Caveline Detection at the Edge for Autonomous Underwater Cave Exploration and Mapping

Mohammadreza Mohammadi
CSE, University of South Carolina
Columbia, SC 29208, USA
mohammm@email.sc.edu

Sheng-En Huang
ECE, University of Florida
Gainesville, FL 32611, USA
huang.sh@ufl.edu

Titon Barua
CSE, University of South Carolina
Columbia, SC 29208, USA
baruat@email.sc.edu

Ioannis Rekleitis
CSE, University of South Carolina
Columbia, SC 29208, USA
yiannisr@cse.sc.edu

Md Jahidul Islam
ECE, University of Florida
Gainesville, FL 32611, USA
jahid@ece.ufl.edu

Ramtin Zand
CSE, University of South Carolina
Columbia, SC 29208, USA
ramtin@cse.sc.edu

2023 International Conference on Machine Learning and Applications (ICMLA) | 979-8-3503-4534-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICMLA58977.2023.00210

***Abstract*—This paper explores the problem of deploying machine learning (ML)-based object detection and segmentation models on edge platforms to enable real-time caveline detection for Autonomous Underwater Vehicles (AUVs) used for underwater cave exploration and mapping. We specifically investigate three ML models, i.e., U-Net, Vision Transformer (ViT), and YOLOv8, deployed on three edge platforms: Raspberry Pi-4, Intel Neural Compute Stick 2 (NCS2), and NVIDIA Jetson Nano. The experimental results unveil clear trade-offs between model accuracy, processing speed, and energy consumption. The most accurate model has shown to be U-Net with an 85.53 F1-score and 85.38 Intersection Over Union (IoU) value. Meanwhile, the highest inference speed and lowest energy consumption are achieved by the YOLOv8 model deployed on Jetson Nano operating in the high-power and low-power modes, respectively. The comprehensive quantitative analyses and comparative results provided in the paper highlight important nuances that can guide the deployment of caveline detection systems on underwater robots for ensuring safe and reliable AUV navigation during underwater cave exploration and mapping missions.**

***Index Terms*—Edge Computing, Object Detection, Segmentation, Underwater Robots, Visual Servoing.**

I. Introduction & Background

Underwater caves offer opportunities for scientific research in fields such as archaeology, hydrology, geology and hydrogeology, and marine biology [1]. Cave formations, sediments, and water chemistry can provide insights into past climate conditions and geological processes. They also play a crucial role in monitoring and tracking groundwater flows in Karst topographies; it shall be noted that almost 25% of the world's population relies on Karst freshwater resources [2]. Moreover, underwater caves often present a pristine *capsule* preserved in time with major archaeological secrets [3], [4]. Underwater cave exploration and mapping by human divers, however, is a tedious, labor-intensive, extremely dangerous operation even for highly skilled people [5]. Therefore, enabling Autonomous Underwater Vehicles (AUVs) and Remotely Operated Vehicles (ROVs) to enter, navigate, map, and explore underwater caves is of significant importance [6], [7]; Fig. 1 shows an ROV deployment scenario inside an underwater cave system.

Fig. 1: A BlueROV2 operating inside an underwater cave system by following a *caveline*; note that the umbilical is connecting the ROV to a surface operator.

The underwater caves explored by human scuba divers are marked with a single and continuous line termed *caveline* [8] that goes from open water (no overhead) to all the major parts of the cave. Along with other navigation markers such as *arrows* and *cookies*, the caveline provides the skeleton of the cave (*i.e.*, a one-dimensional retraction of the 3D space) [9] of the main passages marking the depth and orientation of a cave. Thus, detecting and following the caveline as navigation guidance is paramount for robots in autonomous cave exploration and mapping missions. Recently, Yu *et al.* [10] developed a robust Vision Transformer (ViT)-based learning pipeline named *CL-ViT* to detect and track cavelines by underwater robots for vision-based navigation. CL-ViT learning pipeline has two important features: (**i**) robustness to noise and image distortions [11]; and (**ii**) generalized model adaptation to data from new locations. Despite state-of-the-art detection performance, the proposed models are computationally demanding and do not offer real-time performance on edge devices. As most AUVs have limited or no connectivity with the surface, all computations have to be onboard. In addition, the limited onboard space available make the use of traditional GPUs infeasible.

1946-0759/23/$31.00 ©2023 IEEE
DOI 10.1109/ICMLA58977.2023.00210

In this paper, we examine several distinct models, deploy them on various edge AI accelerators, and compare their computational load and accuracy. In particular, we investigate three models, (**i**) U-Net [12] that is a classic model for semantic segmentation, (**ii**) ViT [10] that has achieved the state-of-the-art performance for various computer vision tasks, and (**iii**) a recently developed model from the well-known YOLO family called YOLOv8 [13]. We fine-tune these models for our targeted application and deploy them on three edge platforms, *i.e.*, Raspberry Pi-4, Intel™ Neural Compute Stick, and Nvidia™ Jetson Nano, with distinct hardware characteristics that demand different deployment strategies.

Application scenario. The lightweight caveline detection models developed in this paper will be deployed in the autonomy pipeline of AUVs for underwater cave exploration and mapping applications. The objective here is to enable underwater robots to enter, explore a cave system by following the caveline while mapping its surroundings, and finally safely exit the cave. To this end, achieving real-time caveline detection performance on edge devices is essential for safe AUV navigation – which is the focus of this work.

II. Related Work

A. Object Detection & Segmentation in Underwater Imagery

An essential capability of visually guided AUVs is to identify relevant objects and interesting image regions to make effective navigational decisions in real time [14]. Various model-based techniques are generally deployed in fast visual search [15], [16], enhanced object detection [17], [18], and monitoring applications [19], [20]. For instance, Koreitem *et al.* [15] used a bank of pre-specified image patches to learn a similarity operator that guides the robot's visual search in an unconstrained setting. Besides, model-free approaches are more feasible for autonomous exploratory applications [21]. For instance, Girdhar *et al.* [22] formulated an online topic-modeling scheme that encodes visible features into a low-dimensional semantic descriptor for AUV exploration. More recent works by Modasshir *et al.* combined a deep learning-based classifier model with Visual Inertial Odometry (VIO) to identify and track the locations of different types of corals to generate semantic maps [23] and volumetric models [24].

B. Object Detection & Tracking on Edge Devices

In recent years, various TinyML techniques [25] and lightweight deep visual models [26] have been introduced such as SqueezeNet [27], MobileNet [28]–[30], ShuffleNet [31], [32], PeleeNet [33], MnasNet [34], Once-for-All (OFA) [35], GhostNet [36], MobileVit [37], and more. Iandola *et al.* [27] proposed a lightweight model called *SqueezeNet*, reaching AlexNet-level accuracy on ImageNet with $50\times$ fewer parameters and less than 0.5 MB memory requirement. By employing streamlined architecture with depth-wise separable convolutions, Howard *et al.* [28] introduced the family of *MobileNet* models that are ideal for single-board embedded platforms. The successor MobileNetv2 [29] uses inverted residual structures to reduce computations, while MobileNetv3 [30] adopts a platform-aware automated neural architecture search in hierarchical search space along with NetAdapt [38], which further reduces the components of the network.

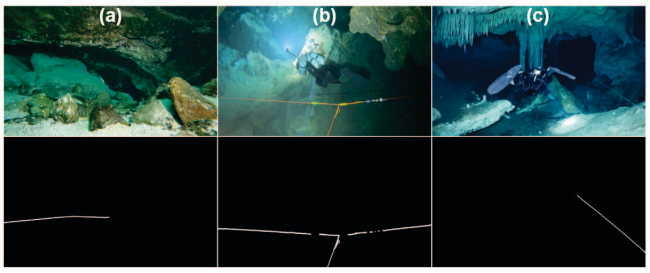

Fig. 2: Sample data training collected from three underwater cave systems in (a) Devil's Spring system, FL, USA; (b) Cueva del Agua, Murcia, Spain; and (c) Dos Ojos Cenote, QR, Mexico – are shown in each column. The corresponding binary labels marking the cavelines are shown in the bottom row.

On the other hand, Zhang *et al.* [31] used the ResNet block coupling with innovations to devise *ShuffleNet*, which is compatible with mobile devices. ShuffleNetv2 [32] further improves the speed and accuracy by adopting a direct metric (speed) rather than indirect metrics like FLOPs. Moreover, Wang *et al.* [33] proposed a *PeleeNet* model by adopting an assortment of computation-conserving methods, making a compelling lightweight network. Tan *et al.* [34] presented a novel mobile CNN-based model using an automated neural architecture search approach. Besides, Cai *et al.* [35] introduced *OFA*, a lightweight network that can meet different hardware requirements. Furthermore, by merging the features of CNN and ViT, Mehta *et al.* [37] presented a lightweight and versatile vision transformer called *MobileVit* for edge devices. Previous works have also explored the deployment of these models on edge AI accelerators [39]–[41].

III. Experimental Setup and Methodology

A. Dataset Preparation

For data-driven training and evaluation, we extract video frames from cave exploration experiments [42], [43] conducted in three different locations: the Devil's Spring System in Florida, USA; the Dos Ojos Cenote, QR, Mexico; and the Cueva del Agua in Murcia, Spain. As illustrated in Fig. 2, the three cave locations exhibit different caveline characteristics in terms of thickness, color, and background patterns. We identify a variety of challenging cases and prepare 1050 images in each set, totaling 3150 training instances. We follow the problem formulation of CL-ViT [10], where the caveline detection in the RGB space is learned as a binary image segmentation task, *i.e.*, identifying pixels with caveline as a semantic map. Four human participants sorted these image samples and then pixel-annotated the cavelines for ground truth generation, which we utilize for the training of all models in consideration. For evaluation, we use the CL-Challenge test set with 200 samples presented in [10].

B. Targeted ML Models

In this paper, we use three ML models: YOLOv8, Vision Transformer (ViT), and U-Net. The following subsections provide the details and characteristics of these models.

1) YOLOv8: YOLOv8 [13] is the cutting-edge YOLO model, which has exhibited promising performance across various ML tasks such as object detection, image classification, and instance segmentation. It is designed by Ultralytics, the developer of the YOLOv5 model that has been influential in the field. Here, we use transfer learning paradigms, according to which we begin with a YOLOv8 model that is pre-trained with the COCO dataset, and fine-tune it for the cave exploration application using our targeted dataset. As we aim to deploy the models on resource-constrained edge devices, we chose a small version of YOLOv8 with 3.4M parameters.

2) Vision Transformer (ViT): The ViT model we used in this paper is inspired by the model developed in [10]. The ViT architecture includes two key components: a streamlined encoder-decoder backbone and a refinement module. It incorporates a MobileNetV3 [30] model as its backbone. The encoder comprises a sequence of convolutional layers consisting of 16 filters, followed by 15 residual bottleneck layers. The decoder employs six convolutional blocks to transform the encoded features into 48 filters, resulting in a 480×270 resolution image.

3) U-Net: U-Net [12] is a well-known architecture for semantic segmentation tasks, comprising two main components: the encoder or contraction path, and the decoder or expansion path. The contraction path mirrors the standard setup of a convolutional network. It involves iteratively applying a pair of 3×3 convolutions, followed by a ReLU activation function and 2×2 max pooling operations using a stride of 2. As for the expansion path, each step involves feature map upsampling, followed by 2×2 convolutions that reduce the feature channel count by half. A concatenation operation connects these layers with the feature maps obtained from the contraction path. Collectively, the network encompasses a total of 23 convolutional layers. The feature extraction component of our U-Net model employs a MobileNetV2 architecture.

C. Edge devices

In this study, we deploy and evaluate our ML models on various edge devices including Raspberry Pi, Intel Movidius Neural Compute Stick 2 (NCS2), and Nvidia Jetson Nano, as illustrated in Figure 3. These devices are commonly utilized in different ML tasks such as image classification [41] and natural language processing [40]. Each edge device has unique attributes and demands a specific deployment methodology, which are explained in the following:

1) Raspberry Pi-4: The Raspberry Pi-4 board is driven by a Broadcom BCM2711 quad-core ARM Cortex-A72 CPU, which operates at different speeds depending on the model – 1.5 GHz for the 2 GB and 4 GB RAM models, and 1.8 GHz for the 8 GB RAM model. Available in various configurations, the Raspberry Pi-4 offers distinct RAM capacities: 2 GB, 4 GB, and 8 GB LPDDR4. Powering the device necessitates a 5V USB-C power supply with the appropriate current rating, which varies according to the intended use case and peripherals connected.

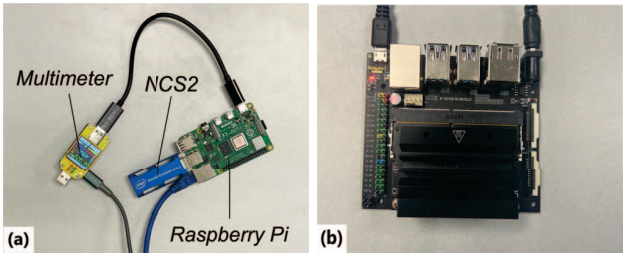

Fig. 3: Experimental setup with (a) Raspberry Pi-4 + NCS2; and (b) Nvidia™ Jetson Nano

2) Intel Neural Compute Stick: The Intel Neural Compute Stick 2 (NCS2) utilizes the Intel Movidius-X Vision Processing Unit (VPU), incorporating 16 programmable cores and a dedicated neural compute engine. The NCS2 operates at 700MHz base frequency and includes a 4GB of RAM. Intel offers the OpenVINO library, accessible in both Python3 and C, to facilitate the deployment of ML models on the NCS2. This library encompasses a model optimizer, which transforms models into a suitable format for NCS2 deployment. Subsequently, utilizing OpenVINO's built-in inference engine API allows for the assessment of latency and power efficiency during inference.

3) Nvidia Jetson Nano: The Jetson Nano is a modular computer powered by a Tegra X1 system-on-chip (SoC) with a ARM A57 quad-core processor, and four 32-CUDA core processing blocks. It also includes 4GB of memory. Nvidia™ offers two operating modes for Jetson Nano; the low power mode (Jetson-low), in which only two cores of the ARM A57 are activated and the clock frequency is set at 0.9GHz. In addition, the GPU's clock frequency is set to 0.64GHz. In the high power mode (Jetson-high), all the four cores of Arm 57 processor are activated at 1.5GHz frequency while the GPU runs at 0.92GHz frequency. The Jetson Nano uses NVIDIA TensorRT as its primary driver for ML model optimization.

IV. Performance Evaluation

A. Deployment Considerations

Once the targeted ML models described in the previous section are trained and fine-tuned for the caveline detection application, our focus shifted to deploying them on various edge platforms that each have specific requirements. Leveraging PyTorch during training, we adopted the ONNX format as an intermediary for model export. The Raspberry Pi-4 employs TFLite models with 32-bit floating-point (FP32) precision, while the Jetson employs TensorRT models with FP16 precision. Additionally, we incorporated the NCS2 accelerator as a co-processor with the Raspberry Pi-4, utilizing OpenVino

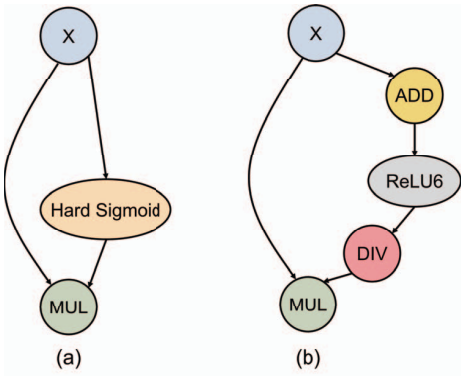

Fig. 4: Computational graphs for Hard-Swish activation function using (a) Hard-Sigmoid (b) ReLU6.

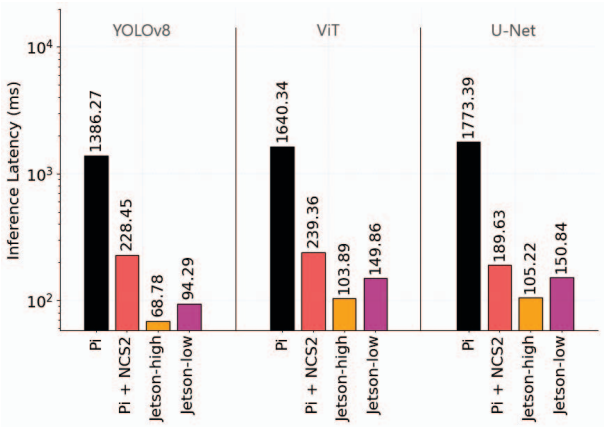

Fig. 5: Inference latency comparison for all models and devices in consideration.

2021 to convert ONNX models to the required format, where FP16 operations are used.

The architecture of the ViT model incorporates a MobileNetV3 model as its encoder backbone, featuring a distinctive *Hard Swish* activation function. This activation function encompasses a component known as the *Hard Sigmoid* as it is outlined in Eq. 1.

$$\texttt{HardSwish}(x) = x \cdot \texttt{HardSigmoid}(x) \tag{1}$$

The OpenVino 2021 lacks support for the *Hard Sigmoid* activation function, which leads to the unsuccessful deployment of the ViT model on the NCS2 platform. However, A noteworthy detail is that the *Hard Sigmoid* employs *ReLU6* function within its equation, as demonstrated below,

$$\texttt{HardSigmoid}(x) = \texttt{ReLU6}(x+3)/6 \tag{2}$$

To surmount the aforementioned deployment limitation, we substituted the Hard Sigmoid layer with equivalent functions, taking advantage of OpenVino 2021's support for ReLU6. The replacement of the Hard Sigmoid is depicted in Fig. 4, which illustrates the revised computation graph to accommodate the ReLU6 activation.

B. Performance Metrics

Here, we evaluate the performance of the ML models using two commonly used metrics: Intersection Over Union (IOU) and F1-score. IOU assesses the accuracy of object localization by calculating the proportion of overlapping area between predicted and actual labels. It is defined as:

$$\texttt{IoU} = \frac{\texttt{Area of Overlap}}{\texttt{Area of Union}} \tag{3}$$

Furthermore, the F1 score evaluates the accuracy of predicted labels in relation to the true data labels, using normalized precision ($\mathcal{P}$) and recall ($\mathcal{R}$) scores, yielding:

$$\texttt{F1-score} = \frac{2 \times \mathcal{P} \times \mathcal{R}}{\mathcal{P} + \mathcal{R}} \tag{4}$$

According to the obtained results, the U-Net model demonstrates superior performance compared to the other two models in both IoU and F1-score evaluations. The results, as listed in Table I, reveal remarkable improvements achieved by U-Net. Specifically, in terms of F1-score, U-Net exhibits 37.32% and 52.63% higher performance compared to the ViT and YOLOv8 models, respectively. Furthermore, when evaluating IoU, U-Net achieves a significant 61.87% improvement compared to YOLO, and a 46.94% improvement compared to ViT. The higher performance of the U-Net model can also be observed in three samples provided in Table II, in which U-Net provides the highest similarity to the labels. After U-Net, as expected from the F1-score and IoU results, the ViT model outperforms the YOLOv8 model.

C. Inference Latency Measurement

To assess the inference latency, we first executed one hundred inference operations for each model on each edge platform and measured the overall latency. Next, we calculated the average inference time for a single image using the measurement results. Fig. 5 shows the inference latency results obtained. As it can be seen in the figure, the edge AI accelerators, *i.e.*, NCS2 and Jetson Nano, could achieve approximately one order of magnitude reduction in inference latency across all models. The Jetson Nano operating in the high-power mode realized the fastest inference operations with latency values of 68ms, 103ms, and 105ms for YOLO, ViT, and U-Net models, respectively. The ranking of edge platforms in terms of inference speed from fastest to slowest is as follows: Jetson in high-power mode, Jetson in low-power mode, NCS2, and Raspberry Pi-4.

TABLE I: Quantitative performance comparisons of all models in terms of F1-score and IoU metrics.

Model	F1-score	IoU
YOLO	32.90	23.51
ViT	48.21	38.44
U-Net	85.53	85.38

TABLE II: Our qualitative evaluations infer that the U-Net has better performance compared to the other models in terms of pixel accuracy and localization performance; three sample results are shown here for each model in comparison.

Original Image	Label	Model Outputs		
		U-Net	ViT	YOLOv8

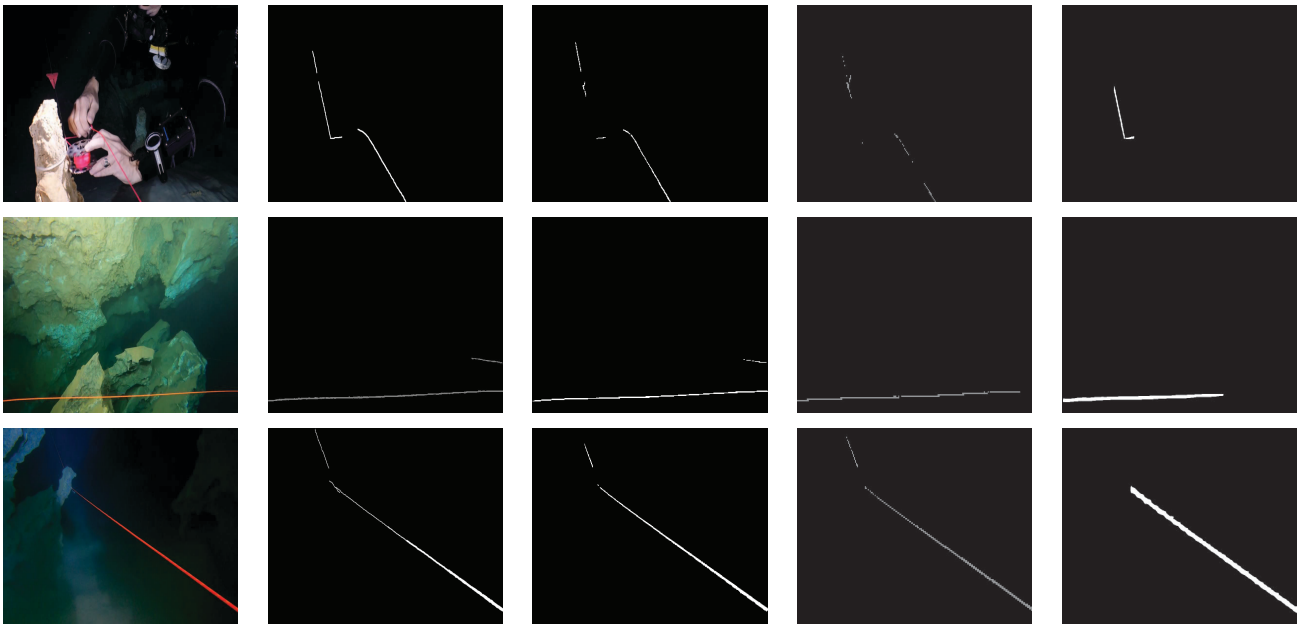

TABLE III: Frame per second (FPS) rates for all models running on the edge devices in consideration.

Model	Frame Per Second (FPS)			
	Jetson-L	Jetson-H	NCS2	Raspberry Pi-4
YOLO	10.29	14.54	4.38	0.72
ViT	6.67	9.63	4.18	0.61
U-Net	6.63	9.50	5.27	0.56

In addition, Table III provides a comparison across all the models and edge platforms in terms of the frames per second (FPS) metric. As listed in the table, the YOLO model executed on the Jetson Nano in high-power mode achieved the highest FPS of 14.54, realizing a near-realtime caveline detection. It is worth noting that AUVs and ROVs move at slow speeds underwater, thus the view does not change drastically, in contrast with aerial vehicles. Therefore, identifying the cave line in a few frames per second is sufficient for safe navigation. Studying the latency and accuracy results together shows that the high accuracy realized by the U-Net model is achieved at the cost of longer inference latency and smaller FPS ratios.

D. Inference Power Measurement

To measure the power dissipation of the inference operation for the different models deployed on Raspberry Pi-4 and NCS2 devices, we used the MakerHawk UM34C USB multimeter as shown in Fig. 3 (a). For the Jetson Nano, however, we used its internal sensors which measure the CPU and GPU power dissipation. Here, we executed the inference operation for each model on each device for three consecutive minutes and measured the power dissipation with a sample rate of one measurement per second.

Figure 6 shows the dynamic power measurements for different models and platforms. The Jetson Nano's high-power mode stands out with the highest power dissipation, ranging from 3.51 watts to 3.72 watts across all models. The remaining hardware platforms demonstrate comparable power dissipation levels ranging from 1.65 watts to 2.07 watts. As shown in the figure, when executing YOLO and U-Net models, the Jetson Nano in low-power mode realized the lowest power dissipation, while for the ViT model, the NSC2 platform achieved the least dynamic power dissipation. In addition, the lowest power dissipation across all models and devices is achieved by the YOLO model deployed on the Jetson Nano in the low-power mode.

E. Inference Energy Measurement

In Fig. 7, we present a comparative analysis of inference energy results. It is evident that all the edge AI accelerators bring a significant improvement in inference energy compared to the baseline Raspberry Pi-4. For instance, the YOLO model running on the Jetson Nano in low-power mode achieves $18.5\times$ inference energy reduction compared to the YOLO model deployed on Pi-4. Furthermore, when examining the YOLO model on the Jetson Nano in high-power mode and the NCS2, we can observe a $11.6\times$ and $6.8\times$ energy reduction compared to the baseline Raspberry Pi-4, respectively. Moreover, for the Vit model, the Jetson Nano's low-power mode demonstrates substantial savings in inference energy, showing

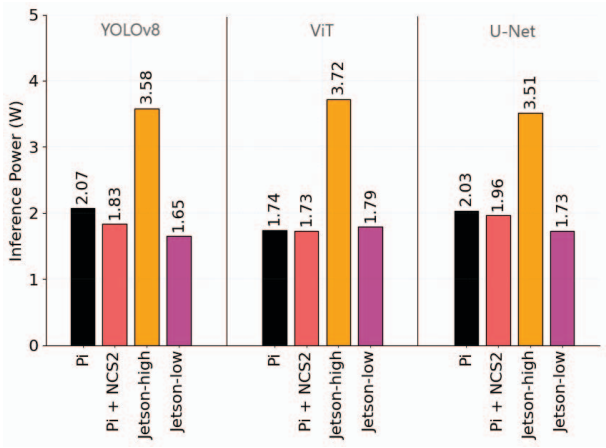

Fig. 6: Dynamic power comparison for all models and devices.

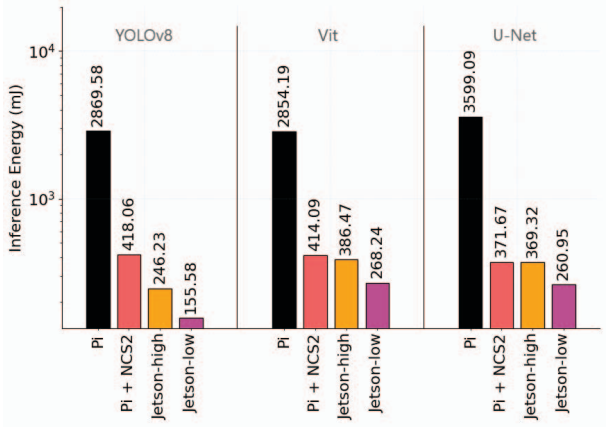

Fig. 7: Dynamic energy comparison for all models and devices.

$10.6\times$, $1.5\times$, and $1.4\times$ energy reduction when compared to the Raspberry Pi-4, Raspberry Pi-4 + NCS2, and Jetson Nano in high-power mode, respectively. In summary, the YOLO model running on Jetson Nano demonstrates the most energy-efficient performance among all models and devices. Both the YOLO and ViT models exhibit similar energy consumption when deployed on Pi-4 and Pi+NCS2 platforms, while the U-Net model stands out as the most energy-demanding option.

V. Conclusion and Future Work

In this paper, we focused on the problem of real-time caveline detection which is crucial for underwater robots in autonomous cave exploration and mapping missions. Due to the limitations in the connectivity of the AUVs with the surface as well as the space limitations, it is not feasible to rely on cloud services or powerful GPUs to run the deep learning-based caveline detection models. Therefore, our study investigated the trade-offs associated with deploying three different object detection and segmentation models on various edge platforms, with a focus on assessing accuracy, latency, power, and energy consumption. Our experimental results affirmed the expected trade-offs such as larger models exhibited higher accuracy at the cost of increased inference times and energy consumption. However, our analysis yielded more intriguing nuances including (**i**) the U-Net model, despite being considered a classic model for object detection and segmentation, could achieve significantly higher F1-scores and IoU values compared to more advanced models like ViT, while maintaining comparable FPS rate and energy consumption across all edge platforms, (**ii**) the FPS rate which is an important metric for realtime operation appeared to be more dependant on the choice of edge platform rather than the ML model employed. For instance, a lightweight YOLOv8 model running on Intel NCS2 realized a caveline detection speed of 4.38 FPS, whereas a significantly larger and more accurate ViT model deployed on Jetson Nano in high-power mode could achieve more than double processing speed of 9.63 FPS.

The comprehensive quantitative analyses presented in the paper offer a guide for making design decisions and managing trade-offs while integrating our cave-line detection systems with underwater robots for cave exploration and mapping missions. In our future research, we will delve into the practical consequences of these trade-offs, including factors such as battery life, accuracy in the wild, meeting real-time mission requirements, and the thermal effects stemming from computing power dissipation.

Acknowledgment

This research has been supported in part by the NSF grants 1943205 and 2024741. The authors would also like to acknowledge the help of the Woodville Karst Plain Project (WKPP), El Centro Investigador del Sistema Acuífero de Quintana Roo A.C. (CINDAQ), Global Underwater Explorers (GUE), and Ricardo Constantino, Project Baseline in collecting data, providing access to challenging underwater caves, and mentoring us in underwater cave exploration. We also appreciate the help from Evan Kornacki for coordinating our field experimental setups.

References

[1] M. J. Lace and J. E. Mylroie, "The biological and archaeological significance of coastal caves and karst features," in *Coastal Karst Landforms*, pp. 111–126, Springer, 2013.

[2] D. Ford and P. Williams, *Introduction to Karst*, ch. 1, pp. 1–8. John Wiley & Sons, Ltd, 2007.

[3] A. H. G. González, C. R. Sandoval, A. T. Mata, M. B. Sanvicente, and E. Acevez, "The arrival of humans on the Yucatan Peninsula: Evidence from submerged caves in the state of Quintana Roo, Mexico," *Current Research in the Pleistocene*, vol. 25, pp. 1–24, 2008.

[4] D. Rissolo, A. N. Blank, V. Petrovic, R. C. Arce, C. Jaskolski, P. L. Erreguerena, and J. C. Chatters, "Novel application of 3D documentation techniques at a submerged Late Pleistocene cave site in Quintana Roo, Mexico," in *Digital Heritage*, vol. 1, pp. 181–182, 2015.

[5] P. L. Buzzacott, E. Zeigler, P. Denoble, and R. Vann, "American cave diving fatalities 1969-2007," *International Journal of Aquatic Research and Education*, vol. 3, no. 2, p. 7, 2009.

[6] B. Joshi, M. Xanthidis, M. Roznere, N. J. Burgdorfer, P. Mordohai, A. Q. Li, and I. Rekleitis, "Underwater exploration and mapping," in *2022 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*, pp. 1–7, IEEE, 2022.

[7] K. Richmond, C. Flesher, N. Tanner, V. Siegel, and W. C. Stone, "Autonomous exploration and 3-D mapping of underwater caves with the human-portable SUNFISH® AUV," in *Global Oceans 2020: Singapore–US Gulf Coast*, pp. 1–10, IEEE, 2020.

[8] S. Exley, *Basic cave diving: A blueprint for survival*. Cave Diving Section of the National Speleological Society, 1986.

[9] K. Siddiqi, S. Bouix, A. Tannenbaum, and S. W. Zucker, "Hamilton-jacobi skeletons," *International Journal of Computer Vision*, vol. 48, no. 3, pp. 215–231, 2002.

[10] B. Yu, R. Tibbetts, T. Barua, A. Morales, I. Rekleitis, and M. J. Islam, "Weakly supervised caveline detection for auv navigation inside underwater caves," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2023.

[11] M. J. Islam, Y. Xia, and J. Sattar, "Fast Underwater Image Enhancement for Improved Visual Perception," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 3227–3234, 2020.

[12] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, Springer, 2015.

[13] "Ultralytics: YOLOv8 Docs, official website = https://docs.ultralytics.com/." Accessed: 2023-09-01.

[14] M. J. Islam, C. Edge, Y. Xiao, P. Luo, M. Mehtaz, C. Morse, S. S. Enan, and J. Sattar, "Semantic Segmentation of Underwater Imagery: Dataset and Benchmark," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE/RSJ, 2020.

[15] K. Koreitem, F. Shkurti, T. Manderson, W.-D. Chang, J. C. G. Higuera, and G. Dudek, "One-shot informed robotic visual search in the wild," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5800–5807, IEEE, 2020.

[16] M. J. Islam, R. Wang, and J. Sattar, "SVAM: Saliency-guided Visual Attention Modeling by Autonomous Underwater Robots," in *Robotics: Science and Systems (RSS)*, (NY, USA), 2022.

[17] J. Zhu, S. Yu, L. Gao, Z. Han, and Y. Tang, "Saliency-Based Diver Target Detection and Localization Method," *Mathematical Problems in Engineering*, vol. 2020, 2020.

[18] M. J. Islam, P. Luo, and J. Sattar, "Simultaneous Enhancement and Super-Resolution of Underwater Imagery for Improved Visual Perception," in *Robotics: Science and Systems (RSS)*, (Corvalis, Oregon, USA), July 2020.

[19] M. Modasshir, A. Quattrini Li, and I. Rekleitis, "Deep neural networks: a comparison on different computing platforms," in *Canadian Conference on Computer and Robot Vision (CRV)*, pp. 383–389, 2018.

[20] T. Manderson, J. C. G. Higuera, R. Cheng, and G. Dudek, "Vision-based Autonomous Underwater Swimming in Dense Coral for Combined Collision Avoidance and Target Selection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1885–1891, IEEE, 2018.

[21] Y. Girdhar and G. Dudek, "Modeling Curiosity in a Mobile Robot for Long-term Autonomous Exploration and Monitoring," *Autonomous Robots*, vol. 40, no. 7, pp. 1267–1278, 2016.

[22] Y. Girdhar, P. Giguere, and G. Dudek, "Autonomous Adaptive Exploration using Realtime Online Spatiotemporal Topic Modeling," *International Journal of Robotics Research (IJRR)*, vol. 33, no. 4, pp. 645–657, 2014.

[23] M. Modasshir, S. Rahman, O. Youngquist, and I. Rekleitis, "Coral Identification and Counting with an Autonomous Underwater Vehicle," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 524–529, Dec. 2018.

[24] M. Modasshir, S. Rahman, and I. Rekleitis, "Autonomous 3D Semantic Mapping of Coral Reefs," in *12th Conference on Field and Service Robotics (FSR)*, (Tokyo, Japan), pp. 365–379, Aug. 2019.

[25] P. P. Ray, "A review on tinyml: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, 2022.

[26] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, "A survey of modern deep learning based object detection models," *Digital Signal Processing*, vol. 126, p. 103514, 2022.

[27] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[28] 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.

[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[30] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, *et al.*, "Searching for mobilenetv3," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1314–1324, 2019.

[31] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856, 2018.

[32] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131, 2018.

[33] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," *Advances in neural information processing systems*, vol. 31, 2018.

[34] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2820–2828, 2019.

[35] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," *arXiv preprint arXiv:1908.09791*, 2019.

[36] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, and C. Xu, "Ghostnet: More features from cheap operations," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1580–1589, 2020.

[37] S. Mehta and M. Rastegari, "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer," *arXiv preprint arXiv:2110.02178*, 2021.

[38] T.-J. Yang, A. Howard, B. Chen, X. Zhang, A. Go, M. Sandler, V. Sze, and H. Adam, "Netadapt: Platform-aware neural network adaptation for mobile applications," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 285–300, 2018.

[39] B. C. Reidy, M. Mohammadi, M. E. Elbtity, and R. Zand, "Work in progress: Real-time transformer inference on edge ai accelerators," in *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 341–344, 2023.

[40] B. C. Reidy, M. Mohammadi, M. E. Elbtity, and R. Zand, "Efficient deployment of transformer models on edge tpu accelerators: A real system evaluation," in *Architecture and System Support for Transformer Models (ASSYST ISCA)*, 2023.

[41] M. Mohammadi, H. Smith, L. Khan, and R. Zand, "Facial expression recognition at the edge: Cpu vs gpu vs vpu vs tpu," in *Proceedings of the Great Lakes Symposium on VLSI 2023*, GLSVLSI '23, (New York, NY, USA), p. 243–248, Association for Computing Machinery, 2023.

[42] B. Joshi, M. Xanthidis, S. Rahman, and I. Rekleitis, "High definition, inexpensive, underwater mapping," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[43] S. Rahman, A. Quattrini Li, and I. Rekleitis, "SVIn2: A Multi-sensor Fusion-based Underwater SLAM System," *International Journal of Robotics Research*, vol. 41, pp. 1022–1042, July 2022.