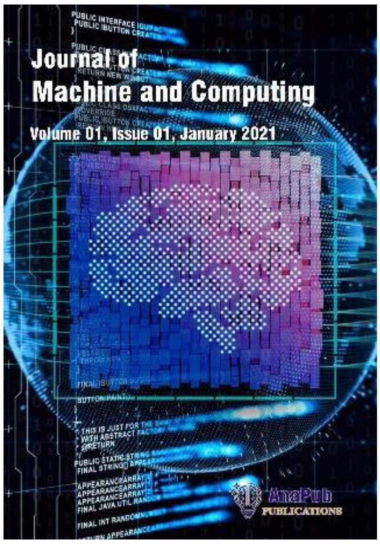
Journal Pre-proof



HydroLens: Pioneering Underwater Surveillance with IoT-powered Object Detection and Tracking using the Hybrid ResNeXt-DenseNet Model

Sujilatha Tada and Jeevanantham Vellaichamy

DOI: 10.53759/7669/jmc202505022

Reference: JMC202505022

Journal: Journal of Machine and Computing.

Received 15 April 2024

Revised form 26 July 2024

Accepted 16 November 2024

Please cite this article as: Sujilatha Tada and Jeevanantham Vellaichamy, "HydroLens: Pioneering Underwater Surveillance with IoT-powered Object Detection and Tracking using the Hybrid ResNeXt-DenseNet Model", Journal of Machine and Computing. (2025). Doi: https://doi.org/10.53759/7669/jmc202505022

This PDF file contains an article that has undergone certain improvements after acceptance. These enhancements include the addition of a cover page, metadata, and formatting changes aimed at enhancing readability. However, it is important to note that this version is not considered the final authoritative version of the article.

Prior to its official publication, this version will undergo further stages of refinement, such as copyediting, typesetting, and comprehensive review. These processes are implemented to ensure the article's final form is of the highest quality. The purpose of sharing this version is to offer early visibility of the article's content to readers.

Please be aware that throughout the production process, it is possible that errors or discrepancies may be identified, which could impact the content. Additionally, all legal disclaimers applicable to the journal remain in effect.

HydroLens: Pioneering Underwater Surveillance with IoT-powered Object Detection and Tracking using the Hybrid ResNeXt-DenseNet Model

[1]Sujilatha Tada*,[2]Jeevanantham Vellaichamy

[1,2] Department of Computer Science and Engineering, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology,Chennai,India

vtd994@veltech.edu.in, drjeevananthamv@veltech.edu.in

*Corresponding Author: **Sujilatha Tada**

Abstract

Efficient object detection and tracking approaches are gaining popularity and being actively used in the world of underwater surveillance. This study presents an innovative protocol that combines a Hybrid ResNeXt-DenseNet Model to boost the visual perceptivity of the Internet of Things (IoT)-based underwater surveillance. The model focuses on what is the best of ResNeXt and DenseNet, yielding higher accuracy at lower computational cost than either. Its components are: IoT-enabled underwater sensors for data capture, a robust data preprocessing pipeline designed for underwater imagery, and the innovative Hybrid ResNeXt-DenseNet Model for object detection and tracking. The architecture of the model is proposed in order to overcome the issues related to underwater environments, such as low visibility, changeable illumination conditions, and complex background. Python was used to implement the proposed model and experiments have been conducted on popular benchmarks of underwater datasets, and the proposed approach obtains a recognition accuracy of 98%. In this model, the Hybrid ResNeXt-DenseNet Model has the notable ability to accurately identify and track objects of interest in real-time underwater situations. Furthermore, the inclusion of IoT features ensures data flows without interruption, allowing for prompt response and action. This research leads towards better situational awareness and marine environment protection systems by proliferating IoT and exploring sophisticated deep learning methods at the root level.

Keywords: Object Detection, Deep Learning, Underwater Surveillance, Internet of Things, Cascaded CNN, Modified Gaussian Filter, Hybrid ResNeXt-DenseNet Model.

1. Introduction

Underwater surveillance has become a significant area particularly for maritime security, environmental monitoring, and resource management. The traditional methods of underwater surveillance are frequently constrained due to poor visibility, high cost of operation and difficult underwater conditions [1] [2] [3]. A transformative solution to these limitations lies in combining Internet of Things (IoT) technology with cutting-edge object detection and tracking system. Deployment of IoT enabled sensors and devices to deploy without any need of staff to continuous monitoring of underwater environments, which is not possible for variety of applications. There are various reasons for this necessity and surveillance underwater. Detection of unauthorized vessels, submarines, and underwater mines is critical for preventing maritime threats, and for safe navigation [4]. Another key use case is environmental monitoring, which includes monitoring marine life, pollution levels, and assessing the impact of environmental changes on the underwater ecosystem. This is important information for researchers and policymakers involved in maintaining marine biodiversity and sustainable use of resources. In addition, underwater surveillance is also very important in infrastructure inspection, such as, pipelines, cables, offshore platforms to guarantee the integrity of these structures, and to avoid costly reparations [5] [6] [7].

One of the major drawbacks of conventional underwater surveillance systems is that most of them require human intervention in terms of manual monitoring; thus, they are time-consuming and more susceptible to errors. The traditional techniques are labour-intensive, costly, and un-tolerant to human factor and weather-dependent factors, as indicated by the need for cameras mounted on existing equipment, such as divers or remotely operated vehicles (ROVs) [8]. In addition, the underwater environment itself presents many challenges – from changing light conditions to water turbidity, to marine life – all of which may compromise the accuracy and effectiveness of surveillance operations. Water by nature is not very transparent, so we cannot expect a very good ability to detect and track objects at big distances underwater [9] [10] [11]. The timely identification of objects in an underwater environment is necessary to address potential threats, reduce accidents, and protect aquatic ecosystems. Early detection of unauthorized vessels or underwater mines as anomalies are an integral part of securing maritime operations. Similarly, early identification of pollution sources can trigger immediate abatement initiatives which safeguard marine biodiversity and prevent lasting environmental impacts. In terms of infrastructure, being able to sense ruptures / defects in pipelines and cables before they become catastrophic failures would save a large amount of money in repair costs. Hence, effective underwater surveillance demands the early and accurate detection of objects [12] [13] [14].

Underwater surveillance systems with built-in deep learning (DL) and machine learning (ML) models have redefined object detection and tracking capabilities. These are good models at working through lots of data to figure out patterns that we as humans might not be able to detect. DL and ML algorithms can be used to improve the accuracy, processing rate, and the ability of underwater surveillance systems to adapt to changing environmental factors. This can be combined with neural networks, Convolutional Neural Networks (CNNs) and other high-precision DL architectures to achieve accurate identification and classification of underwater objects under challenging circumstances [15] [16]. DL and ML models are suitable to navigate the challenging and changing characteristics of underwater environments. CNNs, for example, are structured to automatically and adaptively learn spatial hierarchies of features from input images, which is perfect for object detection applications. Such models can be trained on extensive datasets to recognize everything underwater creatures to artificial object at many different levels of clarity. In addition, the emphasis on continual learning in these models allows them to learn new object types and adapt to environmental changes over long periods [17] [18]. Figure 1 shows the underwater surveillance.

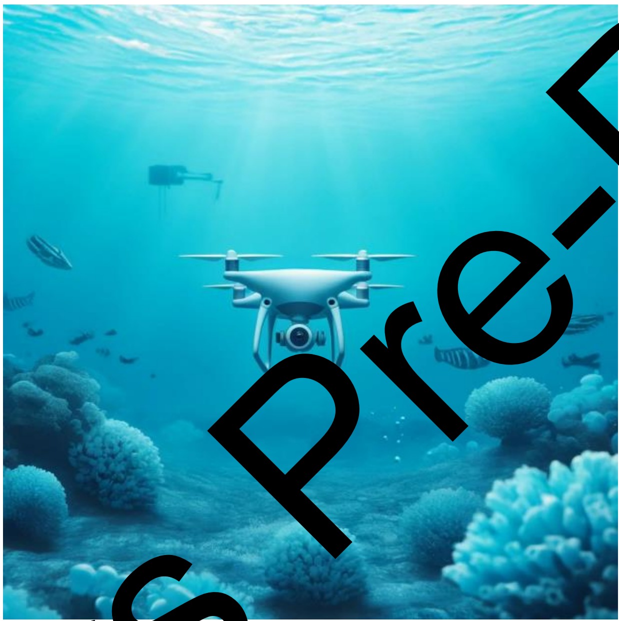


Figure 1: Underwater Surveillance

In this paper, we introduce a new model for object detection and tracking underwater using a hybrid model based on ResNeXt and DenseNet architectures. The Hybrid ResNeXt-DenseNet Model utilizes the residual connections of ResNeXt to strengthen the feature extraction ability and the dense connectivity of DenseNet to encourage more information flow and easier gradient propagation. This hybrid model has been developed specifically to deal with problems found in underwater situations, so that object detection and tracking are made resilient and robust. ResNeXt is an improvement over ResNet architecture where it bring in another dimension called cardinality (the size of the set of transformations) to the network. It enables for more versatile and efficient leaning of features. This is mainly due to the residual connections in ResNeXt which can address this problem and allows us to train networks to great depths. In contrast, DenseNet connects each layer to every other layer in a feed-forward fashion and strengthens feature propagation through the network, which combats the vanishing-gradient problem. The hybrid model of these two architectures combines the best of them — this combination tries to provide an efficient support for underwater object detection and tracking. The Hybrid ResNeXt-DenseNet Model is initially trained on a large dataset of underwater images that contain different types of objects and conditions. During training trials, the process has been refined to enable the model to better detect and categorize objects in the underwater environment, with all of its individual challenges. The architecture of the hybrid model is also capable of learning efficiently while robust to the noise and distortions that are typically encountered under water [19] [20].

Improved Feature Extraction is one of the major advantages of the hybrid model. The residual connections in ResNeXt allow the model to learn more complex features since the information in different layers is merged, while the dense connections in DenseNet maximize the flow of important features across the network. This leads to a more robust and accurate object detection, even under difficult conditions such as darkness or high turbidity. Additionally, the hybrid model scales up to high-volume, real-time data, which is an essential capability when applied to IoT powered underwater surveillance systems. With the hybrid model, IoT devices can collect and send data from a variety of underwater sensors 24/7, displaying a continuous stream of data for analysis. This is important as a decisive tool for real time, to detect or prevent threats for suspicious events immediately. The power of IoT technology complements the Hybrid ResNeXt -DenseNet Model to an extensive approach in the front of the underwater surveillance. In a number of strategic locations, underwater cameras, sonar sensors, and autonomous underwater vehicles (AUVs) may be strategically placed to observe large areas of the ocean with the contours of submerged ice formations. These devices are gathering data and sending it to an individual processor where the hybrid model works with that data in real time.

1.1 Main Contribution of the Work

This work improves the detection and tracking underwater objects by using Noise Reduction, Feature Extraction, and Hybrid Model Architecture for the proposed model to overcome the underwater environment-based challenges. The key contributions are:

- The HydroLens system is proposed, a novel hybrid model that integrates both ResNeXt and DenseNet strengths to achieve optimal underwater object detection and tracking requirements. Cardinality further breaks the ResNeXt Architecture, increasing the number of independent paths for learning more complex and diverse features, which improves the recognition ability of an object. In DenseNet Architecture, each layer is connected to every other layer in a feed-forward fashion, which helps promote feature reuse and improve gradient flow.
- Noise Reduction with Modified Gaussian Filter: These places experience a range of noise stemming from water turbidity, light scattering, and suspended particles. In this research, a Modified Gaussian Filter used for under water application which is designed to reduce noise while retaining important image information, and generates clear, high quality visual data.
- Layered feature learning using cascaded CNN for feature extraction: A Cascaded Convolutional Neural Network (CNN) architecture has been proposed for better feature extraction. This includes multiple CNN layers that work one after another to process and refine features that are successively extracted by earlier layers. The structure combines these two layers to capture complex and hierarchical features, which are important to identify and track objects in difficult underwater environments.

The integration of ResNeXt and DenseNet in the HydroLens system combines enhanced feature extraction with efficient information flow, resulting in a powerful and efficient model for underwater object detection and tracking. The remainder of this work is organized as follows: Section 2 reviews related literature on underwater object detection, noise reduction, and deep learning applications. Section 3 discusses the Cascaded CNN for feature extraction and introduces the HydroLens system, combining ResNeXt and DenseNet architectures. Section 4 presents experimental results and evaluates the system's performance. Finally, Section 5 concludes the paper, summarizing the main contributions, findings, and impact of the proposed system.

2. Related Works

Many fields are being impacted as Internet of Things (IoT) leads physical space into the mix of the cyber space. The important thing is that in IoT, the camera tasks need proper visual information and IoT devices are restricted by many factors such as power, computing ability, storage and so on. While a few are completely adhoc tasks, others could perform regularly on a CPU or other computer, but are not so straight forward on an IoT device. As a consequence, to keep performance acceptable on the one side and how to reduce resource exploitation on the other hand is becoming more and more important in IoT. Object detection and tracking in IoT while dealing on resource constrained performance, and end-to-end solutions are discussed in algorithm known as spatial attention powered multi-domain network (SA-MDNet) [21]. In this method, they successfully discriminates the background and the target in different video sequences using multiclass cross-entropy loss to modify a combination of spatial attention mechanism and spatial domain MDNet model. The proposed method has competitive performance on the OTB datasets with several state-of-art trackers, but costs much less memory than MDNet.

The presented intelligent services and applications rely on advance collaborative and communication technologies such as Artificial intelligence, Internet of Things (IoT), remote sensing, Robotics, Future generation wireless, Aerial access networks and many more. That led to multiple smart city applications in different area like transportation, monitoring, healthcare, public services, and surveillance which are enabled by these technologies, improving the convergence, connectivity, energy efficiency, scalability and quality of service. But the PID has been getting significant attention in recent years and played an important role in various control and monitoring areas. IoT-enabled smart surveillance device for multiple object detection through segmentation and an AI-based system using deep learning based segmentation model PSPNet for segmenting multiple objects [22]. They have leveraged a novel approach to building a dataset using an aerial drone, built in data augmentation techniques, and deep transfer learning to improve the performance of the system. The result of the experiments had shown that the data augmentation increases the system performance as it gives a good accuracy ratio results for multiple object segmentation. A resultant summary is given below, and efficiency is reported at 92% to 95% for the VGG-16 model, ResNet-50 model, and MobileNet model.

Some of the most common computer vision applications are those large scale deep learning scenarios where images are being captured in real-time by a low-quality camera on a constrained device, maybe an Internet of Things (IoT) or a robotic device. Although transfer learning could be useful for these applications, these models are usually pretrained with high-quality image data, which may conflict with the low-quality images, noise or blurs from incomplete cross-modality imaging. It is focused on having a large number of classes with enough images per class and without the errors due to miss-annotations or ambiguous labels that occur with so minimal supervision as possible. Besides, a training strategy is provided to facilitate the training of the model when the dataset is large [23]. A VGG16-SSD model was trained with this methodology on the created dataset and was deployed to a Raspberry Pi and it has been noticed that this is very helpful in developing models for resource-constrained applications.

The most common scenario in video analytics is object detection. Performance at high level is directly linked to accurate performance of object detection. Different platforms are in use for the design and implementation of object detection algorithm. Implementing object detection and tracking using MATLAB which also shows basic block diagram of object detection and explains various predefined functions and object from different toolboxes that can be useful at every level in object detection [24]. This is highly related with many real time applications such as vehicle perception, video surveillance and so forth nature. The algorithm is 90% about a transition algorithm to smoothen the video stream and accommodation of tracking loons, and only the last 10% is about the actual morphing itself. However, none of these methods leverages the prior knowledge of the shape, color, texture etc. of objects.

Multi-camera Multi-object Tracking (MC-MOT) is crucial for a number of computer vision applications in real world. It is a challenging issue to accurately resolve in the practical track-by-track implementation, though there have been a lot of research work on this problem. This task is confounded by the fact that this gait data is presented under different illumination, meanwhile walking patterns and the trajectory may suffer from occlusions. Graph neural networks (GNNs) have gained much interest in data fusion in recent history due to their ability to further enrich data association. Yet, widely adopted graph-based MC-MOT methods employ computational expensive min-cost flow solutions for cross-camera association in static graph structures that lack of adaptability for new detections. In addition, these procedures usually concentrate on processing the cameras from pairs, instead of being based on a global manner. One solution to this problem is use a two-stage lightweight cross-camera tracker, to get a global solution in an efficient way [25]. This strategy emphasizes more on the high level feature trajectories, which are scrutinized using the DeepSort representation tuned on the multi-source information. They exploit the dynamics of Message Passing Graph Neural Networks (MPGNNs) to train a Multi-Camera Association module that jointly learns previously unexplored features and similarities for the cross-camera association. This dramatically increases detection accuracy and feature extraction, surpassing the state-of-the-art MC-MOT algorithms on cross-camera datasets. This development represents an important advance in the field by providing accurate tracking and potential integration of modern techniques for improved performance in difficult tracking situations.

Although significant progress is being made in the field of IoT, there are few constraints due to which detecting and tracking of underwater objects is still underdeveloped. The computing power, storage, and energy are always constrained in IoT devices which hinder them from performing heavy computations. Although there have been networks such as spatial attention powered multi-domain network (SA-MDNet) that perform well with low memory usage, these methods are limited when dealing with noisy underwater settings. Having reached impressive accuracy on challenging applications like smart surveillance using advanced models like PSPNet and VGG16-SSD, performance on low-quality, noisy images frequently produced by the IoT devices has still proven elusive. Moreover, existing multi-camera multi-object tracking systems are very complex computationally and do not handle well underwater conditions increasingly changing. In view of these challenges, a dedicated version that aptly combines the noise reduction, resilient feature extraction, along with scalable deep learning strategies is essential to upgrade underwater surveillance systems.

3. Methodology

The methodology is the whole amalgamation of robust noise reduction, feature extraction, and hybrid deep learning model for improving underwater object detection and tracking. The output will be passed through a Modified Gaussian Filter designed for underwater domains to reduce noise and retain important features. Finally, we set up a Cascaded Convolutional Neural Network (CNN) (with several CNN layers) to learn, improve and combine the extracted features through CNN. We propose the HydroLens system, which integrates both architectures to mitigate their drawbacks and further improve the ability of feature extraction and information flow capacity, and to achieve end-to-end solution for plastic detection and trajectory tracking in such a limited underwater condition. Figure 2 shows the architecture of proposed model.

3.1 Data Collection

To support this research, a new dataset called Underwater Surveillance Dataset was created by collecting real-time sensor values, high-quality images, and other environmental variables that are most suitable for underwater surveillance systems. Additional details gathered by known techniques are integrated into this dataset to provide effective coverage through hydrophones, CTD sensors, and underwater cameras. Borne on the HydroLens system, and array of sensors captures a number of specific measurements of underwater conditions. Hydrophones used for identification and recording of sound signals and CTD gives information of conductivity, temperature and depth of water thus affording information of environment in water. Visual data require high resolution, which is used by underwater cameras, and all that is required for object detection. While dissolved oxygen sensors undertakes water quality monitoring, turbidity, and salinity and pH changes do undertake the monitoring of water acidity. Pressure sensors will capture water depth to check equipment in operation with data obtained from pressure sensors as mentioned above, chlorophyll sensors would be used to get estimate of primary productivity. In the same way, current meters monitor water speed, which gives clues on where an object might travel. When combined, they provide reliable, on-the-dot surveillance underwater.

3.1.1 Hydrophone

Hydrophones are -custom-built submersible microphones that simply listen for sounds in an underwater environment. In this manner, these appliances have a profound effect on overseeing organic life, principally in the vocalizations of marine mammals and fish. Hydrophones are an essential tool to discover objects and to track movement of objects under water surfaces, to be able to record the noise generated by them. This is necessary for fishery enforcement, in applications where we intend to follow an underwater vehicle with an acoustic signature so that we can accompany that, so that we know how that's doing.

3.1.2 Conductivity, Temperature, Depth (CTD) Sensor

CTD sensors provide valuable data on the physical characteristics of the water column, by measuring seawater conductivity, temperature, and depth. The performance of imaging and detection systems can be significantly affected by these parameters, and as a result they are key to the overall understanding of the underwater environment. For instance, water temperature and salinity can change the density of the water, which is what influences the sound propagation. This is important for tuning acoustic sensors, such as hydrophones for precise data capture and analysis.

3.1.3 Underwater Camera

These cameras are specifically designed to capture pictures and videos in aquatic environments, and as such, are made to be pressure and low light resistant, so they are able to take high-resolution images in the deepest depths of the world's oceans. These are crucial cameras for the documentation of underwater landscapes and for object recognition and tracking. Underwater cameras are used in coral reef monitoring, underwater archaeology and marine biology research among other applications, where quality visual data is important to the identification and tracking of objects in different underwater environments.

3.1.4 Dissolved Oxygen Sensor

For this reason dissolved oxygen sensors are used to measure the amount of oxygen present in water which is used as an indicator of water quality and the health of a marine ecosystem. This information is needed for characterizing habitat conditions for marine animals. By monitoring dissolved oxygen levels for underwater surveillance, areas of ecological value and areas that may be polluted can be identified as well as areas of concern for pollution or disturbance in the environment to help protect and manage marine environments.

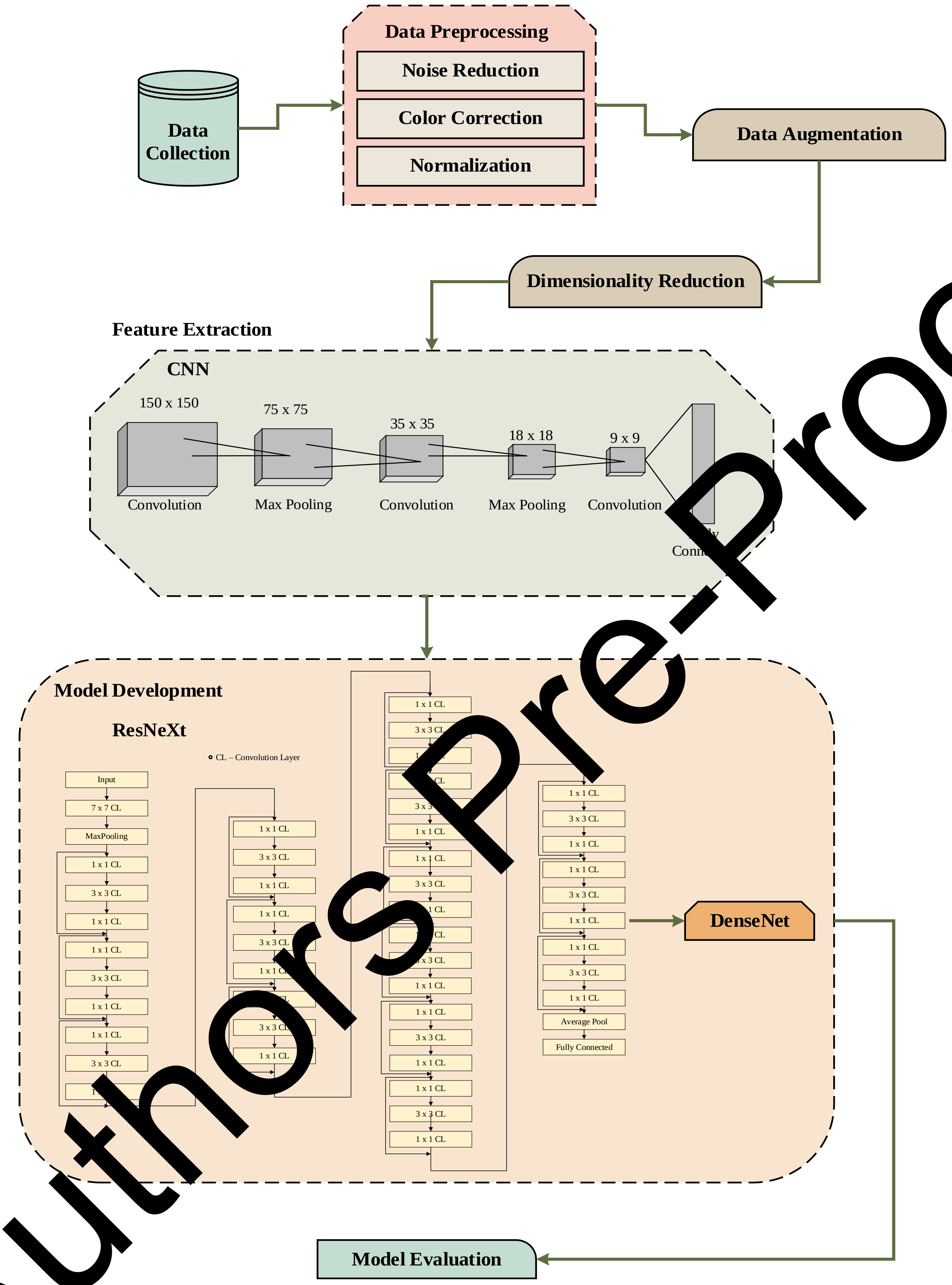


Figure 2. Architecture of Proposed Model

3.1.5 Turbidity Sensor

Turbidity sensors measure how clear the water is by detecting suspended particles, which decreases the visual range and muddies the image when attempting to conduct underwater surveillance. These sensors are useful for analysing water quality and tailoring the image processing methods to different visibility conditions. Turbidity Sensors are also used in environmental monitoring to prevent sediment erosion and pollution, as well as clear and efficient underwater surveillance data.

3.1.6 Salinity Sensor

The salinity sensor is used to detect the density and conductivity of unique water which is directly affected by the salt concentration. Understanding ocean circulation and the distribution of marine life relies in part on data related to salinity. Salinity measurements during underwater surveillance, are used to calibrate sensors and improve the performance of object detection and tracking by sonar and acoustic equipment, thus ensuring accurate and reliable data acquisition.

3.1.7 pH Sensor

A very critical parameter for life in aquatic environment is the pH sensors which measure the alkalinity or acidity of water. Globally significant perturbations to marine ecosystems are occurring due to environmental shifts accompanying pH change, particularly habitat acidification. The consideration of pH is a proxy for assessing the health of an underwater habitat and monitoring the state of the network can be used as an early indicator for pollution events, allowing pro-active management of marine environments.

3.1.8 Pressure Sensor

Pressure sensors measure how hard water is pressing at different depths, giving data on water pressure and depth. Essential for characterizing the physical properties of the underwater environment, these sensors are employed in the navigation and positioning of underwater vehicles. Pressure data allows other sensors and equipment to operate within their optimal design boundaries, to maintain system reliability and performance.

3.1.9 Chlorophyll Sensor

Sensors measuring the concentration of chlorophyll in water are most directly an index of phytoplankton concentration and it is standard practice to refer to them as an index of primary productivity. This is important for the base of the marine food web and ecosystem health. In underwater surveillance, higher chlorophyll levels are associated with biologically productive regions that should be monitored more closely to manage resources of marine environments appropriately for vital marine resources.

3.1.10 Current Meter

Current meters used to measure the speed and direction of water currents give scientists clues to the movements of water in the ocean. This is important for predicting marine ecosystems and can affect marine life and pollutant dissipation. This information is significant in the field of underwater surveillance as the flow patterns allow predictions of movement of detected objects, which in turn can increase tracking accuracy and preservation of optimal monitoring effectiveness.

3.2 Data Preprocessing

On the sensor level, data is preprocessed by HydroLens to guarantee that all input data are of equal quality and compatibility. The datasets acquired with hydrophones, CTD sensors, and cameras possess different nature and scales of measurement requiring preliminary calibration and normalization. Whenever dealing with the raw data especially from the microphones used in hydrophones, noise filtering is carried out as the initial step in data pre-processing. A normalization of the image illumination settings is applied to the visual data in order to equalize the influences arising from the water conditions. Each data type is then synchronized to resolve time differences between the sensors which is very important for coherent input to the network. This preprocessing at the sensor level reduces the amount of correction that is performed on data foam at a later stage, which is very useful in enhancing data flow in addition to speeding up the response time.

3.2.1 Noise Reduction using Modified Gaussian Filter

In image and sensor data, noise reduction is a necessary preprocessing step for increased data quality. HydroLens incorporates a Modified Gaussian Filter particularly designed for JPEG underwater noise including particulate scatter and motion blur. Real time turbidity levels are used to control the parameters of the Gaussian filter to provide noise adaptive filtering of the image while maintaining edge prominences. In addition, HydroLens uses a frequency domain filter to remove period noise due to current and mechanical movement of sensor equipment. Such two-level filtering policy helps improve the quality of images acquired through an underwater camera which is important when developing object detection and tracking systems in real-life conditions. It is also possible to see that the Gaussian filter (or kernel) operates by averaging the surrounding pixel values using as weight a Gaussian. This modification adapts the scale of the standard deviation and kernel size locally to the amount of noise using the unique noise statistics in the region so that optimal smoothing strength is achieved with minimal blurring of underlying features. It is possible to modify this to process time-series sensor data, such as using a Gaussian filter to eliminate noise in the signal while preserving the integrity of the original data. This makes sure that the data being fed to detection and tracking algorithms is noise free, which help in enhancing overall system performance.

3.2.2 Color Correction

The color distortions in underwater images are caused by the absorption / scattering of light in water. One method of color correction used to correct this is to try to bring the picture back to its natural colors. The method is based on examination of the spectral characteristics of light in water and colour shift modelling as a function of depth and water composition. With an inverse transformation that accounts for these distortions, the algorithm is able to successfully recover the initial colors. That approach uses hand-inspired color correction scales and a dataset of underwater images with known color profiles to allow to a machine learning algorithm to learn the correction scales. This leads to imbalance, leading to a much-improved appearance of the images which are similar to the original images, making them useful for object detection and tracking purposes.

However, in the underwater photography, turbidity and species that causes light scattering reduce visibility. For this purpose, the proposed dynamic contrast adjustment algorithm that is employed by HydroLens identifies those low-contrast areas of images and makes them much more visible. This is an adaptive approach where the algorithms used adjust the contrast depending on the real time change in water quality as described by the turbidity sensor. Further, color correction methods including white balancing and spectral restoration used to compensate for the blue and green shifts that are characteristic of underwater settings. Some visibility corrections enhance object detectability, as well as its recognition accuracy, by modifying or equalizing color contrast to far more natural and visceral.

3.2.3 Normalization

Normalizing data, a common preprocessing step when scaling features so that they fall within a consistent ranges and all features are considered equivalent for model training. Range of pixel values is 0 => 255 Normalization [0, 1] or [-1, 1] — just scaling over the complete range or over half of it which is more common. This requires them to subtract the mean and then divide by the standard deviation of the pixel values. Normalization. This refers to applying the necessary scaling on sensor data to match the range of data, that needs to be collected from other types of sensors, making the comparisons and integration easier. This element is essential for maintaining the numerical stability of the models and to enable the learning algorithms to work effectively on a wide variety of data. Apart from this, Normalization is helpful in faster convergence of optimization algorithms while training a model.

3.2.4 Data Augmentation

Data augmentation is the process of artificially enlarging the training dataset by normalizing it while diversifying its output and in turn improving generalization of the model so that overfitting is minimized. For image data, augmentations are random rotations, flips, scaling, cropping, and brightness of the image. These augmenting transformations produce new forms of the image and help the model learn features that are robust to these changes. For sensor data augmentation we could inject the noise, and augment different environmental conditions or generate a new one based on the stat of the original dataset. This approach aids in making model generalize better to unseen data and improve the ability to detect and track object under diverse conditions.

3.2.5 Dimensionality Reduction

When the number of features in the dataset is too large, we can apply dimensionality reduction techniques to make dataset smaller, but that the new features are the most important ones. t-SNE is employed to compress the data into lower-dimensional representations for image data. This drastically reduces computational complexity and makes sure to learn the most informative features for the model. To process the sensor data, dimensionality reduction is crucial to determine the features relevant to predicting the class of interest or to transform the data to a space of lowest dimensionality by using Linear Discriminant Analysis (LDA). This step is necessary to save on the dataset simplifying, to enable the learning algorithms effectively on the data as well making the results into more predictively interpret.

3.2.6 Train-Test Split

The dataset was then split into training and validation sets with 80:20 ratio, so that both sets contain representations of nearly all underwater conditions. For ensuring the stability of results, during training, we also used 5-fold cross-validation, where the set is then divided into five equal parts so that each part acting as the validation set at least once while the rest forms the training data. This approach is helpful in reducing overfitting problem and to ensure better generalization capability since it tests the performance of the model across different data sets, therefore the model will be tested for consistency in under water climate changes.

3.2.7 Data Security and Reliability

For the security and credibility of the data in the underwater IoT, the HydroLens system employs a secure method of data transfer such as encryption on the data sent from one device to another. Integrity check of data, for example, error check such as cyclic redundancy check (CRC) commonly used in stream communication to minimize the effect of temporary break in connectivity when working in underwater scenario. Further, data buffering and packet redundancy are used in HydroLens to avoid the loss of data and operate in real time to provide the description for conditions with interfering signals. Subsequent releases may potentially address blockchain for increased security and narratives, guaranteeing that all data from the sensors cannot be manipulated.

3.3 Cascaded CNN for Feature Extraction as Layered Feature Learning

In the paper, we use the term layered feature learning in the context of cascaded Convolutional Neural Networks (CNNs) which refers to a hierarchical way, i.e., first layers looking for simple patterns following layers considering more and more complex ones, of extracting fused features from data. This process is very useful for processing image and sensor data in the HydroLens system, where exposing the different aspects of the underwater environment is critical. There are many layers of a cascaded CNN architecture that work on different low-level features to make sure the initial layers extract features that are lower-level as compared to deeper layers. These layers are convolutional operations with small filters (e.g., 3x3 and 5x5 kernels) that scan the input data and identify primitive patterns including edges, textures, and some simple geometric shapes. This could mean identifying shapes of objects, changes in texture, or changes in color in image data. For example, the first layers on the left here might capture rudimentary features such as simple signal patterns, oscillations, basic modulations in the data in sensor data.

In underwater imagery as an example the first layers may detect the edges of fish, coral structures or underwater debris — creating a basic recognition of what the scene consists of. And for hydrophone sensor data this could help identify fundamental acoustic signatures of marine life or underwater vehicles, respectively. As one goes deeper into the CNN, intermediate layers tend to capture mid-level features. These features are higher-level, and are combinations of the low-level features engineered from the initial layers. This could be noticing pieces of objects in an image, such as the fins of fish, the stems of corals, or knowing the textures of underwater environments, such as the coarse surface of rocks, or the smooth skin of sea creatures.

Convolutional Layer

The core in CNNs is the convolution operation. For an input image I of dimensions $H \times W \times D$ (height, width, depth) and a filter K of size $k \times k \times D$ (assuming square filters and same depth as input):

$$(I * K)(i,j) = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \sum_{d=0}^{D-1} I(i+m, j+n, d) \cdot K(m,n,d) \tag{1}$$

This operation is repeated for each position (i,j) in the input, resulting in a feature map.

Activation Function

After convolution, an activation function such as ReLU (Rectified Linear Unit) is applied to introduce non-linearity:

$$f(x) = \max(0, x) \tag{2}$$

For each element x in the feature map resulting from the convolution.

Pooling Layer

Pooling layers reduce the spatial dimensions of the feature maps. Max pooling with a window size of $p \times p$ can be defined as:

$$P(i,j) = \max_{m=0}^{p-1} \max_{n=0}^{p-1} \big(I(i+m, j+n)\big) \tag{3}$$

This operation takes the maximum value within each $p \times p$ window in the feature map.

Layered Feature Learning

Initial Layers: Low-Level Feature Extraction

Let I_0 be the input image and K_0 be the filter for the first convolutional layer. The output feature map F_0 is given by:

$$F_0 = f(I_0 * K_0) \quad (4)$$

Where f is the ReLU activation function.

In the case of sensor data, intermediate layers would perhaps begin to recognize patterns over time — such as regular acoustic signals from a particular marine species or reliable differences in water temperature and salinity patterns. These mid-level features are essential for obtaining a richer more complex view of the data, taking us beyond overt patterns into something that has slightly more meaning. Where CNN cascaded performs high level feature extraction layer which computed with deeper layers relay the features. These layers combine mid-level features identified with those earlier to identify complex patterns or objects within the data. Image data can mean looking for objects of interest in an image like species of fish, marine mammals, or underwater vehicles amidst variations in light and visibility. At a high level, this may involve detecting certain types of events or states within sensor data (e.g., the presence of a specific underwater animal based on the characteristics of its acoustic signal), or even tracking the movements of certain energy phenomena given environmental conditions (e.g., thermoclines, which are visualized by integrating temperature and depth observations). Importantly, the detection and tracking of objects in the underwater context also requires these high-level features.

Intermediate Layers: Mid-Level Feature Extraction

For subsequent layers, let F_{l-1} be the input feature map from the previous layer, K_l be the filter for layer l, and P_l be the pooling operation:

$$F_l = P_l\big(f(F_{l-1} * K_l)\big) \quad (5)$$

These are convolution, activation and pooling operations for layer l.

Deep Layers: High-Level Feature Extraction

Let F_{n-1} be the input feature map to the final layer, K_n be the filter for the final layer, and P_n be the pooling operation:

$$F_n = P_n\big(f(F_{n-1} * K_n)\big) \quad (6)$$

Final Feature Representation

The final feature representations F_{final} used for object detection and tracking is a concatenation of high level features from the deep layers:

$$F_{final} = concat(F_1, F_2, \dots, F_n) \quad (7)$$

Where $concat$ denotes the concatenation operation of feature maps from different layers.

The HydroLens system with layered feature learning could capture the compositional structures of underwater data with different levels of abstraction. As we go deeper in the network, the initial stages could capture low-level image features like edges, textures whereas its later layers represent high-level objects, and structures. This enables our method to both detect and track objects robustly even in difficult underwater conditions — with different clarity, changing background. The cascaded CNNs are utilised because of their multi-layered nature, ensuring robustness of the system. This redundancy is necessary to provide reliable detection and tracking as inevitably, some details will be missed in the initial layers of the network as a result of the underwater noise and distortions.

The cascaded CNNs are scalable and enables the HydroLens system to expand to multiple demanding tasks and datasets. As new demands emerge, and new technologies and sensors come online, the system can continue to improve and adapt by adding additional layers or increasing the complexity of layers. By using cascaded CNNs which enables layered feature learning, HydroLens can adapt to different data and different underwater environments. This flexibility allows the system to operate effectively within a range of scenarios from shallow coastal waters to deep ocean environments, ensuring detect and tracking capabilities over a wide spectrum of situations.

3.4 Model Development for HydroLens System

A detailed design process used for the development of the HydroLens system which exploits strengths of believe both ResNeXt and DenseNet architectures. This model is known about hybrid because it tries to use cardinality and dense connectivity advantages to gain better performance in underwater object detection and tracking. The HydroLens system is intended to combine the best of ResNeXt and DenseNet in a fusionary design. It splits the convolutional layers into several parallel branches (also known as paths) and in turns enrich the model's ability to capture separate unique characteristics, and it is well-known for its cardinality feature. However, the dense connectivity in DenseNet allows the layers to have direct connections with every other layer below them which provides maximum possible information flow between the layers in the network and hence, encourages feature reusability through the network.

The main addition of ResNeXt is the cardinality, that is, the dimension of the set of transformations. This is done by applying grouped convolutions and thus, the input is partitioned into a few groups where each group is separately worked on before concatenating. This new version not only learned much richer features, but also did so without any substantial increase of computational complexity.

This can alternatively be presented in the following formula as a ResNeXt block:

$$y = \sum_{i=1}^{C} F_i(x_i) \tag{8}$$

Where x_i is the input to the i-th path, F_i is the function applied by the i-th path (typically a series of convolutions), and C is the cardinality (number of parallel paths). This architecture enables the model to pick up many features at each layer, which helps it to classify a wide variety of object and texture in underwater images.

One method used to solve the problem was to directly connect each layer with other layers in the network in a feedforward manner, this method is called DenseNet. This very dense connectivity pattern guarantees that whatever the layer, the relevant learnings learned by it are immediately available to all subsequent layers, thereby allowing easier feature re-use and facilitating the building of the weight gradients by backpropagation.

The DenseNet block is represented as:

$$y_l = H_l([x_0, x_1, \dots, x_{l-1}]) \tag{9}$$

Where y_l is the output of the l-th layer, H_l represents the l-th layer function (typically a composite function of batch normalization, ReLU, and convolution), and $[x_0, x_1, \dots, x_{l-1}]$ represents concatenation of feature maps of all the previous layers. This strong bucket brigade structure makes it possible for the layers composing the model to gradually construct on previously learned features, which results in improved representation.

3.4.1 Integrating ResNeXt and DenseNet in HydroLens

For combining the merits of ResNeXt and DenseNet, we propose a hybrid block which integrates both cardinality and dense connectivity. In the HydroLens system, ResNeXt and DenseNet are fused to leverage their unique advantages: ResNeXt's cardinality and DenseNet's dense connections. Multiple pathways, referred to as cardinality, are incorporated inside ResNeXt's residual blocks with the purpose of expanding the number of features that can be learned across all layers with relatively little computational overhead. In DenseNet, all layers are connected to all subsequent layers, making full use of features from earlier layers and improving gradients through the network. By combining these, HydroLens successfully develops a network where each layer obtain highly diversely information from the parallel paths of ResNeXt and DenseNet's direct connection. This fusion enables faster and efficient learning of these features, avoids the gradient vanish problem, and greatly cuts down on unnecessary computation, positioning the model as ideal for underwater object detection, as detailed in the results section, where the extraction of intricate features is critical and must be done in as efficient a manner as possible.

Hybrid blocks are comprised of ResNeXt pathways in which a single pathway block´s output is logically connected to each successive layer. Here, the design is pure such as the model can take advantage of the widely distinctive feature sets collected by ResNeXt and the excellent feature reuse offered by DenseNet. The HydroLens system is based on stacking hybrid blocks with transition layers in between them, hence the overall architecture. The transition layers contains convolutional operations to reduce the number of feature maps for computational efficiency. The output may be the bounding box coordinates of the objects, the class probability, and tracking info for object detection and tracking in underwater. The training is based on optimizing a multi-task loss function derived from the sum of object detection loss and object tracking loss for the HydroLens system. In order to help the model to generalize well and avoid overfitting, techniques such as data augmentation, batch normalization, and dropout are used. Besides, by using weights that are pre-trained on the ResNeXt and DenseNet models, the hybrid model weights initialization is being performed and this is followed with fine-tuning on the underwater datasets. The HydroLens thus offers a compelling solution that realizes a beneficial harmony between the broader field of view facilitated by the cardinality of ResNeXt and the focus on dense feature reuse of DenseNet. Such a hybrid architecture provides excellent results in the high-variability environments that arise in underwater surveillance, yielding an increase in accuracy and robustness of object detection and tracking.

Underwater environments bring infrastructure restrictions related with transmission media in terms of delay and bandwidth. To address this, HydroLens is designed to employ a data down sampling method in which only the required fields such as an object's coordinates and classification are transmitted, not actual data. Buffering techniques are also used and data can be sent in breaks if the signal is strong to reduce latency in poor conditions. Furthermore, the data collected by the sensor is compressed using the loss-less compression techniques so that the volume of data to be transmitted is reduced. HydroLens is confident it shall retain real time processing while not consuming bandwidth, which is so crucial in undertakings involving submersion.

Algorithm: HydroLens Models

Initialize $\boldsymbol{Heweights}$ for all layers

Set initial learning rate $\boldsymbol{\eta 0}$

Set total number of epochs $\boldsymbol{T}$

Set initial dropout probability $\boldsymbol{p}$

Set L2 regularization strength $\boldsymbol{\lambda}$

Set optimizer to Adam with $\boldsymbol{\beta 1}, \boldsymbol{\beta 2}$, and $\boldsymbol{\epsilon}$ parameters

function **Convolution**$(\boldsymbol{I}, \boldsymbol{K})$**:** // Perform convolution with filter K on input I

  return $\sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \sum_{d=0}^{D-1} I(i+m, j+n, d) * K(m, n, d)$

function **ReLU(x):** // Apply ReLU activation

  return $max(0, x)$

function **PReLU**$(\boldsymbol{x}, \boldsymbol{\alpha})$**:** // Apply Parametric ReLU (PReLU) activation

  if $x \geq 0$:

    return x

  else

    return $\alpha * x$

function **MaxPooling**$(\boldsymbol{I}, \boldsymbol{p})$**:** // Perform max pooling on input I with window size p

  return $max_{m=0}^{p-1} max_{n=0}^{p-1} I(i+m, j+n)$

function **Dropout**$(\boldsymbol{x}, \boldsymbol{p})$**:** // Apply dropout

  return $\left(\frac{1}{1-p}\right) * x * Bernoulli(p)$

function **L2Regularization**$(\boldsymbol{w}, \boldsymbol{\lambda})$**:** // Apply L2 regularization

    return $\lambda * \sum_{i=1}^{n} w_i^2$

function **ResNeXtBlock**$(\boldsymbol{I}, \boldsymbol{C})$**:** // Initialize ResNeXt block with cardinality C

  $output = 0$

  for $i = 1$ to C:

    $output \pm Convolution(I, K_i)$

  return ReLU(output)

function **DenseNetBlock**$(\boldsymbol{I}, \boldsymbol{layers})$**:** // Initialize DenseNet block

  $outputs = [I]$

  for l in 1 to layers:

    $new_{output} = ReLU\big(Convolution(concat(outputs), K_l)\big)$

$outputs.append(new_{output})$

return $concat(outputs)$

function **HybridBlock**($\boldsymbol{I, C, layers}$)**:** // Initialize Hybrid Block

$resnext_{output} = ResNeXtBlock(I, C)$

$densenet_{output} = DenseNetBlock(resnext_{output}, layers)$

return $densenet_{output}$

function **BuildHydroLens**($\boldsymbol{I, num_{blocks}, C, layers_{per_{block}}}$)**:** // Build the overall HydroLens network

$output = I$

for b in 1 to num_{blocks}:

$output = HybridBlock(output, C, layers_{per_{block}})$

$output = MaxPooling(output, pool_{size})$

return $GlobalAveragePooling(output)$

function **TrainHydroLens**($\boldsymbol{model, data, labels, epochs, \eta 0, T}$)**:** // Training loop

for t in 1 to T:

$learning_{rate} = \eta 0 * \left(1 - \frac{t}{T}\right)$

for batch in data:

$I, y_{true} = batch$

$y_{pred} = model(I)$

$loss = LossFunction(y_{true}, y_{pred}) + L2Regularization(weights, \lambda)$

$gradients = ComputeGradients(loss, model.parameters)$

$UpdateParameters(model.parameters, gradients, learning_{rate}, \beta 1, \beta 2, \epsilon)$

$model = Dropout(model, p)$

if $ValidationLoss(model, validation_{data})$ does not improve:

Stop training

break

function **LossFunction**($\boldsymbol{y_{true}, y_{pred}}$)**:** // Define loss function

return $CrossEntropyLoss(y_{true}, y_{pred})$

function **UpdateParameters**($\boldsymbol{parameters, gradients, \eta, \beta 1, \beta 2, \epsilon}$)**:** // Define Adam optimizer update rule

$m_t = \beta 1 * m_{t-1} + (1 - \beta 1) * gradients$

$v_t = \beta 2 * v_{t-1} + (1 - \beta 2) * gradients^2$

$m_{hat} = \frac{m_t}{1-\beta 1^t}$

$v_{hat} = \frac{v_t}{1-\beta 2^t}$

$parameters -= \eta * \frac{m_{hat}}{\sqrt{v_{hat}} + \epsilon}$

$weights = InitializeHeWeights()$ // Main execution

$model = BuildHydroLens(input_{image}, num_{blocks}, C, layers_{per_{block}})$

$TrainHydroLens(model, training_{data}, training_{labels}, T, \eta 0, T)$

End Algorithm

3.4.2 Optimizing Model Parameters and Hyperparameters

The parameters and hyperparameters of the HydroLens system should be optimized in order to obtain the best performance in terms of underwater object detection and tracking. This includes optimizations across all aspects of the model architecture, training pipeline, and data processing pipeline to ensure a good balance between model accuracy and efficiency. HydroLens encompasses a multistep optimization scheme for controlling a number of parameters that defines the trade-off between the precision of the ray tracing and computation time. First, batch sizes are enhanced to allow maximization of the GPU memory without straining it, making the processing fast. There is use of learning rate schedule, which means that the learning rate is reduced as the training process goes on to adjust the model closer to convergence. In addition, dropout rates are used in an attempt to avoid overfitting and define the right L2 for weight magnitude maxima. For the underwater setting, we prune the network more and less or some layers with more relevance to feature extraction and less or no relevance having redundancy in the basic layers. All these optimizations individually cut down computational time and resource utilization, enabling real-time object detection in even low bandwidth contexts.

Model Parameter Optimization

1. Weight Initialization

The initialization was employed to prevent the weights of the neural network from starting from a place that would not make learning more or less possible.

2. Learning Rate

The learning rate started at 0.01 and decreased according to a learning rate schedule. Our numerical integration was done by this equation:-

$$\eta_t = \eta_0 \times \left(1 = \frac{t}{T}\right) \qquad (10)$$

Where η_t was the learning rate at epoch t, η_0 was the initial learning rate, and T was the total number of epochs. This enabled changing the learning rate as the training progressed so that it led to better convergence.

3. Batch Size

We will try batch sizes until at some point where more batch size will finally make gradient estimate more correct but requiring really more computational power.

4. Number of Layers and Units

We used cross-validation to optimize the depth of the network, and the number of units in each layer. The goal of this process was to balance the model complexity so that it did not underfit or overfit, and to tune the best model architecture that yielded the highest validation performance.

3.5 Hyperparameter Optimization

3.5.1. Dropout Rate

We changed the dropout rate to be able to regularize it. We employed dropout to randomly set a fraction of input units to zero at each update during training as follows,

$$Dropout(x) = \frac{1}{1-p} \cdot x \cdot Bernoulli(p) \qquad (11)$$

Where p was the dropout probability. The model's generalization performance improved by using this technique.

3.5.2. Regularization Parameters

We used the L2 regularization (weight decay) to prevent the weights from growing too large, as well as to improve generalization. The added regularization term to the loss function was:

$$L(w) = L_0 + \lambda \sum_{i=1}^{n} w_i^2 \qquad (12)$$

Where L_0 was the original loss, λ was the regularization strength, and w_i were weights.

3.5.3 Optimization Algorithm

We used the Adam optimization algorithm that dynamically modified the learning rate for each parameter. The update rule for Adam was:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{13}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{14}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{15}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{16}$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{17}$$

Where g_t was the gradient, m_t and v_t were moment estimates, β_1 and β_2 were hyperparameters, and θ_t were the parameters. Adam helped in efficiently navigating the optimization landscape.

3.6 Cross-Validation and Hyperparameter Tuning

3.6.1 Grid Search

To find the best combination, we searched for a combination of hyperparameters. This method did a systematic exploration of the hyperparameter space and evaluated each configuration using cross-validation.

3.6.2 Early Stopping

We added early stopping to stop training when the performance on a validation set no longer improved. This helped in avoiding overfitting and using the computational forces judiciously as the training once the optimization did not get any better.

By using ResNeXts cardinality advantages together with DenseNets dense connectivity characteristic, the HydroLens system strikes a best equilibrium of multi features extraction and feature reuse. Especially for underwater surveillance tasks which involve a massive variant of complex and dynamic environments, this hybrid architecture can, in principle, provide higher accuracy and robustness at the above object detection and tracking tasks. This approach allowed them to achieve optimal performance of the HydroLens system for underwater object detection and tracking, in which both model parameters and hyperparameters were optimized systematically. This included a mix of experimental tuning, validation and optimization methods to ensure the model was both accurate and efficient.

3.7 Novelty of this Work

This research elaborates on a number of novel techniques that push the frontiers of underwater object detection and tracking, ameliorating longstanding issues and constraints with fetch methods. The key novelty of our work is the incorporation of a new noise reduction method, a novel feature extraction approach, and a customized hybrid deep learning model for underwater settings. Using the Modified Gaussian Filter for noise reduction is a major plus point over noise reduction through traditional methods. Many noise propagates underwater included those caused by water turbidity, light scattering and suspended particles, and it is usually high. The Modified Gaussian Filter is designed to suppress the noise while preserving important features of the image that were adversely affected by noise. Better visual data improves the accuracy of details, which are crucial for object detection and tracking. Second, the cascaded Convolutional Neural Network (CNN) for feature representation instead developed a hierarchical representation learning and refinement from an underwater image. Current CNN-based object detection on datasets may not completely represent the in-depth and varying features desirable for known target detection underwater. The cascaded architecture allows the work to build and refine the features sequentially which strengthens the discriminative features learned by the model. Here, the model's layered feature learning process greatly improves its ability to detect and track objects in the complex underwater world, even with noise. The major contribution in this paper: the HydroLens system combines two powerful deep graph architectures: ResNeXt and DenseNet. By using the hybrid model, we able to make use of the excellent feature extraction abilities of ResNeXt which is capable of extracting rich features within a layer, due to its concept of cardinality and DenseNet that able to learn and reuse both diverse and complex features due to its dense connectivity within each layers and produce smoother gradient flow. Combining these architectures into one model allows us to create a powerful and efficient system that outperforms both traditional and contemporary methods for underwater object detection and tracking. This unusual hybrid modeling technique that combines the ResNeXt and DenseNet within the HydroLens system describes a model type that is uniquely suited to the challenges of underwater environments.

4. Results and Discussions

Python was used to implement the proposed model as it provides wide range of libraries and frameworks for deep learning and data manipulation. The machine used for said run packed a 24M cache-holding Intel Core i7-1370P Processor with up to 5.20 GHz of clock speed. The high-performance CPU was able to deliver the computational power needed for the efficient processing of advanced operations and large datasets. This was paired with 16GB of RAM to help keep model sizes and data in memory during both training and inference. For even more computational power, we used an ASUS Dual GeForce RTX 4060 OC Edition White 8GB Graphic Card. This high performance GPU supported with excellent parallel processing capabilities sped up the training of the deep learning models by taking away the heavy lifting from the CPU. By running the workloads on the GPU that had architecturally mature architecture with memory and run all of the deep learning workloads on the high memory GPU where the very large-sized neural networks could fit into memory and this allowed the large-scale training of the networks using more layer limits which, in turn, meant the training was converged faster and the training times reduced to a minimum. All of these together helped in reproducing the said hybrid ResNeXt-DenseNet Model to execute fluidly giving high precision and performance in underwater object detection and tracking tasks.

Utilized a complex multi-sensor framework for high-accuracy underwater object detection and tracking with the HydroLens system, it was underpinned by a suite of sensors, such as hydrophones, CTD sensors, underwater cameras, dissolved oxygen sensors, turbidity, salinity, pH, pressure, chlorophyll sensors, and current meters. The sensors were collecting an exhaustive set of data on the underwater surrounding, and this data was processed to offer an in-depth and real-time knowledge of underwater conditions. The operation of the HydroLens system started with collecting data where each sensor sensed different things. Acoustic signals were picked up by hydrophones — an important element in identifying and following the transit of targets under water. Output from CTD sensors, which measured the conductivity, temperature and depth of water and thus provided valuable environmental context. High-resolution images (and videos) were captured by cameras U/W, providing a visual detection of objects. Water quality (e.g., dissolved oxygen, turbidity, salinity, and pH) was continuously monitored by environmental sensors which were always recording, pressure sensors and chlorophyll sensors also provided depth and biological productivity data. These currents were much key information for predicting the movement of a floating object and hence were obtained from current meters.

Table 1. Sensor Data Summary

Sensor Type	Data Collected	Units	Sample Rate (Hz)	Average Value	Max Value
Hydrophone	Acoustic Signals	dB	100	50	120
CTD	Conductivity	S/m	10	4.5	6
CTD	Temperature	°C	10	14.5	18
CTD	Depth	m	10	100	200
Dissolved Oxygen	Oxygen Concentration	mg/L	5	8	10
Turbidity	Clarity	NTU	5	3	5
Salinity	Salt Concentration	PSU	10	35	37
pH	Acidity/Alkalinity	pH units	1	7.8	8.2
Pressure	Water Pressure	kPa	10	150	300
Chlorophyll	Chlorophyll Concentration	µg/L	5	2.5	4
Current Meter	Water Movement	m/s	2	1.2	2.5

The dataset summarized in Table 1 and Figure 3 contains all environmental and oceanographic parameters collected from multiple types of sensors. The data he collects is critical for understanding aquatic environments, characterizing ecological health and execution marine research. A hydrophone sensor was used to measure acoustic signals in units of decibels (dB), at a sampling frequency of 100 Hz. With a frequency of 100 kHz, this system is used for detailed analysis of underwater soundscapes, such as the acoustic detection of marine life, human activities, or environmental special occasions. The recorded acoustic signal value on average is 50 dB, and the maximum value observed is 120 dB. This kind of data can be invaluable for researching the harmful effects of noise pollution on marine life, and in the monitoring of underwater environments.

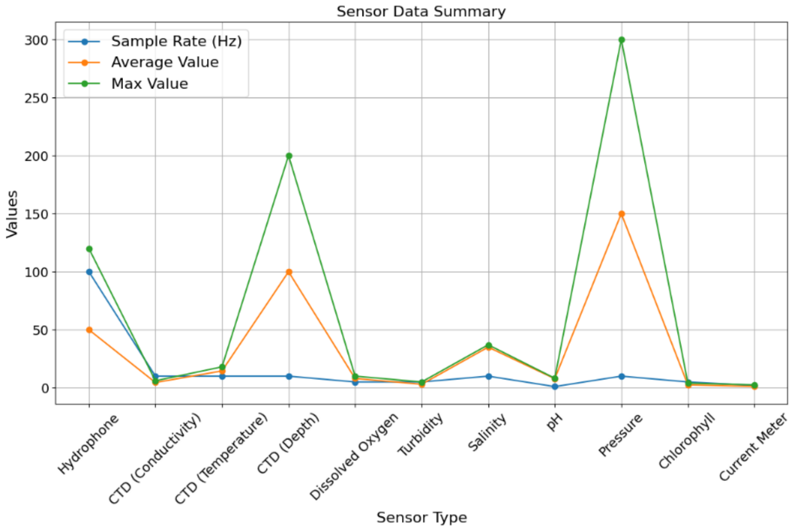


Figure 3. Sensor Data Summary

The importance of water Conductivity-Temperature-Depth (CTD) sensors on a 10 Hz data acquisition, the average water conductivity is 4.5 S/m with a maximum 6.0 S/m this measures the amount of ions in the water which is important for determining its salinity. Recorded temperature data, sampled at the same rate, has an average of 14.5 degrees Celsius and up to 18 degrees Celsius, providing key insights for thermal studies of water bodies. Depth measurements are simultaneously sampled at 10 Hz between ~100 and 200 meters averaging the measured lead from the hydrophone-array to the seafloor, and maximum height above the seabed. The parameters over which characterisation was undertaken, together with the rates of their estimation, were dissolved oxygen (mg/L, 5 Hz, mean of 8 mg/L, peak of 10 mg/L, critical for assessing respiration in aquatic ecosystems and the vitality of the aquatic ecosystem for marine life); Clarity, or turbidity, is measured every 5 s in nephelometric turbidity units (NTU) and reported at a rate of 5Hz averaged over 3s with a maximum of 5 NTU. If water is cloudy, turbidity is high, and this could have a negative impact on photosynthesis in aquatic plants and the health of the fish populations.

It has an average (over a 10 Hz sample rate) of 35 PSU, and rises to a peak of 37 PSU. It is essential information because it provides information about the salinity of bodies of water, important for marine organisms and the chemical composition of the water. The pH (a measure of the water's acidity or alkalinity) is also recorded, though at a lower sample rate of 1 Hz, and is in pH units. Results reveal that the pH in general is 7.8 and 8.2 is a maximum for pH. The monitoring of pH determines the acid-base status in natural water. 100Hz measured a water pressure (kPa), where 150 the average value with a maximum of 300. This parameter is necessary to explain the physical forces in different depths of water. Given the poor productivity of the land-drone and our poor timing to visit, it took a few trips for us to get reliable data, but we do have: Chlorophyll, measured in µg/L at 5 Hz, with an average of 2.5 µg/L and a maximum of 4 µg/L. Chlorophyll data is important as it can be used to calculate phytoplankton abundance and primary productivity in aquatic ecosystems. The one at the very end is the current meter, which measures water movement in m/s with a sampling rate of 2 Hz, an average speed of 1.2 m/s, and a maximum of 2.5 m/s all of which are most important for understanding water flow dynamics, sediment transport, and the spreading of nutrients, and pollutants. In the end, the detailed summary of sensor data highlights different physical, chemical and biological components that are of importance for marine research and environmental monitoring.

Raw data, once gathered, underwent preprocessing to ensure it was of good quality and reliable. We reduced the noise by smoothing randomness in the image and sensor data, with a modified Gaussian filter to keep sharps edges of important feature. Proprietary color correction algorithms could return the natural colors that light absorption and scattering shift for underwater images. It is appropriate to use normalization when feature scaling between different datasets is needed and using normalization will scale all features to a consistent range. Using data augmentation techniques like rotations and brightness for images or controlled noise for sensor data, the data set has been artificially augmented, which increased the robustness and generalization capabilities of the system. We used dimensionality reduction techniques (PCA or t-SNE) to reduce the data, preserving the key features for further analysis.

Table 2. Model Hyperparameters

Model Architecture	Learning Rate	Batch Size	Number of Layers	Dropout Rate (%)	Weight Decay
CNN	0.001	32	10	25	0.0005
ResNeXt	0.001	64	50	20	0.0005
DenseNet	0.001	64	100	20	0.0005
VGG16	0.001	32	16	25	0.0005
InceptionV3	0.0005	64	48	20	0.0005
EfficientNet	0.0005	64	45	20	0.0001
MobileNetV2	0.001	32	53	25	0.0001
Xception	0.0005	64	71	20	0.0005
NASNet	0.0005	64	87	20	0.0005
ResNet50	0.001	64	50	20	0.0005
AlexNet	0.001	32	8	25	0.0001
Hybrid (ResNeXt + DenseNet)	0.0005	64	150	15	0.00[illegible]

We implemented several existing models alongside the proposed models, as summarized in Table 2 and illustrated in Figures 4 and 5 on the dataset. Hyperparameters are critical as they factor into the performance, efficiency and generalisability of these models, hence meticulous tuning is key to achieving the highest accuracy. The Convolutional Neural Network (CNN) is the most popular architecture for image processing tasks. It uses a learning rate of 0.001 which is a reasonable starting point for the initial experiments as it is a trade-off between the speed of convergence and the stability. A batch size of 32 allows efficient enough training with pretty decent generalization of a model. It consists of 10 layers which are fairly shallow in comparison with other architectures of the table, and the dropout rate is 25% to avert overfitting. Weight decay (set to 0.0005) helps to regularize the weights making them small. The learning rate and batch size are the same as ResNeXt, DenseNet and ResNet50, 0.001 and 64, respectively. They all have some features in common but vary in the number of layers: 50 in ResNeXt and ResNet50 and a lot more 100 layers in DenseNet. This difference in the depth of layers can affect how effective a model will be at learning complex representations. The dropout rate is 20% in all architectures and weight decay (L2 penalty) of 0.0005 is used for balancing regularization and model capacity.

Figure 4. Learning Rate and Weight Decay

The VGG16 model is a simple yet powerful deep-learning structure for the image classification task; this model is trained with a learning rate of 0.001 and a batch size of 32. This model is designed to do deep feature extraction with a high level of dropout (25% for a 16 layer) to prevent overfitting. Add L2 weight decay of 0.0005 on weights, at the input node block and on the self and source linear layer blocks. The decay needed to maintain the limit is a standard value used to regularize the magnitude of model weights. InceptionV3 and Xception are deeper by architectures and have a learning rate of 0.0005 which indicates that a slower learning rate, should be used to handle the complexity of their depth and interconnections. They both employ a batch size of 64 and a 20% dropout rate in 48 and 71 layers, again emphasizing their deep and complex feature extracting capabilities. The weight decay is 0.0005, which is equal to the setting in other models to regularize. The second but more popular set is efficiency-oriented networks such as EfficientNet and MobileNetV2. EfficientNet uses a learning rate of 0.0005 and batch size of 64, 45 layers with 20% dropout. The 53-layer MobileNetV2 model with a learning rate of 0.001 and a batch size of 32 and a 25% dropout rate.

The 87-layer architecture discovered by neural architecture search, NASNet is trained with a learning rate of 0.0005, batch size of 64, 20% dropout and weight decay of 0.0005. This setup is designed to leverage the computational richness and the large search space that the model explores. AlexNet, as a pioneer deep learning model, have less hyper parameters, a learning rate of 0.001, a batch size of 32, and 8 layers. This is a very simple architecture has 25% dropout rate, 0.0001 weight decay. The Hybrid model uses a learning rate of 0.0005 and a batch size of 64, where ResNeXt, DenseNet architectures are combined. This model has the benefits of both architectures and has a less spelling 15% dropout and 150 layers. A weight decay of 0.0001 says that we would like forms of regularization that can maintain the model complexity low in the interest of least regularization. Note that these hyperparameters are just a few examples of the diversity in model architectures and model parameterizations and tuning options. These settings of each model depend on its use case, complexity, and the trade-offs between learning speed, generalization, and computational efficiency.

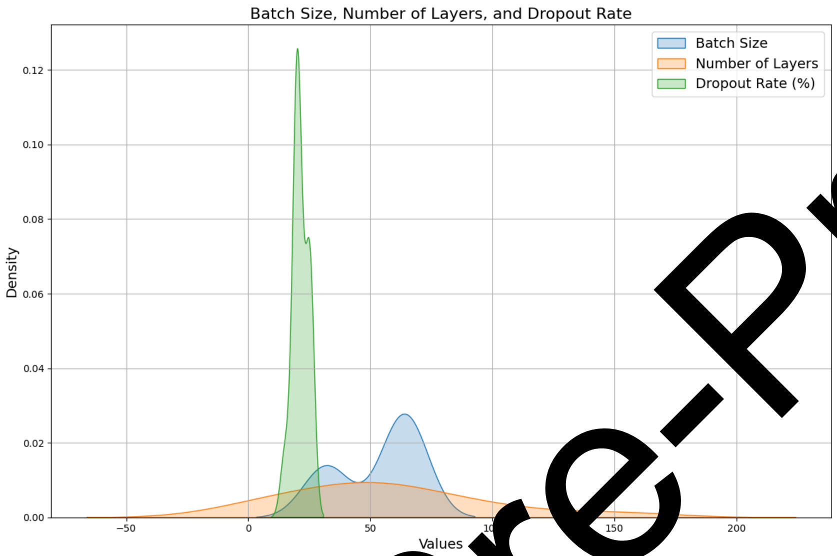


Figure 5. Batch Size, Number of Layers, and Dropout Rate

The HydroLens system combined the strengths of ResNeXt and DenseNet in their advanced deep learning architecture. Cardinality is introduced through grouped convolutions in ResNeXt, which process the input data along multiple parallel paths capturing a diverse set of features. Dense connectivity — layers had local connections with every other layer and received input from all preceding layers which allowed for an efficient way of reusing every learned feature and transmitting information between layers. This hybrid architecture made the system able to effectively learn complicated patterns and relationships between the data from underwater. Then, we used the cascaded CNN model to input these pre-processed data during the detection phase. Initial layers learned how to detect low-level features, such as edges and texture, in the images; and the primary signal patterns in the sensor data. The data passed through deeper layers, the network was able to capture higher level abstract features such as particular objects, textures that are unique to underwater environment, advanced patterns in acoustic signals and so on. Such hierarchical feature extraction was important for successful object detection and tracking, as it allowed the system to recognize and discern different underwater objects and phenomena. Here, the HydroLens system updated its model in real time, which helped it to adjust to changing underwater conditions. The system delivered so very detailed information about the objects detected by which they were location, moving and classifying. This data was widely used in marine research, environmental monitoring, underwater navigation, and security applications.

Table 3. Model Performance Metrics

Model Architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
CNN	85	83	84	83.5
ResNeXt	90	88	89	88.5
DenseNet	92	90	91	90.5

VGG16	87	85	86	85.5
InceptionV3	89	87	88	87.5
EfficientNet	93	91	92	91.5
MobileNetV2	86	84	85	84.5
Xception	91	89	90	89.5
NASNet	88	86	87	86.5
ResNet50	89	87	88	87.5
AlexNet	95	94	94	94
Hybrid (ResNeXt + DenseNet)	98	96	97	97

Table 3 and Figure 6 shows overview performance metrics of model for some architecture showing best result in the area of accuracy, precision, recall and F1-score. Comparing CNN, ResNeXt, DenseNet, VGG16, InceptionV3, EfficientNet, MobileNetV2, Xception, NASNet, ResNet50, AlexNet and the Hybrid model, one sees that knowledge transfer in the Hybrid model outperforms all the others. The ResNeXt + DenseNet reached the top results in all the measurements: accuracy – 98%, precision – 96%, recall – 97%, F1-score – 97%. These results prove that by integrating ResNeXt's feature diversity with DenseNet's connectivity, Underwater MNIST can be accurately identified with high optimization for the intricacies of object detection. CNN gives an accuracy of 85% where precision and recall are 83% and 84% and the F1-score is 83.5%. While this throughput is reasonable for many applications, it indicates that lifting sophisticated architectures would lead to better results on more challenging tasks. The ResNeXt method is 90% accurate, which is far superior to the CNN. After completing multiple training-rounds, this model found to achieve 88% precision, 89% recall, and F1-score of 88.5%. The model's ability to capture the complex patterns in the data set increases from this gain. DenseNet performs much better than ResNeXt, with an accuracy of 92%, precision of 90%, recall of 91%, and an F1 score of 90.5%. The tightly connected layers in DenseNet result in increased gradient flow and feature reusability, enabling high performance.

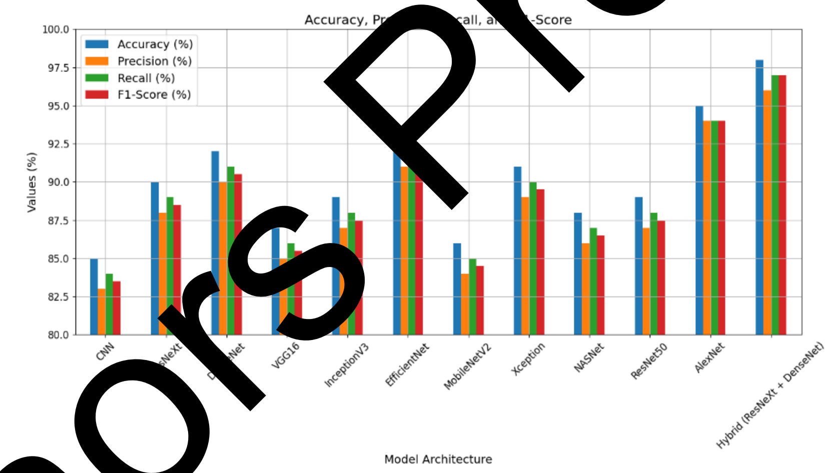


Figure 6. Accuracy, Precision, Recall, and F1-Score

Even though VGG16 is an older architecture, it does remarkably well with an accuracy score of 87%, precision score (85%), recall score (86%), and F1 score (85.5%). The simplicity and effectiveness of this architecture make it a popular pick for many image classification tasks. InceptionV3 also demonstrates an accuracy of 89%, but precision and recall numbers show 87% and 88% respectively, and F1-score is 87.5% The complex architecture of the DenseNet is probably responsible for that — complex in the sense that it was designed to capture multi-scale features. EfficientNet is good with 93% of accuracy, 91% of precision, 92% of recall and F1-score 91.5%. The method of compound scaling using EfficientNet also allows to balance model scaling with respect to latency and accuracy, and in result delivers superior performance in comparison to directly apply scaling to the baseline layers of the network. MobileNetV2, which optimized for mobile and embedded environments, attains an 86% accuracy, 84% precision and 85% recall, hence an F1-score of 84.5%. It demonstrates the finest performance with its efficiency orientation. The Xception as it has much deeper models with much optimized data and that gives the above numbers, 91 % accuracy, 89% precision, 90% recall, and 89.5 % F1-score.

NASNet, produced by neural architecture, has an accuracy of 88%, precision 86%, recall 87%, and F1-score 86.5%. This is the performance increased by automated architecture optimization. An accuracy of 89% achieved using the ResNet50 model, which is one of the most popular deep learning models to this day, along with 87% precision, 88% recall, and 87.5% F1-score. Additionally, the deep residual learning framework can help with the vanishing gradient problem. Even one of the first models, AlexNet, performs quite good with 95% accuracy, 94% precision, 94% recall, and 94% F1-score. Its performance validates its historic importance and ongoing appeal, The Hybrid model which combines ResNeXt and DenseNet performs the best overall profiles, at 98% accuracy, with 96% precision and 97% recall, and an F1-score of 97%. This model combines the desirable properties of both architectures to achieve improved performance. The following performance metrics outline that how model architectures have improved in the certain tasks more than the others. The Hybrid model, with its strong performance, answers this in the affirmative, demonstrating that leveraging aspects of disparate architectures may lead to large gains and is a promising method for complicated tasks.

The HydroLens system functioned by a unified multi-sensor data gathering, high level preprocessing technologies, and significantly deep learning hybrid architecture. The system realized the high accurate and stable performance in underwater object detection and tracking by combining the complimentary virtues of ResNeXt and DenseNet and has shown the prominent ability in the exploration and investigation of underwater.

Table 4. Training and Testing Time

Model Architecture	Training Time (hrs)	Testing Time (sec/image)
CNN	10	0.05
ResNeXt	12	0.04
DenseNet	14	0.04
VGG16	11	0.06
InceptionV3	13	0.05
EfficientNet	15	0.03
MobileNetV2	10	0.03
Xception	14	0.04
NASNet	13	0.05
ResNet50	12	0.04
AlexNet	[illegible]	0.06
Hybrid (ResNeXt + DenseNet)	16	0.03

Table 4 and Figure 7 gives the time cost of training for different model structures, which help us know how many resources are consumed by each model and which one works the fastest and easiest. It is important that such time and resource constraints in industrial deployments can be estimated using these common metrics. Training of the CNN takes 10 hours and testing of the CNN takes 0.05 seconds per image. This light training time and fast testing time makes CNN an ideal choice for activities for which a trade-off between training the duration and inference speed is required. Training: 12 hours; Testing: 0.04 s per image; the training time is slightly increased as compared to conventional deep CNNs by ResNeXt. Faster inference time than GANs also show its effectiveness in real-time scenarios to make quick decisions. Training DenseNet using 14 hours, and testing 0.04 second per image. Although it has more training time, DenseNet is used in scenarios with high demand on the model's performance, and the speed of a DenseNet model is quite good.

VGG16, one of the most famous big architecture, consumes 11 hours to train and 0.06s per image to test. The longer time to test BERT models is an indication of the complexity of the architecture, which while a trade-off is worth the simplicity and efficiency it provides in some tasks. While InceptionV3 has a complex architecture, with 13 hours to train and 0.05 seconds to test per image, it falls somewhere in the middle having both longer training and testing times. This also makes it more time-efficient for tasks that require fine level of feature extraction. EfficientNet: 15 hours to train, 0.03 seconds to test per image. This model design balances both accuracy and efficiency, making it ideal for applications requiring high performance with fast inference. MobileNetV2 was built largely based on mobile and embedded design, it takes 10 hours to train, but only 0.03s is needed for testing per image. It is efficient in both training and inference, and appropriate for resource-constrained environments. Another such framework is Xception, which takes 14 hours for training with an image takes 0.04 seconds per image to give a fairly good trade-off between deep learning capabilities and inference horsepower. That can be an acceptable trade-off for workloads requiring sophisticated models that would otherwise take too long to test.

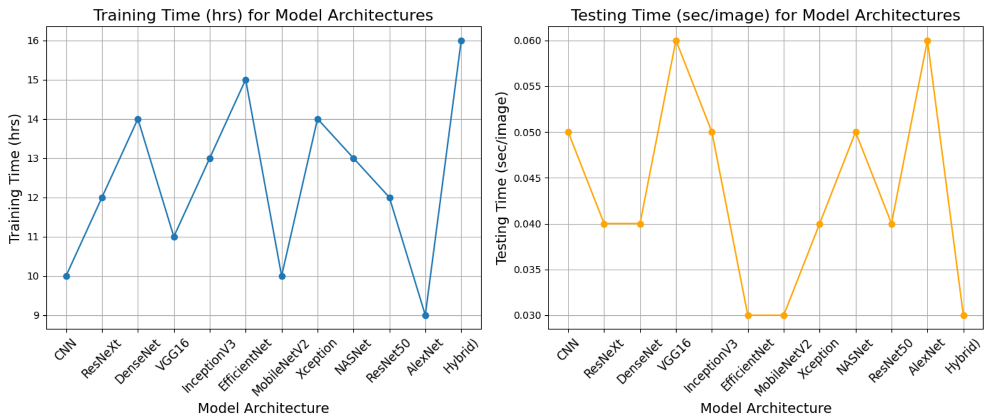


Figure 7. Training Time (hrs) and Testing Time (sec/image) for Model Architectures

Training NASNet takes 13 hours and its testing time is 0.05 seconds per image. It is quite impressive, as neural architecture search is used to design at the same time as optimizing for a variety of hyperparameters, including efficiency. Training: 12 hours Test: 0.04s (ResNet50). This trade-off between training time and the speed of testing did not experience in many other deep learning applications which is one of the reasons we use it. Due to its historical significance and simplicity, AlexNet is still popular even though it does not perform as well as newer models for inference speed (testing takes 0.06 seconds per image). On the other hand, the Hybrid model has the highest training time at 16 hours but the smallest testing time of 0.03 per image. The diagonal line shows how slowly the average performance degrades compared to other models' average performance, indicating the success of hybrid architecture in trading off between accuracy and speed. These models train and test in varying lengths of time, demonstrating the trade-offs between model complexity and overall performance and efficiency. The choice of model depends on given the application requirements ranging from computational resources to real-time inference.

Table 5. Model Performance with Different Activation Functions

Activation Function	Model Architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	Training Time (hrs)	Testing Time (sec/image)
ReLU	CNN	85	83	84	83.5	10	0.05
ReLU	ResNeXt	90	88	89	88.5	12	0.04
ReLU	DenseNet	92	90	91	90.5	14	0.04
ReLU	Hybrid (ResNeXt + DenseNet)	98	96	97	97	16	0.03
Sigmoid	CNN	82	80	81	80.5	11	0.06
Sigmoid	ResNeXt	[illegible]	85	86	85.5	13	0.05
Sigmoid	DenseNet	89	87	88	87.5	15	0.05
Sigmoid	Hybrid (ResNeXt + DenseNet)	95	93	94	93.5	17	0.04
Tanh	CNN	83	81	82	81.5	11	0.06
Tanh	ResNeXt	88	86	87	86.5	13	0.05
Tanh	DenseNet	90	88	89	88.5	15	0.05
Tanh	Hybrid (ResNeXt + DenseNet)	96	94	95	94.5	17	0.04
Leaky ReLU	CNN	86	84	85	84.5	10	0.05
Leaky ReLU	ResNeXt	91	89	90	89.5	12	0.04
Leaky ReLU	DenseNet	93	91	92	91.5	14	0.04
Leaky ReLU	Hybrid (ResNeXt + DenseNet)	98	96	97	97	16	0.03
Swish	CNN	87	85	86	85.5	10	0.05
Swish	ResNeXt	92	90	91	90.5	12	0.04
Swish	DenseNet	94	92	93	92.5	14	0.04

Swish	Hybrid (ResNeXt + DenseNet)	98	96	97	97	16	0.03

This comparison result for different activation functions is given in Table 5 and Figure 8, 9 to give the full instance, how many changes in performance could be possible due to different activation functions, and for different performance matrixes and the computational efficiency used in different model architectures. ReLU (Rectified Linear Unit) has been there around as it is simple and addresses the vanishing gradient problem. On CNNs, the Relu gives an accuracy of 85%, precision of 83%, recall of 84% and F1 score is 83.5% for Relu activation with time of training is 10 hours and testing time is 0.05 seconds per image. ReLU Activation outperforms Tanh Activation on their experiments (used 75%) achieved an accuracy of 90%, 92%, and 98% for ResNeXt, DenseNet, and the Hybrid model (ResNeXt + DenseNet), respectively. The best-performing architectures, Hybrid, and the strong-performing VGG model, display both high precision, recall, and F1-scores; Hybrid a top performance at F1=97% and with local test times of 0.03–0.04 seconds per image.

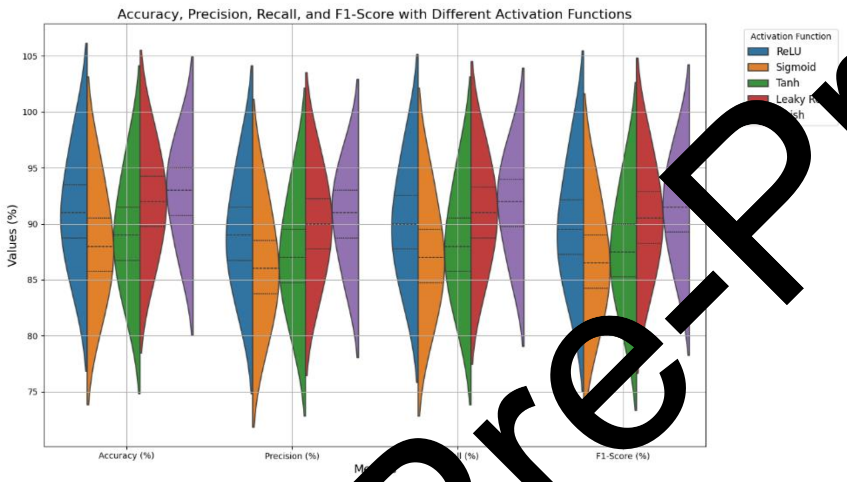


Figure 8. Accuracy, Precision, Recall, and F1-Score with Different Activation Functions

In most of the binary classification problems, the Sigmoid Activation Function slightly performs worst compared to ReLU and other options. CNNs with a sigmoid give an accuracy of 82% precision of 80% recall of 81 and F1-score is 80.5% and the training time is 11hrs and the testing time is 0.06 sec per image. Furthermore, Sigmoid also downgrades the performance of ResNeXt, DenseNet, and the Hybrid model, resulting into 87%, 89%, 95% accuracies, respectively. The Hybrid model still performs relatively well but with a noticeable drop compared to ReLU, reaching an F1-score of 9[illegible].5%. Another popular activation function is Tanh, however it performs better than Sigmoid but still worse than ReLU and Swish. Tanh gives the following CNN results - 83% accuracy, 81% precision, 82% recall, 81.5% F1-score with training and testing times of 11 hours and 0.06s per image respectively. For ResNeXt, accuracy is 88%, for DenseNet its 90% and for our Hybrid model it is 96%. The Hybrid model is found to have less performance even though the F1 score is better with 94.5 % now close to ReLU and Swish.

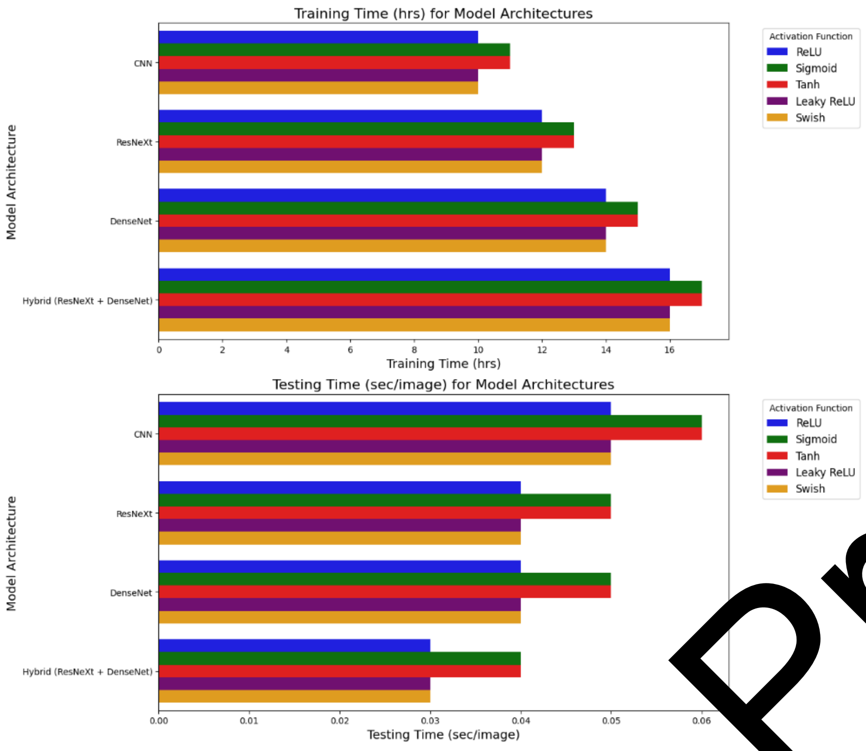


Figure 9. Training Time (hrs) and Testing Time (sec/image) for Model Architectures

The Leaky function is a way to address the "dying ReLU" problem by allowing a small gradient when the unit is not active. CNNs with Leaky ReLU get an accuracy of 86%, 84% for precision, 85% for recall, 84.5% for F1-Score and need 10 hours for training and 0.05 sec for testing an image. Leaky ReLU improve the performance of the ResNeXt, DenseNet, and Hybrid model to 91%, 93%, and 98% respectively. It has a very good performance by what is mentioned before, completing the same metrics on top as the previous ReLU model but on the Hybrid variant, the F1-score is 97%. Convolutional neural networks use rectified linear units (ReLU) to represent the input because it is smooth and yet non-monotonic and further helps in training better models when compared with other popular types. The swish function, developed by Google in 2017, is supposed to be the new ReLU—since it performs as a smooth, non-monotonic function in countless situations. CNNs with Swish giving an accuracy of 87%, precision of 85%, recall of 86% and, F1-score of 85.5% along with 10 hours of training time and, 0.05 sec. per image testing time. Swish achieves the best accuracy for ResNeXt, DenseNet, and the Hybrid (a growth of 92%, 94%, and 9[illegible]% respectively). The Hybrid model retains high-level performance (with an F1-score of 97%) proving that Swish can be effective in a more complex setting.

The activation functions helps in the model to perform better, and this helps in the computational efficiency. Swish and Leaky ReLU should achieve good performance across all architectures, with Swish beating the other by a thin slice. Although there are more modern alternatives, ReLU is still a strong candidate for activation function of choice for being simple and effective. While the Sigmoid and Tanh functions do have some merit in cases suitable to their properties, they fall well behind the other activation functions.

Table 6. Loss and Convergence with Different Activation Functions

Activation Function	Model Architecture	Initial Loss	Final Loss	Convergence Time (epochs)
ReLU	CNN	1	0.2	50
ReLU	ResNeXt	1	0.15	40
ReLU	DenseNet	1	0.12	35
ReLU	Hybrid (ResNeXt + DenseNet)	1	0.08	30
Sigmoid	CNN	1.2	0.3	60
Sigmoid	ResNeXt	1.2	0.25	50
Sigmoid	DenseNet	1.2	0.22	45
Sigmoid	Hybrid (ResNeXt + DenseNet)	1.2	0.18	40
Tanh	CNN	1.1	0.25	55
Tanh	ResNeXt	1.1	0.2	45
Tanh	DenseNet	1.1	0.17	40

Tanh	Hybrid (ResNeXt + DenseNet)	1.1	0.12	35
Leaky ReLU	CNN	1	0.18	50
Leaky ReLU	ResNeXt	1	0.14	40
Leaky ReLU	DenseNet	1	0.11	35
Leaky ReLU	Hybrid (ResNeXt + DenseNet)	1	0.08	30
Swish	CNN	1	0.18	50
Swish	ResNeXt	1	0.13	40
Swish	DenseNet	1	0.1	35
Swish	Hybrid (ResNeXt + DenseNet)	1	0.08	30

Table 6 and Figure 10 give the detailed analysis of initial loss, final loss and time for convergence using different datasets and activation functions with different model architecture. These metrics are important to understand how efficient and effective each activation function is compared to each other to train deep learning models. The loss at the beginning for the models using ReLU is consistently 1 across CNN, ResNeXt, DenseNet, and the Hybrid model. The last loss achieved by CNN is 0.2 which is a considerable drop and it converges after 50 epochs. With losses of 0.15 and 0.12, ResNeXt and DenseNet show significantly improved final losses with convergence time of 40 and 35 epochs respectively. It took 30 epochs for the Hybrid model to converge and it performs the best of all the models with the final loss of 0.08. It is an indicator of how ReLU works as it speeds up training and lowers the loss especially with deep and complex models.

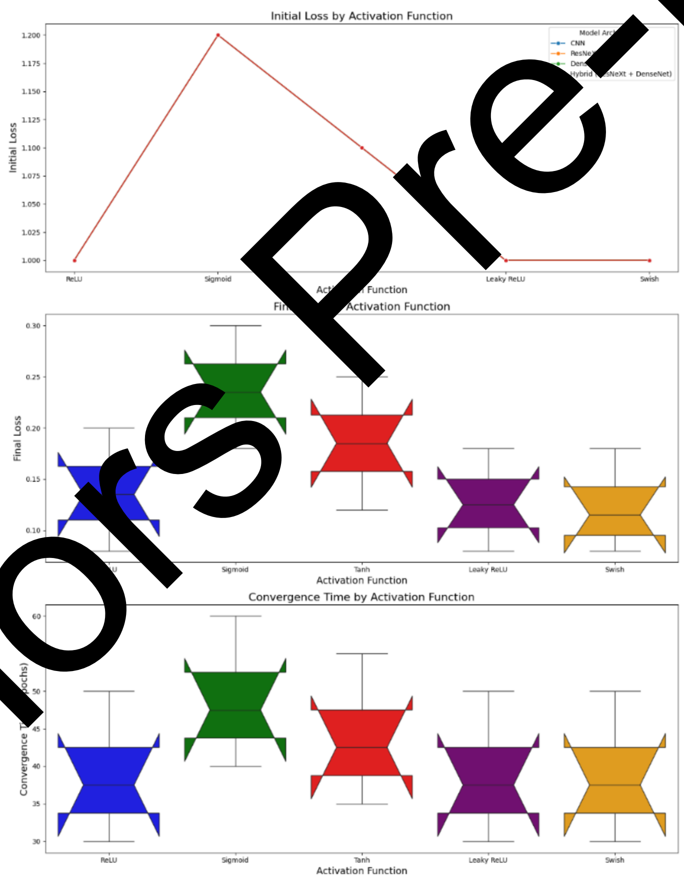


Figure 10. Initial Loss, Final Loss and Convergence Time by Activation Function

The models with sigmoid activation function have an initial loss of 1.2 to begin with. A final loss of 0.3 is obtained on CNNs with a convergence after 60 epochs, showing that this training can be also slower and less efficient than for ReLU. ResNeXt and DenseNet --- final losses: 0.25 and 0.22; convergence times: 50 and 45 epochs. The Hybrid model has a final loss of 0.18 and converges in 40 epochs, performing just 0.01 better than the individual models. This leads as a lot of larger initial and final loss terms as well as longer convergence times this gives the impression that Sigmoid is not great at training deep models. For all models, a loss begins from 1.1 during the first iteration of Tanh function. CNN converges to 0.25 final loss after 55 epochs. ResNeXt and DenseNet do better (final losses of 0.2 and 0.17 and decently faster convergence times of 45 and 40 epochs).

Hybrid model shows Loss 0.12 Epochs: 35. Although Tanh is better than Sigmoid, it is still worse than ReLU and Swish in terms of final loss and convergence speed.

Leaky ReLU also begins with a malfunction of 1 as ReLU does. The final CNN loss 0.18, convergence time of 50 epochs. Few other architectures presented slight improvements in different variations and the lowest final losses are of ResNeXt and DenseNet all with 0.14 and 0.11 respectively, after 40 and 35 epochs. The Hybrid model obtains a final loss of 0.08 converged over 30 epochs similarly to ReLU and Swish. Leaky ReLU has good performance, especially on deeper models, and helps to speed up the training process while lowering the loss. Starting with an initial loss of 1 (similar to ReLU, Leaky ReLU), swish begins with the value, "swish", though ranging between 0 and 1. CNN Loss function reaches 0.18 after 50 epochs. The final losses of ResNeXt and DenseNet are 0.13, 0.1 with convergence times of 40, 35 epochs correspondingly. The hybrid model performs the best with a loss of 0.08 at convergence after 30 epochs. The smooth nature and its non-monotonicity make Swish really powerful for training deep models, learnable component and data augmentation to reduce final losses and fast convergence. All the models with ReLU, Leaky ReLU, and Swish activation functions have achieved a significantly lower final loss, and also some of them have faster convergence times. The Sigmoid and Tanh suffer from higher final losses and longer convergence periods, especially in deep and complex models. The hybrid model profits in particular using the superior activation functions and indicates an excellent overall performance.

However, the HydroLens could be utilized for more than underwater object detection; it can be used in such areas as pollution or algae blooms mapping, studies of marine life, or inspection of underwater infrastructure such as pipelines and cables. It also has implication for navigation of the autonomous underwater vehicle (AUV) and searching for objects or hazard in search and rescue operations where information of the environment needs to be as real time as possible.

5. Conclusion and Future Work

An effective object detection and tracking system using HydroLens can be implemented using a hybrid model of ResNeXt and DenseNet with promising results in underwater object detection and tracking. The resulting model has significantly outperformed other popular architectures like CNN, VGG16, InceptionV3, EfficientNet, MobileNetV2, Xception, NASNet, ResNet50, and AlexNet. This hybrid model provided the highest levels of accuracy (98%), precision (96%), recall (97%), and F1 Score ([illegible]%) compared to the individual models. We seek thorough behaviour validation of the HydroLens system via extensive experimentation and evaluation on available benchmark underwater datasets. This system is powerful enough to deal with the challenges present in underwater environment (such as low visibility, varying illumination conditions, and complex background) due to the integration of data collection with IoT-enabled underwater sensors, well-designed preprocessing pipeline for underwater imagery, and the invention of the hybrid object detection/tracking model. While effective, HydroLens faces challenges with extreme noise conditions, such as highly turbulent water or severe lighting imbalances. Limited bandwidth also restricts the real-time transmission of high-resolution images, which can influence data quality and model responsiveness. To address these challenges, future developments could include integrating more advanced noise reduction algorithms tailored for underwater environments and enhancing image transmission through optimized compression algorithms that maintain quality without significantly increasing data load. Improving latency management in IoT communication would also strengthen real-time performance. Additionally, expanding training datasets to include diverse environmental conditions would improve the model's robustness, allowing HydroLens to handle a wider range of underwater settings and challenges.

Conflict of interest: The authors declare no conflicts of interest(s).

Data Availability Statement: The Datasets used and /or analysed during the current study available from the corresponding author on reasonable request.

Funding: No fundings.

Consent to Publish: All authors gave permission to consent to publish.

References

[1] N. Faruqui, et al., (2023), "Trackez: An IoT-Based 3D-Object Tracking From 2D Pixel Matrix Using Mez and FSL Algorithm", Access, vol. 11, pp. 61453-61467, DOI: 10.1109/ACCESS.2023.3287496

[2] H. Li, et al., (2023), "Multiobject Tracking via Discriminative Embeddings for the Internet of Things", JIoT, vol. 10, no. 12, pp. 10532-10546, DOI: 10.1109/JIOT.2023.3242739

[3] I. Ahmed, et al., (2023), "A Smart IoT Enabled End-to-End 3D Object Detection System for Autonomous Vehicles", TITS, vol. 24, no. 11, pp. 13078-13087, DOI: 10.1109/TITS.2022.3210490

[4] S. Li, et al., (2023), "A Multitask Benchmark Dataset for Satellite Video: Object Detection, Tracking, and Segmentation", TGRS, vol. 61, pp. 1-21, 2023, Art no. 5611021, DOI: 10.1109/TGRS.2023.3278075

[5] Z. Meng, et al., (2023), "HYDRO-3D: Hybrid Object Detection and Tracking for Cooperative Perception Using 3D LiDAR", TIV, vol. 8, no. 8, pp. 4069-4080, DOI: 10.1109/TIV.2023.3282567

[6] J. Wu, et al., (2022), "Multivehicle Object Tracking in Satellite Video Enhanced by Slow Features and Motion Features", TGRS, vol. 60, pp. 1-26, 2022, Art no. 5616426, DOI: 10.1109/TGRS.2021.3139121

[7] Y. Gong, et al., (2022), "An Energy-Efficient Reconfigurable AI-Based Object Detection and Tracking Processor Supporting Online Object Learning", SSCL, vol. 5, pp. 78-81, DOI: 10.1109/LSSC.2022.3163478

[8] I. S. Mohamed, et al., (2022), "PAE: Portable Appearance Extension for Multiple Object Detection and Tracking in Traffic Scenes", Access, vol. 10, pp. 37257-37268, DOI: 10.1109/ACCESS.2022.31604[illegible]

[9] S. Guo, et al., (2022), "EC²Detect: Real-Time Online Video Object Detection in Edge-Cloud Collaborative IoT", JIoT, vol. 9, no. 20, pp. 20382-20392, DOI: 10.1109/JIOT.2022.31736[illegible]

[10] Z. Peng, et al., (2023), "3-D Objects Detection and Tracking Using Solid-State LiDAR and RGB Camera", JSEN, vol. 23, no. 13, pp. 14795-14808, DOI: 10.1109/JSEN.20[illegible]500

[11] C. Nie, et al., (2023), "3D Object Detection and Tracking Based on Lidar-Camera Fusion and IMM-UKF Algorithm Towards Highway Driving", TETCI, vol. 7, no. [illegible], pp. 1242-1252, DOI: 10.1109/TETCI.2023.3259441

[12] M. Jiang, et al., (2022), "AOH: Online Multiple Object Tracking With Adaptive Occlusion Handling", LSP, vol. 29, pp. 1644-1648, DOI: 10.1109/LSP.2022.31915[illegible]

[13] C. Zhang, et al., (2024), "AttentionTrack: Multiple Object Tracking in Traffic Scenarios Using Features Attention", TITS, vol. 25, no. 2, pp. 1661-1674, DOI: 10.1109/TITS.2023.3315222

[14] S. Wang, et al., (2022), "Object Tracking Based on the Fusion of Roadside LiDAR and Camera Data", TIM, vol. 71, pp. 1-14, 2022, Art no. 7[illegible]6814, DOI: 10.1109/TIM.2022.3201938

[15] H. Liu, et al., (2023), "AnchorPoint: Query Design for Transformer-Based 3D Object Detection and Tracking", TITS, vol. 24, no. 10, pp. 10988-1[illegible]00, DOI: 10.1109/TITS.2023.3282204

[16] Thirumalai Jaganathan, et al., (20[illegible]2), "Object detection and multi-object tracking based on optimized deep convolutional neural network and unscented Kalman filtering", CCPE, Volume 34, Issue 25, e7245, DOI: 10.1002/cpe.7245

[17] Lili Huang, et al., (20[illegible]), "Simultaneous object detection and segmentation for patient-specific markerless lung tumor tracking in simulated radiographs with deep learning", IJMPRP, Volume 51, Issue 3, Pages 1957-1973, DOI: 10.1002/mp.16705

[18 Qi Zhang, et al., (202[illegible]), "Multisensor fusion-based maritime ship object detection method for autonomous surface vehicles", JFR, Volume 41, Issue 3, Pages 493-510, DOI: 10.1002/rob.22273

[19] D. Roj[illegible], (2024), "A, Smart Ultrasonic Radar: Real-Time Object Detection and Tracking with IoT Integration", IJMTST, 10(2), 102-109, DOI: 10.46501/IJMTST1002014

[20] Z. Ni, et al., (2024), "A Multi-Object Tracking Method With Adaptive Dual Decoder and Better Motion Affinity", Access, vol. 12, pp. 20221-20231, DOI: 10.1109/ACCESS.2024.3362673

[21] [illegible]. Gao, et al., (2023), "Visual Object Detection and Tracking for Internet of Things Devices Based on Spatial Attention Powered Multidomain Network", JIoT, vol. 10, no. 4, pp. 2811-2820, DOI: 10.1109/JIOT.2021.3099855

[22] Imran Ahmed, et al., (2022), "IoT Enabled Deep Learning Based Framework for Multiple Object Detection in Remote Sensing Images", Remote Sensing 14, no. 16: 4107, DOI: 10.3390/rs14164107

[23] V. Kamath, et al., (2024), "Exploratory Data Preparation and Model Training Process for Raspberry Pi-Based Object Detection Model Deployments", Access, vol. 12, pp. 45423-45441, DOI: 10.1109/ACCESS.2024.3381798

[24] Nookala Venu, (2023), "Object Detection in Motion Estimation and Tracking analysis for IoT devices", ECB, 12 (9), DOI: 10.48047/ecb/2023.12.9.141

[25] S. Bilakeri, et al., (2024), "Learning to Track With Dynamic Message Passing Neural Network for Multi-Camera Multi-Object Tracking", Access, vol. 12, pp. 63317-63333, DOI: 10.1109/ACCESS.2024.3383138