Towards Robust Visual Diver Detection Onboard Autonomous Underwater Robots: Assessing the Effects of Models and Data*

Karin de Langis[1], Michael Fulton[2] and Junaed Sattar[3]

***Abstract*— Deep neural networks are the leading solution to the object detection problem. However, challenges arise when applying these networks to the kind of real-time, first-person video data that a robotic platform must process: specifically, detections may not be consistent from frame to frame, and objects may frequently appear at viewpoints that are particularly challenging for the model, resulting in inaccurate detections. In this paper, we present our approach for addressing these challenges for our particular vision problem: diver detection onboard autonomous underwater vehicles (AUVs). We begin by producing and releasing a dataset of approximately 105,000 annotated images of divers sourced from videos in order to address the challenge of learning a wide variety of object rotations and translations. This is one of the largest and most varied diver detection datasets ever created, and we compare models trained and tested on both our dataset and a previous dataset to demonstrate that our dataset improves the state-of-the-art in diver detection. Then, in order to choose an object detection model that produces detections that are consistent from frame to frame, we evaluate several state-of-the-art object detection models on the temporal stability of their detections in addition to the typical accuracy and efficiency metrics, mean average precision (mAP) and frames per second. Importantly, our results showed that models with the highest mAP do not also have the highest temporal stability.**

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) are invaluable tools with great potential in advancing underwater science and engineering. AUVs serve oceanographers and marine biologists by charting biological habitats [1] and marine geology [2], observing climate change underwater [3], inspecting and repairing undersea infrastructures such as pipelines or cables [4], and helping to control invasive species [5]. Many of these applications require AUVs to work alongside and aid human workers underwater by carrying loads, recording data, or mapping environments while the human does other critical work. A key capability of an AUV intended to work alongside humans is diver detection: an AUV must have an understanding of where humans are in order to safely move in the environment, communicate with its operator, or follow a human to a location. Diver detection is typically achieved through deep neural networks for object detection.

Deep neural networks [6], particularly single-frame convolutional neural networks (CNNs), have achieved great success in many object detection applications in the underwater space, including diver detection [7], coral identification [8], fish species identification, and trash detection [9]. However, not enough has been done to address the diver detection problem in terms of the practical considerations of deployment on robots. In a robotic deployment, these detectors will be operating in the context of a video stream, with consequences to missed detections (*e.g.*, becoming separated from the diver, swimming in the wrong direction, etc).

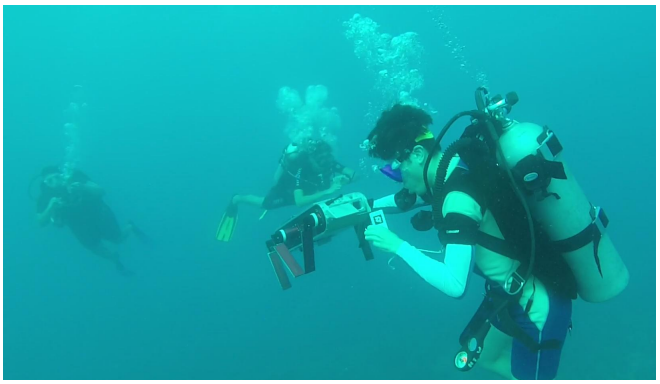


Fig. 1: A diver operating an Aqua AUV in Barbados.

CNNs, despite their strong performances on benchmark vision datasets, have been shown to struggle at times with images from video streams [10], [11], [12], [13]. This is particularly problematic for robotic applications, where vision mistakes have physical consequences [14]. There are several factors that explain this phenomenon, which are discussed in more detail in Section II. In short, videos present objects at a wide variety of translations and rotations, and CNNs can struggle to generalize to the full range of translations and orientations found in video data [15], [16].

Most previously presented diver detection systems achieve relatively high accuracies in the single image sense, but in our work, we have observed that these metrics can hide a tendency for instability in detection bounding boxes when viewed in a temporal context: the diver may be detected in one frame and undetected in the next or the diver's bounding box may have an inconsistent position or scale (relative to the ground truth) from frame to frame, making it difficult to build an accurate understanding of the diver's location. The simplest applications of diver detection, diver following, can often make do with low-stability detections. Other human-robot collaboration capabilities "down the pipeline," however, such as gesture detection, attention and intention detection, and action recognition may not be able to cope with bounding boxes which frequently change in scale and position, or simply fail to reliably detect a given diver. In

The authors are with the Department of Computer Science and Engineering and the Minnesota Robotics Institute, University of Minnesota Twin Cities, Minneapolis, MN, USA. {[1]dento019, [2]fulto081, [3]junaed}@umn.edu

*This work was supported by the US National Science Foundation Awards IIS-#1845364 & #00074041 and the MnRI Seed Grant.

this paper, we address these challenges by (1) creating a large dataset of divers sourced from videos in order to learn more diver poses and rotations, and (2) analyzing detector performance not just in terms of traditional accuracy and efficiency metrics, but also in terms of stability metrics.

Specifically, we create and release the Video Diver Detection dataset (VDD-$\bar{C}$[1]), a large dataset comprised of approximately $105,000$ fully annotated images of divers, drawn from videos taken in pool and field environments. (The dataset can be found at http://irvlab.cs.umn.edu/vddc.) This dataset improves on previous datasets in size (approximately 17 times more images than our previous work [7]), but also by providing images in video context, allowing for analysis of the temporal stability of single-image detectors, and also allowing models to learn a wider variety of diver translations and rotations. We then train a variety of single-image object detection networks on this dataset, as well as one video object detection network, and evaluate them all. In order to build a full picture of their capabilities, we evaluate each network not only in terms of traditional accuracy metrics such as precision and recall, but also on a number of video stability metrics and in terms of their inference speeds on embedded platforms.

Contributions In this paper we present our approach to addressing shortcomings in our diver detection models:

- We create, process, and release a $105,000$ image dataset of fully annotated videos of divers.
- We demonstrate that this dataset improves the state-of-the-art of diver detection over our previous dataset.
- We analyze several state-of-the-art deep models in terms of not only their accuracy and efficiency, but also in terms of their stability over sequential frames. In particular, we demonstrate that detectors with high accuracy do not necessarily have high temporal stability.

II. RELATED WORK

Object detection is a computer vision task that involves identifying and localizing objects. Convolutional neural networks (CNNs) are by far the highest performing object detection models [17], [18] and can generally be divided into two groups: two stage region-based detectors, which propose object regions in stage one and extract features from these regions in stage two (*e.g.*, Region CNN [19] and its descendants Fast R-CNN [20], Faster R-CNN [21], Region FCN [22], Mask R-CNN [23]), and one stage grid-based detectors, which skip the region proposal step and instead extract features over a dense, static grid of possible object locations in the image (*e.g.*, SSD [24], YOLO [25]). One stage detectors are less accurate but fast enough for realtime inference, while two stage detectors are more accurate but often unsuitable for realtime deployment.

State-of-the-art CNNs perform impressively well on vision benchmarks. However, these CNNs can stumble on images that come from a video stream [10], [11], [12], [13], which is particularly problematic for robotic applications [14]. Research investigating this phenomenon points to multiple reasons behind this performance deficit. One issue is that as objects move in a video, they appear at a variety of locations. Image translations as small as one pixel can result in a radically different image representation at the deepest layers of state-of-the-art CNNs [10], which means that CNNs can struggle to generalize to the wide range of translations seen in video data. In fact, even small translations of input images can be effective adversarial attacks on CNNs [10], [15], [26]. It is also important to note that CNNs are often trained on datasets like ImageNet [27] that have demonstrable location bias: the photographed objects' locations are not equally distributed throughout the dataset, and traditional data augmentation strategies do not sufficiently address this problem [10], [15].

Similarly, objects in videos appear in a variety of orientations, which is also challenging for CNNs. Datasets typically present relatively head-on views of objects, whereas videos typically capture objects from a wide range of vantage points [16]. There is significant evidence that state-of-the art object detectors generalize very poorly to certain rotations [15], [16].

Learning to detect objects well in video is motivated by many applications, from robotics to surveillance. Since the release of ImageNet VID [28] in 2015, researchers have developed many models for video object detection. These detectors can learn to exploit temporal information in video streams in order to make better detections on video data, and they typically outperform static image detectors on video datasets [29]. Video detectors utilize a variety of strategies for leveraging temporal information, most notably linking static detections across frames in tracklets/tubelets [30], optical flow [31], and spatio-temporal feature memory [32], [33], [34]. However, many video detectors cannot perform inference in real time, making them unsuitable for robotic applications. Most realtime-capable video detectors (*e.g.*, [29], [35], [33]) achieve faster speeds by focusing intensive computational efforts on periodic "key frames" and propagating some of these computed features to subsequent frames, rather than computing features for every single frame.

Evaluation metrics are under-studied in video object detection. Video object detectors are typically evaluated with the same metrics used for static images detectors, *e.g.*, mean average precision (mAP). Notably, these metrics do not take into account the temporal nature of video data. Recently some metrics [36], [37] have been proposed that evaluate video detectors not only on mAP, but also on the stability of bounding box location and scale for a given object across frames (*i.e.*, how much the bounding box jitters around the ground truth) and on how fragmented detections are for each object in the video (*i.e.*, during the duration of an object's presence in the video, how many times does an object's status change from "detected" to "undetected"), although these metrics have not yet been widely adapted.

Diver detection is of significant interest to researchers in marine robotics, and previous work has collected various diver datasets for the purpose of diver detection. The

[1] $\bar{C}$ is the Roman numeral for 100,000, denoting the number of images.

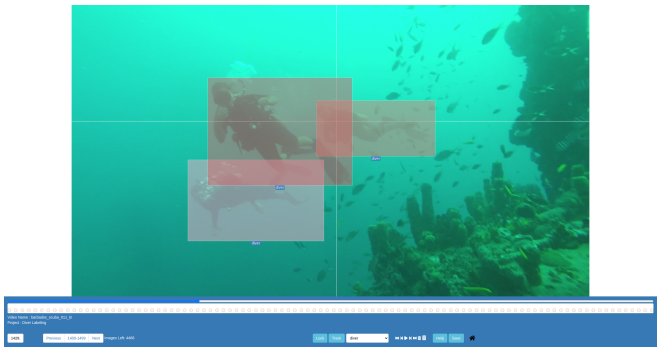


Fig. 2: The EVA labeling tool.

(a) Unlabeled (b) Labeled

Fig. 3: An image from VDD-$\bar{C}$, with and without labels.

Cognitive Autonomous Diving Buddy (CADDY) project is a broad collection of data for underwater vision research that includes a diver pose estimation dataset and a diver gesture recognition dataset [38], [39]. [40] develop a diver detection algorithm that uses nearest-class-mean random forests.

III. DATASET CREATION AND PROCESSING

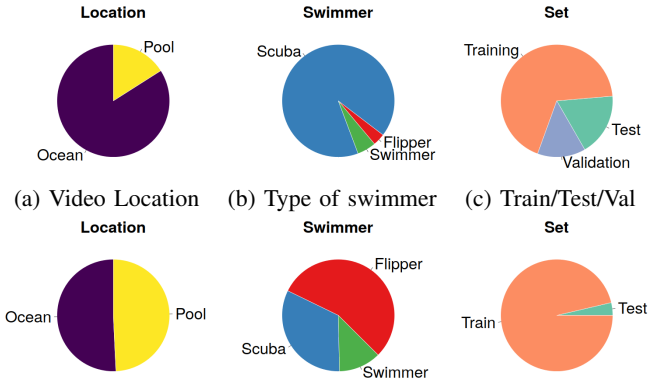
Our previous work produced a small dataset of images of divers in various environment [7]. This dataset will be referred to in this paper as the Deep Diver Dataset (DDD). Our motivation to create a new dataset that improved upon the existing DDD dataset stems from the following reasons:

(i) DDD is a relatively small dataset from a deep learning perspective, with $6,011$ images in its training set.
(ii) While many of the images are from videos, the organization of the dataset does not lend itself to temporal stability testing or training video detection methods.
(iii) The majority of the training images are biased to the application of diver following: a single diver swimming away from the camera is a common image.

To address these shortcomings, we present a new dataset, the Video Diver Detection dataset (VDD-$\bar{C}$), available at http://irvlab.cs.umn.edu/vddc.

A. Source Data

With the goal of temporal stability testing and video detection in mind, we chose to create our dataset out of videos, extracted into images at a rate of 20 frames per second for annotation. The majority of the videos were from dives off the coast of Barbados in the Caribbean Sea, but a sizeable number of videos were taken in pool environments. The percentage of the dataset containing images from different environments (ocean/pool), images featuring different diver gear types (scuba/flippers/no gear), and images allocated to training, test, or validation sets is visualized in Figure 4 for VDD-$\bar{C}$ (4a - 4c) and DDD (4d - 4f). Note that



(a) Video Location (b) Type of swimmer (c) Train/Test/Val

(d) Video Location (e) Type of swimmer (f) Train/Test/Val

Fig. 4: Distribution of VDD-$\bar{C}$ *(a-c)* and DDD *(d-f)* data.

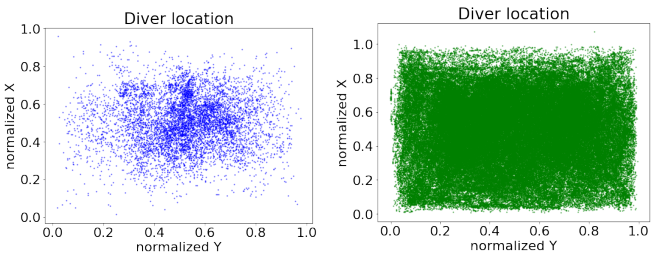
the the Figure shows percentages rather than number of frames. For instance, while a smaller percentage of VDD-$\bar{C}$'s total data is from pool environments, it has nearly three times as many images of pool environments $(16,657)$ as DDD has images of any type. Additionally, while there is less variety in what equipment divers are wearing, a much wider array of viewpoints are represented: divers were recorded swimming with or without a robot, viewed from many different angles, and sometimes merely treading water. Analysis of bounding box centroid location (Figure 5) clearly shows the greater variety in diver locations present in our new dataset. In comparison to previous datasets, our VDD-$\bar{C}$ dataset is more numerous, contains a sufficient amount of diver and environment variation, and has a much wider range of viewpoints and diver activities represented.

B. Labeling Process

Once the videos for the dataset had been selected and extracted to frames, the task of labeling them was addressed. Labeling $105,000$ images one by one was a difficult and time-consuming task, but it was improved by our choice of labeling tool. We used EVA [41], a web-based tool for labeling video data, shown in Figure 2. EVA is a rebuild of the popular Beaver-Dam tool, with the addition of tracking capabilities. Annotation is completed normally for the first frame in a video, with a user drawing a bounding box around every object they wish to label (Figure 3). Then, the user clicks the track button, and the initial annotations are used to initialize a Kernelized Correlation Filter tracker [42] which propagates those bounding boxes over the following frames. Depending on the difficulty of the tracking, the generated bounding boxes need to be adjusted and re-tracked somewhere between every frame and every 30 frames.

C. Post-Processing

With the initial labels generated, we began post-processing our data, beginning with a significant proofreading effort. To proofread, we watched every labeled video from start to finish to look for labeling errors, and we then corrected all observed labeling errors. Following the correction of these errors, a number of sections of video were cut due to



(a) Deep Diver Dataset (b) VDD-$\bar{C}$

Fig. 5: Distribution of bounding box centers.

significant motion blur or degradation of the visual quality. Additionally, any frame in a pool video which contained no diver was cut, as these frames were almost entirely from portions of the video with the camera out of water. A significant number of images were cut, bringing the total number of images down from approximately $114,000$ to the $105,000$ images previously mentioned. Finally, the exported annotations were automatically filtered for bounding box coordinates out of the acceptable range of the image size before being converted to YOLO [43]-style labels, TFExample [44] and TFSequence [45] records.

IV. Detection Models

We trained four models (some with a few variants) on VDD-$\bar{C}$ with the aim of answering the following questions:

1) Do models trained on VDD-$\bar{C}$ generally perform better on both the VDD-$\bar{C}$ and DDD test sets than models trained on the DDD dataset?
2) Do models with the highest accuracy (as measured by mAP) also have the highest temporal stability?

In the following sections we briefly explain these models, along with the training process we used for each, and any variants for which we report results.

A. Faster R-CNN

Faster R-CNN [21] is a two-stage object detector in the R-CNN family and a staple high accuracy object detector. Although it is not fast enough for use on a robotic platform, we chose to train this model to loosely represent a "top end" accuracy of state-of-the-art CNNs on our dataset. We utilize the Tensorflow Object Detection API [46] and its Faster R-CNN with an Inception-ResNet-v2 [47] feature extractor for training. Two hyperparameters, learning rate and batch size, were tuned with the validation set.

B. SSD with Mobilenet

SSDs [24] are among the most accurate real-time object detectors and therefore are good candidates for eventual deployment on a robotic platform. We train SSD320 (*i.e.*, SSDs for inputs sized 320×320) models with multiple Mobilenet [48] backbones to find the optimal model for our use case. We utilize the Tensorflow Object Detection API and the provided models for training. Learning rate and batch size were again tuned with the validation set.

C. YOLO

You Only Look Once (YOLO) [43] is a well established object detection model, valued for its high accuracy and speed. YOLO predicts a set of bounding boxes in a grid across the image, with confidences for each, and class probabilities for each grid box, matching class probabilities to the boxes with the highest confidences. We evaluate a variety of versions of YOLO (v2 [49] and v4 [50]) in this work, although YOLOv4 is our primary version for comparing with other networks. For every version of YOLO we train, we also train Tiny-YOLO, which reduces the number of convolutional layers and filters to improve the inference runtime of the network. To train these networks, we fine-tune them using initial weights trained on Imagenet.

D. LSTM-SSD

The only video object detection network evaluated in this work is the LSTM-SSD detector proposed by Liu and Zhu [29]. This model is based on Mobilenet SSDs, but adds a number of Bottleneck LSTMs [29] after the feature extraction network, followed by output layers. On the next frame's inference, features extracted by the convolutional layers will be combined with the LSTM's state, propagating feature maps through time. In order to train the LSTM-SSD, we initialize the convolutional portion of the network from a fine-tuned MobileNetV1 SSD, then train the LSTM portion of the network in order to improve the feature propagation.

V. Temporal Stability

In robotic applications, it is often desirable to have temporal stability for object detections. That is, detections for a given object should be stable over time with respect to:

- *Translation.* If detected bounding boxes are not consistently located with respect to the ground truth bounding boxes from frame to frame, it is difficult to estimate the object's location and trajectory.
- *Scale and aspect ratio.* Similarly, if detected bounding boxes have inconsistent scales and aspect ratios, it is difficult to estimate the object's location and trajectory.
- *Fragmentation.* For any given diver that appears in the video, the diver should be consistently detected (*i.e.*, the object should not be undetected in one frame, then detected in the next, and so on). At worst, fragmentations make it difficult for the robot to confidently determine that the object is present, and at best, they increase uncertainty of estimations of the object's location.

We adapt methods from [36] and [37] to evaluate diver detectors with respect to these three aspects of temporal stability. As in [37], we do not have ground truth tracks for the divers in our dataset and therefore computationally calculate ad-hoc tracklets for each diver by matching ground truth annotations from frame to frame with intersection over union (IOU) as in [51]. Using these tracklets, we then calculate the stability metrics from [36] as follows:

	Trained on VDD-$\bar{C}$						Trained on DDD					
	Tested on VDD-$\bar{C}$			Tested on DDD			Tested on VDD-$\bar{C}$			Tested on DDD		
Model	**AP$_{50}$**	**AP$_{75}$**	**IOU**	**AP$_{50}$**	**AP$_{75}$**	**IOU**	**AP$_{50}$**	**AP$_{75}$**	**IOU**	**AP$_{50}$**	**AP$_{75}$**	**IOU**
SSD(MobileNetV2)	82.43	41.07	39.61	90.12	**33.09**	65.14	69.14	25.61	45.73	85.90	19.8	65.80
SSD(MobileNetV3-Small)	81.43	34.03	29.50	60.20	11.10	58.50	60.24	11.10	31.23	83.90	17.46	65.80
SSD(MobileNetV3-Large)	**88.47**	**44.16**	40.42	91.29	27.94	66.21	**77.01**	**26.19**	40.01	89.81	**30.20**	65.85
YOLOv2	86.73	38.04	**68.25**	93.30	23.89	68.19	76.27	19.3	**60.92**	87.84	21.93	68.76
YOLOv2-Tiny	72.69	8.82	59.48	63.50	3.14	60.12	66.51	6.17	55.47	81.95	3.27	63.95
YOLOv4	83.65	34.13	64.16	81.38	21.64	**71.13**	70.47	20.86	59.25	**91.57**	18.23	71.13
YOLOv4-Tiny	81.93	34.7	58.82	**92.15**	26.26	68.00	75.56	19.43	57.83	84.41	11.44	**73.56**

TABLE I: Comparison between performance on test sets of the VDD-$\bar{C}$ and DDD with training on either train set.

1) Translation error: The translation error of each tracklet's detection is measured with the center position error e_c. To calculate e_c, for each detection d in the tracklet, we find the standard deviation of the distance between the normalized center x bounding box coordinate, x_d, and the normalized ground truth x_g. We do the same for the y_d and y_g coordinates. The translation error is the mean of these standard deviations across all tracks. Formally, for each tracklet t:

$$e_c(t) = \sigma(x_d - x_g) + \sigma(y_d - y_g), \forall d \in t$$

Then the detector's overall translation error is

$$\frac{1}{N}\sum_{t=1}^{N} e_c(t)$$

2) Scale and aspect ratio error: For each detection, the aspect ratio error is defined as the ratio between the bounding box aspect ratio and the ground truth aspect ratio. The scale error is defined as the square root of the bounding box area over the ground truth area. To find the scale and aspect ratio error, we find the average standard deviations of each track's summed scale error $e_s(t)$ and aspect ratio error $e_r(t)$. Formally, for each tracklet t:

$$e_s(t) = \sigma\left(\sqrt{\frac{w_d h_d}{w_g h_g}}\right), \forall d \in t$$
$$e_r(t) = \sigma\left(\frac{w_d}{h_d} / \frac{w_g}{h_g}\right), \forall d \in t$$
$$e_{sr}(t) = e_s(t) + e_r(t)$$

Then the detector's overall scale and aspect ratio error is

$$\frac{1}{N}\sum_{t=1}^{N} e_{sr}(t)$$

3) Fragmentation error: For each track, we count the number of fragments as the number of times the track's status changes from detected to undetected or vice versa. Then the fragmentation error is the average number of fragments f per track, normalized by track length l:

$$\frac{1}{N}\sum_{t=1}^{N} \frac{f_t}{l_t - 1}$$

Because the translation and scale metrics rely on standard deviations of a tracklet's detections, they become meaningless for tracklets with only one detection. Our analysis therefore excludes any tracklets with only one detection.

VI. RESULTS

A. Dataset Comparisons

To quantify how effective the new VDD-$\bar{C}$ dataset is in training deep vision models compared to previous datsets, we train one version of each SSD and YOLO variant on the VDD-$\bar{C}$ dataset and a second version on the existing DDD dataset. (We do not train Faster R-CNN or LSTM-SSD on DDD, since these models are not likely to be deployed and training is time- and resource-intensive.) We then compare the models' respective performances on each test set as shown in Table I. Results show that models trained on VDD-$\bar{C}$ outperform those trained on DDD on both the VDD-$\bar{C}$ test set and, to a lesser extent, the DDD test set. These results support our expectations that VDD-$\bar{C}$'s more complex data will lead to more successful detectors, because the VDD-$\bar{C}$ trained models outperform the DDD-trained models with few exceptions. Additionally, the fact that our VDD-$\bar{C}$ dataset is more challenging is reflected in these results, as DDD-trained detectors perform more poorly on the VDD-$\bar{C}$ test set than they do on the DDD test set.

B. Average Precision and IOU

To evaluate the accuracy of each model, we calculate the average precision (AP) of each model on diver identification. The average precision is found by evaluating the model's precision at different recall values. Specifically, since models output a confidence score for each detection, model recall values can be manipulated by changing the confidence threshold required for a detection; the AP is the weighted mean of the model's precision values at each recall value, where the weight for the recall at a given confidence threshold is the increase in recall from the previous threshold. Note that since our models are only trained to identify divers, the diver AP is equivalent to the mean average precision (mAP), which is a widely used object detection metric [18]. For each model, we pick a confidence threshold that results in the best precision and recall scores. Using that confidence threshold, we calculate AP at IOU thresholds of 0.5 and 0.75, average IOU, and an average of APs with thresholds between 0.5 and 0.95 with a step size of 0.05 (0.5-0.95).

Model	AP	AP_{50}	AP_{75}	IOU
Faster R-CNN	55.50	90.18	60.50	49.81
SSD(MobileNetv2)	43.45	82.43	41.07	39.61
SSD(MobileNetv3-Small)	39.81	81.43	34.03	29.50
SSD(MobileNetv3-Large)	**47.05**	**88.47**	**44.16**	40.42
YOLOv4	41.01	83.65	34.13	**64.16**
YOLOv4-Tiny	33.39	81.93	34.70	58.81
LSTM-SSD	39.00	79.80	33.10	51.40

TABLE II: Precision and IOU.

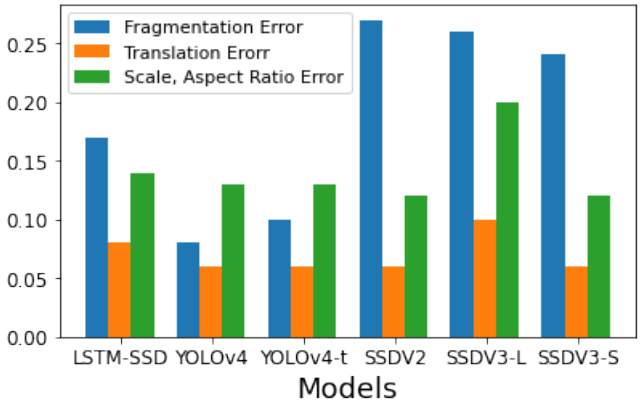


Fig. 6: Measured stability errors for different models.

C. Stability Results

The average translation, scale, and fragment errors across all diver tracklets are calculated for each model using the equations discussed in Section V. Most of the models perform very similarly with respect to translation error and scale error, with the exception SSD-MobilenetV3-L, whose errors are higher than the other models'. Fragmentation error varies more across models. The YOLOv4 and YOLOv4-tiny models have the lowest fragmentation errors, despite not having the highest AP. Notably, while the LSTM-SSD and SSD-Mobilenets have comparable AP, the LSTM-SSD has a lower fragmentation error, indicating that it outperforms SSD in detecting divers consistently. Finally, we note that despite SSD-MobilenetV3-L having the highest AP, it also has the highest scale and aspect ratio error and the second highest fragmentation error, which suggests that it may not be the best choice for deployment.

D. Efficiency Results

YOLO, SSD, and LSTM-SSD are high speed models designed for real-time inference use cases. While Faster R-CNN has demonstrated high accuracies, it is computationally involved, and is not quite suitable for real-time inference as shown in [7]. In order to quantify the usability of our real-time models on robotic platforms, we quantify their inference run-time in terms of frames processed per second (FPS). The results of these tests can be seen in Table III. Due to the size of the test dataset, we only tested a portion of the test set for runtime calculation: 5,000 randomly selected frames. We tested each network on two devices: an Nvidia 1080 GPU and an Nvidia Jetson TX2. These results do not represent the maximum inference speed possible, as no platform-specific optimization was done, but they provide a guide to the applicability of these networks in embedded contexts, on board AUVs. While the SSD variants achieve relatively high framerates on embedded devices, the clear standout is YOLOv4-Tiny, which achieves real-time performance with accuracy closer to other methods. LSTM-SSD also performs quite well, surpassing the traditional SSD variants.

Model	FPS(GPU)	FPS(TX2)
SSD(MobileNetv2)	50	9
SSD(MobileNetv3-Small)	52	9
SSD(MobileNetv3-Large)	51	8
YOLOv4	50	5
YOLOv4-Tiny	88	35
LSTM-SSD	73	19

TABLE III: Frames per second for inference.

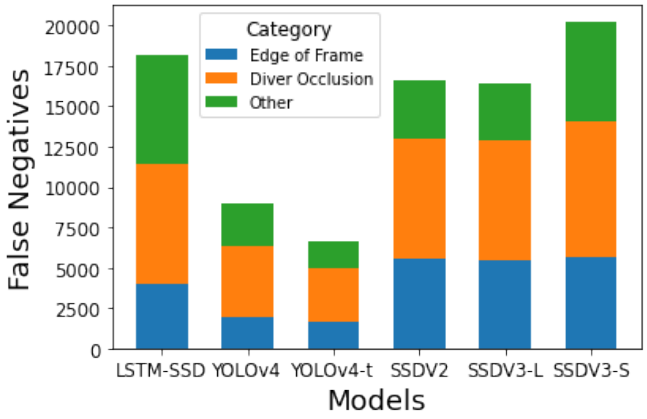


Fig. 7: The source of false negative errors in different models.

E. Failure Scenarios

When considering a diver detector for use on an AUV, there is information of interest beyond accuracy, stability, and efficiency: when and why the detector fails. By inspecting the instances of false negative detections, we can gain some intuition on the circumstances of detector failures in the diver detector task. A significant portion of false negatives stem from one of two cases: divers not fully in the frame, or diver occlusions, as shown in Figure 7. We define a diver as not fully in the frame if an edge of their ground truth bounding box is on the edge of the frame. We define a diver occlusion as two ground truth bounding boxes with an IOU above 0.

VII. Conclusion

In this paper, we address some challenges we have faced in deploying object detectors onboard AUVs to detect divers. First, we create VDD-$\bar{C}$, a large dataset of annotated videos of divers that presents divers at a variety of translations and orientations. We show that our dataset improves the quality of detection results significantly, while simultaneously providing a stronger challenge in the test set. In terms of precision and efficiency results, we reproduced previous work, showing that SSD and YOLO networks have similar AP and mobile inference runtimes, though LSTM-SSD outpaces all but Tiny-YOLO for mobile inference. Critically, we also evaluate models on their temporal stability, since consistent detections

are important for several robotic applications. We found that while SSD models generally had an edge over YOLO models in terms of AP, YOLO models had better detection stability in terms of fragmentation error and scale and aspect ratio error. In particular, SSDs had more than double the fragmentation error of YOLO models. This suggests that those whose work depends on consistent detections across frames may wish to evaluate vision models on stability metrics in addition to traditional metrics. Lastly, we recommend further exploration of video object detection methods for future work. While LSTM-SSD did not outperform other models in terms of accuracy, its low inference times and high stability reveal the potential benefits of video object detection for robotic applications.

REFERENCES

[1] S. B. Williams, O. Pizarro, M. Jakuba, and N. Barrett, "AUV Benthic Habitat Mapping in South Eastern Tasmania," in *Field and Service Robotics*, A. Howard, K. Iagnemma, and A. Kelly, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 275–284.

[2] R. B. Wynn, V. A. Huvenne, T. P. Le Bas, B. J. Murton, D. P. Connelly, B. J. Bett, H. A. Ruhl, K. J. Morris, J. Peakall, D. R. Parsons, E. J. Sumner, S. E. Darby, R. M. Dorrell, and J. E. Hunt, "Autonomous underwater vehicles (auvs): Their past, present and future contributions to the advancement of marine geoscience," *Marine Geology*, vol. 352, pp. 451–468, 2014.

[3] K. R. Reisenbichler, M. R. Chaffey, F. Cazenave, R. S. McEwen, R. G. Henthorn, R. E. Sherlock, and B. H. Robison, "Automating MBARI's midwater time-series video surveys: The transition from ROV to AUV," in *OCEANS 2016 MTS/IEEE Monterey*, 2016, pp. 1–9.

[4] Y. R. Petillot, S. R. Reed, and J. M. Bell, "Real time AUV pipeline detection and tracking using side scan sonar and multi-beam echo-sounder," in *OCEANS '02 MTS/IEEE*, Oct 2002, pp. 217–222 vol.1.

[5] M. Allison, H. Dawson, and G. Rusin, "Towards an AUV swarm based mobile underwater sensor network for invasive species data acquisition," in *2018 4th International Conference on Universal Village (UV)*, 2018, pp. 1–4.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[7] M. J. Islam, M. Fulton, and J. Sattar, "Toward a generic diver-following algorithm: Balancing robustness and efficiency in deep visual detection," *IEEE Robotics and Automation Letters*, vol. 4, no. 1, pp. 113–120, 2019.

[8] M. Modasshir and I. Rekleitis, "Augmenting coral reef monitoring with an enhanced detection system," in *IEEE International Conference on Robotics and Automation*, Paris, France, 2020, pp. 1874–1880.

[9] M. Fulton, J. Hong, M. J. Islam, and J. Sattar, "Robotic Detection of Marine Litter Using Deep Visual Detection Models," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 5752–5758.

[10] A. Azulay and Y. Weiss, "Why do deep convolutional networks generalize so poorly to small image transformations?" *Journal of Machine Learning Research*, vol. 20, no. 184, pp. 1–25, 2019.

[11] V. Shankar, A. Dave, R. Roelofs, D. Ramanan, B. Recht, and L. Schmidt, "Do image classifiers generalize across time?" *arXiv preprint arXiv:1906.02168*, 2019.

[12] K. Gu, B. Yang, J. Ngiam, Q. Le, and J. Shlens, "Using videos to evaluate image model robustness," *arXiv preprint arXiv:1904.10076*, 2019.

[13] R. Taori, A. Dave, V. Shankar, N. Carlini, B. Recht, and L. Schmidt, "Measuring robustness to natural distribution shifts in image classification," *arXiv preprint arXiv:2007.00644*, 2020.

[14] N. Sünderhauf, O. Brock, W. Scheirer, R. Hadsell, D. Fox, J. Leitner, B. Upcroft, P. Abbeel, W. Burgard, M. Milford, *et al.*, "The limits and potentials of deep learning for robotics," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 405–420, 2018.

[15] L. Engstrom, B. Tran, D. Tsipras, L. Schmidt, and A. Madry, "Exploring the landscape of spatial robustness," in *International Conference on Machine Learning*, 2019, pp. 1802–1811.

[16] M. A. Alcorn, Q. Li, Z. Gong, C. Wang, L. Mai, W.-S. Ku, and A. Nguyen, "Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 4845–4854.

[17] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[18] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[20] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[22] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 379–387.

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2961–2969.

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 21–37.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[26] M. Manfredi and Y. Wang, "Shift equivariance in object detection," *arXiv preprint arXiv:2008.05787*, 2020.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[29] M. Liu and M. Zhu, "Mobile video object detection with temporally-aware feature maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 5686–5695.

[30] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3038–3046.

[31] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei, "Flow-guided feature aggregation for video object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 408–417.

[32] G. Bertasius, L. Torresani, and J. Shi, "Object detection in video with spatiotemporal sampling networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 331–346.

[33] M. Liu, M. Zhu, M. White, Y. Li, and D. Kalenichenko, "Looking fast and slow: Memory-guided mobile video object detection," *arXiv preprint arXiv:1903.10172*, 2019.

[34] Y. Chen, Y. Cao, H. Hu, and L. Wang, "Memory enhanced global-local aggregation for video object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 337–10 346.

[35] X. Zhu, J. Dai, L. Yuan, and Y. Wei, "Towards high performance video object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7210–7218.

[36] H. Zhang and N. Wang, "On the stability of video detection and tracking," *arXiv preprint arXiv:1903.10172*, 2017.

[37] X. Chen, Z. Wu, J. Yu, and L. Wen, "Rethinking temporal object detection from robotic perspectives," *arXiv preprint arXiv:1912.10406*, 2020.

[38] A. Gomez Chavez, A. Ranieri, D. Chiarella, E. Zereik, A. Babić, and A. Birk, "Caddy underwater stereo-vision dataset for human–robot

interaction (hri) in the context of diver activities," *Journal of Marine Science and Engineering*, vol. 7, no. 1, p. 16, 2019.

[39] N. Mišković, M. Bibuli, A. Birk, M. Caccia, M. Egi, K. Grammer, A. Marroni, J. Neasham, A. Pascoal, A. Vasilijević, *et al.*, "Caddy—cognitive autonomous diving buddy: Two years of underwater human-robot interaction," *Marine Technology Society Journal*, vol. 50, no. 4, pp. 54–66, 2016.

[40] A. G. Chavez, M. Pfingsthorn, A. Birk, I. Rendulić, and N. Misković, "Visual diver detection using multi-descriptor nearest-class-mean random forests in the context of underwater human robot interaction (hri)," in *OCEANS*. IEEE, 2015, pp. 1–7.

[41] Ericsson Open Source Software, "Github - ericsson/eva," https://github.com/Ericsson/eva, (Accessed on 10/26/2020).

[42] J. a. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *Proceedings of the 12th European Conference on Computer Vision (ECCV)*, Berlin, Heidelberg, 2012, p. 702–715.

[43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[44] TensorFlow, *tf.train.Example*, 2020, https://bit.ly/3jO6tuR. Accessed 10-31-2020.

[45] ——, *tf.train.SequenceExample*, 2020, https://bit.ly/3oMDdYX. Accessed 10-31-2020.

[46] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7310–7311.

[47] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 4278–4284.

[48] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[49] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[50] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint ArXiv:2004.10934*, 2020.

[51] X. Chen, J. Yu, and Z. Wu, "Temporally identity-aware SSD with attentional LSTM," *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2674–2686, 2019.