as compromised as well. As of conducting this experiment, 18,461 *dead* scallops were annotated contained in 12,293 images from 32 missions. 37,423 scallops of this dataset were labelled as *alive*. The images were split 80/20 for training/testing: 9,834 images (and all of their scallops) in the training set, and 2,459 images in the test set.

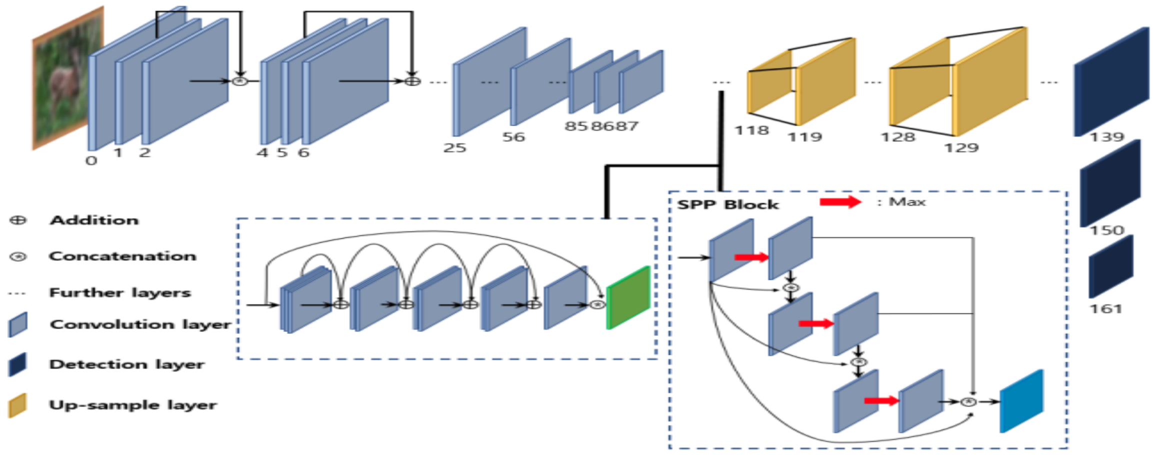
Training Architecture and Procedure

We considered both YOLOv3 and YOLOv4 architectures [107] as well as their lightweight versions (i.e. YOLOv3 tiny and YOLOv4 tiny) as our classifier. YOLOv3 uses Darknet-53 (Figure 1.7c) as the feature extractor and stacks another 53 layers on top of it for the task of detection resulting in a 106 fully convolutional underlying architecture (Figure 2.11a). The YOLOv4 (Figure 2.11b) consists of the CSPDarknet53 backbone, a spatial pyramid pooling module, a PANet path-aggregation neck and the YOLOv3 head giving us a complete 161-layer architecture. YOLOv3-tiny (Figure 2.12a) and YOLOv4-tiny (Figure 2.12b) are compact versions of YOLOv3 and YOLOv3 respectively, with simpler structure and reduced parameters. The number of convolutional layers in the backbones are largely scaled down producing a 23-layer structure for YOLOv3-tiny and a 37-layer structure for YOLOv4-tiny. For small datasets, the compressed network architectures are convenient for fast training and detection without the need of huge memory usage.

In addition to experimenting with different network structures, we explored the influence of batch size on training dynamics. Batch size decides the number of training samples propagating in one forward pass and is one of the most important hyperparameters that controls the error gradients estimation in training modern neural networks. Larger batch size means greater gradient updates which may make the model take longer to converge and too large of a batch size may result in poor generalization to new datasets. On the other hand having a proper large batch size allows us to take advantages of the parallelism of GPUs to speed up the training process. Small batch size increases the computational costs but is able to provide higher training stability and better generalization performance due to its regularization effect by adding noise to the training data. We trained the networks on different batch sizes (i.e. 32, 64, 128) for 20K iterations. Input images were resized to 832 x 832 (instead of 416 x 416 by the default setting) to boost up the detection accuracy with higher input resolution.

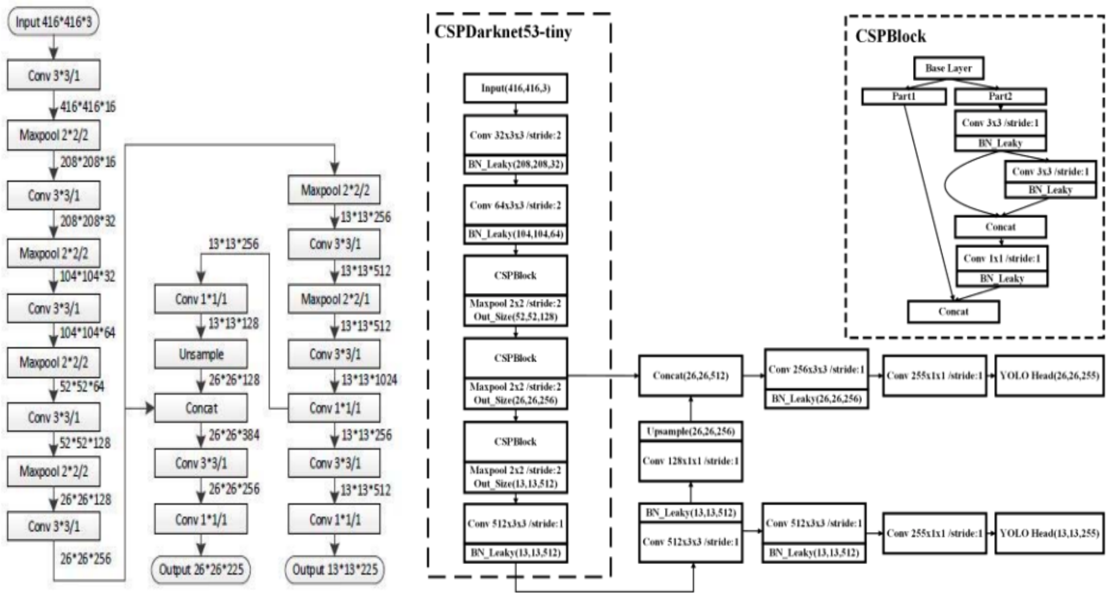


(a)



(b)

Figure 2.11: The YOLOv3 (a) and YOLOv4 (b) network architectures [61].



(a) (b)

Figure 2.12: The YOLOv3-tiny (a) [5] and YOLOv4-tiny (b)[56] network architectures.

Results

We used the test accuracy to interpret the generalization gap of each model. Table 2.3 reveals the average precision (AP) of each class, mean average precision (mAP) across 2 classes and the detection speed (fps) of being compiled and running with GPU acceleration on 3 NVIDIA GeForce GTX 1080 Ti GPUs. The IoU threshold was set to 0.2 during testing. From Table 2.3 we can see that the lightweight model structures fitted the data better than the full-sized models given the size of our dataset. AP for the *alive* class were generally higher than the *dead* class among all classifiers as there were 2x more *alive* examples in the dataset and the *dead* contained disparate features. And for different model structure, the optimal batch size alters, for instance batch size of 32 worked the best for YOLOv3 whereas large batch size (i.e. 128) increased the AP, mAP for YOLOv4. This further confirmed that the optimal batch size is necessary to be experimented with in order to maximize the process capacity. Figure 2.13 illustrates the training curves of the YOLOv4-tiny models on different batch sizes where we can see that smaller batch size converges faster but larger batch size is able to provide stability dusting training under the same learning rate. YOLOv4-tiny outperformed all the other models and with optimal batch size of 128, the AP achieved 0.734 and 0.888 for *dead* and *alive* classes respectively. The mAP achieved 0.811 with IoU threshold of 0.20. We further trained this model lowering the ignore threshold from 0.7 to 0.5 which controls the scale of detection bounding boxes involving in the loss calculation. The mAP slightly increased to 0.815 with AP of 0.740 and 0.890 for *dead* and *alive* classes respectively.

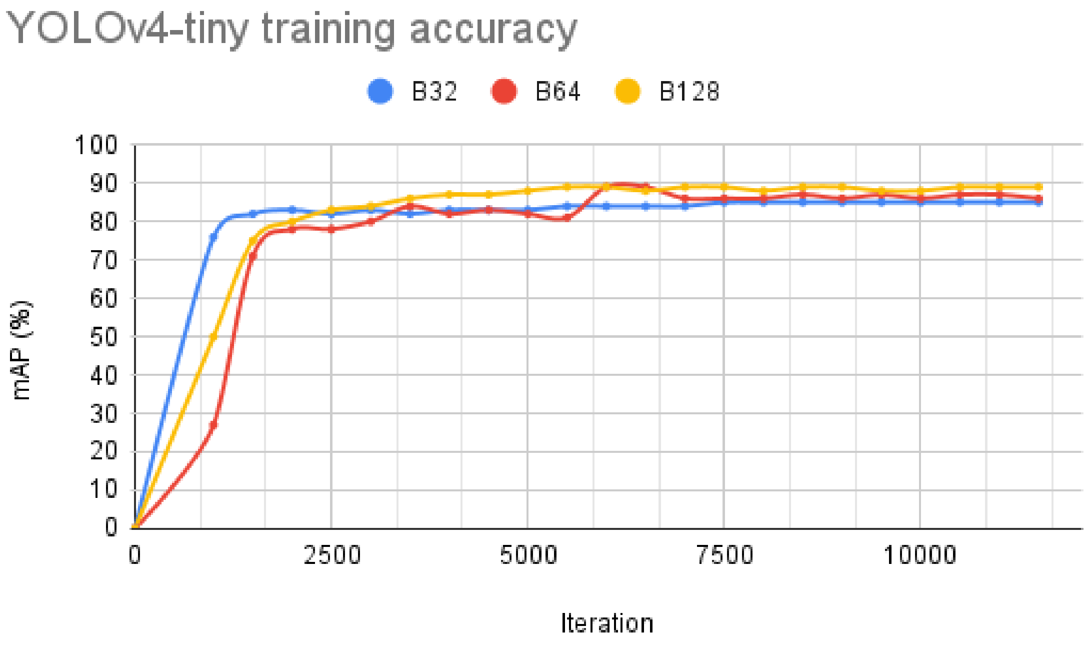


Figure 2.13: YOLOv4-tiny training curves of different batch sizes

In Figure 2.14 we show the example classification results from our best model (i.e. YOLOv4-tiny with ignore threshold = 0.5) on the test set with confidence threshold of 0.25, corresponding to a precision of 73% and recall of 84%. Green boxes indicate the ground truth, purple boxes are predictions for the *alive* class and pink boxes are predictions for the *dead* class with detection confidence indicated on top. It demonstrates that the network is capable of distinguishing compromised scallops from healthy ones. We also observed some labeling inconsistency and ambiguity among the ground truth annotations. For example, On the top left corner of Figure 2.14e the scallop predicted as *alive* with a confidence score of 85% is labeled as *compromised* however it shares similar characteristics with the one at its bottom right with confidence score of 96% which is labeled as *healthy*. The one on the top of Figure 2.14g predicted as *dead* with confidence score of 37% is labeled as *healthy* but appears like having a broken shell.

	YOLOv3				YOLOv4		
	32	64	128		32	64	128
AP(*dead*)	0.699	0.693	0.673		0.650	0.674	0.700
AP(*alive*)	0.870	0.864	0.843		0.837	0.860	0.869
mAP@0.20	0.785	0.778	0.758		0.744	0.767	0.784
Detection Speed (fps)	19	24	20		11	17	17
	YOLOv3 -tiny				YOLOv4 -tiny		
	32	64	128		32	64	128
AP(*dead*)	0.690	0.676	0.727		0.696	0.701	**0.734**
AP(*alive*)	0.859	0.846	0.883		0.871	0.868	**0.888**
mAP@0.20	0.775	0.761	0.804		0.783	0.785	**0.811**
Detection Speed (fps)	85	89	91		49	66	85

Table 2.3: Test accuracy and detection speed of YOLO models trained on different batch sizes

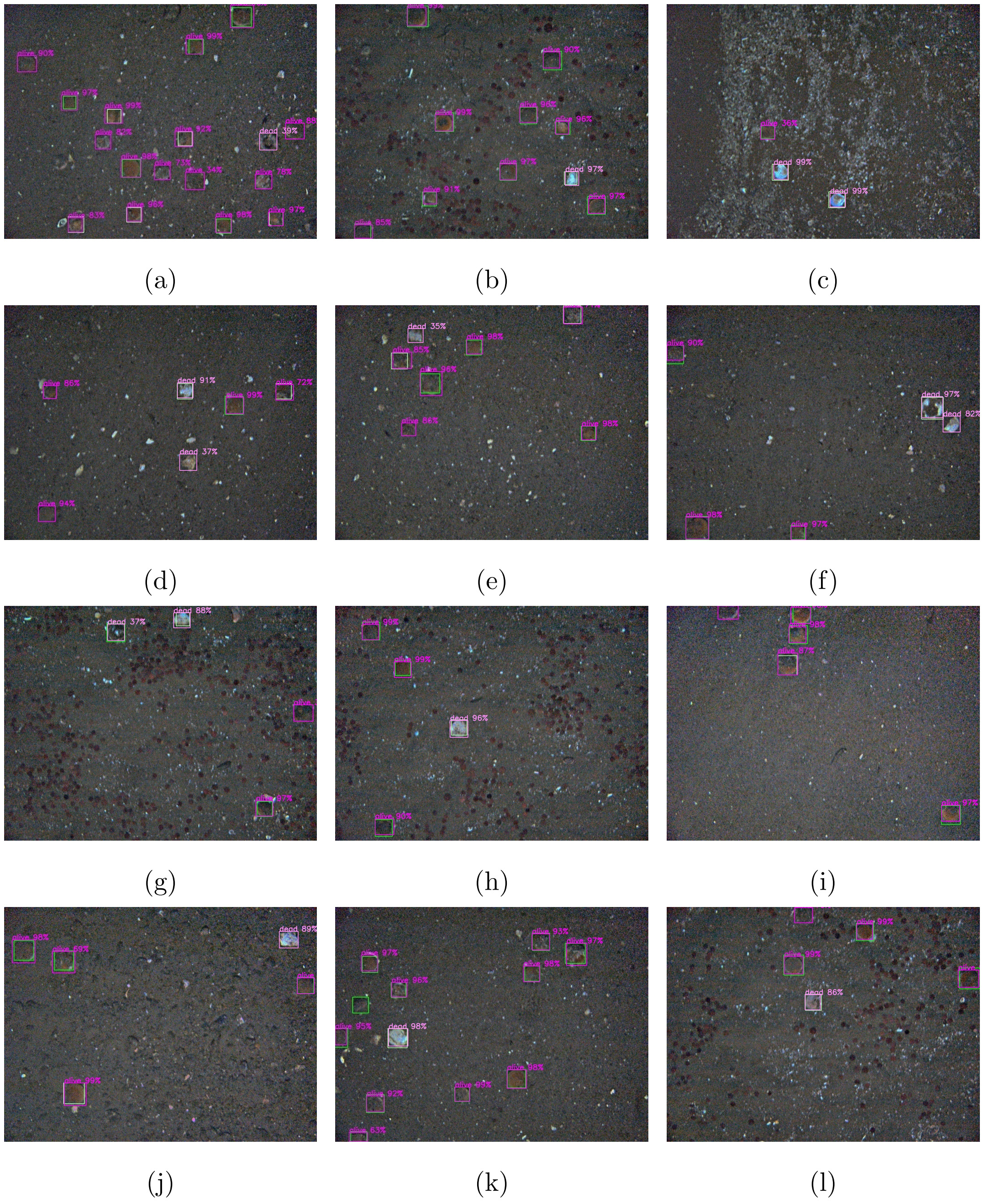


Figure 2.14: Example classification results from the YOLOv4-tiny network on the mortality rate estimation test set (Green boxes indicate ground truth, purple boxes are predictions for the *alive* class and pink boxes are predictions for the *dead* class)

2.2.2 Dynamics of Predation

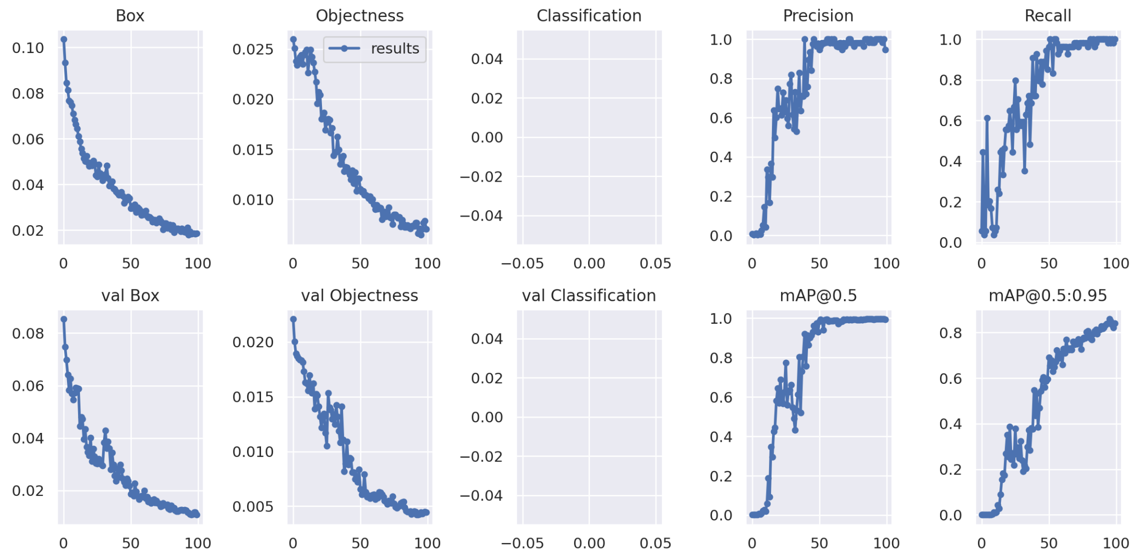
Population of any creature do not remain constant but changes, sometime drastically, within a period of time. One of the factors of such fluctuations is the dynamic predator-prey cycles. Predators influence the numbers of their prey and vice versa. The interaction between the two forms creates a cyclical pattern and urges changes in populations over time. For this reason, we added 2 more categories of relevant creatures to sea scallops: monkfish and sea star. Sea stars are the primary predator of sea scallops (Figure 2.10i) and monkfish are opportunistic feeders that prey whatever is most available at the time. Monkfish usually swim close to the seafloor with an unified appearance to the surface of seabed (as shown in Figure 2.10j) which is a substantial challenge to our detection network.

Dataset and Data Augmentation

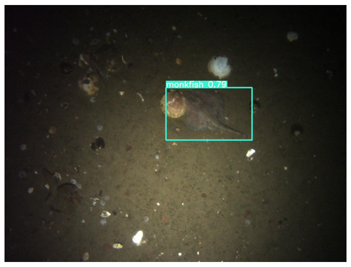
For the class of *Monkfish*, there were only 74 images annotated from `M59`, `M60`, `M62`, `M63`, `M66`, `M68`, `M70` and `M71` resulting in 97 monkfish annotations. Due to the limited number of images in this class, data augmentation was required to reduce the negative effect of class imbalance. We first expanded the original dataset by flipping the images horizontally and vertically generating a dataset that consists of 296 images and 388 annotations. We then trained a YOLOv5 [58]-based neural network on this lightly enhanced dataset for a 1-class detection task recognizing monkfish only. The images were split 80/20 for training/validation (236 images for training, 60 images for validation). We trained the monkfish detector for 100 epochs and the validation accuracy peaked at the 67th epoch achieving mAP@.5 and mAP@[.5:.95] of 0.995 and 0.861, respectively. Training curves are shown in Figure 2.15a. Next we run the trained monkfish detector on our original scallop dataset (i.e. 97,344 images from 67 missions) in order to discovery as much monkfish as possible from un-annotated images. 86 more images with 90 more monkfish objects (Figure 2.15b - Figure 2.15d) were found by using a IoU threshold of 0.5 to filter out most of the false positives .

Given the fact that the data size of the *Monkfish* class was yet relative small

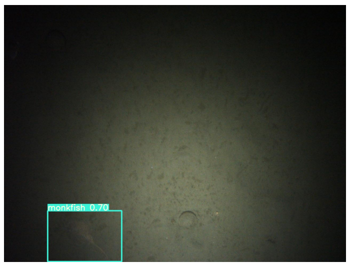
H



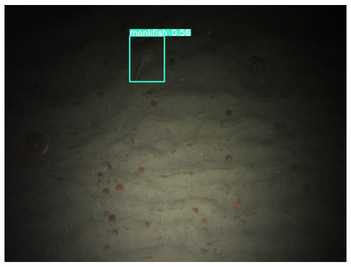
(a)



(b)



(c)



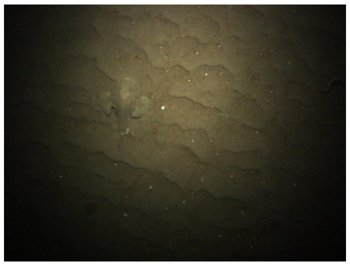
(d)

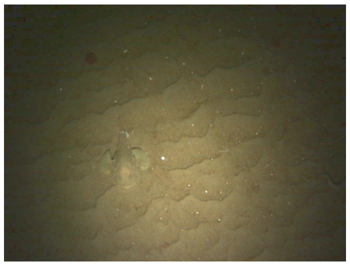
Figure 2.15: (a): Monkfish detection training curves; (b)-(d): Example detection results with IoU threshold = 0.5.

compared to the other two classes, we further augmented the dataset to increase its size and diversity. Techniques we adopted include ColorJitter that randomly adjust the brightness, saturation, and contrast of the images (Figure 2.16d - Figure 2.16f); Adding Gaussian, Speckle, and Salt-and-Pepper noises Figure 2.16g - Figure 2.16i); Applying Mean, Gaussian, and Median filters to the images (Figure 2.16j - Figure 2.16l). Eventually 5,760 more *Monkfish* images were generated after augmentation. Our final dataset for the 3-class classification task consists of 11,013 images containing 29,206 annotated objects (7,676 stars, 15,023 scallops, 6,507 monkfishes). Noted that we only considered alive/healthy scallops for this problem.

(a) Original

(b) Horizontal-flip


(c) Vertical-flip


(d) Brightness-jitter

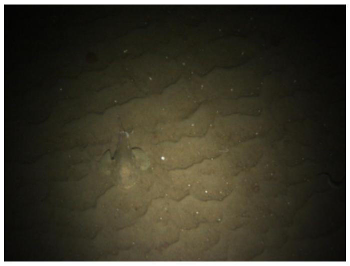
(e) Saturation-jitter

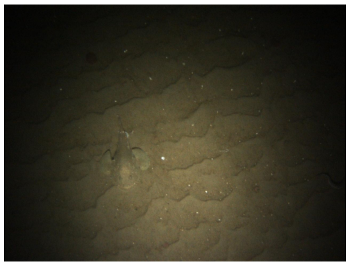
(f) Contrast-jitter

(g) Gaussian noise

(h) Speckle noise

(i) Salt-and-pepper noise


(j) Mean filtering


(k) Gaussian filtering

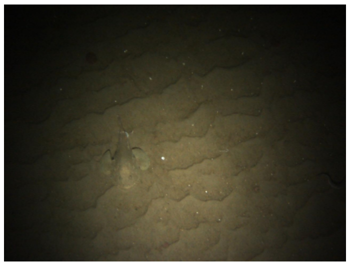

(l) Median filtering

Figure 2.16: Example of augmented Monkfish images

Training Procedure

We first experimented with a various of training configurations and weights for the training architectures discussed in Section 2.2.1. The combination that fits the best for our data is YOLOv4 with pre-trained weights on COCO [70], learning rate of 0.0001, batch size of 64, and ignore threshold of 0.7. Input images were resized to 832 x 832. We trained the network for 20,000 iterations and the images were split 80/20 for training/testing: 8,811 images in the training set, and 2,204 images in the test set. Training loss and validation accuracy curves are plotted in Figure 2.17.

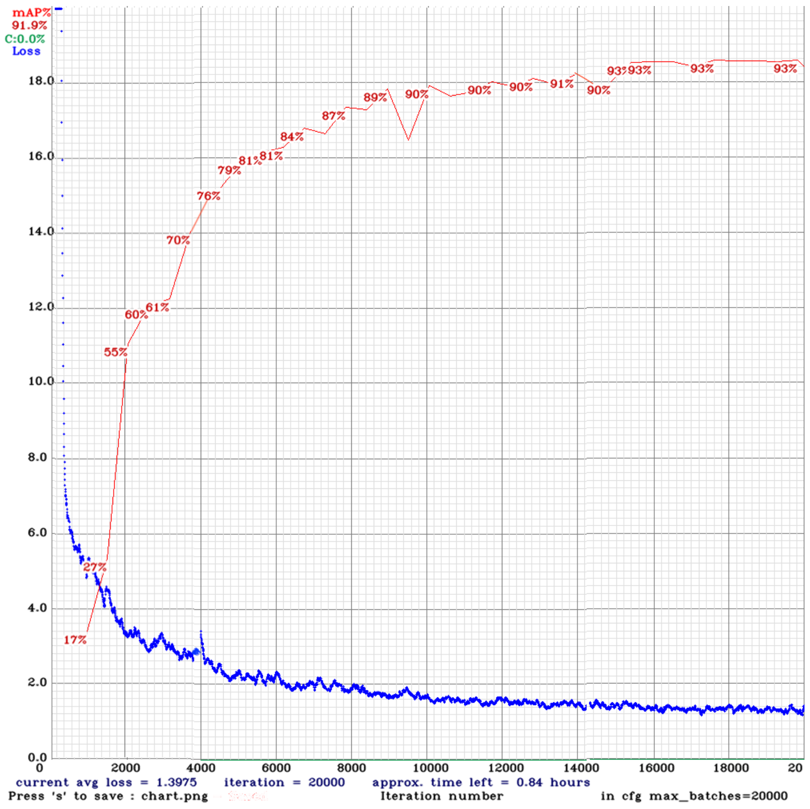
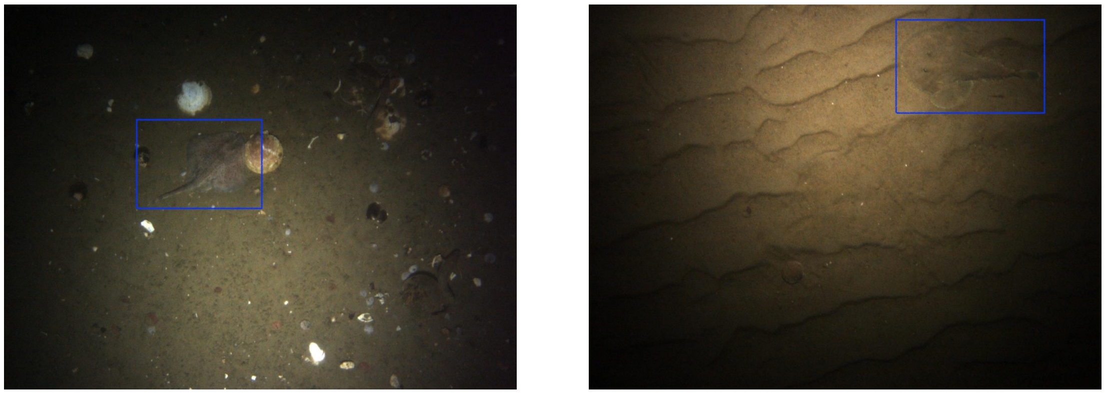


Figure 2.17: Training loss and validation accuracy for YOLOv4 on 3-class classification

Results

The YOLOv4 classification network exhibits a mAP of 0.937 with IoU threshold of 0.2. Precision and recall are both 0.89. Corresponding APs are 0.882, 0.969 and .960 for *Monkfish*, *Scallop* and *Star* classes respectively. Total detection time is 97 seconds equivalent to 22 fps. The results shows that the classification network is quite competitive. Figure 2.19 displays example test results where green boxes indicate the groundtruth annotations; Pink, yellow and blue boxes illustrate the predictions for *Scallop*, *Star* and *Monkfish*. Figure 2.19a - 2.19d demonstrate that the classifier was able to detect and localize monkfishes even under extreme lighting conditions. And the network was capable of learning features of objects in various sizes simultaneously from low-contrast and low-brightness images. Noted that Skate fish presents high similarity with Monkfish (shown in Figure 2.18) which was confounded during both annotating and detecting processes. It would be interesting to address this problem in the future with improvements for the network architecture on detecting objects with analogous appearances.



(a) Skate (b) Monkfish

Figure 2.18: Example to show the similarity between Monkfish and Skate.

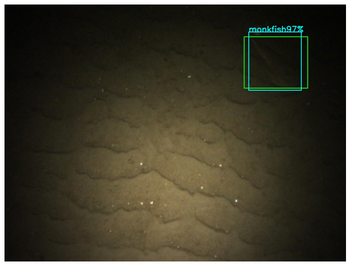
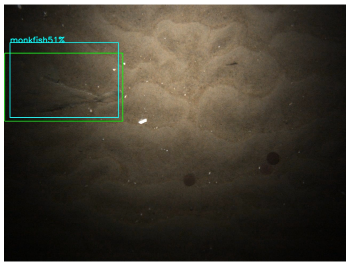
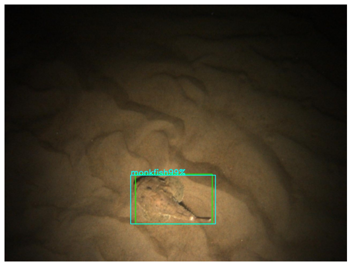
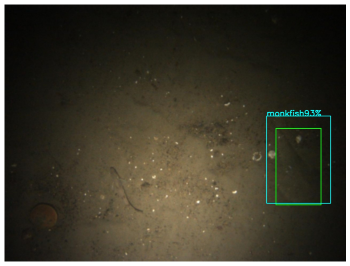
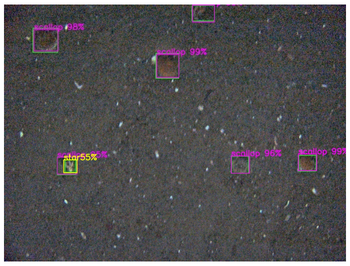
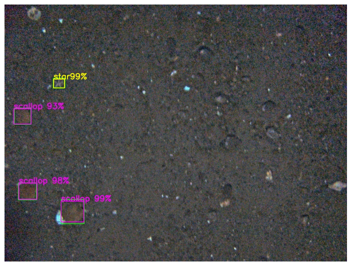
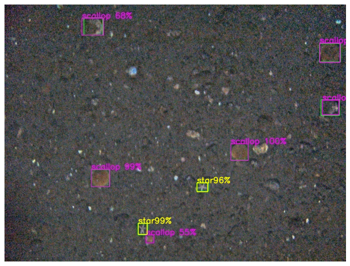
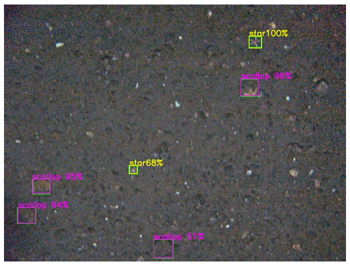
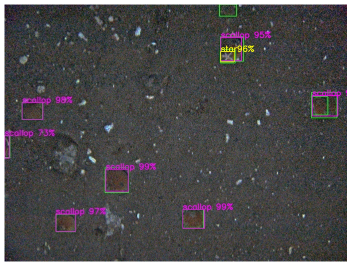
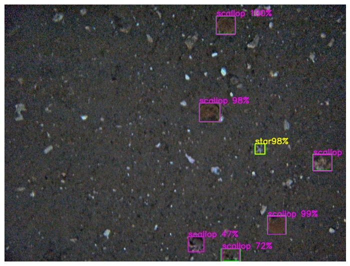
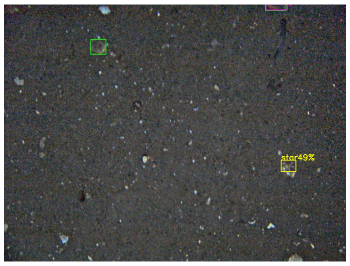
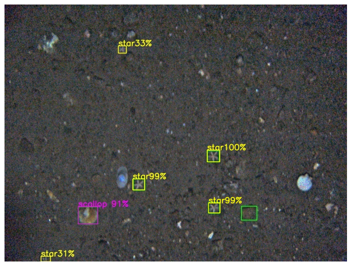

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Figure 2.19: Example classification results from the YOLOv4 network on the Predation test set (Green boxes indicate ground truth, pink boxes are predictions for the *Scallop* class; yellow boxes are predictions for the *Star* class; and blue boxes are predictions for the *Monkfish* class)

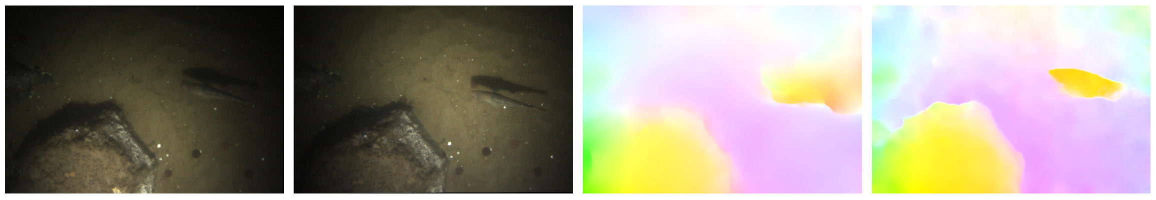
Chapter 3

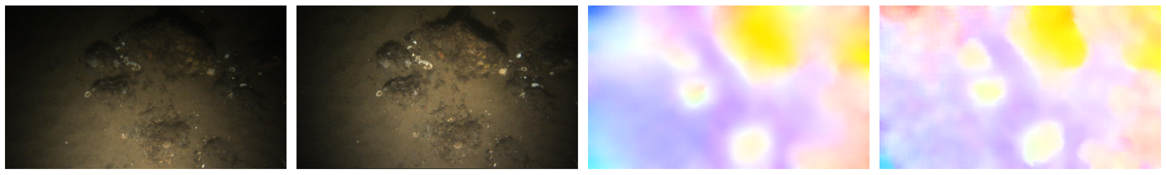
IMAGE REGISTRATION FOR LONGITUDINAL DETECTION

3.1 Optical Flow

In this section we focused on motion-based segmentation to identify landmarks and separate moving objects from the background. Moving animals are considered as objects of interest for detection, tracking or monitoring animal behaviors. The dormant objects, such as prominent rocks that are distinctive can be contemplated as landmarks to relate data derived from disparate sources. Optical flow often serves as a good representation for motion of individual pixels on the image plane. It is a way to calculate the motion of image intensities over time, of which high intensity indicates regions of significant change. The flow fields that are associated with moving objects can then be analyzed to segment objects of interest.

We choose the models of LiteFlowNet [53] pretrained on KITTI [41] and Sintel benchmarks as our motion detectors. Each image pair generated from our previous registration model (Section 3.2) is cropped and rescaled to the same size (First and second columns from the left in Figure 3.1) to be fed into the flow estimation network. Examples of the flow field predictions are shown in Figure 3.1. The KITTI [41] dataset contains frames of real world scenes of traffic and roadways captured by cameras and 3D laser scanner. The Sintel [15] are generated from animated sources of artificial scenes. Third and forth columns of Figure 3.1 are the results generated by the models pretrained on KITTI [41] and Sintel [15] respectively where we can see that both networks are able to detect motionless large objects or cluster of small objects. The Sintel-model produces clearer boundaries around objects, for example, the fish on the top-left corner in Figure 3.1a and the small whitish scallop on the bottom in Figure 3.1c.


(a)


(b)

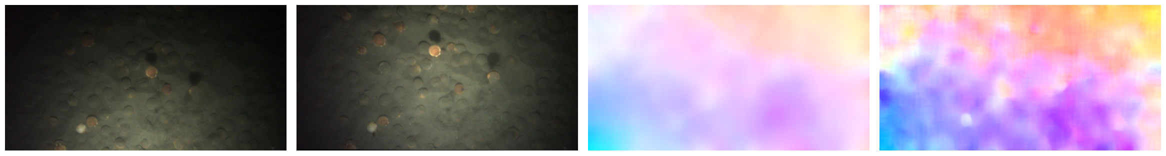

(c)

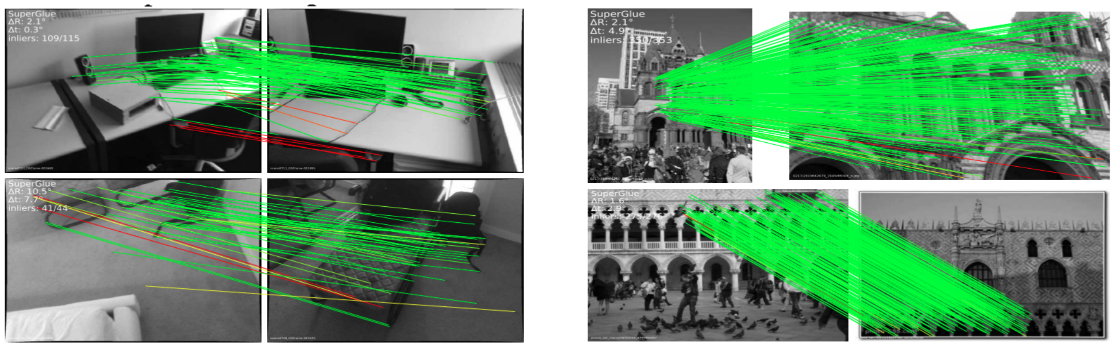
Figure 3.1: Examples of flow fields computed by LiteFlowNet [53] models pretrained on KITTI [41](3rd column) and Sintel [15](4th column)

This could be due to that the KITTI [41] dataset contains limited motion types and does not have distant objects captures. In addition, the Sintel [15] dataset has 10x more data frames and double the groundtruth density per frame than KITTI [41]. Even so, both networks have difficulties with illumination changes from the AUV spotlight that neither is capable of distinguishing flat seafloor as background and recognizing small moving objects.

3.2 Image Registration

Image registration is convenient in many computer vision applications for image mosaicing, target localization, motion tracking and environmental monitoring. It aims to transform different set of data from various circumstances into a particular reference coordinate system in order to obtain the integrated information from divergent sources [89]. Basic steps involved in image registration include feature detection that is to find the common features shared between the images, feature matching that decrypts and measures the similarities among the features besides spatial relationships, as well as transformations of points mapping from one image to the referred image.

Since it was hard to find corresponding matching points based on optical flow from image pairs with low contrast, we then adopted deep learning based image registration techniques to transform the pairs of images into one coordinate system. SuperGlue [116] was proposed with an attention-based context aggregation mechanism to learn geometric transformation and jointly correspondences as well as rejecting non-matchable points using an Graph Neural Network (GNN) combined with an Optimal Matching layer. It performs context aggregation, matching, and filtering in a single end-to-end fashion. The architecture of SuperGlue [116] was trained on both ScanNet [24] dataset of indoor scenes as well as MegaDepth [69] dataset developing an outdoor model (Figure 3.2).



(a) Indoor (b) Outdoor

Figure 3.2: SuperGlue keypoints matching example of two environments [116].

In order to evaluate the performance of SuperGlue [116] architecture on our dataset with challenging attributes, we first selected a set of consecutive frames with interesting objects, such as discriminative objects (large or cluster of rocks shown in Figure 3.3a and 3.3d, Figure 3.3b and 3.3e) or moving animals (floating fishes in Figure 3.3a and 3.3d, a swimming scallop appears on a flat seabed in Figure 3.3c and

3.3f). We specifically chose the SuperGlue [116] network settings for outdoor estimation as it was designed to better deal with illumination changes and occlusion. Given a pair of images the model extracts matching features across the image pair in 7.5 sec on average (0.1 fps running speed). The matching threshold was set to 0.5 for keypoints selection in order to identify positive correspondences between the images pairs.

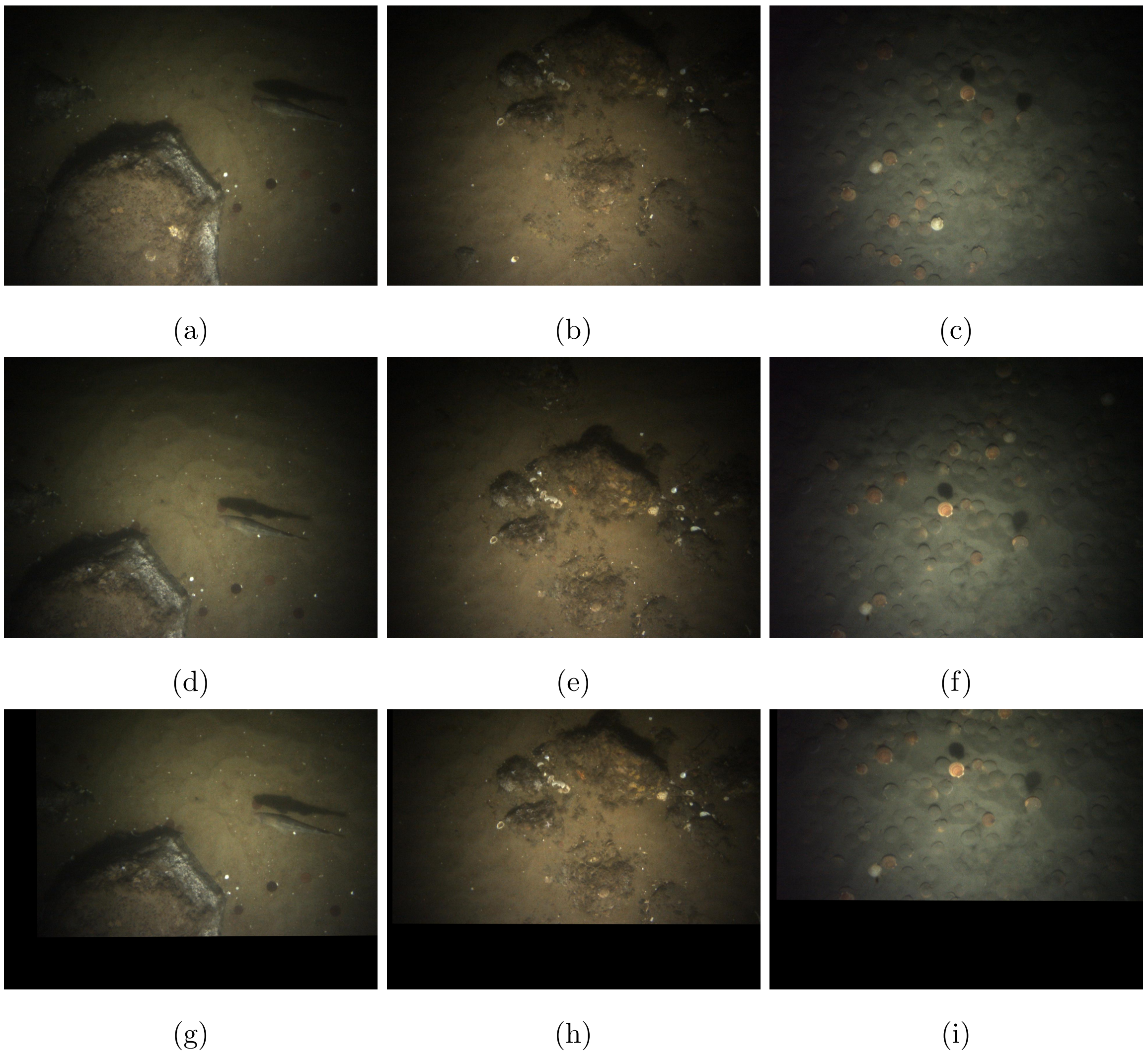
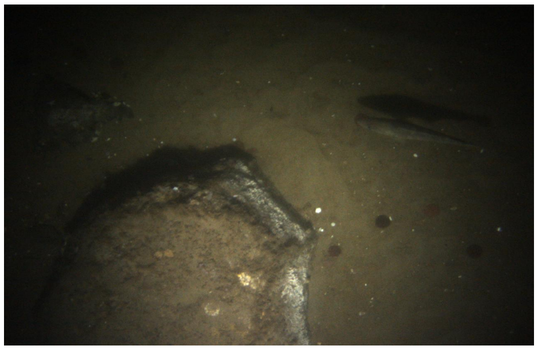


Figure 3.3: First two rows: examples of image pairs with distinctive objects; Third row: homographies computed from SuperGlue [116]

Figure 3.4 gives the example correspondences output from SuperGlue [116] where the matches are colored by their predicted confidence in a jet colormap (i.e.

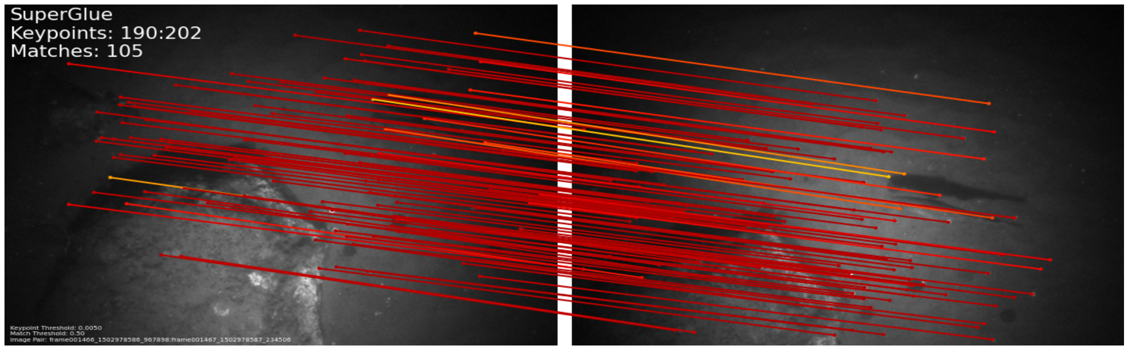
the red line means higher confidence and the blue line indicates less confident). We generated homographies of the image pairs by registering each image pair to be in the same coordinate systems to further validate the accuracy of matching feature keypoints extracted from the model. The homography should be perfectly aligned with meticulous feature matching. Example results can be found in Figure 3.3g - 3.3i, we show the homographic images computed from the image pairs in the first two rows in Figure 3.3. The homographic image contains the overlap area only and projects the second image in the same coordinate frame of the first image. Based on the point correspondences results, we can tell that SuperGlue [116] generalized well on images with low contrast and unconventional illumination changes. Small register deviation happened commonly among the areas that had apparent motion variations. Reasons that account for such changes in the images are: 1). Independent movement of animals; 2). Parallax due to objects that are not co-planar with the rest of the features; 3). Radio distortions; 4). Possible registration miscalculations.

After observing the promising results from SuperGlue [116] on our selected test image sequences, we continued running the registration network on the entire datasets of images from other missions for our later image mosaicing tasks (see Section 3.3).

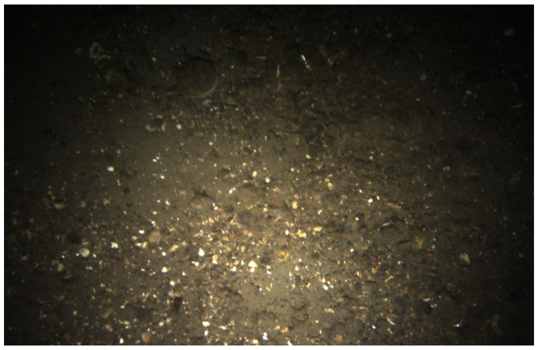


(a)

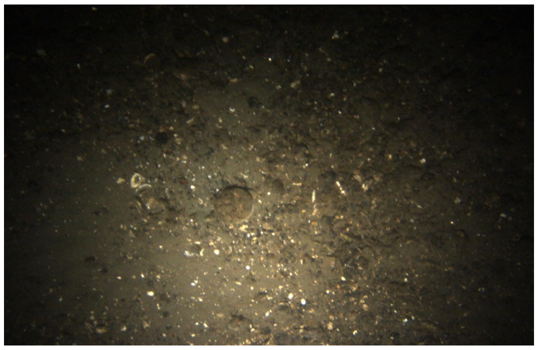
(b)


SuperGlue
Keypoints: 190:202
Matches: 105
Keypoint Threshold: 0.0050
Match Threshold: 0.50
Image Pair: frame001466_1502978586_967898:frame001467_1502978587_234506

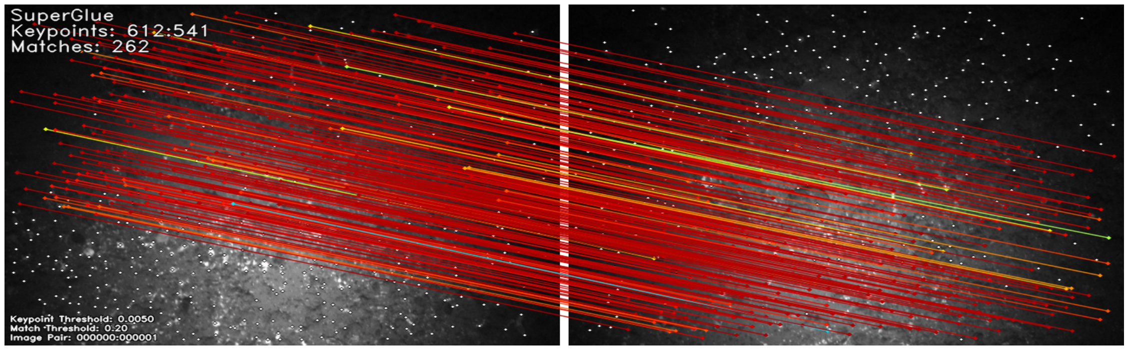
(c)



(d)



(e)



(f)

Figure 3.4: Example outputs from SuperGlue [116] showing feature point matches of image pairs

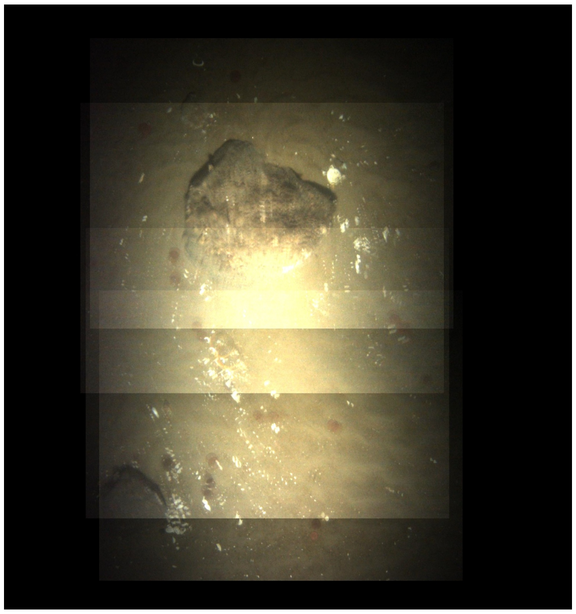
3.3 Image mosaicing

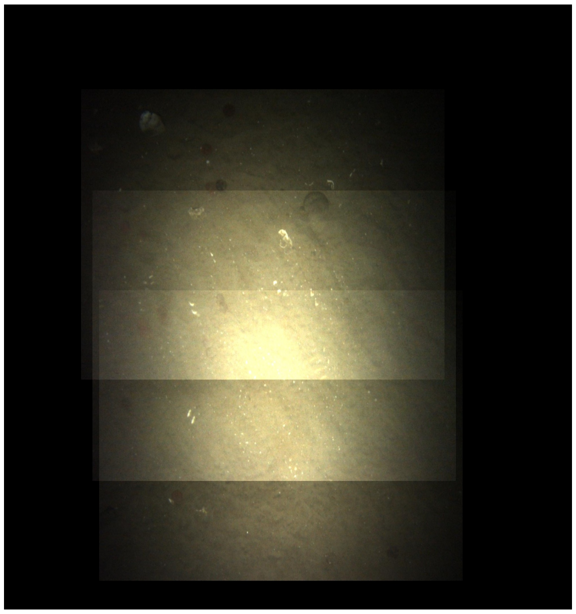
Since SuperGlue [116] has shown its capability of finding matching points between adjacent images with distinct features we further run the registration network on large datasets to evaluate its robustness. Campaigns and missions we evaluated SuperGlue [116] on:

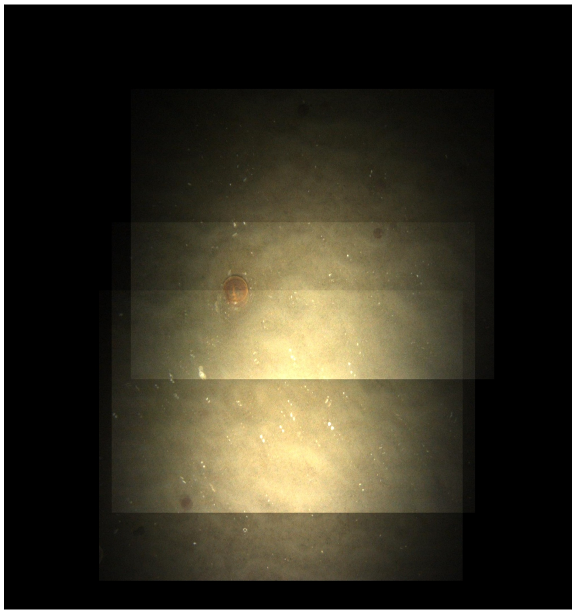
- `20170817_IM`: 17 *legs* containing 7,791 images.
- `20170824_VIMS`: 25 *legs* containing 103,528 images.
- `M107`: *Leg* `20170825235249` containing 1,450 images.

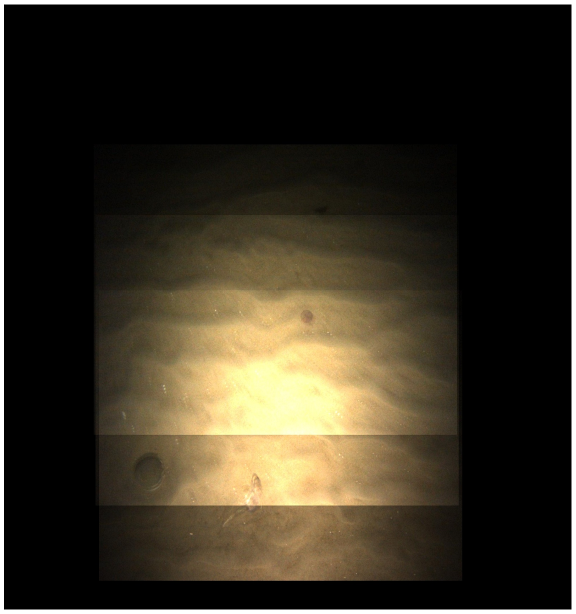
As discussed previously,the AUV followed a preset "lawn-mowing" pattern during each mission in order to photographically cover the study areas and images were captured at 3.75 Hz with the AUV traveling at a certain speed which resulted in an overlap between consecutive images in a line. In our previous experiments, we skipped some in-between images and only picked every fourth image in the datasets however there were overlaps that still exist between successive images. The same scallops were double counted if they appeared in overlapping areas of successive images which led to an inaccurate population estimation. Hence we intended to integrate multiple overlapped images in order to acquire more accurate scallop counts over the sequences as well as removing the distortions between images caused by the movement of the AUV.

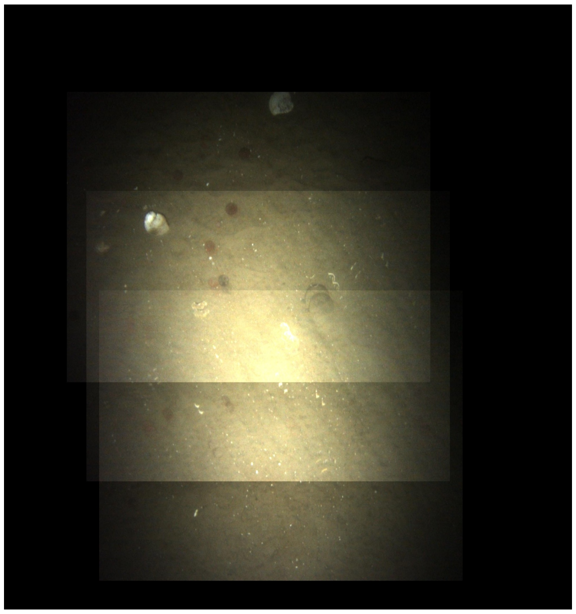
(a)

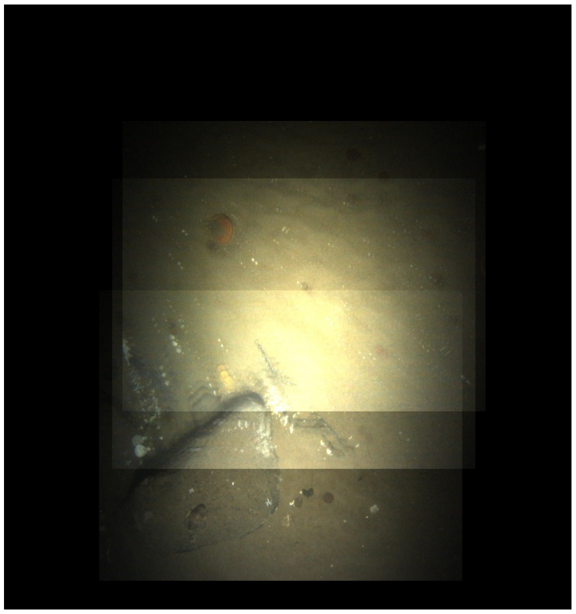

(b)

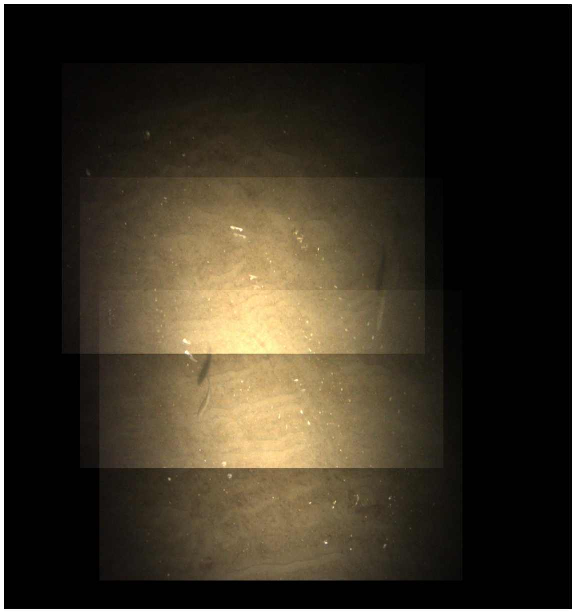

(c)

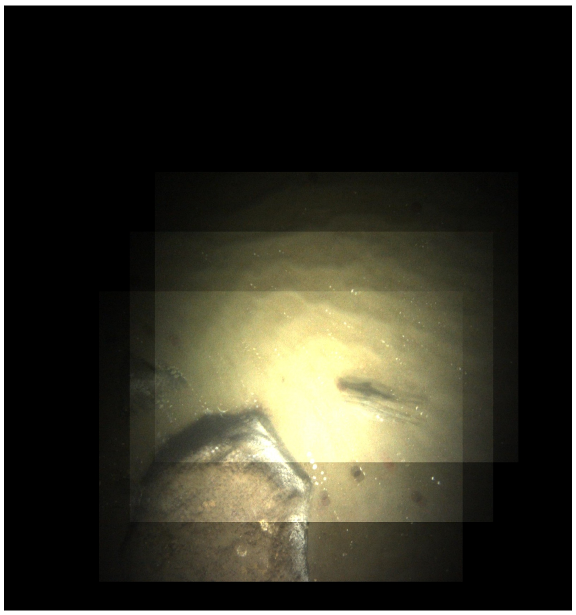

(d)


(e)

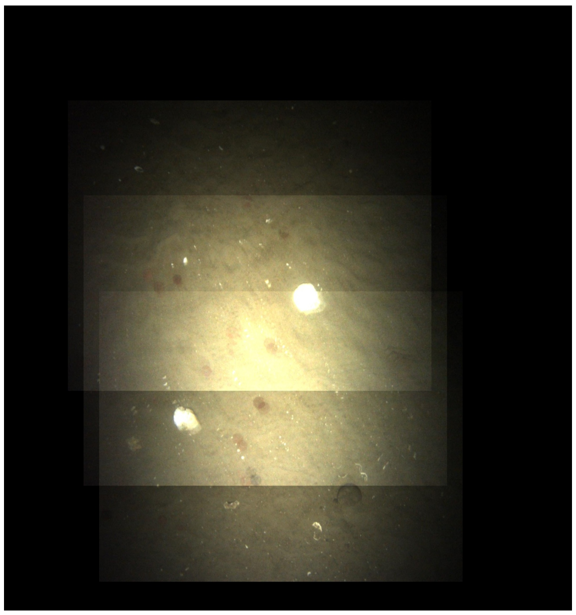

(f)


(g)

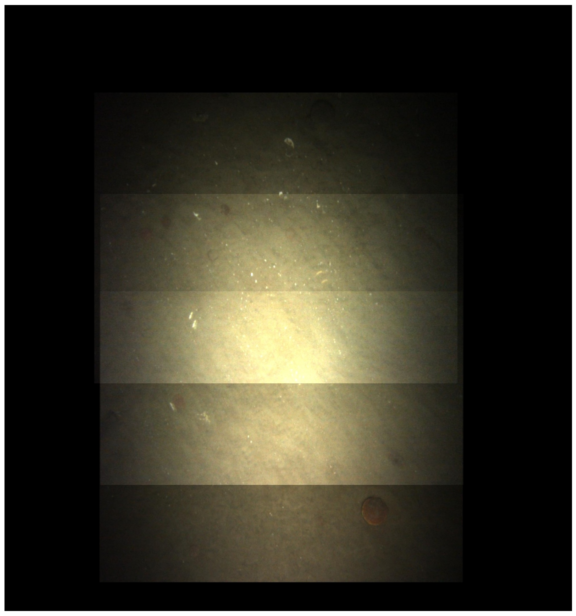

(h)



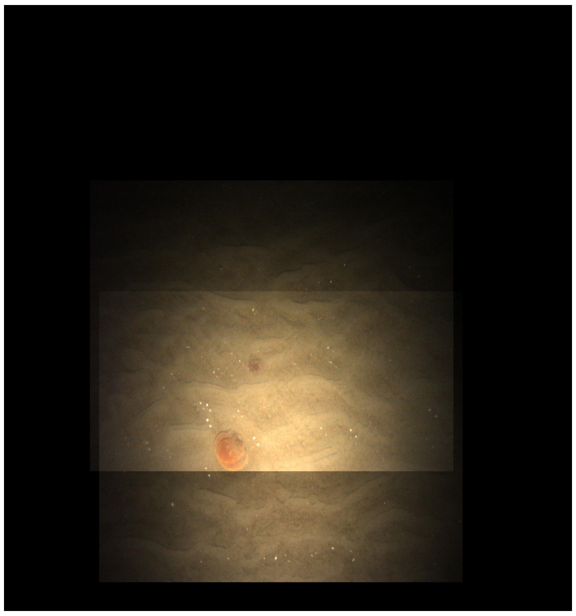
(i)



(j)



(k)



(l)

Figure 3.5: Example image mosaics from mission `20170817_IM`

From the keypoints and matches generated after registration, we iterated through all neighbouring images of the current image in order to find the sequences of successive images that contain sufficient overlapping areas. Based on the UTM ((Universal Transverse Mercator) coordinates we were able to align multiple scenes to a single unified frame. Figure 3.5 gives example mosaic images generated from mission `20170817_IM`. Even though mission `20170817_IM` is an area of low scallop density with sparse gravel and rocks laying on the seabed , SuperGlue [116] was still able to find sufficient matching keypoints (e.g. Figure 3.5a, 3.5c). As the center of images are typically clearer than the corners due to lighting configuration of the AUV, the mosaics are beneficial on improving the illumination condition around the corners of the images where the objects may be too dark to detect. Generating the mosaics also allows us to obtain a wider view of objects that are too large for a single image (Figure 3.5b, 3.5k, 3.5g) which can be used as landmarks for correlation with data from other sources.

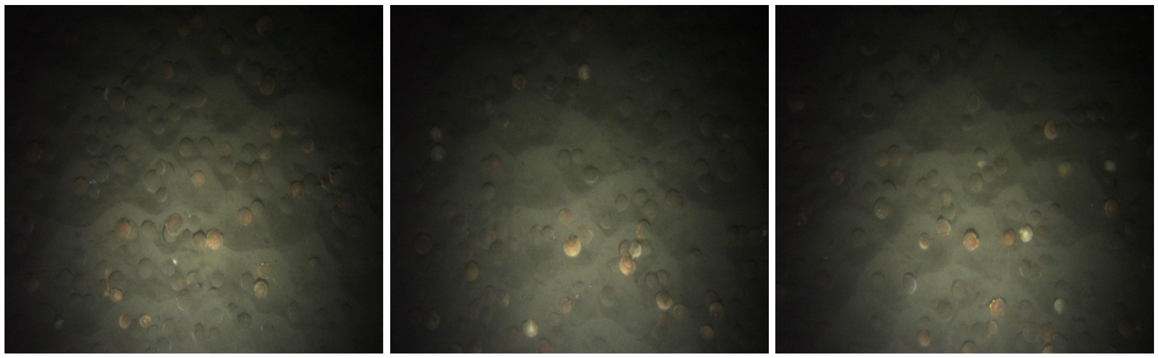

(a) (b) (c)

Figure 3.6: Example images of high scallop density areas from mission `M107`

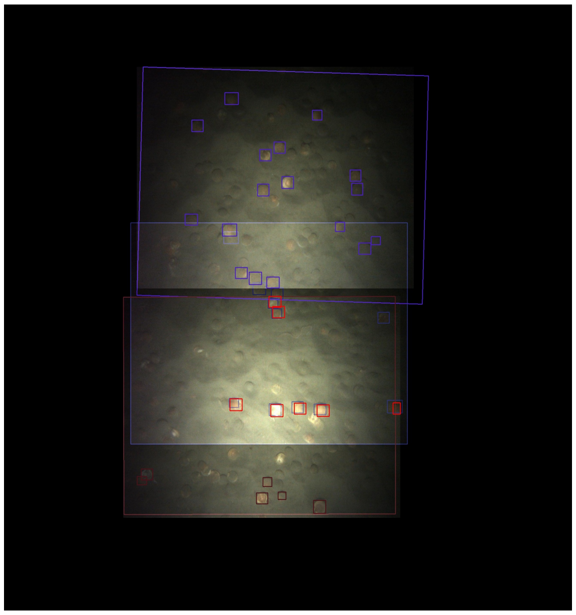
In order to evaluate our hypothesis on correcting scallop counts, we performed image mosaicing on images from one leg (i.e. `20170825235249`) of mission `M107` which is an area of sandy substrate with high scallop density (Example images of these areas are shown in Figure 3.6). We only considered *healthy* scallops for this experiment and run the 2-class classifier trained with ignored threshold of 0.5 from Section 2.2.1 on a total of 1,449 images. There were in a total of 23,364 *healthy* scallop detections collected on individual image frames. Noted that the detection results shown are different from our previous one-class detectors in Chapter 2 since we excluded the *compromised* detections classified by the network. For the detections in the overlapped areas between consecutive image, we used a IoU threshold of 0.2 to determine whether the two detection boxes imply the same scallop. As illustrated in Figure 3.7 we draw the detection bounding boxes on each individual image with the same color of the image outline and use red boxes to indicate duplicate detections. It is obvious to notice that nearly every scallop detection within the overlap areas are counted multiple times implying an over-calculated scallop counts. By removing the duplicates, there are 14,522 *healthy* scallop detections in `20170825235249` of mission `M107`.

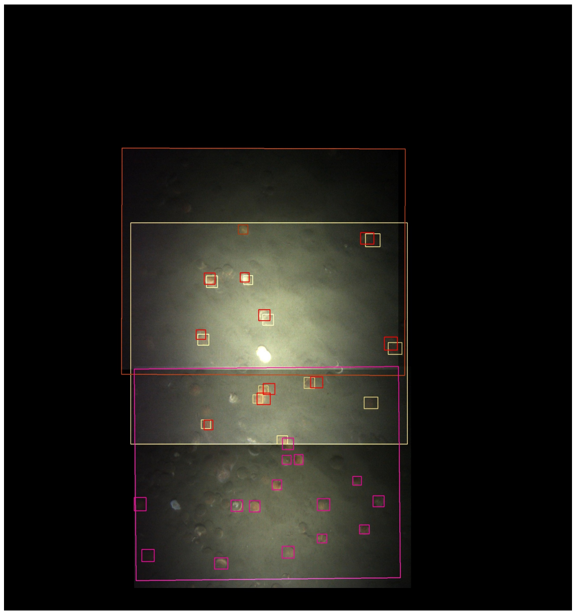
	Num. of Original detections	Num. of *Corrected* detections	Double-counted percentage
`20170817_IM`	1,219	702	42%
`20170824_VIMS`	50,363	21,032	58%
`M107-20170825235249`	23,364	14,522	38%

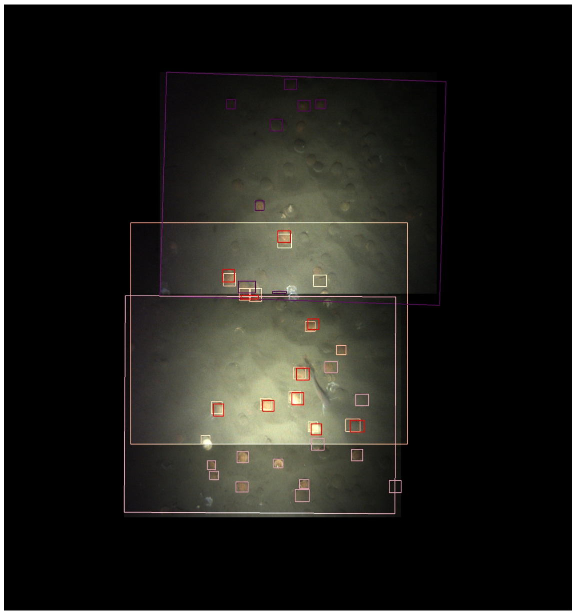
Table 3.1: Number of detections and the corresponding double-counted rates for the 3 separate datasets.

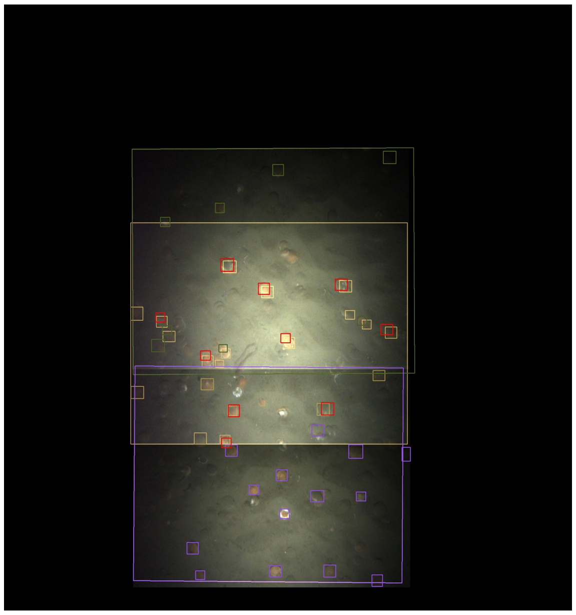
We continued performing the same approach on images from `20170817_IM` and `20170824_VIMS`. `20170817_IM` is dominated by gravelly and rocky sediments with less amount of scallops. 1,219 scallop detections were collected by the classifier from which 517 overlapped boxes were removed resulting in 702 *corrected* detections. For `20170824_VIMS`, the areas are heavily sand-covered with high scallop density. After correction, the total number of detections was reduced from 50,363 to 21,032. Results are summarized in Table 3.1 from which we show the double-counted percentage for the 3 datasets and we can see that nearly half of the detections generated from individual images were double counted resulting in a over-calculated assessment.

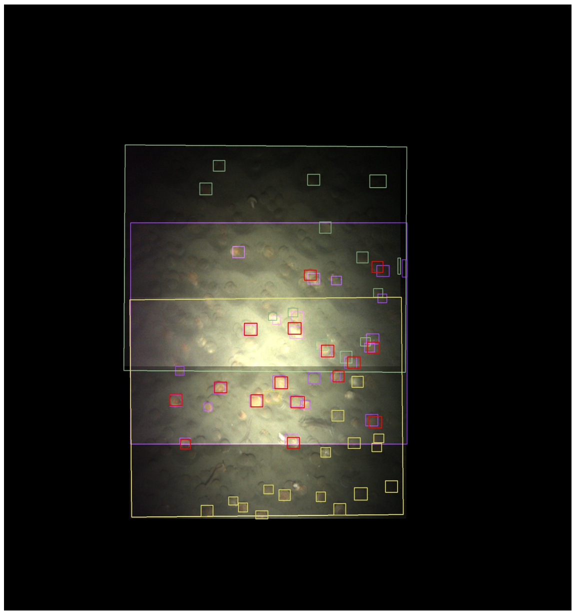
Given the results from this experiment, we believe that besides the importance of detection accuracy on individual images, it is crucial to avoid double-counting the same scallop by the adoption of robust registration and proper alignment of consecutive images for the purpose of achieving the most accurate scallop population estimation.

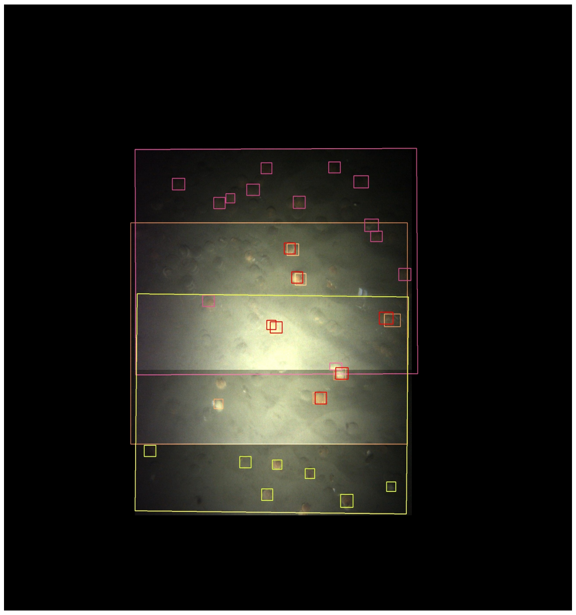

(a)

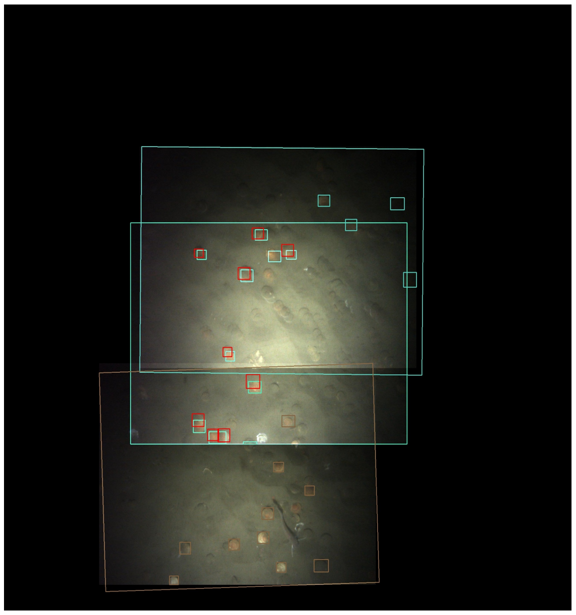

(b)

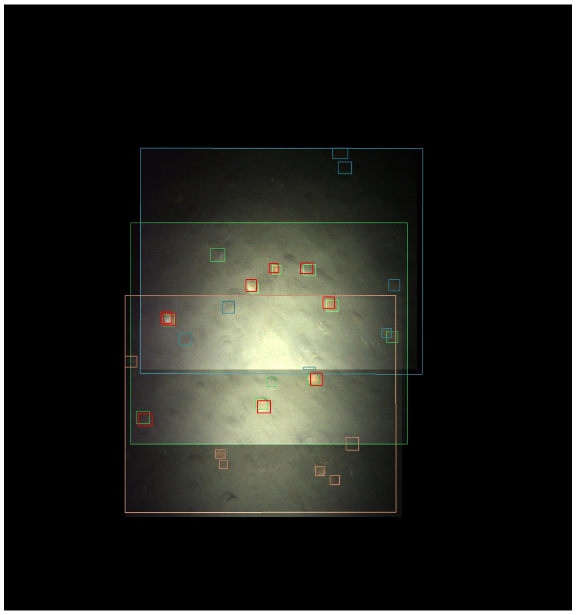

(c)

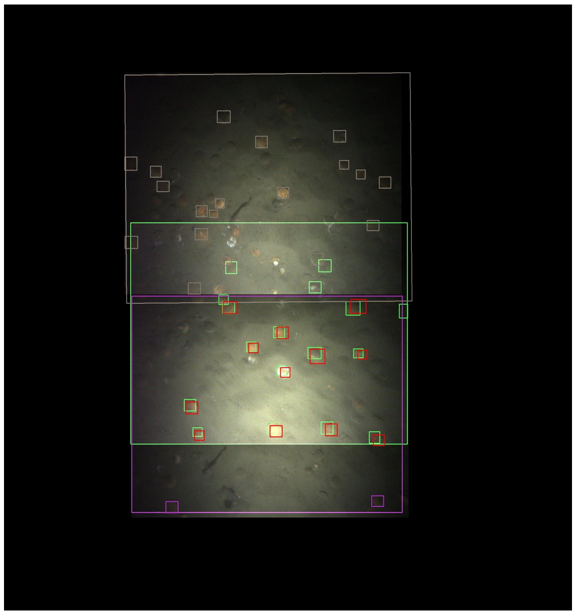

(d)

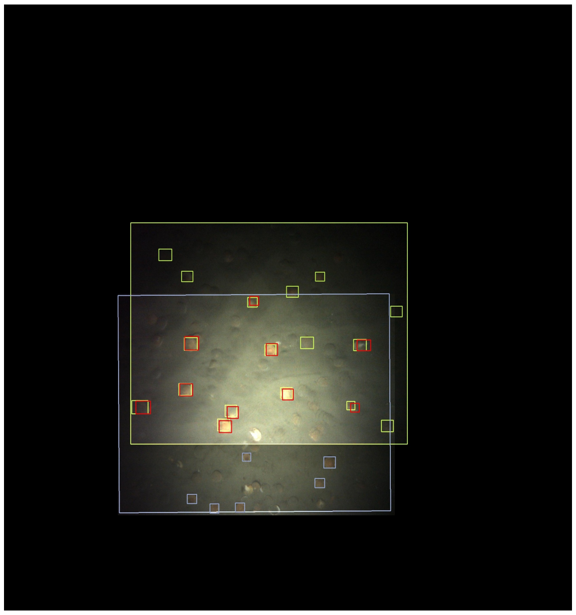

(e)

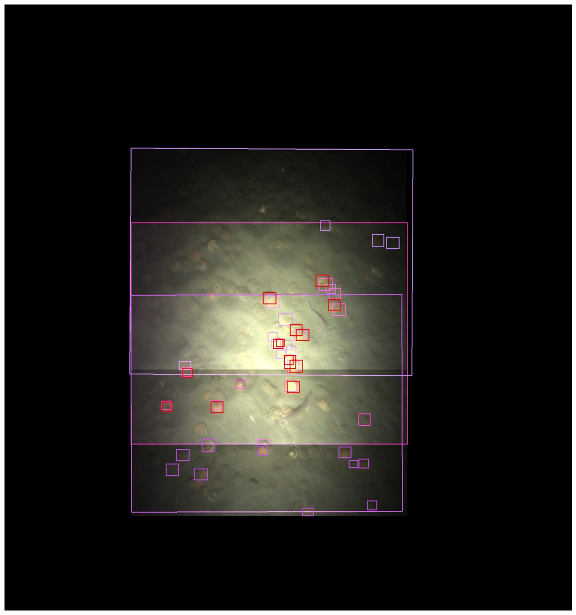

(f)


(g)


(h)


(i)


(j)


(k)

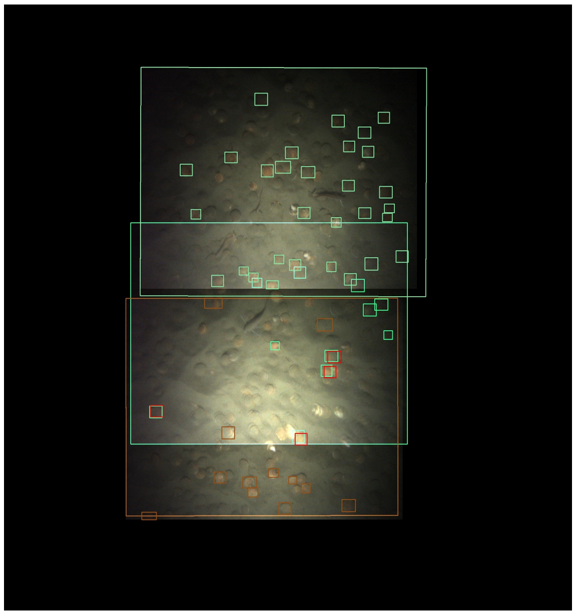

(l)

Figure 3.7: Example image mosaics with scallop detection bounding boxes from `M107-20170825235249` ; Detection bounding boxes are color-coded by which image they belong to; Red boxes indicate the double-counted detections in overlapping areas.

3.4 Classification on NEXRAD Data sequence

In this section we discuss a project of multi-class classification conducted for biological applications on data collected from next-generation weather surveillance radars (NEXRAD) (shown in Figure 3.8) during 1996-2017. The NEXRAD system is composed of 160 radar sites throughout the United States for monitoring atmospheric and biological movements [91]. A standard NEXRAD radar that has a diameter of 9.1 m and an aperture diameter of 8.5 m operates in three-dimensional at a frequency of 2800 MHz [144]. There are currently two types of data collected from the NEXRAD. The base Level-II data contains three meteorological quantities and the Level-III data is an upgrade from Level-II with vertical polarization added to distinguish additional meteorological conditions and movements.

Figure 3.8: Example image of the next-generation weather surveillance radars (NEXRAD)

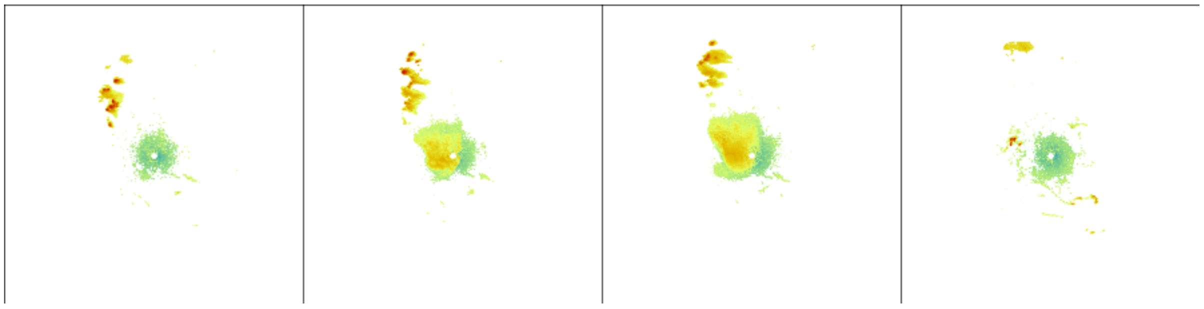
In Section 3.4.1 we explain the data format and the preprocessing steps required to analysis the dataset; Section 3.4.2 comprises the experiments we conducted and the corresponding results;

3.4.1 The NEXRAD Dataset

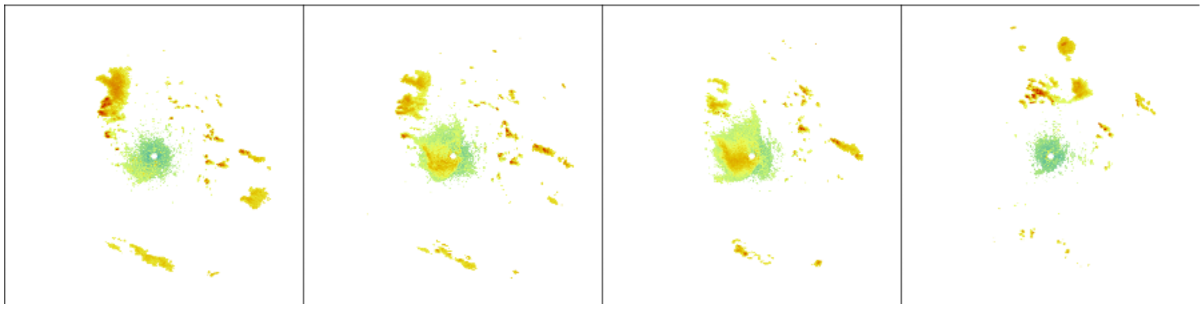
The Level II NEXRAD radar data consists of 3 products of Radial Velocity, Reflectivity, Spectrum Width from 1995-2012 (Legacy) and 6 products of Correlation Coefficient, Differential Phase, Differential Reflectivity, Radial Velocity, Reflectivity, Spectrum Width from 2012-present (Dual-pol). There are 44 radar cites collected a total of 36,196 data instances in spring and autumn. Each data instance contains a number of radar format files recorded during a certain period of time at night.

There are four major classes: *Bird-dominated Movement* (B), *Insect-dominated Movement* (I), *Contamination* (C) and *No-bird* (NB). The contamination class can be further divided into four types: *Precipitation* (P), *Trans-gulf movement* (T), *Anomalous Propagation* (AP), and *Clutter* (CL). Light precipitation are shown in green on the radar display with yellow, red and magenta indicating the increased amount of precipitation. The trans-gulf movement occurs during biological migration near coastal areas where the targets (e.g., birds) are present over water and gradually move inland. Anomalous propagation is often characterized by echoes at far distances from the radar and spreading over time far outward from the normal range of the radar. The clutter type usually indicates smoke, sea breezes or other radar echoes with no distinct type of structure.

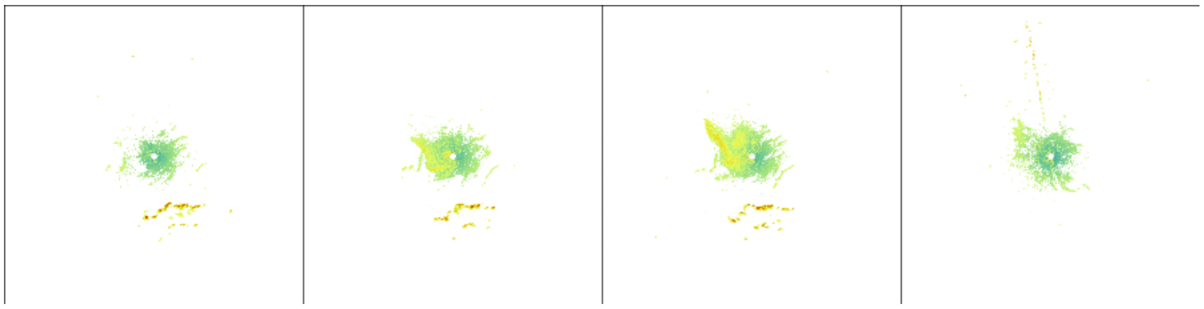
We used PyArt [47], an open source Python library to visualize and convert the raw radar data into color-mapped image sequences using the *reflectivity* feature. Each image was cropped and scaled from the center to a 224×224 sub-image and then used as input to the networks. Example image sequences of classes *Bird-dominated Movement*, *Insect-dominated Movement* and *Contamination* are shown in Figure 3.9. The first column and fourth column are the first and last frame from the sequence. And we pick two frames in between to illustrate the transformation. We also noticed that the number of frames varies from each data instance and Figure 3.10 illustrates the distribution of the number of frames across the entire dataset.



(a)



(b)



(c)

Figure 3.9: Example Image sequences of the three classes from KAMX radar site in Spring. (a) Bird ; (b) Insect ; (c) Contamination.

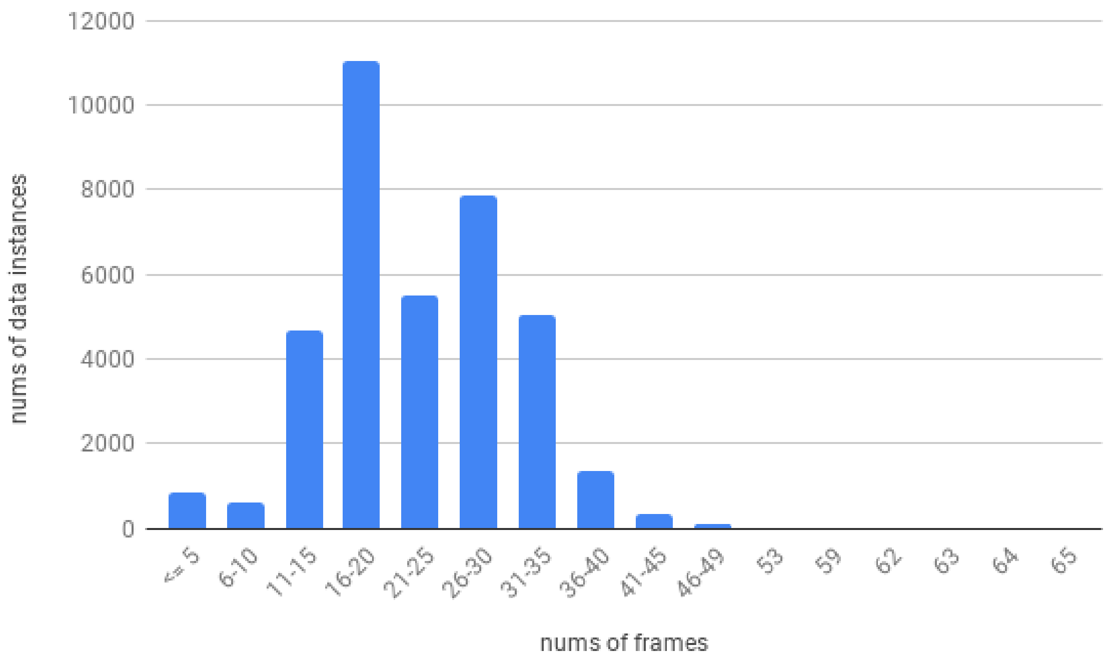


Figure 3.10: A distribution of the number of frames in each sequence across the entire dataset.

3.4.2 Experiments and Results

For our experiments we only considered *Bird-dominated Movement*, *Insect-dominated Movement*, *Contamination* as a 3-class problem. We split the dataset 80/20 for train/test. There were 29,305 sequences of which 656,465 frames in the training set: 4,992 sequences (98,307 frames) of *Bird*; 2,769 sequence (53,851) of *Insect*; 21,544 sequence (504,307 frames) of *Contamination*. To deal with the class imbalance issue, we also created a balanced dataset that had around 2,600 number of sequences in each class and every sequence was augmented to over 40 frames. This balanced dataset was spit 80/20 for train/test as well. There were 7,315 sequences of 309,862 frames in total in the training set and 1,855 sequences of 78,517 frames in the test set.

CNN for single frame classification

We first used a CNN to classify single frame at a time. The architecture we adopted was a pre-trained Inception-V3 network from Tensorflow Hub [4]. Additionally we deployed a voting mechanism that aggregate the predictions of each frame within a sequence and assign a final predicted class to this sequence by majority votes. The network trained on our original training set for 25K steps.

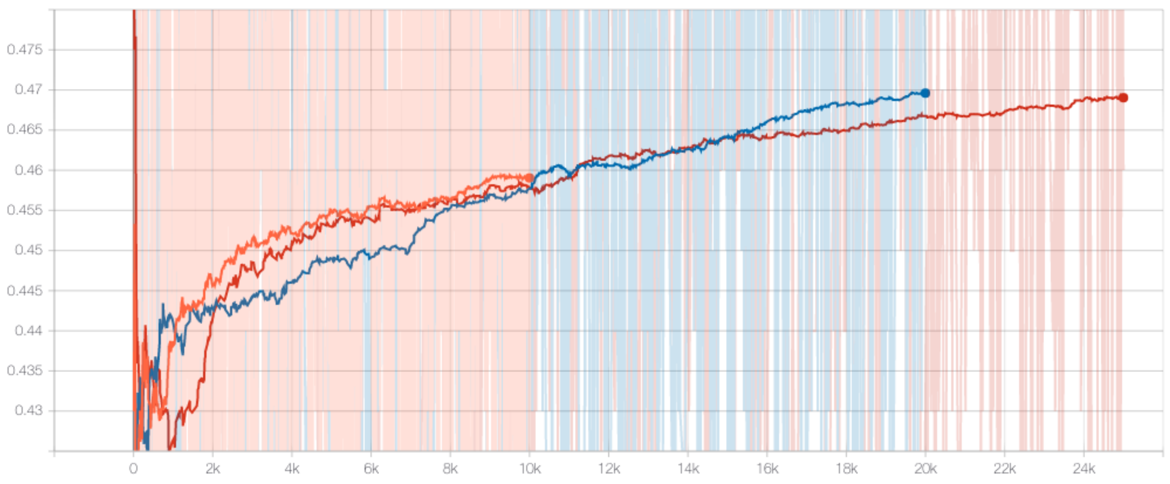
We stopped the training process at both 10K steps and 25 steps to run a validation on the test set. As illustrate in Figure 3.11 (a), the training accuracy (0.470) did not improve from 20K to 25K steps. And the test accuracy peaks at 20K steps of 0.653 and 0.686 after voting. We also trained the Inception network on the balanced dataset for comparison with the other methods described below.

LRCN

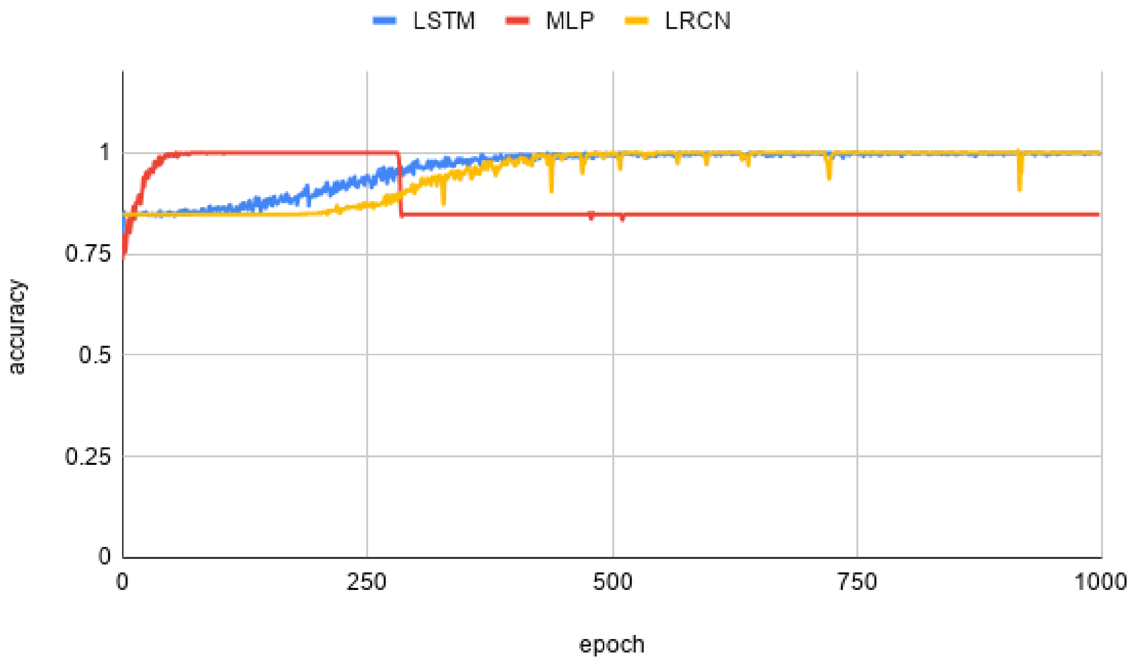
In this experiments we built a CNN into a RNN, as known as the long-term recurrent convolutional network (LRCN), in consideration of the temporal features of the data. We used a smaller VGG-style [125] network that consists of 10 convolutional layers with batch normalization and ReLU activation and 5 max-pooling layers for the CNN part of the model. And then followed by a LSTM network. This model was trained from scratch for 10k epochs on the balanced dataset.

Extracted Feature from CNN

In the following two experiments we explored if the feature extracted from a CNN would benefit the classification results of other networks. We first had every frame from the balanced dataset run through the Inception network and saved the outputs from the last pooling layer. The features extracted are 2048-d vectors that can be used as input for other networks. The extracted features were then fed into a separate RNN: a 2048-wide LSTM network with dropout and ReLU activation. The second experiment was to flatten the feature and then fed into a 5-layer MLP that has dropout and ReLU activation in between as well. Both of this network were trained for 10k epochs.



(a)



(b)

Figure 3.11: (a): Training accuracy of Inception-v3 on the original training set for 10K (orange), 20K (blue) and 25K (Red) steps; (b) Training accuracy of LRCN, LSTM, MLP models for 1K epochs.

Results

The training accuracy of LRCN, LSTM, MLP is shown in Figure 3.11 (b). We only plot results of the first 1k epochs since all the model has already converged within this period. The LRCN method reached 0.416 accuracy during testing. We noticed that the validation accuracy dropped since the 50th epoch while the training accuracy kept increasing that may indicate overfitting. The LSTM network has a test accuracy of 0.482 and the MLP model achieves 0.456 test accuracy which shows that features extracted by a CNN would be of helpful in processing temporal data. Our baseline approach of using Inception network reached 0.433 test accuracy and 0.443 after voting mechanism.

Generally our approaches did not perform well on this dataset. One may be due to the high similarity between samples in different the categories. And we only used reflectivity to convert the radar data to images. As there are other products such as correlation,velocity, spectrum that can be used as channel to generate high-dimensional imagery. And we also expect to adopt a more sophisticated 3D network that can utilize the spatial and temporal feature of the data.

Chapter 4

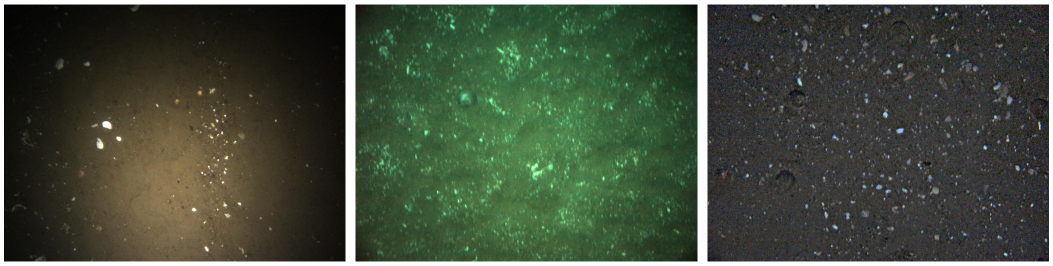
MULTI-SENSOR TERRAIN ANALYSIS

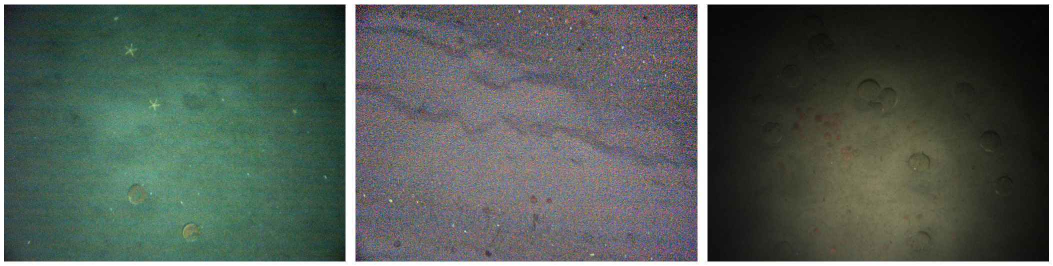
4.1 Image-based Terrain Analysis

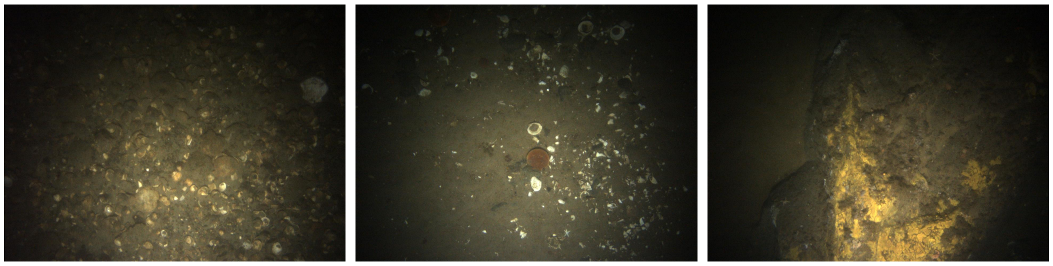
In this section we investigate the terrain type distribution of the seabed in the areas of the BACI studies. Section 4.1.1 disuses the multi-class classifiers we built to distinguish various forms of sediments. In Section 4.1.2 we correlated scallop quantity with terrain types to demonstrate the habitat preference of sea scallops for our population estimation study.

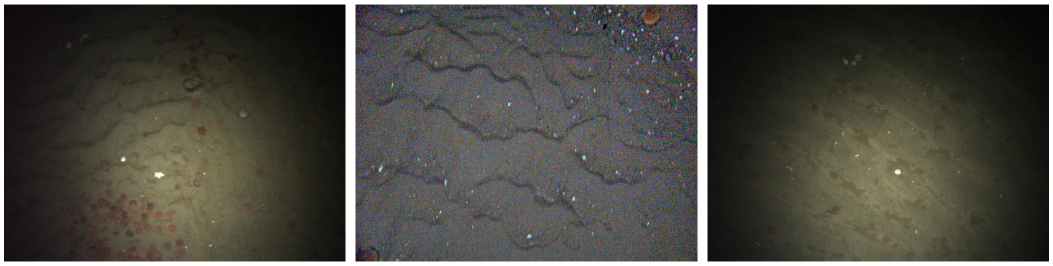
4.1.1 Terrain Classification

From our prior knowledge and observations, images collected from different missions show diverse textures or appearances of the seabed and scallops or other relevant creatures are very likely to aggregate on certain types of benthic substrates. Figure 4.1 shows examples of seabed sentiment as well as terrain categories, such as gravelly areas with large amounts of debris, rocks and shell hashes (Figure 4.1c), feature-less (Figure 4.1b) or rippled (Figure 4.1d) sandy substrate. Mounds of sand dollars (Figure 4.1d) usually appear in both gravelly and sandy areas. To further improve our marine species population study, we developed a whole-image terrain classifier such that the priori contextual information of the seafloor would be beneficial to the AUVs for path planning, target detection, and other perceptual tasks in real-world environments.


(a) Gravel


(b) Sandy


(c) Rocky


(d) Ripple

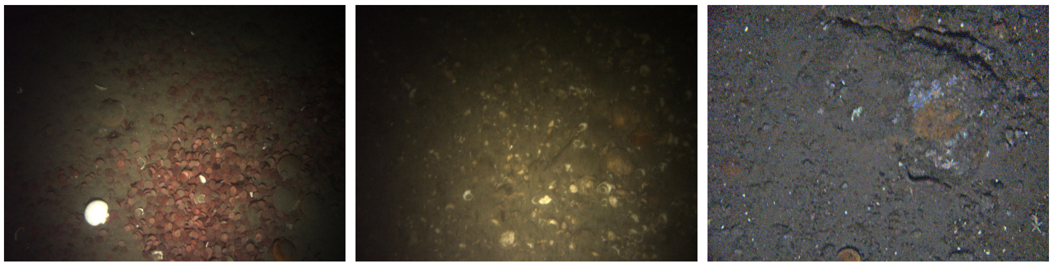

(e) Mounds

Figure 4.1: Example of different terrain categories

The dataset we used for this implementation consists of images of the seabed that are annotated with terrain labels according to their benthic texture. There are six categories and five of them are different terrain types: *Gravel* (Figure 4.1a), *Sandy* (Figure 4.1b), *Rocky* (Figure 4.1c), *Ripples* (Figure 4.1d) and *Mounds* (Figure 4.1e). Images that do not belong to either of the terrain types (e.g. blurry images or images taken in poor reflection) are categorized into the *Null* class (Figure 4.2). The dataset is formed from 241,951 images, a mixture of raw images and Retinex-enhanced images, and 26,481 (*Gravel*), 93 (*Mounds*), 27,766 (*Null*), 1,360 (*Ripples*), 435 (*Rocky*), 185,816 (*Sandy*) images in each class respectively. Note that not all images in the entire dredging dataset were annotated with terrain labels. We select 4 campaigns (`20150711_IM`, `20170817_IM`, `20170822_VIMS`, `20170824_VIMS`) where a large ration of images have terrain annotations to demonstrate numerical proportions of each terrain type in Figure 4.3. From where we learn that *Sandy* is the dominant class among all terrain types and *Gravel* is the second largest class in the newest 2017 AUV campaigns. For earlier campaigns, a numerous number of images were labeled as *Null* which could be due to the camera we used for earlier missions was inclined to capture low quality images.

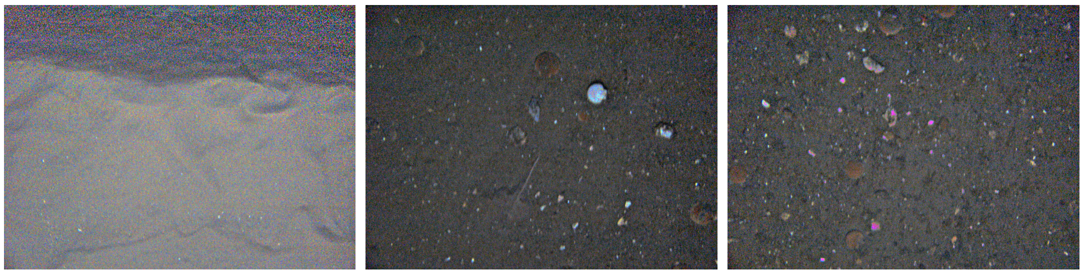


Figure 4.2: Example images in *Null* class

We conducted three multi-class image classification experiments and a additional binary classification task. We trained Inception-V3 [4] classifiers using the same train/test split ratio (80/20) for all the experiments.

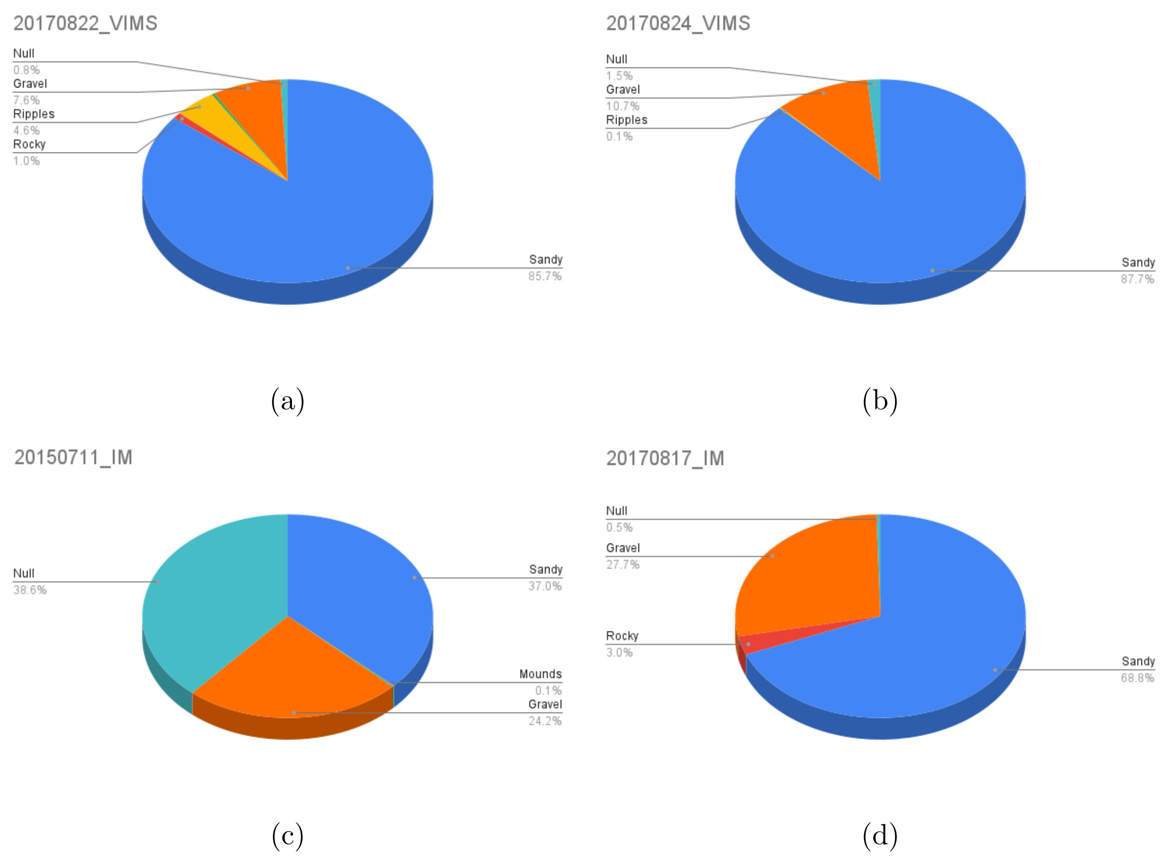


Figure 4.3: Pie chars of terrain type distributions by campaign

In the first experiment, we trained the classifier on subset of images collected in 2017 which was the latest annotated data. This subset contains approximately 20% of images of the entire terrain-labeled dataset. The training set had 38,668 images with 9,848(*Gravel*), 27 (*Mounds*), 714 (*Null*), 378 (*Ripples*), 303 (*Rocky*), 27398 (*Sandy*) images in each class respectively. The test set had 12,890 images. We trained the network (denoted as **2017-subset** in Table 4.1) for 10,000 steps. It achieves an average precision of 0.357, an average recall of 0.794 and overall test accuracy of 0.696. Precision and recall values for each class achieved are presented in the first two columns in Table 4.1.

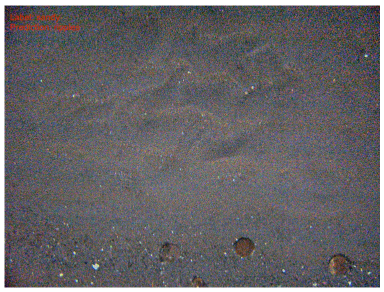
In the second experiment, we trained the classifier on the entire terrain-labeled dataset and used 20% of the training data for validation. The validation accuracy peaked at 20,000 steps and the model exhibits the test accuracy of 0.667 with an average precision of 0.325, an average recall of 0.682. Precision and recall values for each class achieved are presented in the third and forth columns in Table 4.1.

Since the *Null* class was made up of disordered images which could be confusing to the classifier leading to a number of false positives, for the third experiment we excluded the *Null* class and trained a 5-class classifier for 20,000 steps. The test accuracy of this network increased to 0.741 as predicted. The last two columns in Table 4.1 give the individual precision and recall values of every class.

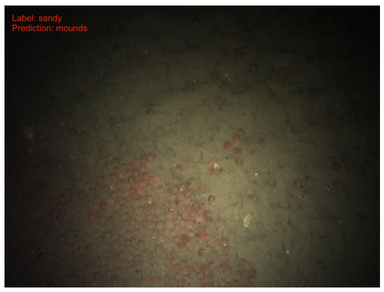
	2017-subset		**All-image**		**5-class**	
	Precision	Recall	Precision	Recall	Precision	Recall
Sandy	0.929	0.645	0.952	0.673	0.957	0.748
Rocky	0.142	0.821	0.081	0.908	0.085	0.897
Ripples	0.136	0.889	0.085	0.868	0.087	0.912
Mounds	0.053	0.75	0.008	0.368	0.009	0.421
Gravel	0.608	0.787	0.453	0.462	0.399	0.684
Null	0.271	0.874	0.373	0.813	-	-
Average	0.357	0.794	0.325	0.682	0.307	0.732

Table 4.1: Precision and recall values from the terrain multi-class classifiers.

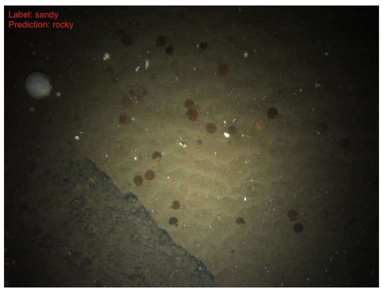
After reviewing the prediction results, we conclude three reasons for misclassification: 1). Images possess features of more than one category: For example Figure 4.4a shows a sandy sediment with ripple patterns, Figure 4.4b has mounds of sand dollars on half of a sandy image, Similarly, rocky texture appears on one third of Figure 4.4c and the rest of the image has ripple/sandy appearance; 2). Ambiguity of labeling: we noticed a inconsistent labeling across the categories and in such case some predictions are in fact correct but were recognized as false positives/negatives. Sand ripples are showing in Figure 4.4d and mounds of sand dollars appear all over Figure 4.4e but the two images are labeled as *Gravel*. Figure 4.4f is labeled as *Sandy* but apparently has the features of *Rocky*; 3). Skewed dataset: 77% of images in the data set are labeled as *Sandy* which has 7x more images than the second largest category and is 2 to 4 orders of magnitude more massive than the other categories. Class imbalance is the major issue that impact the performance of the classifiers.



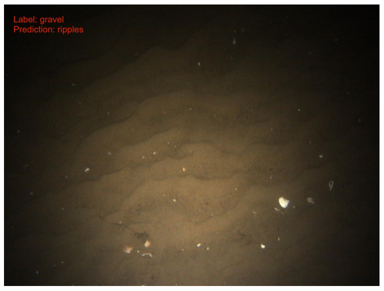
(a)



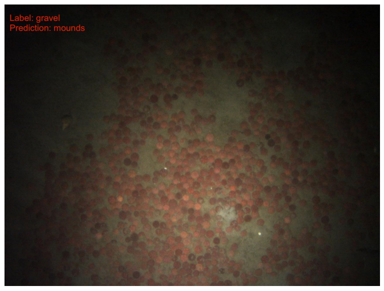
(b)



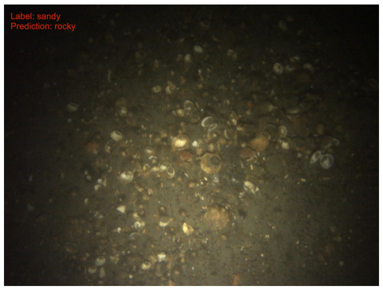
(c)



(d)



(e)



(f)

Figure 4.4: Examples of misclassified terrain images.

Under these circumstances our next step was to create a more balanced dataset by combining classes with high similarities. Class *Ripples* were merged into *Sandy* class as we witnessed that most of the *Ripples* images indeed had a sandy sediment. From Figure 4.1a, 4.1c and 4.1e we can see that images in *Gravel*, *Rocky* and *Mounds* have considerably similar textures among them hence it was intuitive to us to combine the three classes together. We trained a *Sandy*-*Gravel* binary classifier including all images from the 2017 dataset. There were a total of 50,606 images where 13,571 images belonged to the *Gravel* class and 37,035 images were *Sandy*. Images were split 80/20 for train/test and the classifier were trained for 10,000 steps. The network achieved an accuracy of 0.897 on the test set with the corresponding average precision of 0.859 and average recall of 0.914. Precision and recall values for the *Sandy* class are 0.876 and 0.978, for the *Gravel* class are 0.738 and 0.952, respectively. From the results we can tell that by combining the related categories it alleviated the impact of class imbalance and labeling inconsistency therefore reduced the number of false positives to a great extend. Comparison of the performance of all the classifiers can be found in Table 4.2.

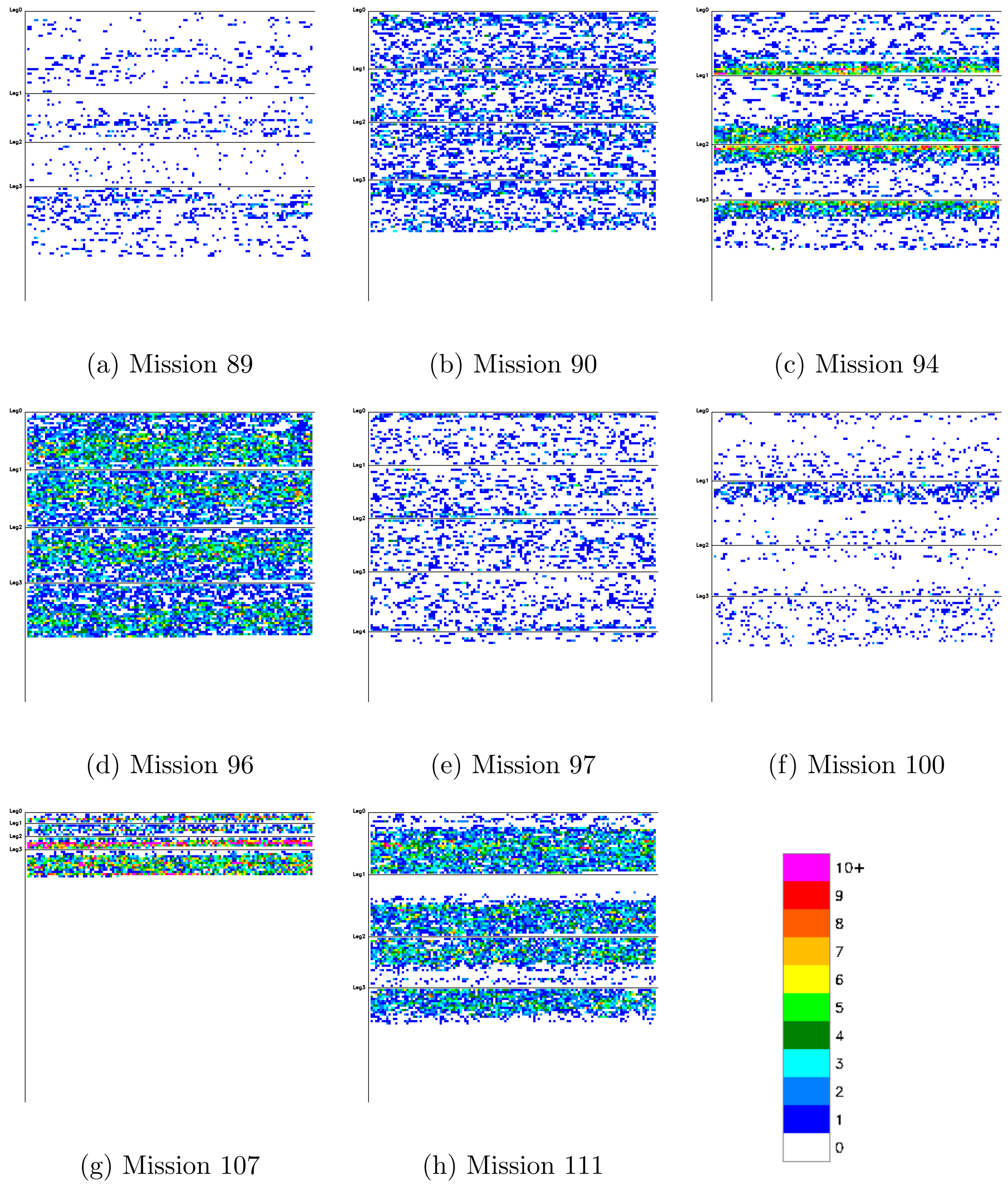
	6-class 2017-subset	6-class All-image	5-class All-image	Balanced 2-class 2017-subset
Avg. Precision	0.357	0.325	0.307	0.859
Avg. Recall	0.794	0.682	0.732	0.914
Test Accuracy	0.696	0.667	0.741	0.897

Table 4.2: Average precision,recall and test accuracy of the terrain classifiers.

4.1.2 Scallop Habitat Study

Establishing scallop-habitat relationship is crucial for scallop population census as well as protecting the native living environment. Terrain types is one of the most important characteristics in habitat structures that determines the abundance of sea scallops. We have observed from past experiments that are likely to aggregate at certain types of benthic substrate. In order to further investigate the habitat preference of healthy scallops, we related scallop density with the corresponding terrain types.

We first analyzed scallop density by missions based on the prediction results generated from our previous one-class detector in Chapter 2. True positive scallop detections were accumulated for each individual image in the missions on which we run the YOLO detectors. A visualization system was then developed that illustrates the quantitative distributions of scallops at different areas from each mission. We looped over every image with its corresponding scallop counts and arrange them geographically based on the image metadata. We adopted heatmap-like color-coding structure to represent different values from 0,1,2,...,9,10+ indicating the number of scallops within one camera image covered area. Warm colors under the red-yellow spectrum represent higher value points namely heavier scallop population and cool colors form the blue-green spectrum mean lower value points indicating sparse distributions. Example 2D histograms are shown in Figure 4.5. The y-axis indicates mission leg numbers and the x-axis represents the geographical location across study areas. Each row demonstrates the AUV trackline from one side to another. The density histograms provide us a more comprehensive overview of scallop population distributions among each mission as well as allowing us to segment and filter the data of interesting areas. From the example plots in Figure 4.5 we can tell that Mission 96, Mission 111 has a more extensive quantitative distribution of scallops across the AUV travelling areas than the other missions. Leg0, leg1 in Mission 107 and the bordered regions of legs in Mission 94 all have a high scallop density.



(a) Mission 89 (b) Mission 90 (c) Mission 94

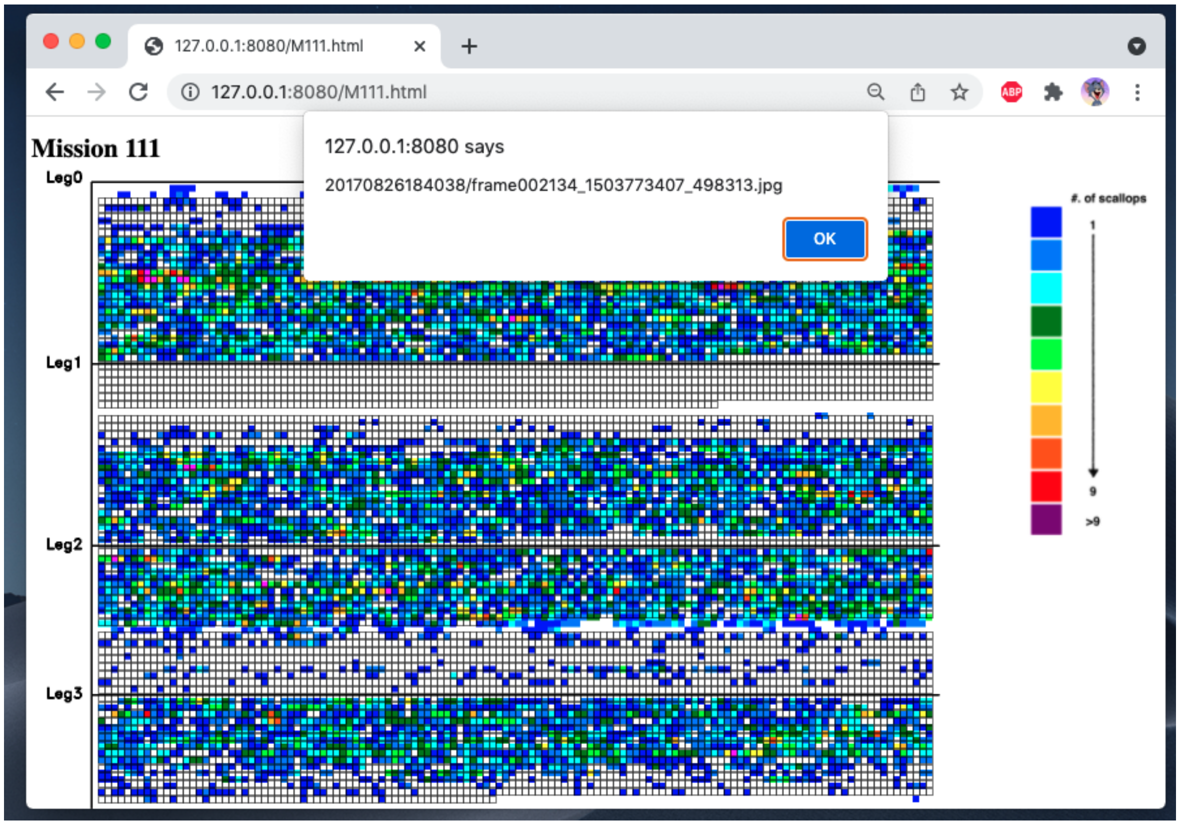
(d) Mission 96 (e) Mission 97 (f) Mission 100

(g) Mission 107 (h) Mission 111

Figure 4.5: Example 2D histograms of scallop population distribution by *Mission*

Additionally we extended our mission visualization system to a web application. Figure 4.6 presents an example of the user interface for Mission 111. The 2D histograms were indexed by their mission number which was easy to switch between missions within one cruise to get a general quantified distribution of scallops of the area. The paths of where significant images (i.e. high density) were located could be sought out simply by clicking on the corresponding squares.

Next we examined the correlation between scallop distribution and the terrain type. First of all, A quantitative analysis on scallop counts for each terrain type was conducted. As we mentioned above not all images in our dredging dataset have terrain labels therefore the images used in this section are only a subset and mainly from later (i.e. 2017) campaigns. In Figure 4.7a we plot the scallop counts categorically with the highest (Max), lowest (Min) and median values for each terrain type across the entire terrain dataset. The highest number scallops per image frame for each category are represented by the blue bars and the values are 280, 6, 3, 31, 6, 69, corresponding to *Sandy*, *Ripples*, *Rocky*, *Gravel*, *Mounds* substrates and *Null* category. The median values represented by the yellow bars for each category in the same order are 2, 1, 2, 2, 1, 3. All images of each terrain type have a least one healthy scallop. Furthermore we display a more detailed spread of scallop counts over different substrates as well as the skewness and variance of the data in Figure 4.7b. For each category, the minimum is shown at the left “whisker” and the maximum is at the far right. The yellow lines inside every box indicate the median values. Figure 4.7b gives us an overall pattern visualization for all categories. We can see that class *Sandy* has the most outliners and a extremely wider distribution compared to other classes. *Null* class includes all types of substrates and the ones with high scallop counts areas can potentially be dominated with sandy sediment.

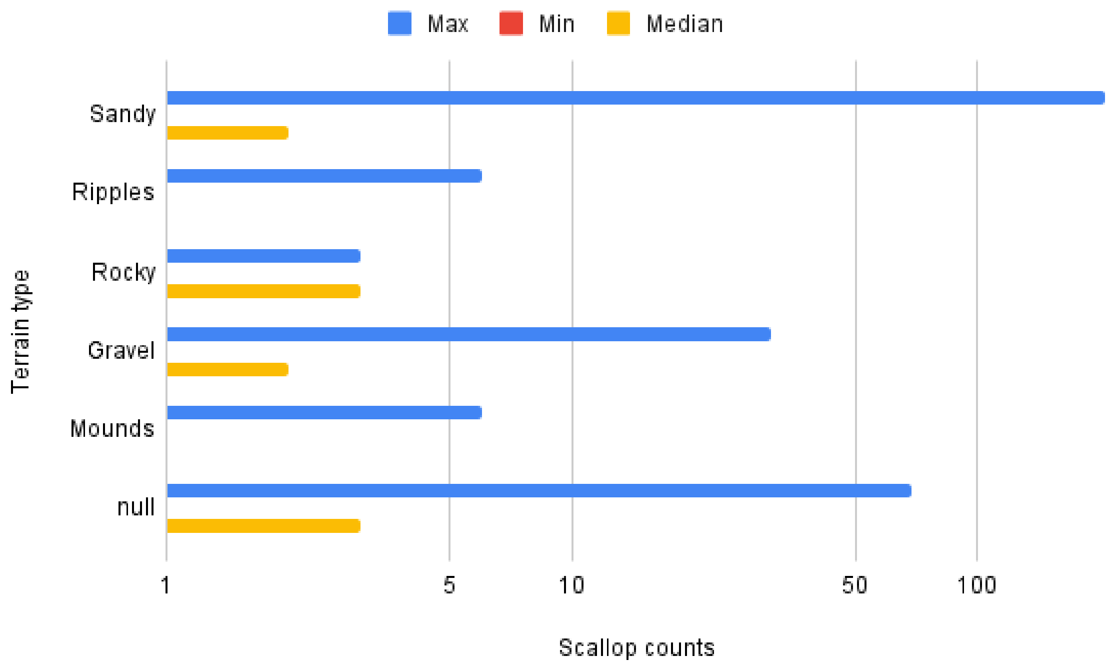


127.0.0.1:8080 says

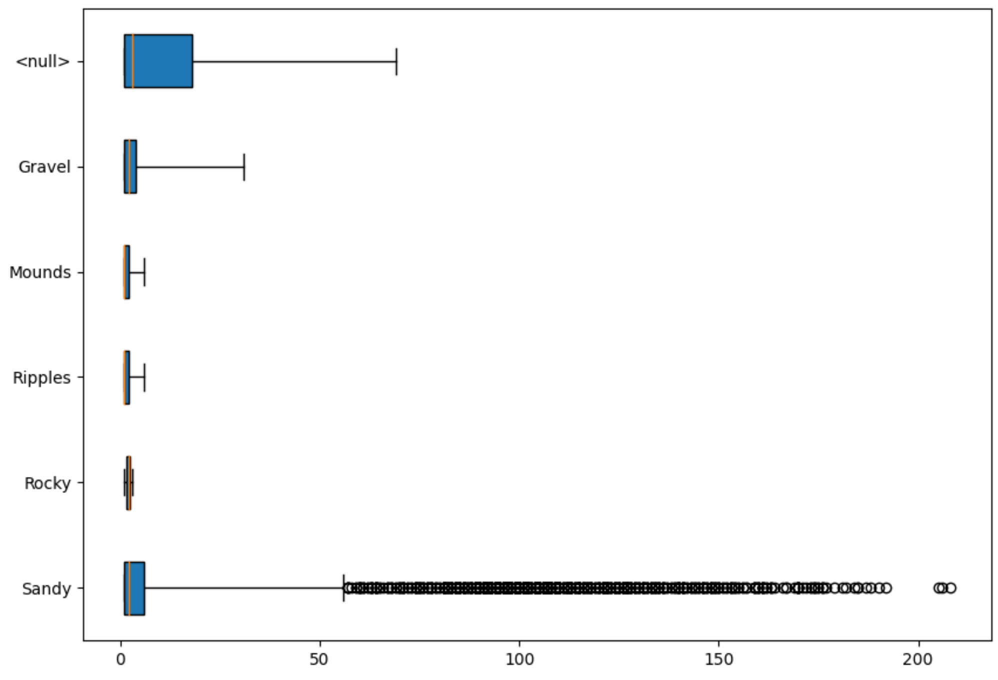
20170826184038/frame002134_1503773407_498313.jpg

Figure 4.6: Example user interface for mission visualization

From the mission visualization with scallop density, we associated it with terrain types for 6 separate campaigns (`20140711_IM`, `20170822_VIMS`, `20170823_VIMS`, `20170824_VIMS`, `20170825_VIMS`, `20170826_VIMS`) as shown in Figure 4.8. Similarly the y-axis of each 2d histogram indicates mission legs of every campaign. The x-axis represents the AUV trackline with the geographical positions of where the image frames were captured. The background color of each rectangle inside suggests the corresponding terrain type: orange-yellow colors indicate sandy substrate and blue colors specify gravelly areas; *Null* class is interpreted in grey color and for those which have not been designated with any terrain labels we describe in color white. The number of green dots inside each rectangle illustrates the total number of scallops within this image where positions of these dots were randomly generated. The grid plots in Figure 4.8 first reveal the substrate composition and the dominant type of the study areas. For example areas campaign `20170823_VIMS` mostly consist of sandy bottom types except the fourth leg showing a gravelly area. And by combining with scallop counts per image, we can directly recognize high-density vs. intermediate-density vs. low-density areas. Such that campaign `20170825_VIMS` covers the most high-density areas compared to others and the last leg in `20170824_VIMS` contains immensely dense scallops. For the analysis we show that sea scallops are most abundant on soft sandy substrate as opposed to coarse rocky areas. And they are likely to aggregate on light gravelly bottoms that are close to sandy areas.

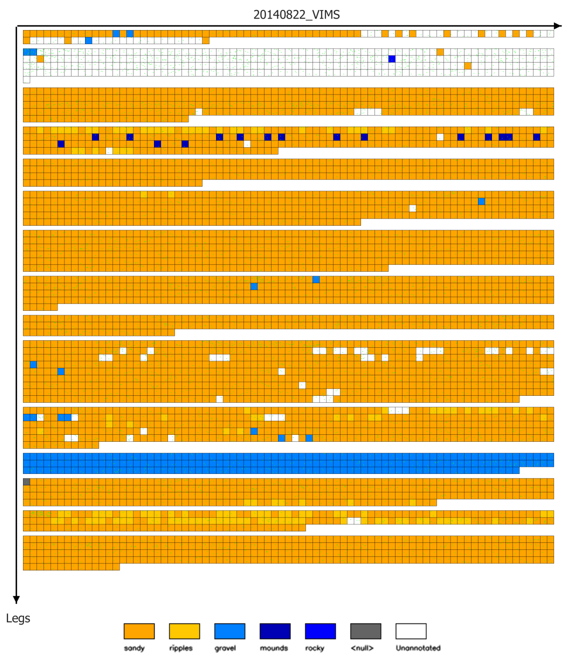


(a)

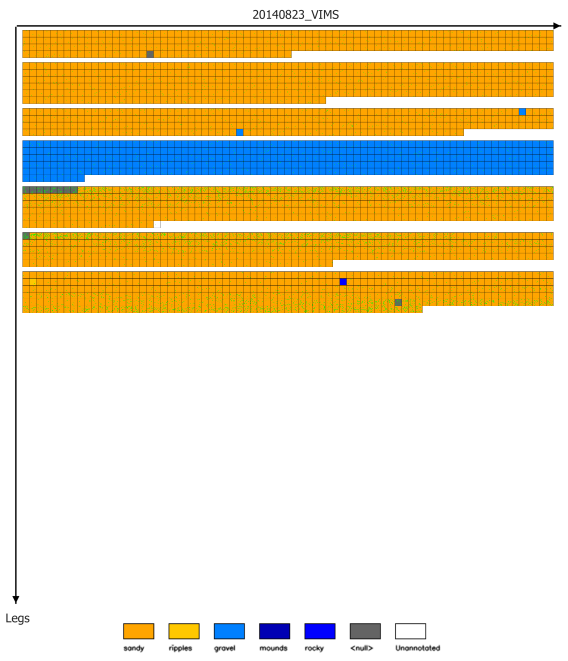


(b)

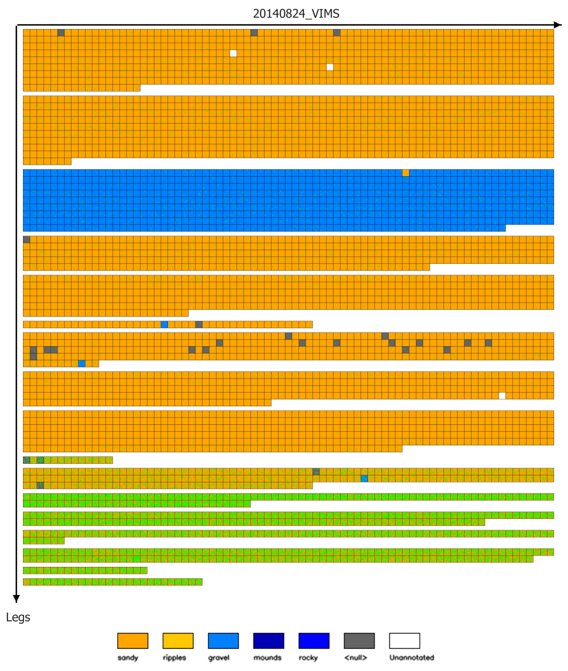
Figure 4.7: Scallop counts analysis on different terrain categories.



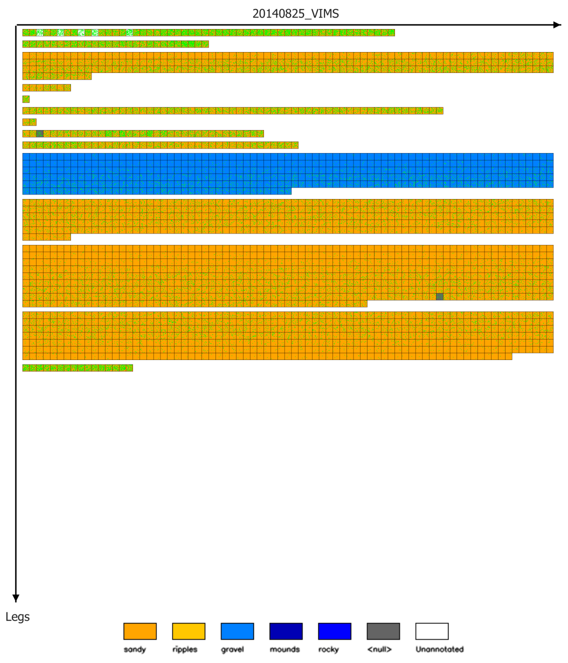
(a) 0170822_VIMS



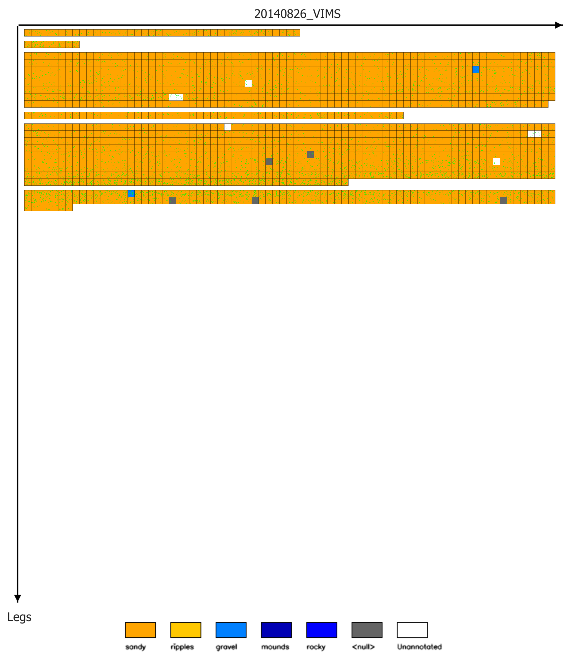
(b) 20170823_VIMS



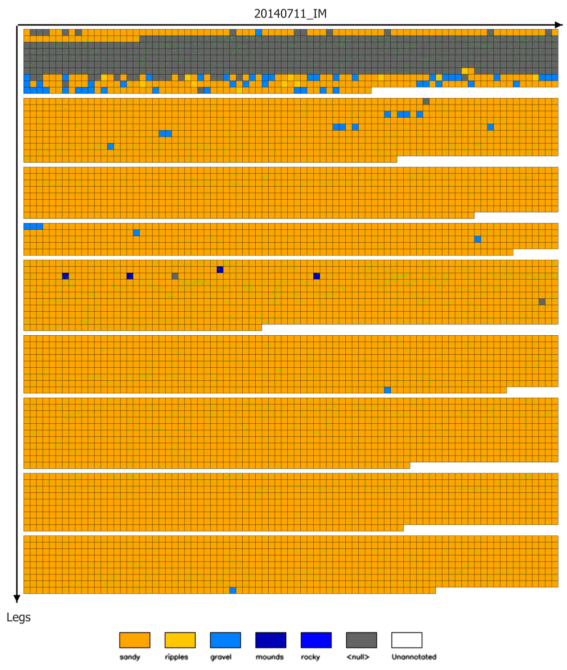
(c) 20170824_VIMS



(d) 20170825_VIMS



(e) 20170826_VIMS



(f) 20140711_IM

Figure 4.8

4.2 Patch Classification

Besides the IMU (inertial measurement unit) information and UTM coordinates that have been collected along with sonar scans and camera images, we need to recover distinguishable objects from the images as well in order to precisely correlate data produced from both types of sensors. Rocks and boulders are substantial objects that are adequate to be considered as the most eligible landmark objects. Therefore our main task of this section is to gather all images that incorporate in which there are dominant rocks. In Section 4.1.1, we discussed that 435 images are labeled as *Rocky* which means that each of these images has at least one large rock or chunks of small rocks and sea shells. However based on our observation and the labeling inconsistency issue discussed in Section 4.1.1, as well as the fact that a large percentage of images in our dredging dataset have not yet been annotated with terrain labels, there are sure of more *rocky* images than the number of known. Going over the entire dataset of 1M+

images to hand-pick the missed *rocky* images is extremely expensive and tedious. For this reason we built patch classifiers to filter out the non-rocky images.

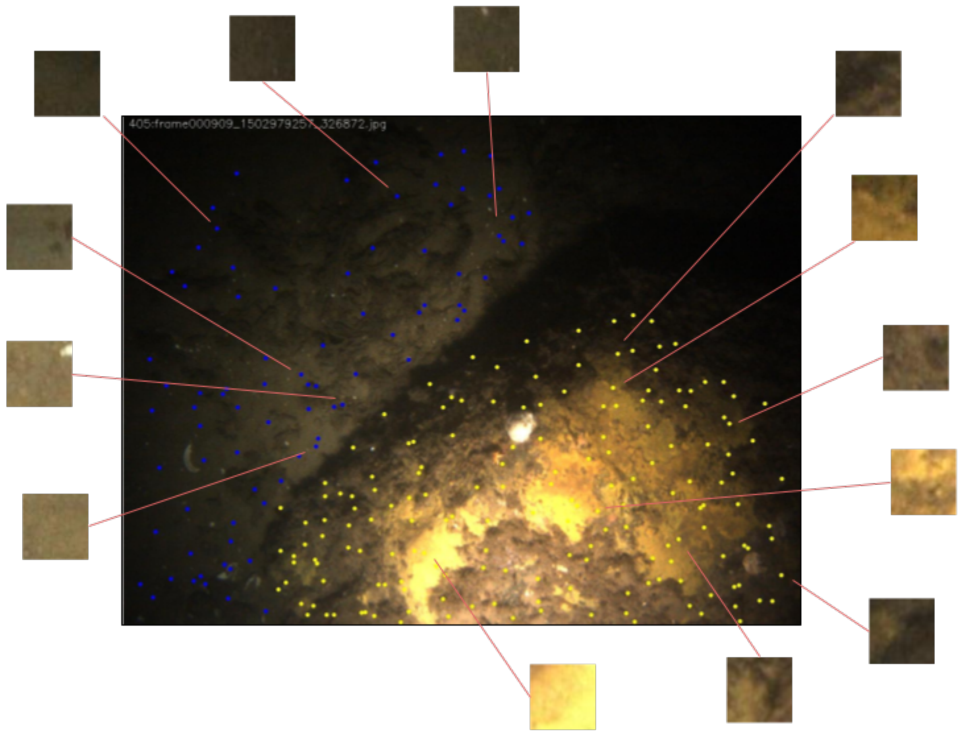


Figure 4.9: Annotation clicks and examples of corresponding patches

We conducted three major experiments on rock patch classification. In the first experiment, we started with annotating patches on the rocky images only. The annotation process was performed as follows: we randomly clicked on rock objects and the seabed. The clicks were interpreted as the center of patches from where we relaxed a couple of pixels to generate 50 x 50 square patches. From 435 rocky images, we generated 90 patches per image on average resulting in a total of 38,920 patches in which 13,514 are rocky and 25,406 are non-rocky. Figure 4.9 gives an example of patch annotations on a rocky image. Yellow dots and the patches on the right side denote the rocky class and blue dots stand for non-rocky patches. We trained the Inception-V3

[4] classification network on this dataset for 10,000 steps with train/test split ratio of 80/20. The test accuracy achieved 0.894 with corresponding precision of 0.793 and recall of 0.823. The confusion matrix can be found in Table 4.3-(a).

	Predicted		
Actual		Rock	Seabed
	Rock	1989	438
	Seabed	522	4096
Accuracy	0.864		

(a)

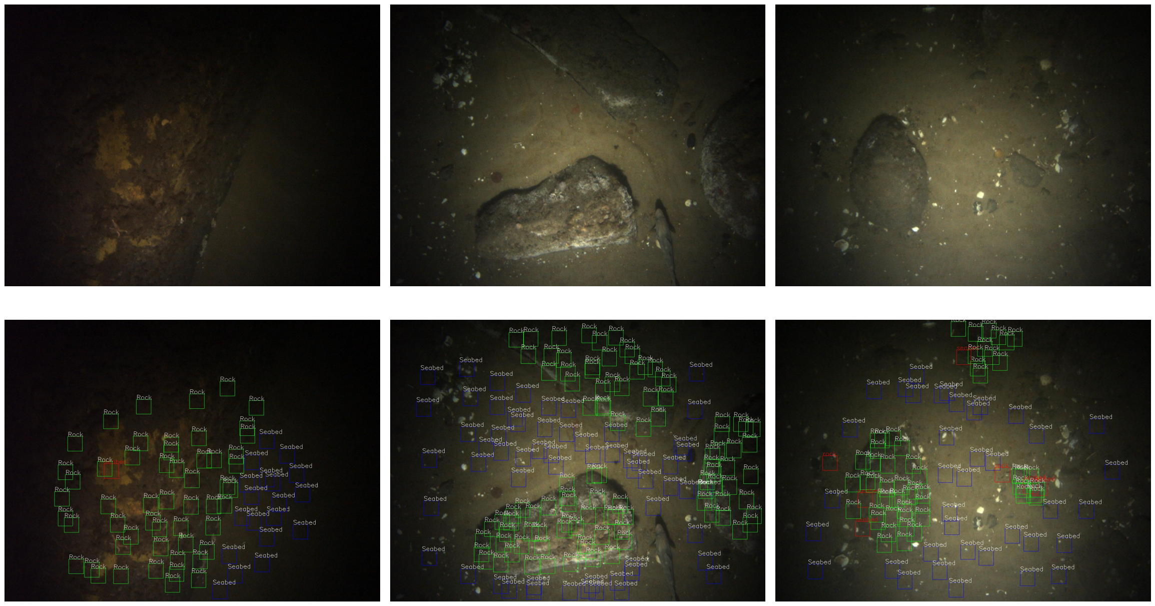
	Predicted		
Actual		Rock	Seabed
	Rock	2529	503
	Seabed	1136	6888
Accuracy	0.855		

(b)

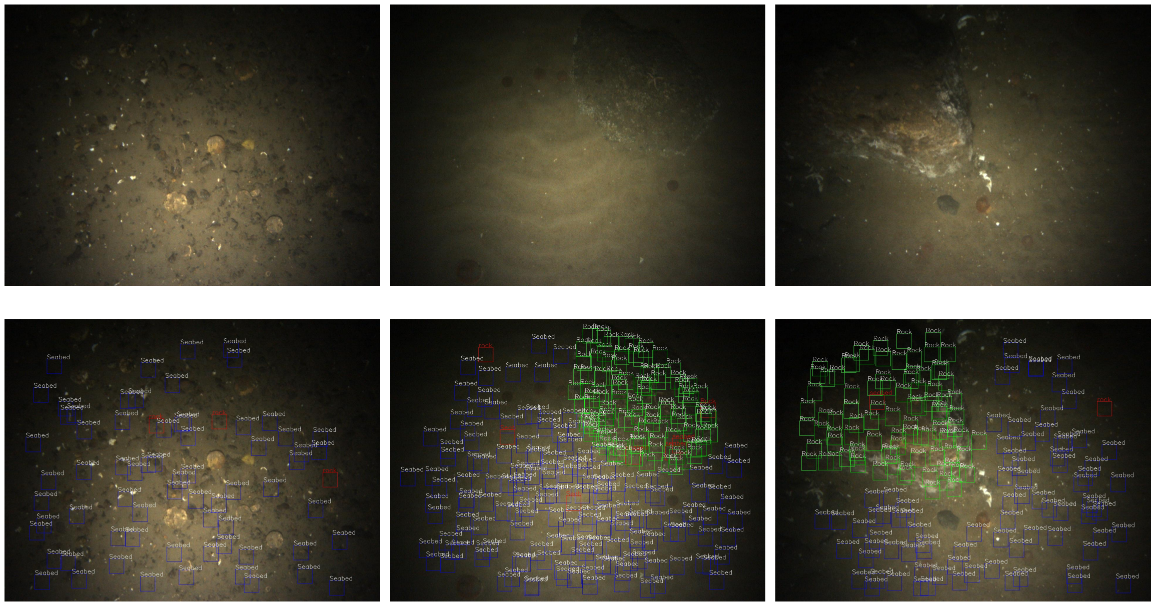
Table 4.3: Confusion matrix

For the second experiment we added images that presents other types of substrate rather than gravelly only in order to add diversity in the features (Figure 4.10c). The dataset (denoted as *combined*) for reference) consisted of 825 images and a total of 62,034 patches generated though the same annotation procedure described in the first experiment. 16,807 rocky patches and 45,277 non-rocky patches were split 80/20 for train/test. The training process was done in 25,000 steps. We evaluated the performance of the classifier on the test set that contained 3,032 rocky patches and 8,024 non-rocky patches. The confusion matrix is summarized in Table 4.3-(b). Precision and recall arrived at 0.690 and 0.834, respectively. Test accuracy reached 0.855.

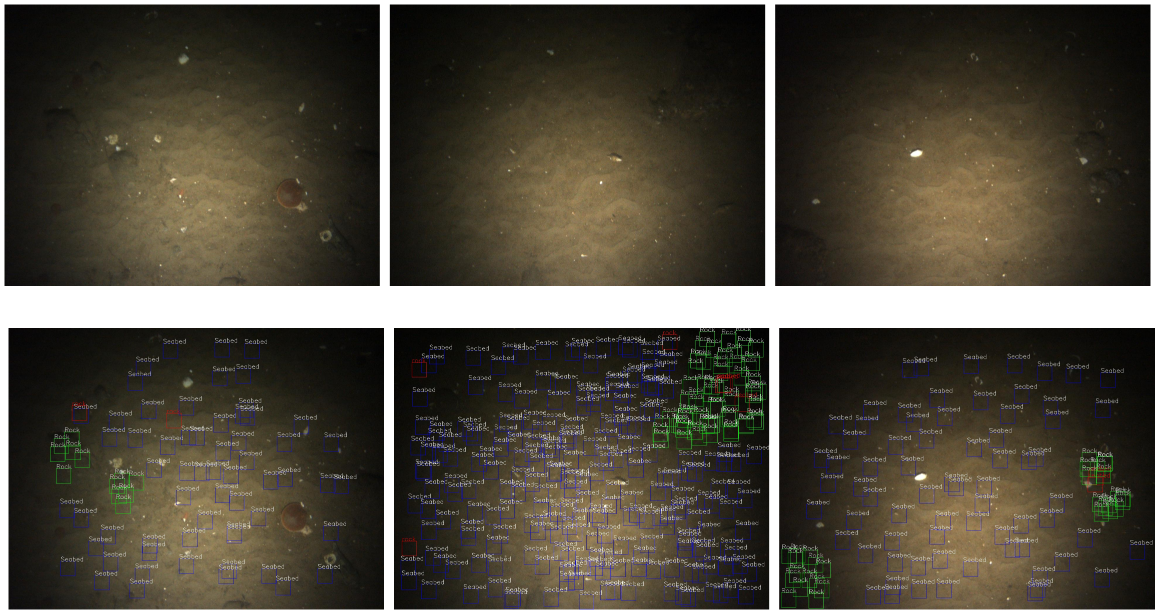
Figure 4.10 shows example test results from our patch classifier. We present image pairs of predictions with their corresponding groundtruth images for a clearer visualization to evaluate the classification performance of the model. Green and blue boxes indicate the correct patch prediction for rocky and non-rocky respectively. Red boxes pointed out the misclassified samples. We can see that the classifier is proficient in segmenting rocks from the seafloor background. It is even capable of recognizing small rocks or gravels laying on the seabed (The third image in Figure 4.10a). We notice that some false negatives often appear near the edges of the rocks where the borders are fairly blend in with the background making it harder to be detected. Furthermore as shown in the examples in Figure 4.10, the images are typically brighter in the center and darker around the corners due to the searching flashlight installed on the AUV. Therefore patches located at the corners are more likely to be misclassified. We combined the grey scale intensities of all images from the test set and extracted the max, median and min values, as shown in Figure 4.11, for the purpose of demonstrating the illumination distribution. For this reason, we proposed to investigate the test accuracy on locations in the image. As the lighting pattern is constant across the dataset, we iterated all patches in the test set and summarized the predictions on a 1280 x 960 blank (Figure 4.12) image that was the same size as the camera images. Green dots are the center of correct prediction patches and red dots means the center of misclassified prediction patches. There are a total of 11,056 predictions of which 9417 are correct (green) and 1639 are misclassified (red). We divided the summary image into 8 x 6 grid and calculated the accuracy (showing in the center) of each grid. From where we can see that the accuracy is generally higher in center grids but drops typically when comes to the corner grids. This summary image further confirms that the lighting variation plays an important role in affecting the performance on patch classification.



(a)

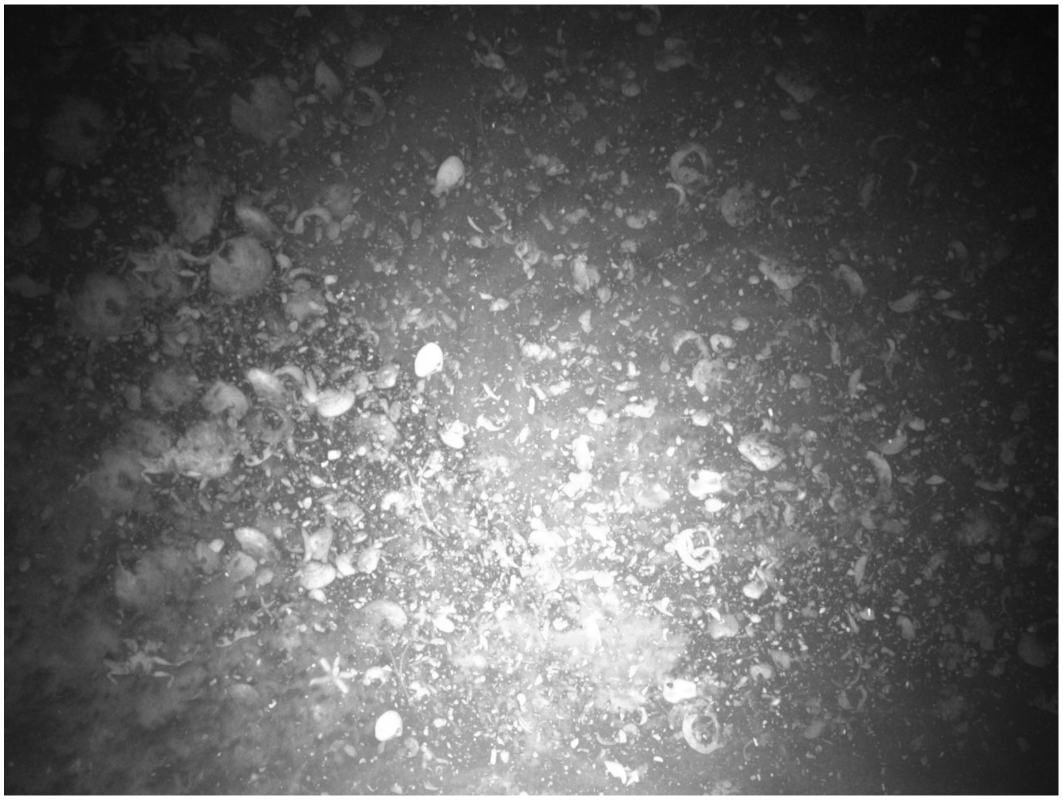


(b)



(c)

Figure 4.10: Example predictions from the patch classifier with corresponding groundtruth images.


(a) Max

(b) Median

(c) Min

Figure 4.11: Summary of max, median and min grey intensity values from images in *combined* test set.

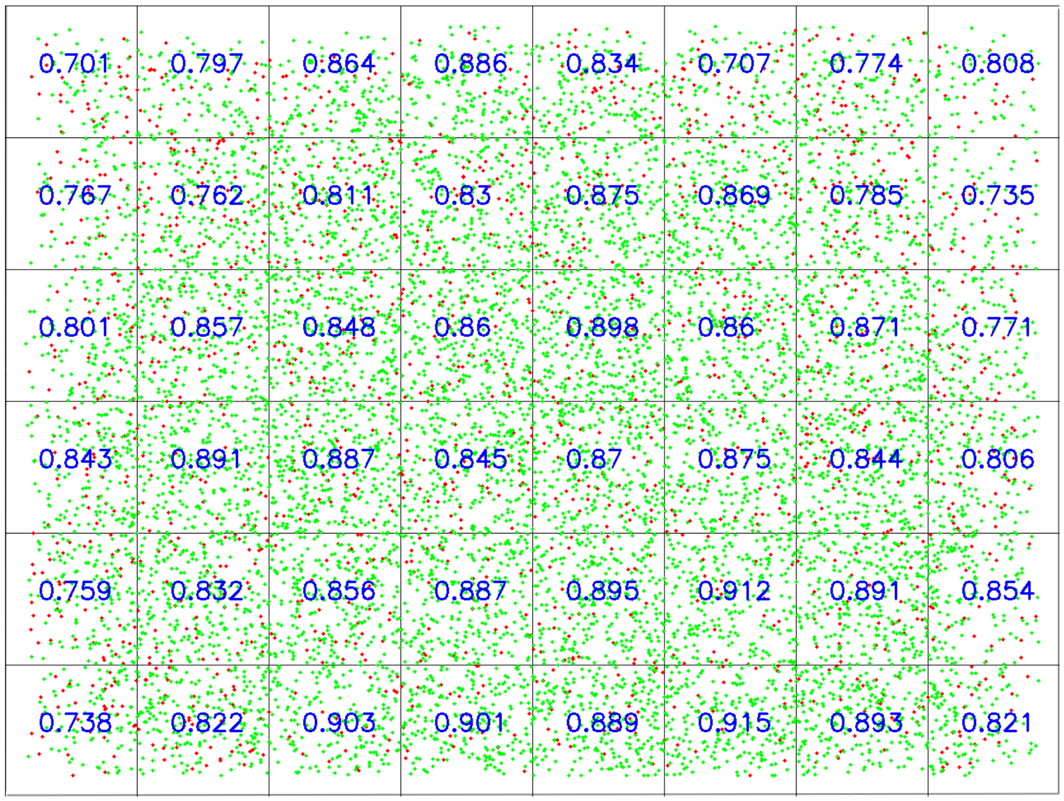
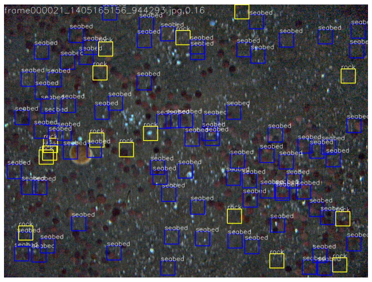
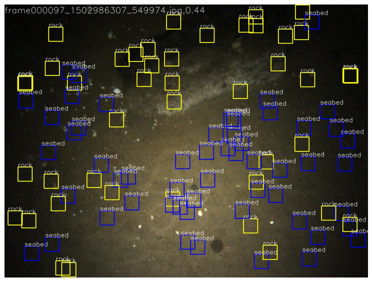
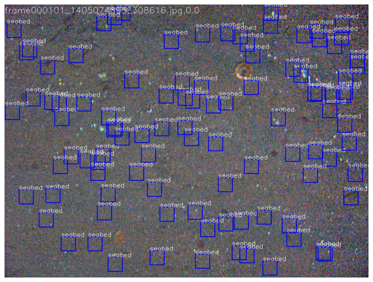


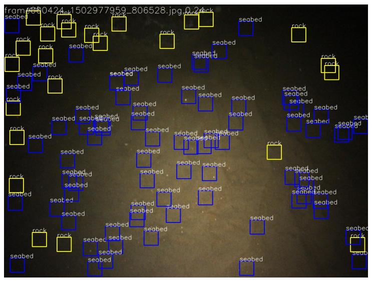
Figure 4.12: Patch classifier: summary of predictions by location and accuracy per grid.

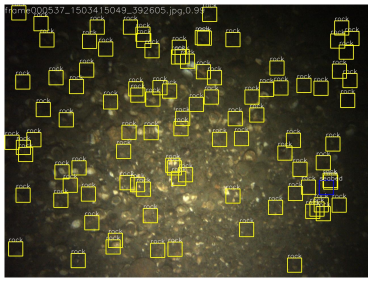
In the third experiment we evaluated our patch classifier on a larger dataset where there was not any terrain label associated with the images. Instead of manually annotating groundtruth we randomly generated 100 patches evenly across the images in the dataset. Meanwhile we introduced a voting mechanism that given a threshold, if the rocky percentage among all patch predictions is greater than the threshold, it implies that there is a large rock or gravelly areas presenting in the image. We can also adjust the threshold based on the kind of images we are looking for to conduct different research studies. The dataset consisted of 152,624 images. We trained the classifier for 55,000 steps (i.e. until the validation loss stopped decreasing).Figure 4.13 shows example predictions. We also explored the distribution of image counts by rocky patch percentages, see Figure 4.14.


(a)


(b)


(c)


(d)


(e)

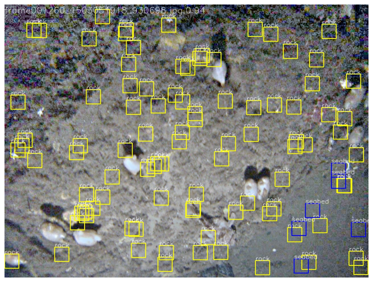

(f)

Figure 4.13: Example patch predictions

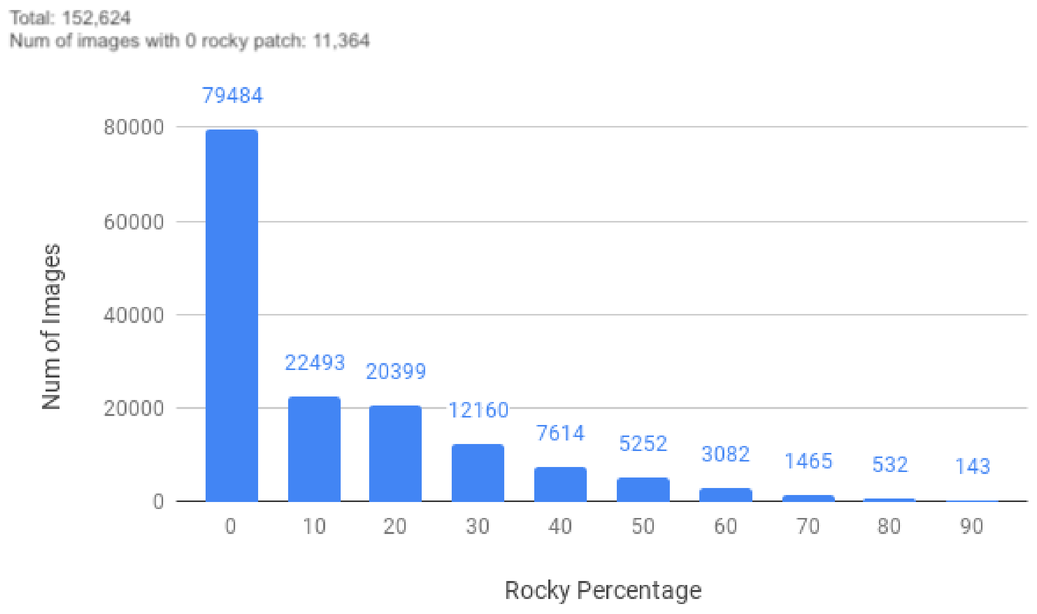


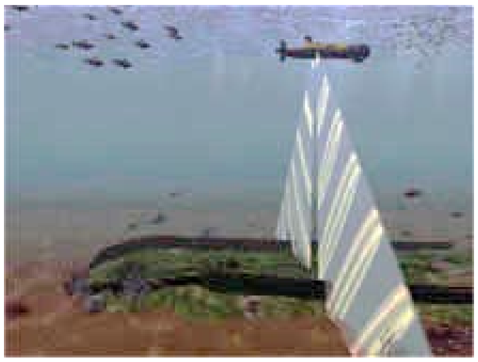
Figure 4.14: Histogram of image distribution by rocky patch percentage

4.3 Side-scan Sonar

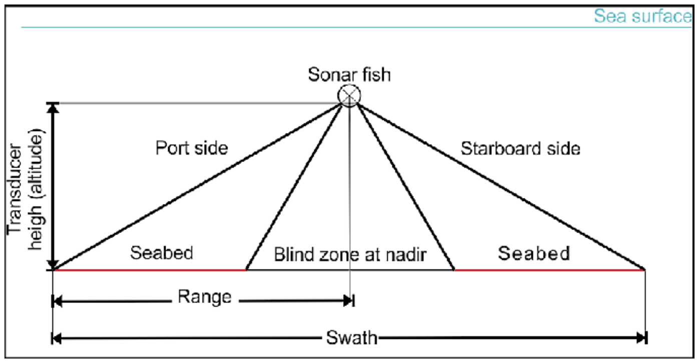
Side-scan sonars are commonly used on boats or AUVs by marine corps and oceanographic researchers for detection in deep water. Sonar uses acoustic waves to detect object in the environment by emitting and receiving reflected sound echo. It out-stands camera-captured data in some respects such that it is capable of capturing a wider range of the environment and sound waves tend to attenuate slower in water. Challenges of interpreting sonar data is that it normally has low-resolution with noise and uneven reflections.

Figure 4.15a shows a schematic diagram of a sonar-mounted AUV scanning an object above the seabed. While in operation, the side scan sonar emits fan-shaped beam of high-frequency acoustic pulse to the seabed at both ends along the tracks. A transverse view of the side-scan sonar system is demonstrated in Figure 4.15b. The side-scan sonar used in our research covers a range of 10 meters on each side and has a frequency of 900kHz. The intensity of the acoustic reflections is recorded in a series of cross-track slices and by attaching the slices together along the direction of motion we can generate an imagery of the seabed within the swath of the beam (Figure 4.16). The black region showing in the center of the imagery represents the nadir gap directly under the path of the vehicle.

The sonar dataset is made up of 774 data recordings collected in August, 2017. We generated sonar waterfall images on each side separately using a Matlab tool resulting in 14,880 images.

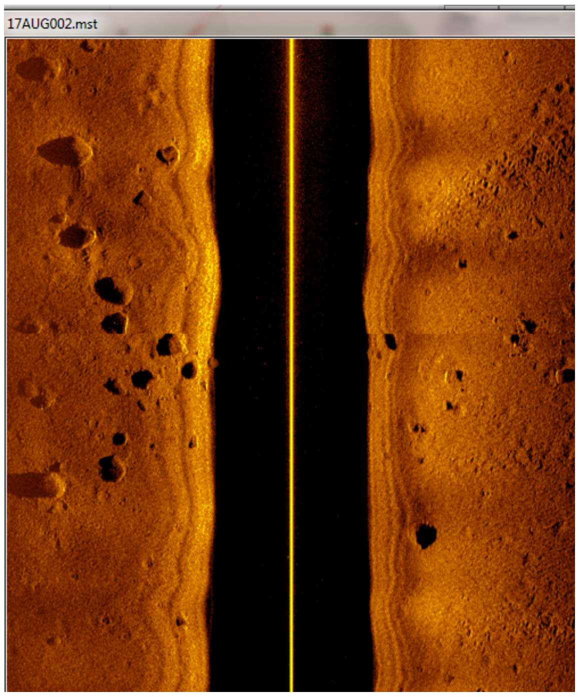


(a)

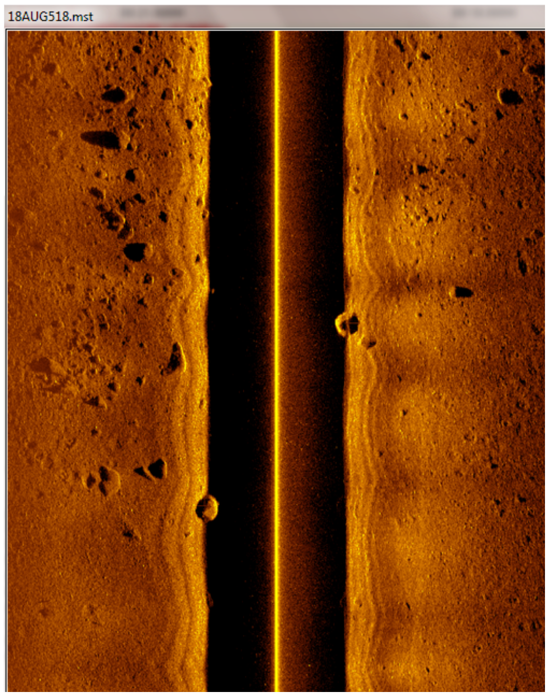


(b)

Figure 4.15: (a) Schematic diagram of AUV sonar scanning an object; (b) Transverse view of a side-scan sonar system [65].



(a)



(b)

Figure 4.16: Examples of seabed scanned by side-scan sonar

4.3.1 Object Recognition on Sonar Imagery

As our sonar data has converted and stored in common image data format, it is intuitive for us to develop a detector for object recognition directly on sonar imagery. Identifying large and distinguishable objects on sonar imagery is also one of the necessary steps to find corresponding optical images.

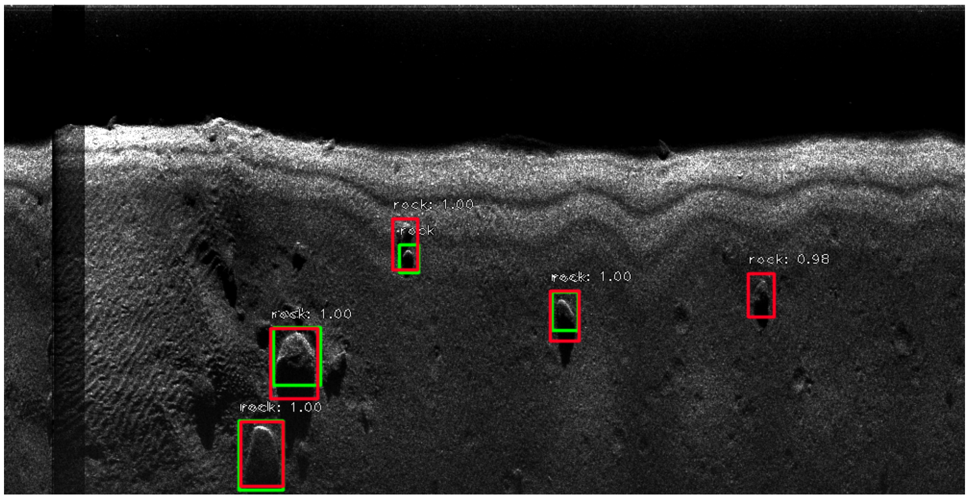
Traditional object recognition methods on sonar images [88, 54, 35] use Template Matching (TM) to measure similarities on the features of shallow and highlight shapes of objects [97]. However TM methods always require to generate templates at first that define the appearance of objects of interest and then find small parts of images that match with the template. [88] proposes to create target signatures from a acoustic model and generate templates using a ray-tracing method. A generalized cross-correlation based function is then used to match regions of an object image to a template. These TM-based methods take strong assumptions about the input images to perform the matching procedure and typically have trouble in generalizing over small objects which in fact are very common to be found on the seabed.

The idea of adopting convolutional neural networks (CNNs) is beneficial in solving tasks that involve optical images and high-dimensional patterns have been proved in the past [67, 63, 106], however there is not extensive research work has been done on CNN-based object recognition for raw sonar imagery. Recent works [135, 154, 146, 145, 90] have explored in this area and proven the potential advantages of using CNNs on sonar imagery for underwater target classification. [90] introduces a sonar image processing pipeline that consists of a data augmenter, feature extractors and filters, followed by a CNN with a target extractor. Input images are cropped before going through data augmentation. Features are extracted separately and then fed into a CNN for a binary classification of whether an image contains an object of interest or not. If there exists an object a target extractor is followed to give the location of the object. [154] shows that using deep learning techniques for feature extraction on sonar images can boost the accuracy of the classifier. Images are first preprocessed and segmented by target objects using different matched filters. Features are extracted with a pre-trained AlexNet [63] and passed to a linear SVM classifier. Their approach outperforms other traditional methods in feature extraction. However common problems of these works are that their datasets are relatively small containing limited number of objects of interest and they only adopted shallow CNN architectures for the detection task. [146, 145] use deeper CNNs of 10 layers for binary classification problems and achieve substantial performance whereas their data are collected by synthetic aperture sonar (SAS) which result in images of higher resolution than the side-scan sonar images in our dataset.

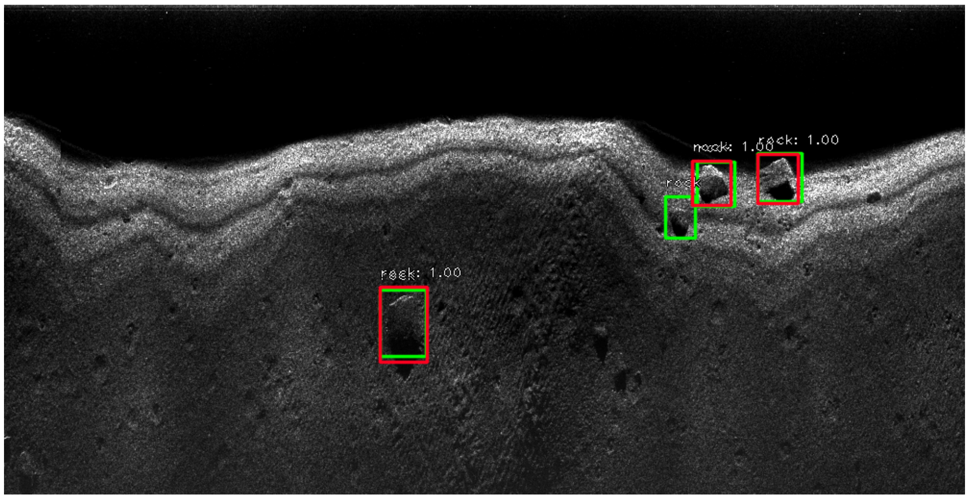
We proposed to use a sophisticated CNN architecture, RetinaNet [46], to train a rock detector. 1,155 of the sonar waterfall images contained at least one rock or boulder, of which there were 2,372 total. All images were converted to grayscale and annotated manually with groundtruth bounding boxes (As the green boxes shown in Figure 4.17). We added 1,000 background images as hard negative examples. The images were split 80/20 for training/testing: 1,725 images with 2,664 objects in the training set and 430 images with 693 objects in the test set. We trained the network with pretrained weights on ImageNet[115] for 50 epochs and 10,000 steps per epoch. Examples of detection results on the test set can be found in Figure 4.17. The green boxes are the groundtruth annotations and the red boxes indicate predictions from the network. Each prediction box has a confidence score written on top to specify the probability that the box contains an object. IoU threshold was set to 0.5 to filter out false positives on the test set. The network reaches a test accuracy of 0.713, with a precision of 0.86 and recall of 0.43

Based on the test results we can see that the network performed well on detecting rocks of different sizes and shapes from low-contrast low-resolution sonar waterfall images. We also notice that some of the false positives could be caused by inaccurate annotations. For example the object on the rightmost in Figure 4.17a and the two objects near the bottom of Figure 4.17c were all detected by the network with quite high confidence scores but overlooked as groundtruth during the annotation process.

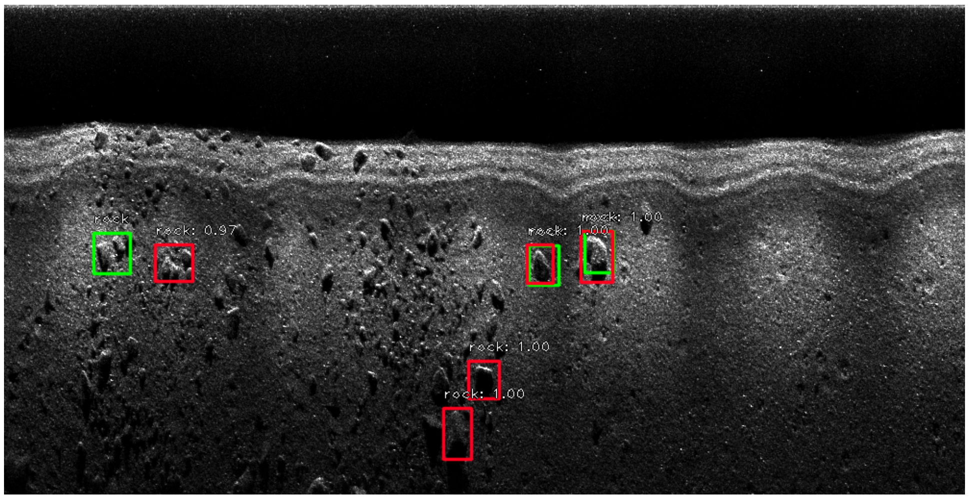
Another case is that it is tricky sometimes for human eyes to tell rocks from bumps or mounds of the seabed, such as the green box on the right-top in Figure 4.17b indicating a false negative. Shatters showing in Figure 4.17c were not annotated individually at this time due to their marginal sizes therefore it was hard for the annotators to distinguish them. However the network appears to be able to find rock pieces from the shatters (object with confidence score of 0.97 in Figure 4.17c). These errors in the groundtruth annotations should be carefully addressed with a strict labeling criteria for a better assessment of the model.



(a)



(b)



(c)

Figure 4.17: Example detection results of RetinaNet on sonar waterfall images.(Green boxes: Groundtruth; Red boxes: Predictions.

4.3.2 Multi-sensor registration

Since the AUV has been collecting data from side-scan sonars and the optical camera simultaneously during each mission, it provides us an opportunity to utilize the output from both sensors for multi-modal analysis. Combining data from different sensors would not only be beneficial for our terrain analysis where sonar scans are able to bring in additional suggestions on substrate composition, but also can improve our object detection performance by including the temporal and depth information.

In this section we explain an application we developed that correlates optical images with sonar waterfall imagery employing the geographical information associated with the images. We utilized the UTM parameters and time-series data from the metadata to extract location coordinates correspondences between neighbouring images. Figure 4.18 illustrates an example of the interface of the application. The current image is specified by index from user input and opened in a separate window along with the geographic map. Images overlapping with the current one are drawn in rectangles at their matching locations. The rest of the images of the current mission are denoted in green dots making up the AUV trackline. Corresponding sonar scan displays at background and rotates the same degree as the AUV. It allows us to navigate to next and previous camera images or sonar scans, jump to images on adjacent legs using keyboard inputs as well as selecting zoom levels. Examples of the 5 zoom levels are shown in Figure 4.19. The lowest zoom level allows us to examine the complete mission trajectory and the highest zoom level reveals the detailed seabed scene. This geographic map application make it more accessible to discover the correspondences between camera images and sonar imagery. In Figure 4.20 we depict the camera images of the seafloor that match with the selected areas in the sonar scans.

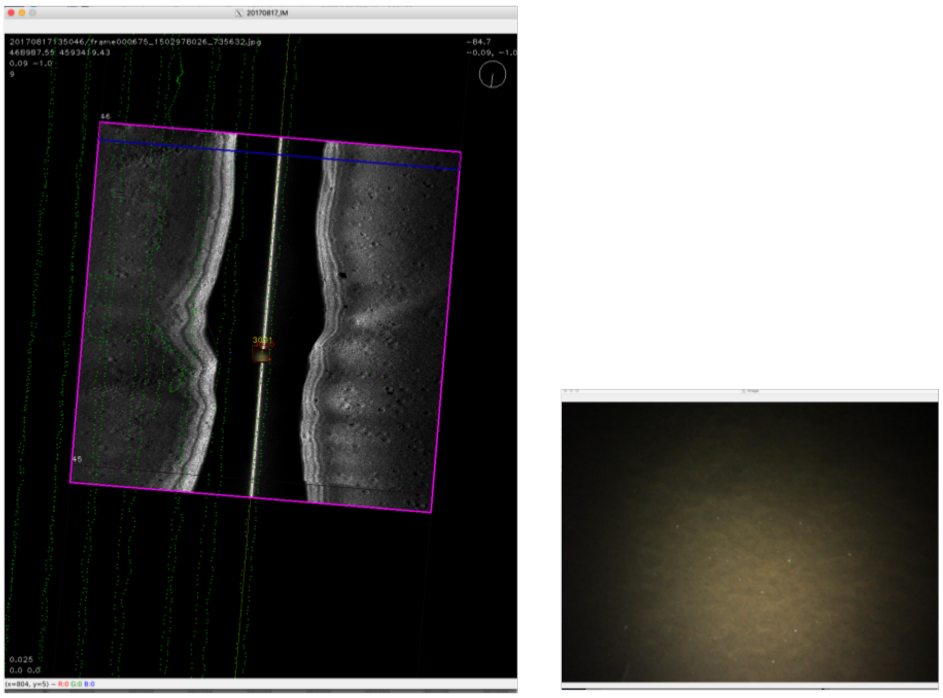


Figure 4.18: Geographic map that correlates optical images and sonar scans.

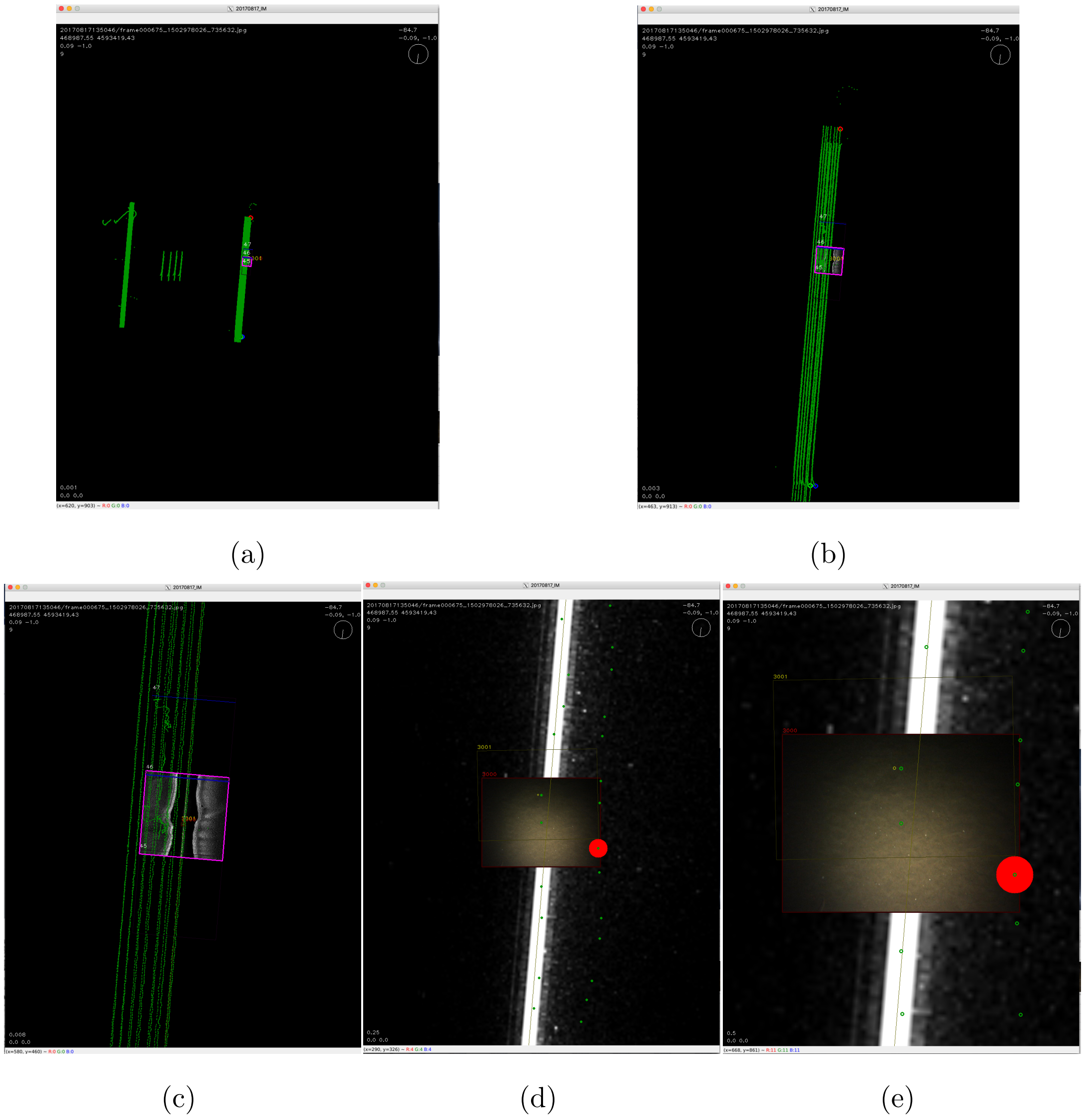
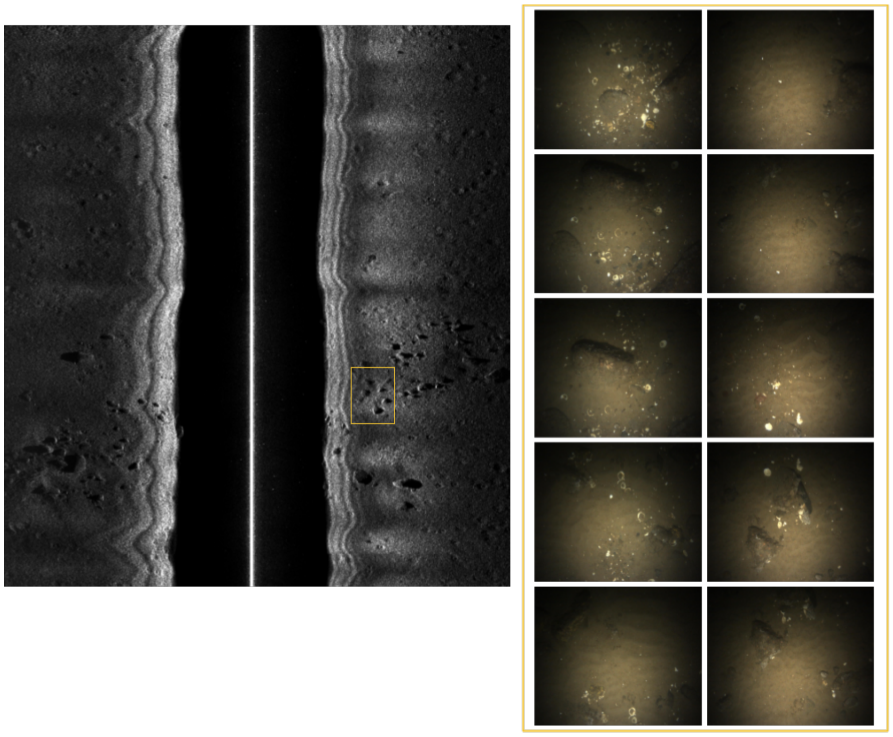
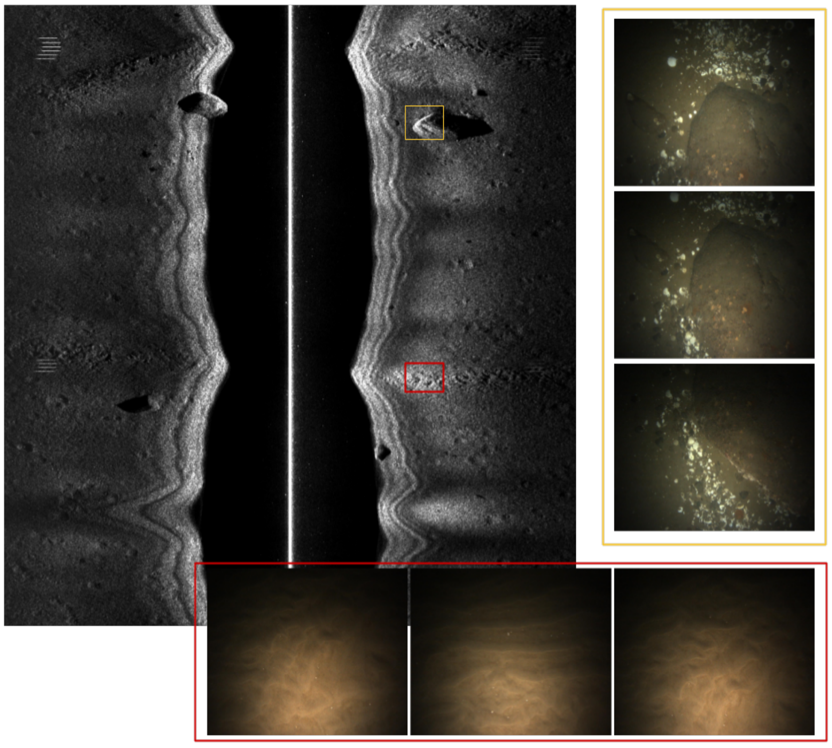


Figure 4.19: 5 different zoom levels of the geographic map



(a)



(b)

Figure 4.20: Example of correspondences between optical images and areas in sonar scans.

Chapter 5
CONCLUSION AND FUTURE WORK

5.1 Conclusion

For this dissertation we focused on developing deep learning based approaches to automate and optimize the process of biological population assessment. As the traditional dredging methods for species population monitoring was expensive and time-consuming, we were motivated to develop CNN-based visual detectors for completing such measurements of one or multiple species with higher efficiency.

In our Image-based Marine Species Population Monitoring study, we started with studying the behavior and performance of different CNN-based visual detectors (including YOLOv2, YOLOv3, RetinaNet) on one-class detection. Results shows that the YOLOv3 detector achieved the best performance on detecting small objects in low-contrast and cluttered scenes. We also evaluated the performance of YOLOv3 on a wide range of scallop density distribution. An auto encoder-decoder neural network was also developed along with the one-class detection experiments to automatically upgrade the original rough positions to groundtruth bounding boxes. We also investigated the influence of batch size on training dynamics in our 2-class mortality rate estimation experiments. We demonstrated that the optimal batch size varies for different model structure and has a great impact on the model performance. Analysis on compromised scallops count can further provide critical statistic information for dredge-included incidental mortality rate estimation. In our 3-class classification, we aimed at learning how the predator-prey cycles affect the populations. We successfully balanced the skewed dataset using data augmentation techniques and achieved competitive results on this task. Our analysis and results from this chapter would benefit future population study of marine animals.

In our Image Registration for Longitudinal Detection study, we first experimented optical flow estimation on motion-based segmentation to identify landmarks and separate moving objects from the background. However the illumination condition on the images in our dataset made it exceptionally hard even for state-of-the-art architectures to catch sufficient movement between image pairs. We then adopted a deep learning based image registration technique generating accurate keypoint matches between successive images and based on matches, we were able to stitch multiple overlapped images into a single unified coordinate system. We further proved that robust registration and proper alignment of consecutive images were necessary for calculating a precise scallop population estimation.

In our Multi-sensor Terrain Analysis study, our terrain classifier demonstrated the relationship between scallop density and substrate types. Our visualization tools provided a clear and straightforward way to recognize the quantified distribution of scallops and the associated terrain types. Our patch classifier was proficient in segmenting rocks from the seafloor background and with the voting mechanism we could easily adjusting the threshold to exclude the images of unwanted terrain types. Instead of traditional templates matching method, our deep learning based detector was able to recognizing objects of various sizes and shapes directly from low-resolution sonar-waterfall images. Our geographic map application can be used effectively as tool for visualizing correlations between images and generating correspondences between optical images and sonar scans.

5.2 Future Work

For our image-based classification project, besides increasing the diversity of our dataset, we noticed that the network commonly confused *Skate* with *Monkfish*. A multi-stage classifier can be developed where the second stage is for learning the features of objects with similar appearance. Fine-tuning and modification in the loss function are also necessary in order to distinguish extremely similar objects. Since we have proved the importance of robust registration between consecutive images, we can further evaluate our technique on the entire dredging dataset to generate the corrected scallop counts for past missions. Additionally the NEXRAD side project we worked on can be improved by converting to high-dimensional data so that more features including spatial and temporal can be captured and fit into a 3D network.

With our multi-sensor registration results, the next step will be improving the performance of object detection utilizing the correlation between data from different source. The sonar is able to measure the distance accurately and observe the environment in a respectively larger scope while optical images give bettor resolution. Therefore a proper fusion approach should be beneficial to provide detailed environment description for object detection. One of the common fusion techniques is to add depth information to tho normal RGB data, producing a 4-channel RGB-D image. From [92] we learned that we can append two subnets at the top of the detection network to extract features from RGB image and depth stream, then concatenate the feature maps at the following fusion layer.

As our ultimate goal is to provide onboard intelligence for the AUV to complete searching tasks in a full- or semi-automatic manner, our future work should be focusing on motion planning so the the AUV would be able to autonomously explore areas and search for target object of interest. Dynamic path planning of unknown environment is one of the critical problems for robotic navigation systems design. It requires a strict driving regulation for the AUV to follow the targets of interest and avoid obstacles. We first propose a modification on the current searching path scheme where the AUV continuously gather information of adjacent legs from the side-scan sonars and use the observations to determine whether skip or keep going on to the next leg. We believe that the sonar detection network described in Section 4.3.1 is capable of providing sufficient information of the unvisited path for monitoring the population of objects of interest. And this alteration on the search path planning intends to reduce mission times and memory usage for image collection.

Deep reinforcement learning (RL) methods proposed in the recent have shown the capability of decision-making problems in multi-dimensional spaces. Hence a vision-based deep RL model that allows the AUV to perform autonomous target-driven exploration in an given mission area would be interesting to probe. The model takes camera-captured RGB images and sonar information as inputs and outputs actions for the next movement. Typical deep RL methods have the target location imbedded in the parameters of the model which requires to retrain the model for each new target. However in our case there are normally more than one interest locations within a mission area results in multiple navigation targets thus retraining the model for every target would be tedious. [156] suggests that both current state and target destination should be included as the input to learn a stochastic policy and generate probability distribution across actions. Involving target locations in the input provides generalization capability to the model for navigation to new targets without retraining. For the navigation model, the downward-facing camera at the front collecting RGB images provides representations for current state and the side-scan sonars observes the rest of the scene. Outputs from the sonar detection network produce information of which the next target is located. As there are common low-level mechanical limitations of real-world robots, we portray the environment as a grid map and analyze command-level actions including moving forward, turning left, and turning right. The turning radius depends on body length of the vehicle and its current speed so we consider two turning angle: 90 degree and 45 degree. As the goal of this agent is to navigate to the target location in shortest path, we design a reward function that gives a large positive reward when the agent reaches the goal destination as well as a small penalty for every step it takes to encourage the agent always seeking for minimum distance from current spot to the destination.pppppppppppppppppppppppp

BIBLIOGRAPHY

[1] An efficient registration method for partially overlapping images. *Procedia Engineering*, 15:2266 – 2270, 2011. CEIS 2011.

[2] Multiscale registration of remote sensing image using robust sift features in steerable-domain. *The Egyptian Journal of Remote Sensing and Space Science*, 14(2):63 – 72, 2011.

[3] A rapid automatic image registration method based on improved sift. *Procedia Environmental Sciences*, 11:85 – 91, 2011. 2011 2nd International Conference on Challenges in Environmental Science and Computer Engineering (CESCE 2011).

[4] Tensorflow hub: A library for transfer learning by reusing parts of tensorflow models., 2018. Available at `https://github.com/tensorflow/hub`.

[5] P. Adarsh, P. Rathi, and M. Kumar. Yolo v3-tiny: Object detection and recognition using one stage improved model. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 687–694, 2020.

[6] J. Aguzzi, C. Costa, K. Robert, M. Matabos, F. Antonucci, S. Juniper, and P. Menesatti. Automated image analysis for the detection of benthic crustaceans and bacterial mat coverage using the venus undersea cabled network. *Sensors*, 2011.

[7] E. Ahmed, S. Cohen, and B. Price. Semantic object selection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3157, 2014.

[8] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 26–33, July 2001.

[9] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the Point: Semantic Segmentation with Point Supervision. *arXiv e-prints*, page arXiv:1506.02106, June 2015.

[10] T. Birgui Sekou, M. Hidane, J. Olivier, and H. Cardot. From Patch to Image Segmentation using Fully Convolutional Networks – Application to Retinal Images. *arXiv e-prints*, page arXiv:1904.03892, Apr. 2019.

[11] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv e-prints*, page arXiv:2004.10934, Apr. 2020.

[12] J.-Y. Bouguet. Pyramidal implementation of the lucas kanade feature trackerdescription of the algorithm. 2000.

[13] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In T. Pajdla and J. Matas, editors, *Computer Vision - ECCV 2004*, pages 25–36, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[14] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011.

[15] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pages 611–625. Springer-Verlag, Oct. 2012.

[16] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[17] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan. BlendMask: Top-Down Meets Bottom-Up for Instance Segmentation. *arXiv e-prints*, page arXiv:2001.00309, Jan. 2020.

[18] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. Change Loy, and D. Lin. Hybrid Task Cascade for Instance Segmentation. *arXiv e-prints*, page arXiv:1901.07518, Jan. 2019.

[19] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Computer Vision – ECCV 2018*, pages 833–851, Cham, 2018.

[20] X. Chen, R. Girshick, K. He, and P. Dollár. TensorMask: A Foundation for Dense Object Segmentation. *arXiv e-prints*, page arXiv:1903.12174, Mar. 2019.

[21] Y. Chen, A. Mimori, and C. Lin. Hybrid particle swarm optimization for 3-d image registration. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1753–1756, 2009.

[22] Y. Chen, Z. Wang, Y. Peng, Z. Zhang, G. Yu, and J. Sun. Cascaded Pyramid Network for Multi-Person Pose Estimation. *arXiv e-prints*, page arXiv:1711.07319, Nov. 2017.

[23] B. Cheng, U. Uiuc, L.-C. Chen, Y. Wei, Y. Zhu, Z. Huang, J. Xiong, T. Huang, W.-M. Hwu, and H. Shi. Spgnet: Semantic prediction guidance for scene parsing. pages 5217–5227, 10 2019.

[24] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. *arXiv e-prints*, page arXiv:1702.04405, Feb. 2017.

[25] J. Dai, K. He, and J. Sun. BoxSup: Exploiting Bounding Boxes to Supervise Convolutional Networks for Semantic Segmentation. *arXiv e-prints*, page arXiv:1503.01640, Mar. 2015.

[26] M. Dawkins, C. Stewart, and A. York. Automatic scallop detection in benthic environments. In *IEEE Workshop on Applications of Computer Vision*, 2013.

[27] D. DeTone, T. Malisiewicz, and A. Rabinovich. Deep Image Homography Estimation. *arXiv e-prints*, page arXiv:1606.03798, June 2016.

[28] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. *arXiv e-prints*, page arXiv:1712.07629, Dec. 2017.

[29] D. DeTone, T. Malisiewicz, and A. Rabinovich. Toward Geometric Deep SLAM. *arXiv e-prints*, page arXiv:1707.07410, July 2017.

[30] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[31] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. *arXiv e-prints*, page arXiv:1905.03561, May 2019.

[32] C. en Lin. Introduction to motion estimation with optical flow, 2019.

[33] K. Enomoto, M. Toda, and Y. Kuwahara. Scallop detection from sand-seabed images for fishery investigation. In *International Congress on Image and Signal Processing*, 2009.

[34] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 2010.

[35] R. Fandos, A. M. Zoubir, and K. Siantidis. Unified design of a feature-based adac system for mine hunting using synthetic aperture sonar. *IEEE Transactions on Geoscience and Remote Sensing*, 52(5):2413–2426, 2013.

[36] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional Two-Stream Network Fusion for Video Action Recognition. *arXiv e-prints*, page arXiv:1604.06573, 2016.

[37] D. Ferraro, A. Trembanis, D. Miller, and D. Rudders. Estimates of sea scallop (placopecten magellanicus) incidental mortality from photographic multiple before-after-control-impact surveys. *Journal of Shellfish Research*, 36:615–626, 12 2017.

[38] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *arXiv e-prints*, page arXiv:1504.06852, Apr. 2015.

[39] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, 1995.

[40] H. Gaiser. Keras retinanet, 2017. Available at https://github.com/fizyr/keras-retinanet.

[41] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[42] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[43] R. B. Girshick. Fast R-CNN. In *IEEE International Conference on Computer Vision*, 2015.

[44] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[45] A. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24:8–12, 2009.

[46] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] J. J. Helmus and S. M. Collis. The python arm radar toolkit (py-art), 2016. Available at https://github.com/ARM-DOE/pyart.

[48] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[49] B. K. Horn and B. G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203, 1981.

[50] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] L. Huang and Z. Li. Feature-based image registration using the shape context. *International Journal of Remote Sensing*, 31:2169 – 2177, 2010.

[52] T.-W. Hui, X. Tang, and C. Change Loy. A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization. *arXiv e-prints*, page arXiv:1903.07414, 2019.

[53] T.-W. Hui, X. Tang, and C. C. Loy. LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8981–8989, 2018.

[54] N. Hurtós, N. Palomeras, S. Nagappa, and J. Salvi. Automatic detection of underwater chain links using a forward-looking sonar. In *OCEANS-Bergen 2013 MTS/IEEE*, 2013.

[55] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks. *arXiv e-prints*, page arXiv:1612.01925, Dec. 2016.

[56] Z. Jiang, L. Zhao, S. Li, and Y. Jia. Real-time object detection method based on improved YOLOv4-tiny. *arXiv e-prints*, page arXiv:2011.04244, Nov. 2020.

[57] B. Jin, M. V. Ortiz Segovia, and S. Süsstrunk. Webly supervised semantic segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1705–1714, 2017.

[58] G. Jocher. Pytorch framework of yolov5, 2020. Available at `https://github.com/ultralytics/yolov5`.

[59] P. Kannappan, J. Walker, A. Trembanis, and H. Tanner. Identifying sea scallops from benthic camera images. *Journal of Limnology and Oceanography*, 2014.

[60] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[61] S. Kim and H. Kim. Zero-centered fixed-point quantization with iterative retraining for deep convolutional neural network-based object detectors. *IEEE Access*, 9:20828–20839, 2021.

[62] S. Korman, D. Reichman, G. Tsur, and S. Avidan. Fast-match: Fast affine template matching. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2331–2338, 2013.

[63] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Conference on Neural Information Processing Systems*, 2012.

[64] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: A large video database for human motion recognition. In *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[65] A. V. Lazar, A. D. Bugheanu, G. Ungureanu, and A. Ionescu. Side-scan sonar mapping of anthropic influenced seafloor: a case study of mangalia harbour. *Geo-Eco-Marina*, 19:59–64, 01 2013.

[66] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006.

[67] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[68] Z. Li, Q. Chen, and V. Koltun. Interactive image segmentation with latent diversity. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 577–585, 2018.

[69] Z. Li and N. Snavely. MegaDepth: Learning Single-View Depth Prediction from Internet Photos. *arXiv e-prints*, page arXiv:1804.00607, Apr. 2018.

[70] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, 2014.

[71] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[72] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[73] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. SSD: Single shot multibox detector. In *European Conference on Computer Vision*, 2016.

[74] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. *arXiv e-prints*, page arXiv:1411.4038, Nov. 2014.

[75] D. Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision*, 1999.

[76] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[77] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–, 11 2004.

[78] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision (ijcai). volume 81, 04 1981.

[79] Z. Luo, T. Shen, L. Zhou, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan. ContextDesc: Local Descriptor Augmentation with Cross-Modality Context. *arXiv e-prints*, page arXiv:1904.04084, Apr. 2019.

[80] Z. Luo, T. Shen, L. Zhou, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan. ContextDesc: Local Descriptor Augmentation with Cross-Modality Context. *arXiv e-prints*, page arXiv:1904.04084, Apr. 2019.

[81] K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[82] N. Mekky and S. Kishk. Wavelet-based image registration techniques: A study of performance. 2011.

[83] E. Memin and P. Perez. Dense estimation and object-based segmentation of the optical flow with robust techniques. *IEEE Transactions on Image Processing*, 7(5):703–719, 1998.

[84] K. Moo Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. *arXiv e-prints*, page arXiv:1603.09114, Mar. 2016.

[85] K. Moo Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to Find Good Correspondences. *arXiv e-prints*, page arXiv:1711.05971, Nov. 2017.

[86] S. M. Moorthi, I. Misra, D. Dhar, and R. Ramakrishnan. Automatic image registration framework for remote sensing data using harris corner detection and random sample consensus (ransac) model. 2012.

[87] F. Moreno-Noguer, V. Lepetit, and P. Fua. Pose priors for simultaneously solving alignment and correspondence. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008*, pages 405–418, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[88] V. Myers and J. Fawcett. A template matching procedure for automatic target recognition in synthetic aperture sonar imagery. *IEEE Signal Processing Letters*, 17(7):683–686, 2010.

[89] S. Nag. Image Registration Techniques: A Survey. *arXiv e-prints*, page arXiv:1712.07540, Nov 2017.

[90] N. Nayak, M. Nara, T. Gambin, Z. Wood, and C. M.Clark. Machine Learning Techniques for AUV Side Scan Sonar Data Feature Extraction as Applied to Intelligent Search for Underwater Archaeological Sites. In *12th Conference on Field and Service Robotics*, 2019.

[91] N. Oceanic and A. A. (NOAA). Nexrad | national centers for environmental information (ncei) formerly known as national climatic data center (NCDC), 2014.

[92] T. Ophoff, K. Van Beeck, and T. Goedemé. Exploring rgb+depth fusion for real-time object detection. *Sensors*, 19:866, 02 2019.

[93] D. Papadopoulos, J. Uijlings, F. Keller, and V. Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017.

[94] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly- and Semi-Supervised Learning of a DCNN for Semantic Image Segmentation. *arXiv e-prints*, page arXiv:1502.02734, Feb. 2015.

[95] E. G. Parmehr, C. S. Fraser, C. Zhang, and J. Leach. Automatic registration of optical imagery with 3d lidar data using statistical similarity. *ISPRS Journal of Photogrammetry and Remote Sensing*, 88:28 – 40, 2014.

[96] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732, 2016.

[97] Y. Petillot, Y. Pailhas, J. Sawas, N. Valeyrie, and J. Bell. Target recognition in synthetic aperture sonar and high resolution side scan sonar using AUVs. *Proceedings of the Institute of Acoustics*, 32, 01 2010.

[98] T. Pfister, J. Charles, and A. Zisserman. Flowing ConvNets for Human Pose Estimation in Videos. In *IEEE International Conference on Computer Vision*, 2015.

[99] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to Segment Object Candidates. *arXiv e-prints*, page arXiv:1506.06204, June 2015.

[100] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollàr. Learning to Refine Object Segments. *arXiv e-prints*, page arXiv:1603.08695, Mar. 2016.

[101] J. Pont-Tuset, F. Perazzi, S. Caelles, P. Arbeláez, A. Sorkine-Hornung, and L. Van Gool. The 2017 DAVIS Challenge on Video Object Segmentation. *arXiv e-prints*, page arXiv:1704.00675, Apr. 2017.

[102] R. Qian, Y. Wei, H. Shi, J. Li, J. Liu, and T. Huang. Weakly Supervised Scene Parsing with Point-based Distance Metric Learning. *arXiv e-prints*, page arXiv:1811.02233, Nov. 2018.

[103] F. Radenović, A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Revisiting Oxford and Paris: Large-Scale Image Retrieval Benchmarking. *arXiv e-prints*, page arXiv:1803.11285, Mar. 2018.

[104] M. Rajchl, M. C. H. Lee, O. Oktay, K. Kamnitsas, J. Passerat-Palmbach, W. Bai, M. Damodaram, M. A. Rutherford, J. V. Hajnal, B. Kainz, and D. Rueckert. DeepCut: Object Segmentation from Bounding Box Annotations using Convolutional Neural Networks. *arXiv e-prints*, page arXiv:1605.07866, May 2016.

[105] A. Ranjan and M. J. Black. Optical Flow Estimation using a Spatial Pyramid Network. *arXiv e-prints*, page arXiv:1611.00850, Nov. 2016.

[106] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[107] J. Redmon. Darknet neural network framework, 2017. Available at https://github.com/pjreddie/darknet.

[108] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once:unified,real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[109] J. Redmon and A. Farhadi. Yolo9000: Better, Faster, Stronger. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[110] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *arXiv e-prints*, 2018.

[111] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN:towards real-time object detection with region proposal networks. In *Conference on Neural Information Processing Systems*, 2015.

[112] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. *arXiv e-prints*, page arXiv:1501.02565, 2015.

[113] G. Rosenkranz, S. Gallager, R. Shepard, and M. Blakeslee. Development of a high-speed, megapixel benthic imaging system for coastal fisheries research in alaska. *Fisheries Research*, 92:340–344, 08 2008.

[114] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23:309–314, 08 2004.

[115] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. In *International Journal of Computer Vision(IJCV)*, 2015.

[116] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning Feature Matching with Graph Neural Networks. 2020.

[117] J. Sarvaiya. Automatic image registration using mexican hat wavelet , invariant moment , and radon transform. *International Journal of Advanced Computer Science and Applications - IJACSA*, Special:75–84, 05 2011.

[118] T. Sattler, B. Leibe, and L. Kobbelt. Scramsac: Improving ransac's efficiency with a spatial consistency filter. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2090–2097, 2009.

[119] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3), 1999.

[120] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. In *International Journal of Computer Vision*, 2002.

[121] L. Schwab, M. Schmitt, and R. Wanka. Multimodal medical image registration using particle swarm optimization with influence of the data's initial orientation. In *2015 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, pages 1–8, 2015.

[122] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations*, 2014.

[123] A. Shrivastava, R. Sukthankar, J. Malik, and A. Gupta. Beyond Skip Connections: Top-Down Modulation for Object Detection. *arXiv e-prints*, page arXiv:1612.06851, 2016.

[124] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *Advances in Neural Information Processing Systems*, 1, 06 2014.

[125] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In *International Conference on Learning Representations*, 2015.

[126] A. Stewart-Oaten, W. Murdoch, and K. Parker. Environmental impact assessment: "pseudoreplication" in time? *Ecology*, 67(4):929–940, 1986.

[127] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *IEEE International Conference on Computer Vision*, 2017.

[128] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference Computer Vision and Pattern Recognition (CVPR)*, 2015.

[129] M. Tang, A. Djelouah, F. Perazzi, Y. Boykov, and C. Schroers. Normalized Cut Loss for Weakly-supervised CNN Segmentation. *arXiv e-prints*, page arXiv:1804.01346, Apr. 2018.

[130] R. Taylor, N. Vine, A. York, S. Lerner, D. Hart, J. Howland, L. Prasad, L. Mayer, and S. Gallager. Evolution of a benthic imaging system from a towed camera to an automated habitat characterization system. In *OCEANS Conference*, 2008.

[131] J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation. In *Conference on Neural Information Processing Systems*, 2014.

[132] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[133] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[134] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *arXiv e-prints*, page arXiv:1607.08022, July 2016.

[135] M. Valdenegro-Toro. Improving Sonar Image Patch Matching via Deep Learning. In *European Conference on Mobile Robots (ECMR)*, 2017.

[136] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention Is All You Need. *arXiv e-prints*, page arXiv:1706.03762, June 2017.

[137] Vinividyadharan and Subusurendran. Automatic image registration using sift-ncc. 2012.

[138] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[139] P. Viola and W. M. Wells. Alignment by maximization of mutual information. In *Proceedings of IEEE International Conference on Computer Vision*, pages 16–23, 1995.

[140] M. P. Wachowiak, R. Smolikova, Yufeng Zheng, J. M. Zurada, and A. S. Elmaghraby. An approach to multimodal biomedical image registration utilizing particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):289–301, 2004.

[141] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. *arXiv e-prints*, page arXiv:1608.00859, 2016.

[142] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li. SOLO: Segmenting Objects by Locations. *arXiv e-prints*, page arXiv:1912.04488, Dec. 2019.

[143] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. pages 1385–1392, 12 2013.

[144] Wikipedia contributors. Nexrad — Wikipedia, the free encyclopedia, 2019. [Online; accessed 22-December-2019].

[145] D. P. Williams. Underwater target classification in synthetic aperture sonar imagery using deep convolutional neural networks. In *International Conference on Pattern Recognition (ICPR)*, 2016.

[146] D. P. Williams. Demystifying deep convolutional neural networks for sonar image classification. In *Underwater Acoustics Conference*, 2017.

[147] J. Wu, Y. Zhao, J.-Y. Zhu, S. Luo, and Z. Tu. Milcut: A sweeping line multiple instance learning paradigm for interactive image segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 256–263, 2014.

[148] Xiaolei Huang, Yiyong Sun, D. Metaxas, F. Sauer, and Chenyang Xu. Hybrid image registration based on configural matching of scale-invariant salient region features. In *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 167–167, 2004.

[149] N. Xu, B. Price, S. Cohen, J. Yang, and T. Huang. Deep GrabCut for Object Selection. *arXiv e-prints*, page arXiv:1707.00243, July 2017.

[150] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville. Describing videos by exploiting temporal structure. In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[151] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. In *European Conference on Computer Vision (ECCV)*, 2014.

[152] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao. Learning Two-View Correspondences and Geometry Using Order-Aware Network. *arXiv e-prints*, page arXiv:1908.04964, Aug. 2019.

[153] S. Zhang, J. H. Liew, Y. Wei, S. Wei, and Y. Zhao. Interactive object segmentation with inside-outside guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12234–12244, 2020.

[154] P. Zhu, J. Isaacs, B. Fu, and S. Ferrari. Deep learning feature extraction for target recognition and classification in underwater sonar images. In *IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017.

[155] Y. Zhu, Z. Lan, S. Newsam, and A. G. Hauptmann. Hidden Two-Stream Convolutional Networks for Action Recognition. *arXiv e-prints*, page arXiv:1704.00389, 2017.

[156] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *arXiv e-prints*, page arXiv:1609.05143, Sep 2016.

[157] C. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, pages 391–405. Springer, 2014.