Distributed Deep Learning and Energy-Efficient Real-Time Image Processing at the Edge for Fish Segmentation in Underwater Videos

MOHAMMAD JAHANBAKHT¹,², (Student Member, IEEE), WEI XIANG³, (Senior Member, IEEE), NATHAN J. WALTHAM¹, AND MOSTAFA RAHIMI AZGHADI¹,², (Senior Member, IEEE)

¹Centre for Tropical Water and Aquatic Ecosystem Research (TropWATER), James Cook University, Townsville, QLD 4811, Australia
²College of Science and Engineering, James Cook University, Townsville, QLD 4811, Australia
³School of Computing, Engineering and Mathematical Sciences, La Trobe University, Melbourne, VIC 3086, Australia

Corresponding authors: Wei Xiang (w.xiang@latrobe.edu.au) and Mostafa Rahimi Azghadi (mostafa.rahimiazghadi@jcu.edu.au)

This work was supported in part by the Australian Government through the Australian Research Council’s Discovery Projects Funding Scheme under Project DP220101634, and in part by the Australian Government Research Training Program Scholarship.

ABSTRACT Using big marine data to train deep learning models is not efficient, or sometimes even possible, on local computers. In this paper, we show how distributed learning in the cloud can help more efficiently process big data and train more accurate deep learning models. In addition, marine big data is usually communicated over wired networks, which if possible to deploy in the first place, are costly to maintain. Therefore, wireless communications dominantly conducted by acoustic waves in underwater sensor networks, may be considered. However, wireless communication is not feasible for big marine data due to the narrow frequency bandwidth of acoustic waves and the ambient noise. To address this problem, we propose an optimized deep learning design for low-energy and real-time image processing at the underwater edge. This leads to trading the need to transmit the large image data, for transmitting only the low-volume results that can be sent over wireless sensor networks. To demonstrate the benefits of our approaches in a real-world application, we perform fish segmentation in underwater videos and draw comparisons against conventional techniques. We show that, when underwater captured images are processed at the collection edge, 4 times speedup can be achieved compared to using a landside server. Furthermore, we demonstrate that deploying a compressed DNN at the edge can save 60% of power compared to a full DNN model. These results promise improved applications of affordable deep learning in underwater exploration, monitoring, navigation, tracking, disaster prevention, and scientific data collection projects.

INDEX TERMS Big marine data, deep learning, edge computing, fish image segmentation, Internet of Underwater Things, real-time video processing, U-Net convolutional neural networks.

I. INTRODUCTION

As Deep Neural Network (DNN) models grow to have billions of learning parameters, while the training data volumes expand to petabytes, model training on local computers becomes highly inefficient, if not impossible, demanding the use of Distributed Computer Systems (DCS). Besides, a well-designed DNN training on DCSs may achieve higher accuracy by disentangling the weight optimization into separated nodes [1]. On this basis, the next generation of cloud-based distributed computer networks may enable DNN processing on edge devices for both improved model training and efficient model inferencing [2], ultimately leading to enhanced decision making.

To that end, in this paper we first explore cloud-based distributed DNN training and analyze its benefits and shortcomings in training a large-scale DNN performing fish segmentation in real-world underwater videos. Next, we deploy
our trained DNN on an embedded edge processor to show the benefits it provides for the Internet of Underwater Things (IoUT) and its wireless communication technology.

Wireless communication plays an important role in all branches of the Internet of Things. However, this type of communication faces many challenges, when it comes to the IoUT [3]. These challenges include high attenuation, multipath fading, frequency dispersion, and signal distortion of electromagnetic waves, which cannot penetrate and propagate deep in underwater environments.

These harsh underwater conditions led scientists toward the Underwater Acoustic Sensor Network (UASN), which by far is the dominant wireless technology in IoUT [3]. UASN is defined as sonic interconnection of marine objects that enables maritime exploration and monitoring activities. However, UASN has some limiting technical characteristics such as low transmission bandwidth, high signal attenuation, and high propagation delays [4].

To overcome these UASN drawbacks, the relatively new concept of edge processing seems to provide a promising solution [3]. In edge computing, endpoint devices perform parts of the required computations on their own data. The results of these computational processes have smaller volumes, compared to the raw input data. By sending the results instead of the initial unprocessed data, underwater network traffic will significantly reduce. Meanwhile, the lower data transmission rate will consequently result in lower communication latency and efficient energy consumption [5].

To get the most out of the edge computing, efficient computational processes must be employed. This is specially important when dealing with the marine high-resolution image and real-time video data streams. To address this problem, modern deep learning processes can be used. State-of-the-art use cases of DNNs in underwater image/video applications are ranging from image enhancement [6] to object detection and classification [7], [8], and further to the vision-based undersea navigation and tracking [9].

However, DNNs usually consist of large architectures requiring large computational resources and power consumption, which are not readily available at the smaller edge. To overcome this challenge, here we propose two strategies. The first is optimizing and compressing the DNN models deployed at the edge. We show that this can result in significant power saving and reduced processing time when compared to remote processing on land. The second strategy we suggest is to use practical approaches to reduce power consumption or harvest environmental energy to power the edge device.

To summarize, in this paper we use cloud-based distributed DNN training and edge DNN inferencing for a real-world underwater image processing task. This area has not been widely explored [10], and deserves further research. To fill this gap,

- We propose, to the best of our knowledge, the first DNN-based edge processor for real-time fish segmentation in remote underwater videos, where no cable nor underwater vehicle is involved. This is implemented on an embedded Graphics Processing Unit (GPU) and is benchmarked against traditional HTTP inference on a GPU-powered computer on land.
- The fish segmentation model is trained on distributed cloud infrastructure to make it more suitable for big marine data training, while also slightly improving its learning accuracy. The distributed training is benchmarked against an on-premises standalone computer, in terms of speed and accuracy.
- To improve the energy requirement of our system in the targeted underwater environment, we compress our DNN model to two quantized weight resolutions. We benchmark these against the full model and show the compressed networks can result in significant improvements. In addition, we propose an efficient energy management plan, consisting of renewable energy resources and motion detection technologies.

We discuss how the proposed underwater edge computing platform may improve the big data processing barrier in the Internet of Underwater Things.

To better illustrate the contributions of this article, the amalgamation of its diverse components into a state-of-the-art IoUT application is illustrated in Fig. 1. This figure shows the logical flow from accurate model training to low-energy model deployment, and further to GPU-based high-speed underwater video inferencing. The proposed technology can involve as many high-quality cameras as needed. Besides, the
FIGURE 2. Modified architecture of the U-NET convolutional neural network for underwater image segmentation. The weights of the encoder part are transfer-learnt from a pretrained MobileNetV2 on the ImageNet. The decoder part employs Pix2Pix for upsampling.

similar GPU-powered edge technology with DNNs enabled IoUT to unlimitedly expand and involve real-time analysis of sensors and hydrophones timeseries.

The rest of this article is organized as follows. In Section II, a modified U-Net model will be designed for fish segmentation in real-life underwater images. This model will be accurately trained on a cloud-based DCS in Section III. Later in Section IV, energy reduction techniques will be introduced, and the trained model with compressed weights will be deployed on a GPU-enabled edge device for fast and efficient inferencing. We will also investigate how edge computing makes underwater big data processing possible. The paper is concluded in Section V.

II. FISH SEGMENTATION

Image segmentation is a major topic in computer vision, which assigns a label to each pixel of the image. Pixels with the same label belong to the same semantic object (i.e., class). Image segmentation is applied in several domains including object localization, video surveillance, scene understanding, augmented reality, and many other image processing applications. To have an accurate image segmentation model, a wide verity of DNNs have been reported in the literature, and they are comprehensively surveyed by Minaee et al. [11].

However, the underwater environment is not image-friendly. The suspended particles in seawater absorb, backscatter, and forward scatter light rays towards the camera, which creates hazy images with low contrast and faded colors [6]. Besides, seawater reacts differently to different light spectra, according to its frequency-dependent power absorption. These and many other extreme imaging limitations make the underwater environment a nonuniform imaging medium [12]. These nonuniform image degradation processes make underwater image segmentation a challenging task.

To address this problem, some authors tend to use statistical models. For example, Kannan [13] has attempted to recognize objects in seawater images using a Gaussian mixture model in combination with optimization methods like the genetic algorithm. Although these traditional inner distance shape matching techniques have made some progress in certain specific situations, they usually have lower underwater image recognition accuracy, when compared to the most recent deep learning algorithms.

One of the most recent works in underwater image segmentation combines the multi-scale Retinex image enhancement algorithm and the Mask R-CNN framework to recognize marine echinoderm [12]. While it is limited to echinus, holothurian, and starfish, the work in [12] can be expanded to incorporate more sea creatures. Besides, the trained model is big and consumes high energy for inferencing. This makes it suitable for deployment on land-side servers for non-real-time applications. However, for real-time underwater video inference, we need a lighter model with lower energy demand that better suits our edge device.

A. MODIFIED U-NET ARCHITECTURE

To address the fish segmentation problem in underwater images, we employed a modified version of the U-Net model [14]. U-Net is a Convolutional Neural Network (CNN)-based deep learning algorithm which has been applied in a number of applications ranging from medical imaging [15], to fish segmentation [16]. Similar to the original U-Net [15], modified U-Net in this paper consists of two consecutive encoder and decoder paths. While the contracting encoder path captures contextual features, the symmetric and expanding decoder path enables segment localization. However, our modified U-Net makes use of a better upsampling block [14].

Both the encoder and decoder parts of the modified U-Net architecture are illustrated in Fig. 2. The encoder part follows the typical CNN architecture with five consecutive convolution layers, each followed by a ReLU activation function and a max-pooling operator. The $7 \times 7 \times 320$ output of
the encoder part feeds to the following decoder part, which consists of four consecutive upsampling and concatenating blocks. Unlike the original U-Net, a Pix2Pix block [17] is used for upsampling in the modified U-Net. Pix2Pix was initially used in the generative adversarial networks for their capability of generating high-quality images across a variety of image translation tasks [17]. Since then, it is widely used as an upsampling block in a wide range of other applications. At the last stage of the modified U-Net model, a simple 2D convolution layer was used to map the extracted 64 features to the desirable two segmentation classes (i.e., fish or not fish).

To reduce the number of trainable parameters in our modified U-Net to improve its training speed, we used the transfer learning technique and froze the transferred weights. To elaborate, the weights and biases of a pretrained MobileNetV2 model on the ImageNet dataset were reused in the encoder part [18].

B. DeepFish DATASET

Supervised DNNs are data hungry models. They require hundreds of labeled data in each class, to succeed in their training phase. A successful training will result in accurate predictions at the subsequent inferencing phase. To train our modified U-Net, the open-access DeepFish dataset was used. DeepFish is a realistic dataset of fish images in their natural habitat [19]. The images have been accurately labeled for classification, localization, and segmentation algorithms evaluation.

DeepFish’s segmentation dataset contains 620 images with accurately labeled fish segments. These images are collected from 20 habitats in the tropical Australian marine environment with and without fish presence. To avoid overfitting, 10% of all images are separated for validation. The images with one or more fish have been augmented by 180° rotation, X and Y flipping, blue and green color degradation, and random noise addition. To have a balanced dataset, the images with no fish are not augmented. Furthermore, the DeepFish images are originally shaped in a 1920 × 1080 pixel rectangle. To feed these images into our modified U-Net model, they are resized into a square shape of 224 × 224 pixels. This resizing is done by 49 zero-padded pixels both on top and bottom of each image.

III. DISTRIBUTED TRAINING

Not so long ago, labeled datasets for supervised machine learning were scarce. However, the number of these expert-labeled and open-access datasets have recently been increased by the order of multiple hundreds. For example, none of the tens of labeled underwater datasets introduced in the GitHub repository Awesome Underwater Datasets, existed a few years ago.

With labeled image proliferation in open-access datasets, processing them becomes more challenging and computation-demanding. More specifically, the pixel labeling algorithms (like segmentation DNNs) with petabytes of remotely gathered data, such as underwater fish videos, will require significant computation and memory resources, which are not available on a single computer. Here, Distributed Computer Systems (DCS) with several processing nodes that share the training, and consequently the memory and computing workload, can be considered [20].

Training DNNs on DCSs have led to the relevant concepts of centralized and decentralized deep learning [21], which has attracted significant attention [22]. Centralized deep learning can be divided into data and model parallelization [23], while the decentralized deep learning involves federated learning [21] and fully decentralized swarm learning [24]. Both the federated and swarm learning is privacy-preserving data processing paradigms, which are more suitable in sensitive sectors like healthcare and finance. Both solutions conduct machine learning on edge devices, within a network of nodal clients [22]. This makes them unsuitable for underwater edge devices with low access to energy resources.

By contrast, the data and model parallelization technique tends to use a simple network of land-side computers for machine learning [23]. In data parallelization, distributed learning is conducted by dividing big data into mini-batches. During the training phase, an allreduce operation is employed to average the mini-batch gradients over the entire DCS nodes [23]. However, data parallelization suffers from multiple drawbacks, including

- The DNN model in question must be small enough to fit in a computing device, i.e., a CPU or GPU.
- Training large image segmentation models on high-resolution images can cause diverging gradients, because of small mini-batch sizes.

Considering the above drawbacks, one may use model parallelization as an alternative solution, which breaks the DNN into small partitions and distributes partitions over DCS nodes for training [23]. Overall, model parallelization is the chosen DCS paradigm in this work. Since the volume of the DeepFish dataset used in our work is only 15 GB and our modified U-Net (including ImageNet pretrained MobileNetV2 model) size is 65 MB, the data volume is not a major concern in our chosen task.

However, our DNN can still validate the DCS training concept and benefit from distributed learning by increasing its accuracy. To elaborate,

- DNNs are directed-acyclic-graphs that have unknown weights on the graph vertices. In model parallelization, these vertices will disentangle in distributed computing, and the weights of every vertex will be independently trained on a dedicated computer node [25]. This weight separation will reduce overfitting on the training dataset, and will consequently increase the overall DNN accuracy.
- Each computer node in a DCS collects a random set of images into a batch. The batch has a fixed size and is used as a single step in model training. A larger batch size helps the model optimizer find the global convergence point faster [26]. In one step, if a single computer uses \( N \) random images to form a batch, then \( M \)
TABLE 1. Hardware specifications of the utilized local computer, distributed computers, and the edge device.

|                  | CPU         | GPU                  | RAM | SSD | Price   |
|------------------|-------------|----------------------|-----|-----|---------|
| **Local computer** | Intel 8-core | Nvidia GeForce (1.6 GHz, 8 GB, and 3071 CUDA cores) | 32 GB | 1 TB | $3500   |
| **Distributed computers**: AWS M5.2xLarge Instances | Intel 8-core | — | 32 GB | 61 GB | $0.576 per hour |
| **Edge processor**: Nvidia Jetson Nano | ARM 4-core | Nvidia Maxwell (1.1 GHz, 4 GB, and 2048 CUDA cores) | 4 GB | 64 GB | $100    |

computers of a DCS would use $M \times N$ random images. As a result, more computers in the DCS will result in a bigger batch size per step, which will consequently result in better convergence towards a more accurate model.

In addition, our work presents a proof-of-concept and introduces a platform for use in training distributed deep learning models for large marine data.

A. CLOUD-BASED DISTRIBUTED COMPUTER SYSTEMS

A DCS is a private network of separated computers, each holding a set of software components that collaboratively work as a single system. Despite their many benefits, DCSs need complex experimental and architectural design procedures for distributed operation algorithms, node counts in the system, initial establishment, and continuous maintenance.

To address these difficulties, commercial cloud services like Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform seem to offer a promising solution. For instance, AWS SageMaker is a cloud-based machine learning platform for running a customer’s DNN training, as well as inferencing jobs on a dedicated DCS.

To perform this study and to evaluate the performance of cloud-based DCS for our proposed application, 20 identical training computers (i.e., instances) were rented from the AWS SageMaker. Hardware specifications of each instance are compared in Table 1 with a GPU-powered on-premises computer. The 61 GB capacity of the cloud-based Solid-State Drives (SSD) were enough for the requirements of this research.

It is worth noting that AWS SageMaker offers P3 instances with high-end GPUs in the cloud. These instances are capable of distributed training of DNN models across hundreds of GPUs. However, these instances are significantly more expensive than ordinary CPU-based computers. For example, a P3.2xLarge SageMaker instance with similar specifications to the employed M5.2xLarge in this study, but with one Tesla GPU, will cost $3.825 per hour. This is nearly 7 times the cost of our current setup. Therefore, to have an affordable DCS training system, we chose to use CPU-based computers.

B. DISTRIBUTED TRAINING RESULTS

The modified U-Net model in Fig. 2 accepts a $224 \times 224$ RGB image at input, and returns two $224 \times 224$ integer values for the two possible pixel classes. In other words, the DNN assigns two integer values to every single pixel of the image. These integer values indicate whether each pixel belongs to the body of a fish or not. During inference, an $\text{argmax}(\cdot)$ operator must be employed to produce a single $224 \times 224$ channel with 1 and 0 values for the fish and not-fish classes. However, this is not the case in the training phase, where the exact integer outputs are passed to the Adaptive Moment (Adam) optimizer of the DCS processor.

The Adam optimizer in DCS uses the Sparse Categorical Cross Entropy (SCCE) as its major loss metric for accurate underwater fish segmentation. In addition to SCCE, the Sparse Categorical Focal (SCF) loss values with $\gamma = 2$ is also measured. SCF is particularly useful in our study, where the foreground fish pixels in each image is densely located against the huge number of background pixels [27]. To elaborate, the labeled segmentation masks are mostly zero-valued for the no-fish class, with occasional occurrences of dense one-values for the fish class.
### TABLE 2. SCCE and SCF loss metrics, as well as SCCA and IoU accuracy metrics of the modified U-NET model, which has been trained on a local computer, compared with distributed computer systems with 10, 15, and 20 computer nodes.

| Training venue                  | SCCE  | SCF   | SCCA   | IoU   | Rent   |
|---------------------------------|-------|-------|--------|-------|--------|
| Locally on one GPU-powered computer | 0.065 | 4.65  | 98.78% | 74.5% | N/A    |
| Distributed on 10 CPU-based computers | 0.092 | 7.07  | 98.72% | 74.1% | $9.7   |
| Distributed on 15 CPU-based computers | 0.089 | 6.77  | 98.73% | 84.7% | $14.3  |
| Distributed on 20 CPU-based computers | **0.053** | **3.48** | **98.81%** | **87.6%** | **$19.1** |

The convergence of the SCCE loss metric for both the distributed computers and the local computer are compared in Fig. 3. Here, the DCS spends a lot of time on HTTP data transaction between its nodes. This makes the overall DCS convergence slower than a GPU-powered local computer.

However, the trained model on DCS achieves a lower loss compared to a single GPU training. The SCCE loss metric, SCF loss, Sparse Categorical Crossentropy Accuracy (SCCA) and the Intersection over Union (IoU) accuracy are benchmarked in Table 2 for both the distributed- and the locally-trained DNNs. The values in this table are calculated for the validation dataset. This contrasts with the plots in Fig. 3, which are for augmented train and augmented validation datasets to artificially increase the amount of training data, without actually collecting new data.

As explained earlier, it is expected that due to the DNN weight separation and independent training on dedicated computer nodes [25], as well as bigger batch sizes in DCSs [26], the model is better trained on bigger DCSs, achieving a lower loss and higher accuracy. This is confirmed in the results shown in Table 2. While the SCCA remains almost the same with only slight improvement for 20 computer nodes, both the SCCE and SCF loss metrics and the IoU accuracy metric show better performance for the distributed training scheme with more computer nodes (i.e., 20).

The SCCE and SCF loss metrics in Table 2 also suggest that the training algorithm in DCSs needs improvement, for having trouble on achieving the optimum convergence point, compared to a single computer. However, this drawback can be compensated by increasing the number of nodes in the DCS, which will consequently decrease overfitting and increase the effective batch size, as discussed in Section III. It is worth mentioning that the total renting price for 20 AWS SageMaker training instances was AUD 19.1. To summarize,

- Table 2 also shows that the IoU accuracy of 20-node DCS (87.6%) is 15% better than a standalone computer (74.5%).
- Due to the HTTP transactions in DCS, a GPU-powered local computer is faster than 20 distributed computers without GPU. This was illustrated in Fig. 3.

### IV. UNDERWATER EDGE COMPUTING

Edge computing is an information technology paradigm that brings computation closer to the data collection hardware. In the case of IoUT, edge computing is very beneficial, without which the processing should be performed on local computers or centralized clouds on land. Transferring the data to these remote land processors would require wideband data transactions, which is not readily available or sometime feasible in IoUT. In contrast, by processing raw data on edge, only the low-volume results should be communicated. Communicating only the results consumes narrower bandwidth (data-rate) and requires shorter transaction latency, making it suitable for IoUT [3].

Additionally, engaging the edge devices in data processing, can shift the load from a single centralized processing point to numerous distributed nodes. In this case, the system would not have a single point of failure. Besides, advanced applications such as prompt decision making will be feasible [28]. These advantages of the edge computing are better illustrated in Fig. 4, where the traditional multi-Mbps wired network is replaced by a multi-mps UASN using the edge processing technology. In other words, the edge processing enables UASNs to quickly submit the DNN output results with a very
short delay. However, edge computing in IoUT must tackle the challenge of limited underwater energy resources, as its main drawback.

A. ENERGY MANAGEMENT AT THE EDGE

 Sustainable power cannot be readily delivered to the underwater edge devices. Therefore, wired energy transferring might be considered, which is a robust but limiting solution. Using wired energy will limit IoUT sensor deployment to the shoreline vicinity, and will make the system vulnerable to cable damages by gnawing animals and corrosive environments.

 Due to these drawbacks, the use of wired energy in remote IoUT applications becomes infeasible. To address this challenge, here we propose two parallel techniques to effectively manage the energy at the edge, as shown in Fig. 5 and described below.

1) REDUCING POWER CONSUMPTION AT THE EDGE

 Energy management from the edge’s perspective means lower computation demand and/or more power-efficient consumption. To reduce the demand, ultrasonic motion detectors can be used to only demand imaging and processing when there is movement in the environment, putting the system into low-power sleep mode at all other times. This, of course, adds to the entire system power, the amount consumed by the submersible echosounder transducers, which depends on the quality of their coverage. A typical 115 kHz ping transceiver with 50 m directional range, 0.5% resolution, and 300 m depth rating that operates for 100 milliseconds per every second will take no more than 50 mW.

 However, both sonar-based and vision-based motion detection technologies are sensitive to cavitation (bubbles), vegetation (leaves, sticks, etc.), water depth, and current speed. Consequently, using them in high-dynamic conditions require artificial passageways with careful control of the environmental parameters [29].

 The fabrication and maintenance of passageways for underwater motion detection will add extra cost and time to the project, which is not desirable. Therefore, where possible, other power reduction techniques should be considered. A technique that we investigated in this paper is utilizing a compressed DNN, which consumes less power, due to lighter computations performed at the edge. In this way, the trained weights of the initial DNN in Table 2 that have single-precision (32-bit) floating-point resolution (FP32), can be compressed. However, this compression may lead to a slight decrease in the DNN accuracy.

 To experiment with compressed quantized DNNs in our performed segmentation task, we employed TensorFlow Lite (TFLite) to conduct the following model quantizations:

- **Half-precision (16-bit) floating-point (FP16):** This post-training compression reduces the floating-point size to the IEEE standard float16.
- **8-bit integer (Int8):** This dynamic range compression quantizes the weights from floating-point to integers with 8 bits precision.

 To analyze the DNN weight quantization effect on its energy demand, the Nvidia Jetson nano embedded GPU was selected as our edge device. The average power consumption of our segmentation DNN models with FP16 and Int8 compressed network parameters are compared in Table 3, with the original FP32 model. The System in this table refers to

| Compression level               | Power consumption [mW] |
|---------------------------------|------------------------|
|                                 | System | GPU | Total  |
| Rest mode, with no ongoing processes |        |     |        |
| Single-precision (FP32)         | 1275   | 42  | 1317   |
| Half-precision (FP16)           | 1315   | 45  | 1360   |
| 8-bit integer (Int8)            | 1311   | 44  | 1355   |
| Busy mode, with ongoing real-time video processing |        |     |        |
| Single-precision (FP32)         | 5135   | 2373| 7508   |
| Half-precision (FP16)           | 2900   | 48  | 2948   |
| 8-bit integer (Int8)            | 2888   | 47  | 2935   |
**FIGURE 6.** Example outputs of the modified U-NET segmentation model trained on 20 cloud-based distributed computers of the AWS SageMaker, and then deployed with Int8 parameter resolution on a Jetson Nano device.
the edge device internal components, except GPU (i.e., CPU, RAM, SSD, high-speed IO, etc.).

In the rest mode with no GPU tasks, the TFLite backend of both the FP16 and Int8 demand slightly more energy (around 3%) than TensorFlow backend of FP32. This is because, TFLite does not optimize model size, compared to the FP32 case trained using TensorFlow. Therefore, the edge device requires larger storage both on its system and on its GPU. Besides, the mobile-friendly TFLite libraries do not support all TensorFlow operators. Consequently, some CUDA mounting commands might run inefficiently at the edge.

On the other hand, in the busy mode with heavy GPU inferencing tasks, the proposed FP16 and Int8 DNNs demand significantly lower power. The results in Table 3 show 61% total power reduction compared to uncompressed FP32 model. This improvement can eliminate the need for motion detection technologies, by enabling the edge device to continuously process underwater video frames. Alternatively, if possible, it can be combined with the motion detection methods for further decreased power consumption.

Despite the fact that weight compression reduces power demand at the edge, it can also decrease the DNN accuracy. This concern has been investigated in Table 4, where the loss and accuracy metrics of the compressed FP16 and Int8 models are compared with the original FP32 DNN. According to this table, the reduction in model performance (specially for the IoU metric) is a direct consequence of model’s optimum weight replacement, due to weight compression. Here, the advantage of model training on DCSs is better seen. In other words, the reduced performance of the compressed model in Table 4 is still comparable with single GPU training scheme in Table 2.

Despite the reduction techniques proposed above, the power consumption of the edge processor can be still significant and in the order of a few watts, requiring a sustainable method of power delivery to the edge device. Here, we propose combining the above-mentioned reduction techniques with several energy harvesting approaches described below, as shown in Fig. 5, to make IoUT edge processing feasible.

### Table 4. SCCE, SCF, SCCA, and IoU metrics of the original segmentation DNN with FP32 resolution, compared with FP16 and Int8 compressed resolutions of the same DNN.

| Resolution | SCCE   | SCF    | SCCA  | IoU    |
|------------|--------|--------|-------|--------|
| Single-precision floating-point (FP32) | 0.053  | 3.48   | 98.8% | 87.6%  |
| Half-precision floating-point (FP16)   | 0.053  | 3.59   | 98.8% | 72.4%  |
| 8-bit integer (Int8)                   | 0.054  | 3.61   | 98.8% | 71.9%  |

2) ENERGY HARVESTING
In undersea energy harvesting, solar panels and the marine grade Absorbent Glass Mat (AGM) or Li-ion batteries are well-developed for small-scale applications. For example, today’s rechargeable Li-ion and AGM batteries can provide superior power to support the electrical demands of start-stop edge devices, while being affordable, impervious to seawater, resilient to underwater vibrations, and maintenance-free [30].

Tidal stream energy harvesting systems are another source of power in undersea environments. While multi-megawatt tidal stream turbines can feed power grids with clean energy, small-scale submersible generators [31] and tidal kites [32] can provide efficient power for UASNs in our proposed application. In addition, tidal power is almost perfectly predictable over long timescales, which is an appealing feature for power management systems. The output energy during neap tides is significantly less than that during spring tides, hence, in a realistic system with steady output power, rechargeable batteries are inevitable for covering the lower energy production periods [31].

When the abovementioned energy harvesting methods are combined with the proposed power reduction at the edge, inference can happen efficiently at the edge. However, the underwater edge inference results still need to be submitted overwater for future records and further actions. This requires an energy-efficient communication technique.

3) ENERGY-EFFICIENT UNDERWATER COMMUNICATIONS
After successful implementation of any energy harvesting technique from Section IV-A2, the infrastructure should undertake an energy allocation plan for every underwater acoustic node. The main goal of this plan is to maximize the delivered data over a given time slot [33]. However, there are many parameters that influence this goal.

For example, both the dynamic nature of underwater acoustic channels and the time-varying characteristics of energy harvesting sources should be considered as stochastic finite-state machines (a.k.a. finite-state Markov chain) [34]. One obvious consequence of this stochasticity is the fact that the transmitter might receive channel state information intermittently, or not receiving them at all. While the traditional solutions like stochastic dynamic programming [33] can fairly address this uncertainty, modern energy allocation plans with DNNs are highly desirable. One approach is to borrow the main ideas from the well-developed Internet of Things (IoT) and use them in the relevant sectors of Internet of Underwater Things (IoUT) [3], [35].

However, there are many fundamental radio and acoustic propagation differences between IoT and IoUT that must be considered in every application. This might even require new routing protocol development and new network management schemes. In this regard, previous techniques such as [36] to design a balanced routing protocol could be employed. The DNN-based design of [36] suits the low-energy requirements of our application, while maintaining a low propagation...
latency and considering the void area issue. The communication protocol in [36] is based on the magnetic induction technique, which has steady channel, predictable response, and low propagation delay.

**B. EDGE INFERENCE RESULTS**

To further analyze the benefits of edge processing compared with transferring the underwater video frames to land for processing, we performed some further inference analyses. To perform these analyses, we needed to choose a GPU-enabled edge device to infer video frames on our proposed modified U-NET DNN architecture. Although, a variety of edge (embedded) GPUs exist in the market, the Nvidia Jetson Nano from the Jetson processor family, which is the world’s leading platform for machine learning at the edge [10] was selected. Nano’s specifications are shown in Table 1. The Jetson Nano module is an efficient minicomputer that runs on Linux for Tegra (L4T). L4T is a free distribution of the Ubuntu Linux by Nvidia for its Tegra processor series.

To utilize Jetson Nano for our application, we installed Python, TensorFlow, and many other software packages on L4T to enable fast and accurate DNN inferencing. Additionally, we developed a Flask-based REST-API in L4T to answer remote HTTP queries. Flask is a light-weight microframework that suits the low-power requirements of an edge computing ecosystem.

As described earlier in Sections III and IV-A, our proposed modified U-NET was initially trained on 20 cloud-based distributed computers. Next, the trained model was compressed to Int8 resolution for power reduction. To test the functionality of our developed framework, we deployed it on a Jetson Nano to perform fish segmentation on a number of video frames from the DeepFish dataset. Fig. 6 shows a few sample segmentation outputs. The demonstrated samples in Fig. 6 include some desirable true-positive and true-negative predictions, along with two undesirable false-positive detections. We did not encounter any false-negative report on the images in the DeepFish dataset.

**C. EDGE INFERENCE SPEED**

The inferencing speed on the Jetson Nano edge device was compared to an on-premises land server in Fig. 7. The first frame inferencing in both the land-side server and the FP32 edge device requires much longer time than the TFLite-based FP16 and Int8. This low initialization speed is due to the model storage method in TensorFlow, which requires more time to map a saved DNN model into a directed-acyclic-graphs in GPU.

In the case of on-premises GPU-powered server in Fig. 7, multi-MB video frames must be submitted via a wired LAN connection first to then be processed on the GPU. The transmission significantly increases the inference time as shown in the figure. In contrast, the edge device processes video frames in-place, producing few bytes of results that usually do not require instant wireless transmission. These results may include \((x, y)\) coordinate of the detected fish, its size, and the frame timestamp that are submitted to the land station once in a while.

The GPU-powered landside server with better computing resources (see the first row of Table 1) is 4 times slower than the edge embedded Nano GPU. Meanwhile, the model compression to FP16 and Int8 reduces inference latency at the edge by 18%, compared to the FP32 model. This lower latency means that the edge processor can segment 3-4 times more frames (images). It is worth noting that, reducing latency also reduce the overall energy consumption of the system. Of course, if faster processing rates are required, more powerful but also expensive devices like Jetson Xavier can be employed.

Overall, the presented results show that the proposed methodologies for underwater edge inferencing can significantly reduce the latency and power requirements of any underwater DNN processing tasks, compared to transferring the large data to land for processing. This can significantly advance IoUT ecosystems in various applications ranging from marine ecological studies to disaster prevention.

**V. CONCLUSION**

In this paper, we first proposed a modified U-NET architecture for fish segmentation in underwater videos. We then demonstrated the use of DCS for training our fish segmentation task and discussed the benefits DCS provides for training DNNs using big marine data. In the second part of our paper, we utilized our DCS-trained DNN to show the benefits of edge processing in underwater environments. It was shown that edge processing can result in more than 4 times speedup, compared to the conventional method of remote land processing. Furthermore, we proposed an energy management plan at the edge, which utilized well-known techniques such as motion-detection, energy harvesting, and compressed DNNs to make underwater edge computing feasible. We showed that, by compressing a DNN model for fish segmentation, 61% of power saving can be achieved compared to a FP32 DNN.
The techniques proposed in this paper can be applied in other domains, most notably, in other underwater applications, where sending large data to land for processing is not feasible. These simple but practical techniques can significantly advance current and future underwater processing applications, leading to more informed decisions in remote underwater environments, for applications such as marine research, navigation, and tracking.

REFERENCES

[1] S. Gupta, W. Zhang, and F. Wang, “Model accuracy and runtime tradeoff in distributed deep learning: A systematic study,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Barcelona, Spain, Dec. 2016, pp. 171–180.

[2] M. Chen, D. Gunduz, K. Huang, W. Saad, M. Bennis, A. V. Feljan, and H. V. Poor. “Distributed learning in wireless networks: Recent progress and future challenges,” IEEE J. Sel. Areas Commun., vol. 39, no. 12, pp. 3579–3605, Dec. 2021.

[3] M. Jahanbakht, W. Xiang, L. Hanzo, and M. R. Aghzahi, “Internet of underwater things and big marine data analytics—A comprehensive survey,” IEEE Commun. Surveys Tuts., vol. 23, no. 2, pp. 904–956, 2nd Quart., 2021.

[4] X. Zhong, F. Ji, F. Chen, Q. Guan, and H. Yu. “A new acoustic channel interference model for 3-D underwater acoustic sensor networks and throughput analysis,” IEEE Internet Things J., vol. 7, no. 10, pp. 9930–9942, Oct. 2020.

[5] R. Khali, G. Premkumar, G. Ilahi, M. D. Francesco, A. Mehrabi, and A. Ylä-Jääski. “Video caching, analytics, and delivery at the wireless edge: A survey and future directions,” IEEE Commun. Surveys Tuts., vol. 23, no. 1, pp. 431–471, 1st Quart., 2021.

[6] L. Chen, Z. Jiang, L. Tong, Z. Liu, A. Zhao, Q. Zhang, J. Dong, and H. Zhou. “Perceptual underwater image enhancement with deep learning and physical priors,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 7, pp. 3078–3092, Aug. 2021.

[7] X. Qin, X. Luo, Z. Wu, and J. Jiang. “Optimizing the sediment classification of small side-scan sonar images based on deep learning,” IEEE Access, vol. 9, pp. 29416–29428, 2021.

[8] I. H. Laradji, A. Saleh, P. Rodriguez, D. Nowrouzzahrabi, M. R. Aghzahi, and D. Vazquez. “Weakly supervised underwater fish segmentation using affinity LCFCN,” Sci. Rep., vol. 11, no. 1, pp. 1–10, Dec. 2021.

[9] J. E. Almanza-Medina, B. Henson, and Y. V. Zakharov. “Deep learning architectures for fish classification using forward looking sonar images,” IEEE Access, vol. 9, pp. 33880–33896, 2021.

[10] L. Wang, X. Ye, H. Xing, Z. Wang, and P. Li. “YOLO Nano Underwater: A fast and compact object detector for embedded device,” in Proc. Global Oceans, Biloxi, MS, USA, Oct. 2020, pp. 1–4.

[11] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos. “Image segmentation using deep learning: A survey,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 7, pp. 1–22, Feb. 2021.

[12] S. Song, J. Zhu, X. Li, and Q. Huang. “Integrate MSRCR and mask R-CNN to recognize underwater creatures on small sample datasets,” IEEE Access, vol. 8, pp. 172848–172858, 2020.

[13] S. Kannan. “Intelligent object recognition in underwater images using evolutionary-based Gaussian mixture model and shape matching,” Signal, Image Video Process., vol. 14, pp. 877–885, Jul. 2020.

[14] TensorFlow Tutorials. (Aug. 2021). Image Segmentation. [Online]. Available: https://www.tensorflow.org

[15] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional networks for biomedical image segmentation,” in Proc. Med. Image Comput. Comput.-Assist. Intervent. (MICCAI), Berlin, Germany, 2015, pp. 234–241.

[16] A. Saleh, M. Sheaves, and M. R. Aghzahi. “Computer vision and deep learning for fish classification in underwater habitats: A survey,” 2022, arXiv:2203.06951.

[17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-to-image translation with conditional adversarial networks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2017, pp. 1125–1134.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. “MobileNetV2: Inverted residuals and linear bottlenecks,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Salt Lake City, UT, USA, Jun. 2018, pp. 4510–4520.

[19] A. Saleh, I. H. Laradji, D. A. Konovalov, M. Bradley, D. Vazquez, and M. Sheaves. “A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis,” Sci. Rep., vol. 10, no. 1, pp. 1–10, Sep. 2020.

[20] S. Lee, H. Kim, J. Park, J. Jang, C.-S. Jeong, and S. Yoon. “TensorLightning: A traffic-efficient distributed deep learning on commodity spark clusters,” IEEE Access, vol. 6, pp. 27671–27680, 2018.

[21] S. A. Rahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Tahli, and M. Guizani. “A survey on federated learning: The journey from centralized to distributed on-site learning and beyond,” IEEE Internet Things J., vol. 8, no. 7, pp. 5476–5497, Apr. 2021.

[22] Y. Sun, H. Ochiai, and H. Esaki. “Decentralized deep learning for multi-access edge computing: A survey on communication efficiency and trustworthiness,” IEEE Trans. Artif. Intell., early access, Dec. 10, 2021, doi: 10.1109/TAI.2021.3133819.

[23] Amazon Web Services. (Jun. 2022). Amazon SageMaker Distributed Training Libraries. [Online]. Available: https://aws.amazon.com

[24] S. Warnat-Herresthal, H. Schultz, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, and S. Klena. “Swarm learning for decentralized and confidential clinical machine learning,” Nature, vol. 594, no. 7862, pp. 265–270, 2021.

[25] D. Lunga, J. Gerrand, L. Yang, C. Layton, and R. Stewart. “Apache spark accelerated deep learning inference for large scale satellite image analytics,” IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens., vol. 13, pp. 271–283, 2020.

[26] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le. “Don’t decay the learning rate, increase the batch size,” 2017, arXiv:1711.04089.

[27] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal loss for dense object detection,” in Proc. IEEE Int. Conf. Comput. Vis., Venice, Italy, Oct. 2017, pp. 2980–2988.

[28] H. Ping, H. Aiping, and T. Linwei. “Deep learning-based decision-making model for the submarine evade movement,” in Proc. Oceans, San Diego, CA, USA, Sep. 2021, pp. 1–6.

[29] M. Farzaikhooh, R. T. Kingsford, I. M. Suthers, P. Geelan-Small, J. H. Harris, W. Peisron, and S. Felder. “Attracting juvenile fish into tube fishways—Roles of transfer chamber diameter and flow velocity,” Ecol. Eng., vol. 176, pp. 106544–106559, Mar. 2022.

[30] V. B. Román, G. A. E. Banos, C. A. Q. Solis, M. I. F. Ba ¯nuelos, M. Rivero, and M. A. E. Soberanis. “Comparative study on the cost of hydrid energy and energy storage systems in remote rural communities near Yucatan, Mexico,” Appl. Energy, vol. 308, Feb. 2022, Art. no. 118334.

[31] A. Roberts, B. Thomas, P. Sewell, Z. Khan, S. Balmain, and J. Gillman. “Current tidal power technologies and their suitability for applications in coastal and marine areas,” J. Ocean Eng. Mar. Energy, vol. 2, no. 2, pp. 227–245, May 2016.

[32] E. Ackerman. (Apr. 2021). Underwater Manta Kites for Tidal Power Harvesting. [Online]. Available: https://spectrum.ieee.org

[33] L. Jing, C. He, J. Huang, and Z. Ding. “Energy management and power allocation for underwater acoustic sensor network,” IEEE Sensors J., vol. 17, no. 19, pp. 6451–6462, Oct. 2017.

[34] E. Cui, D. Yang, H. Zhang, and M. Gidlund. “Improving power stability of energy harvesting devices with edge computing-assisted time fair energy allocation,” IEEE Trans. Green Commun. Netw., vol. 5, no. 1, pp. 540–551, Mar. 2021.

[35] B. Zhao and X. Zhao. “Deep reinforcement learning resource allocation in wireless sensor networks with energy harvesting and relay,” IEEE Internet Things J., vol. 9, no. 3, pp. 2330–2345, Feb. 2022.

[36] L. Alsalman and E. Alotaibi. “A balanced routing protocol based on machine learning for underwater sensor networks,” IEEE Access, vol. 9, pp. 152082–152097, 2021.
WEI XIANG (Senior Member, IEEE) received the B.Eng. and M.Eng. degrees in electronic engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 1997 and 2000, respectively, and the Ph.D. degree in telecommunications engineering from the University of South Australia, Adelaide, Australia, in 2004. He was the Foundation Chair and the Head of the Discipline of IoT engineering at James Cook University, Cairns, Australia. He is currently the Cisco Research Chair of AI and the IoT and the Director of the Cisco-La Trobe Centre for AI and IoT, La Trobe University. He has published over 250 peer-reviewed articles, including three books and 180 journal articles. His research interests include the Internet of Things, wireless communications, machine learning for the IoT data analytics, and computer vision. Due to his instrumental leadership in establishing Australia’s first accredited Internet of Things Engineering degree program, he was inducted into Pearcy Foundation’s Hall of Fame, in October 2018. He is an elected Fellow of the IET in UK and Engineers Australia. He was named a Queensland International Fellow by the Queensland Government of Australia, from 2010 to 2011, an Endeavour Research Fellow by the Commonwealth Government of Australia, from 2012 to 2013, a Smart Futures Fellow by the Queensland Government of Australia, from 2012 to 2015, and a JSPS Invitational Fellow jointly by the Australian Academy of Science and Japanese Society for Promotion of Science, from 2014 to 2015. He received the TNQ Innovation Award, in 2016, the Pearcey Entrepreneurship Award, in 2017, and the Engineers Australia Cairns Engineer of the Year, in 2017. He is currently an Associate Editor of the IEEE INTERNET OF THINGS JOURNAL and IEEE ACCESS.

NATHAN J. WALTHAM received the Ph.D. degree in coastal marine science from Griffith University, Australia. Since joining James Cook University, he has continued to develop a deep research interest on the tropical coastal and marine environments, focusing on understanding and solving the most important management and conservation challenges facing humans now, and into the future. He is currently a Senior Principal Research Scientist with the Centre for Tropical Water and Aquatic Ecosystem Research (TropWATER) and a Senior Lecturer in marine science. He is a Queensland Government and a Smithsonian Research Fellow, and is regularly invited to deliver lectures to government, industry, and community on coastal marine science and restoration. He won the Business Higher Education Round Table (BHERT) Award, in 2019 for a long standing research partnership with a major port authority in Queensland (North Queensland Bulk Ports). He is an Associate Editor of Estuaries and Coasts.

MOSTAFA RAHIMI AZGHADI (Senior Member, IEEE) received the Ph.D. degree in electrical and electronic engineering from The University of Adelaide, Australia. He is currently an associate professor at the College of Science and Engineering, James Cook University, Australia, where he researches machine learning software and hardware design for a variety of applications including automation, precision agriculture, aquaculture, marine sciences, mining, and medical imaging. His research has attracted over $6.5 Million in funding from national and international resources. He received the Doctoral Research Medal and the 2015 Adelaide University Alumni Medal. He was the recipient of several accolades, including the 2015 South Australia Science Excellence Award, the 2016 Endeavour Research Fellowship, the 2017 Queensland Young Tall Poppy Science Award, the 2018 Rising Star ECR Leader Fellowship, the 2019 Fresh Science Queensland finalist, and the 2020 Award for Excellence in Innovation and Change. He serves as an Associate Editor for IEEE ACCESS.