Evaluation of Deep Neural Networks for Underwater Imagery Segmentation

1st Ashvin Pidaparti
University of Minnesota Twin Cities
Minneapolis, USA
pidap008@umn.edu

2nd Pranav Julakanti
University of Minnesota Twin Cities
Minneapolis, USA
julak004@umn.edu

3rd Shilpa Deshpande
University of Minnesota Twin Cities
Minneapolis, USA
deshp143@umn.edu

***Abstract*—Image segmentation is a critical process of identifying and classifying sub patterns in an image. It is widely used in Robotics, Autonomous vehicles, Medical Imaging, face detection, machine vision and object detection. Several algorithms and techniques have been developed for image segmentation and there is no general solution to the image segmentation problem, these techniques have to be combined with domain knowledge in order to effectively solve an image segmentation problem for a given domain. This paper presents a comparative study of some of the segmentation techniques. The results shown for SegNet, U-Net, and SUIMNet could not be accurately reproduced, but of all of them, SegNet achieved the highest accuracy both visually and in terms of the loss function over a validation set, and SUIMNet performed best in time taken to validate.**

I. Introduction

Image segmentation refers to the process of segmenting an image in such a way that the result is relevent, either to the human eye or computationally. This can refer to identifying the pixels that make up a known object as opposed to a background, identifying the hue or saturation of a region of an image, or determining the patterns of an image, among other applications.

Image Segmentation is a key element in the application of several fields. In the field of autonomous robotics, a robot takes actions based on its surroundings, including tasks such as path planning, determining goal positions, and many more. For all of these, accurate sensing of occupied space in the field is key.

While there exist laser-based (LiDar) techniques [3] for determining the occupied space in a field, this task is made more complicated when a robot is underwater, as the robot has several more degrees of freedom than a land based robot. Additionally, as the name suggests, laser based techniques use the time taken for a laser to reflect off an object back to a sensor to determine the distance to an object, however water can lead to distortion, resulting in poor occupancy grid population. An underwater robot must be able to determine the occupied volume, rather than the occupied area. As such, researchers have attempted to apply image segmentation in underwater robotics in an attempt to determine the volume that an object occupies out of the space a robot could potentially occupy.

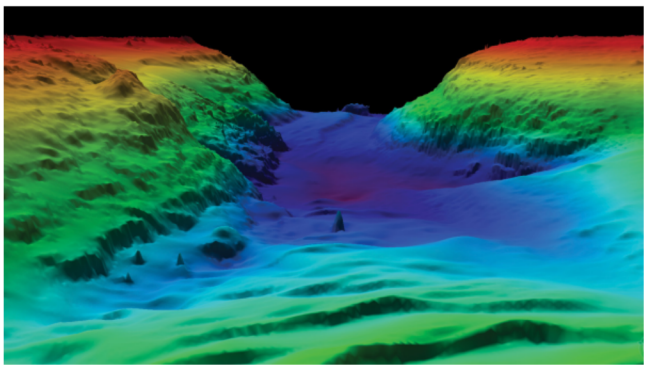

Fig. 1. A 3 dimensional occupancy map created via LiDar. Image from Jeff Moser via powerandmotoryacht.com.

When segmentation techniques are applied with

two or more cameras, depth can be estimated and a point cloud representing occupied space in a 3-dimensional space can be established. This allows for underwater robotics to easily utilize the advantage of existing in 6 degrees of freedom.

Similar to the case of robotics, an autonomous vehicle must understand where, specifically, in space an obstacle exists [4]. While real time object detection significantly aids the development of autonomous robotics, path planning and obstacle avoidance can be made much more efficient through segmentation. For instance, a driverless car driving down a street must understand where pedestrians are in space. If there is no place to stop, the vehicle can be made more efficient by understanding what space is occupied by a pedestrian, rather than a bounding box around a pedestrian, in that, the vehicle may not need to brake as hard or slow down as much if the pedestrian has exited the path but the bounding box has not. In this way, segmentation, as opposed to detection, can be used to make autonomous vehicles more efficient.

Fig. 2. A driverless car performing image segmentation. As described, if a pedestrian were to cross the street, a bounding box would allow the car to avoid the pedestrian, however it may lead to the car slowing too much, resulting in a harder or longer brake than necessary. Image from Dhanoop Karunakaran via Medium.com article.

Beyond autonomous vehicles and robotics, medical imaging benefits highly from image segmentation. One of the key benefits of medical image segmentation is that it allows for a more precise analysis of anatomical data by isolating only necessary areas. It is necessary to segment out certain things, for example in the hip and the knee. Segmentation offers the benefit of removing any unwanted information from the scan and isolating different tissues and bones. When working with CT, MRI, and other types of scans, segmentation generally works by taking information from the background image data and using it to generate a mask. Depending on the task, users may work on their scans in 2D or 3D.

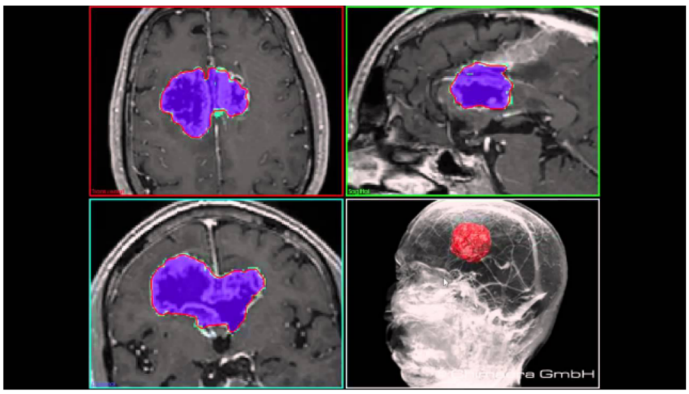

Fig. 3. An example of image segmentation for medical imaging. This model identifies brain tumors from Magnetic Resonance Imaging (MRI).

One of the primitive methods of implementing image segmentation is simply using the color of an image. The classes of images must be set prior to processing an image. The nearest neighbor of the pixel value of an image to a class is treated as the classification.

A more advanced method of image segmentation applies some form of edge detection to separate regions of an image. There exist several methods of edge detection, including the Canny edge detector [7], among others. Upon identifying the edges within an image and treating the boundaries of an image as an edge, a pixel is segmented into the same class as all pixels within the boundaries of the edge they are contained in.

In recent times, deep learning based methods of image segmentation have become increasingly popular. The problem of segmentation can be generally described as a classification problem, in which a given neural network is attempting to classify every pixel in an image as one of several classes. Popular network architectures and backbones are commonly applied, including a method of spatial down-sampling and channel-wise expansion [2] as well as residual components carried between layers and blocks in the architecture [6].

II. Methods

n contrast to the comparison conducted in [1], we implemented 3 model architectures for segmentation, U-Net [2], Segnet [5] with a ResNet50 [6] backbone, and SUIMnet [1].

A. U-Net

U-NET is a U-shaped encoder-decoder network architecture, which consists of four encoder blocks and four decoder blocks that are connected via a bridge. The encoder network (contracting path) half the spatial dimensions and double the number of filters (feature channels) at each encoder block. Likewise, the decoder network doubles the spatial dimensions and half the number of feature channels. U-Net uses two 3X3 convolutional blocks followed by ReLU activation block.

DownBlock and UpBlock help to build the architecture. Both use smaller helper functions that return the correct layer, depending on what arguments are passed.The number of blocks is defined by the depth of the network.U-Net combines these blocks and makes sure that the correct layers from the encoder are concatenated to the decoder (skip pathways). These layers have to be cropped if their sizes do not match with the corresponding layers from the decoder.

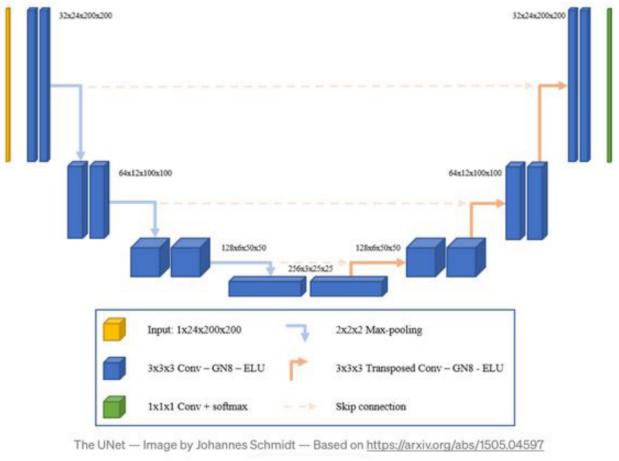

Fig. 4. A visual representation of U-Net.

B. Segnet

The second architecture we implemented is SegNet with a ResNet50 Backbone. We use ResNet as an encoder and SegNet as a decoder.

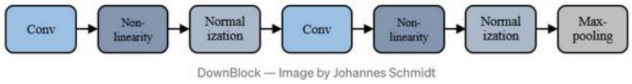

Fig. 5. Spatial down-sampling and channel-wise expansion is a key characteristic of U-Net. The architecture used to apply this function is shown.

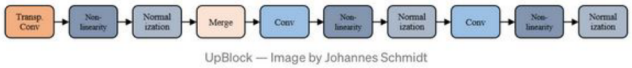

Fig. 6. After spatially down-sampling and a channel-wise expansion, U-Net corrects the dimensions of the image input back to the original shape via spatial up-sampling and a channel-wise reduction.

ResNet stands for “Residual Neural Network”. This type of network employs “skips connections” to pass data across layers. Common ResNet architectures have data that skip multiple layers. The purpose of skip connections is to mitigate the vanishing gradient problem. As we update the weights of a neural network through backpropagation, each successive partial derivative gets smaller. When the derivative becomes sufficiently small, the network may stop improving entirely. Skip connections mitigate this issue by adding non-linearity. Additionally, skipping simplifies networks, allowing the speed of learning to improve.

ResNet50 is an instance of ResNet that uses 50 layers. The depth of this network allows us to classify up to 1000 object categories. We use ResNet in this implementation purely as an encoder. It will take our input images and map them to an abstract feature map space.

The decoder for this project is the SegNet architecture. Segnet is a convolutional network which first convolves the original image to a smaller feature representation before upsampling the image back to its original dimensions. Upsampling is done by storing pooling indices. Since we’re using ResNet as our encoder these indices are not available. Instead, we use bilinear interpolation. SegNet is shown below.

While SegNet has decent performance, it has some limitations. In the case of self-driving cars for example, the output segmentation result has lower resolution than the original input image. A lot of information is lost. In addition, SegNet performance

suffers when presented with novel and variable scenes with a small number of classes. On the other hand, SegNet remains competitive when the training set is robust

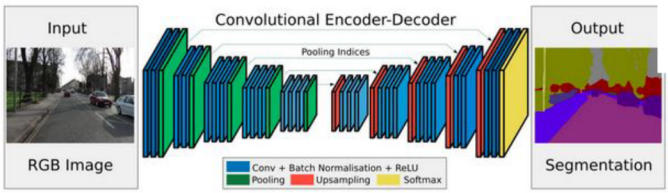

Fig. 7. The model architecture of Segnet.

C. SUIMNet

SUIMNet implements a fully-convolutional encoder-decoder architecture with skip connections between blocks in the encoder and the decoder, shown in figure 8. The base blocks of the encoder implement a residual skip block, in which there are three sets of convolution, batch normalization, and a ReLU activation. The residual component, depending on whether or not a flag for skip is set, is either convolved and batch normalized or simply added to the output of the three convolutions, batch normalizations, and ReLUs, shown in figure 9.

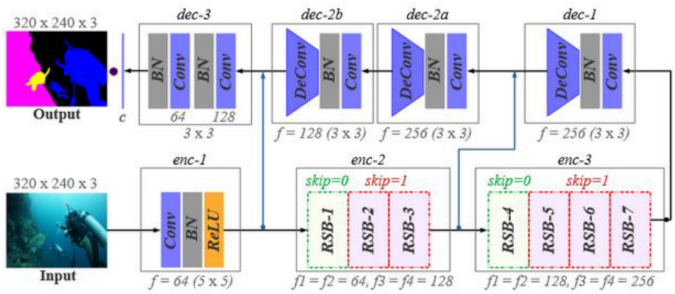

Fig. 8. A visual representation of SUIMNet.

In the encoder of SUIMNet, the image is convolved, normalized, and ReLU activated before being processed by the Residual Skip Blocks. The output of this ReLU is stored for later use in the skip connection to the decoder. There are 7 total Residual Skip Blocks, split into sets of three and four. The first Residual Skip Block of both sets has the skip flag set, indicating that the input should be convolved and summed with the primary convolutions. These are referred to *enc-1* and *enc-2*.

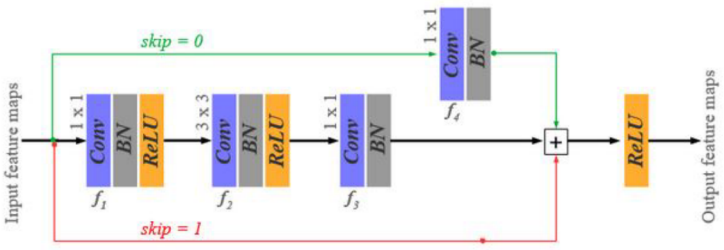

Fig. 9. The Residual Skip Block architecture used in SUIMNet. When the skip flag is set, the input is convolved, normalized by batch, and activated with ReLU.

From image space to *enc-1*, the input is transformed from 3 channels to 64 channels. Each Residual Skip Block maps from input space to a specified output space. The first three residual skip blocks map from input space to 128 channels and the second set of residual skip blocks map from 128 channels to 256 channels. Each first residual skip block spatially downsamples the input such that the output of the encoder is spatially half the size of the input.

The output of the first set of operations (convolution, normalization, and ReLU activation) and the output of the third residual skip block are stored, to be concatenated with intermediary outputs in the decoder architecture.

The decoder architecture has 4 blocks, each spatially upsampling and decreasing the dimensionality. The output of the second encoding block, *enc-2*, is concatenated by channel to the output of the first decoding block, *dec-1*. This quantity is inputted to two sequential spatial upsampling blocks, which decreases the channels and increases the spatial dimensions of the intermediate output. The output of the first encoding block, *enc-1*, is concatenated with the output of the second and third decoding blocks before being imputed to a final decoding block with two convolutions and batch normalizations. This renders the input into the shape (batch size, number of classes, height, width), and the one-hot encoding of the number of classes is used to classify a pixel as the material.

III. Results

SegNet has the advantage of fast computational complexity and memory efficiency. On the other hand, this architecture took a considerable amount

of time to train due to the depth of the encoder network. Our encoder is ResNet50. This 50 layer deep residual network is often used to classify thousands of objects, however since our task only required us to successfully identify 8 classes of objects, it proved costly and time consuming to train this network. As a result we were only able to train 25 epochs with a batch size of 2 images. Given enough time, we can expect these results to improve drastically. The final validation loss of SegNet was 1.085.

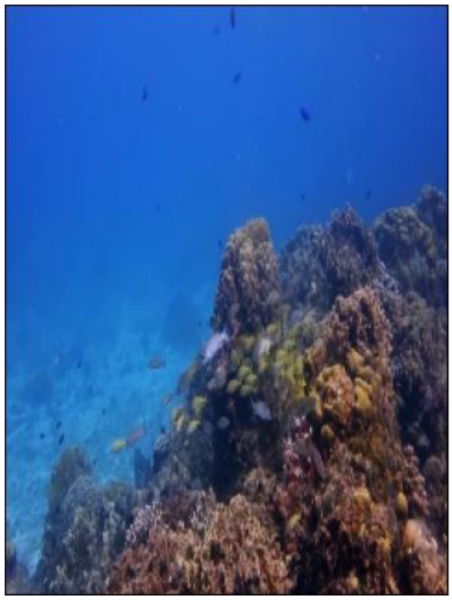

Fig. 10. A test image in the validation set of the SUIM dataset.

Fig. 11. An target mask from the validation set of the SUIM dataset.

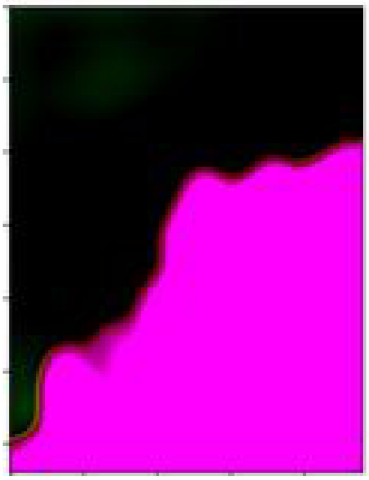

Fig. 12. The result of SegNet after processing the image associated with the above target mask.

U-NET is the next network architecture that we implemented. Much like SegNet, U-Net downsampled to a feature space before returning with an output. However, U-Net also utilizes skip connections by concatenating convolved images in the encoder with upsampled ones in the decoder. The visual results are shown below. The final validation result was a loss of 1.050 using Cross Entropy Loss over 5 epochs.

Fig. 13. An image used in validation of the U-Net model.

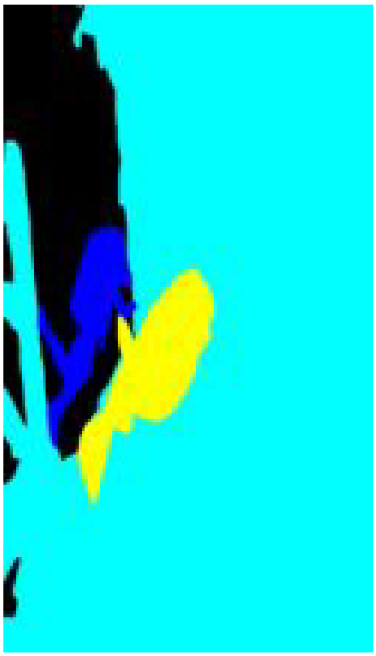

Fig. 14. The mask associated with the image shown used to test U-Net.

Fig. 15. The result of the testing with U-Net.

While SUIMNet shows great promise in the paper put forward by Islam et al, the results could not be reproduced. The test image shown in figure 10 was tested and the results are shown below in Figure 16.

Fig. 16. The result of SUIMNet after processing the image associated with the above target mask.

SUIMNet was trained for 50 epochs and completed training relatively quickly, in under 30 minutes. The ending loss calculated with Cross Entropy Loss was 1.599. As Islam et al writes, the intention of SUIMNet is to balance training speed and real-time image segmentation against the accuracy of the result. Although SegNet outperformed SUIMNet visually, SUIMNet was able to be trained significantly faster, indicating that because the model architecture is simpler, if the model were to be fed live video, there would be significantly less delay in the model output. This would allow for the key application of real-time image segmentation for autonomous vehicles and robotics to make decisions based on the segmentation.

It should be noted that the results indicated here do not undermine the results indicated in [1]. The results shown by Islam et al are achieved via a more complex version of SUIMNet that applies blocks of pre-trained VGG-16 models, rather than Residual Skip Blocks. Because of the added complexity, this would added additional training time and a greater delay for real-time image segmentation, but would

allow for greater accuracy.

IV. Discussion

In this paper we explored 3 approaches to image segmentation using the SUIM underwater dataset. We implemented the SUIM approach, SegNet with a ResNet50 encoder and U-Net. Each method has its respective advantages and drawbacks, however SegNet has achieved the highest visual accuracy of the three.

While the results of the paper by Islam et al could not be reproduced, both in terms of the established models architectures for image segmentation or with SUIMNet, this does not detract from the results they have put forward, nor the effectiveness of SUIMNet for real-time image segmentation.

Rather, the disparity between the results of our work and the work in [1] can be explained by a lack of training time. This project endured computational difficulty in terms of the required amount of time to train each network as well as hardware limitations for training the models.

U-Net proved significantly slow in training, taking as long as 5 hours to train only 5 epochs. Despite this, it achieved some results. Visually U-Net performed well, as seen visually in Figure 15, the network was able to somewhat accurately segment the image shown in Figure 13. With significantly more time and/or better hardware, U-Net would likely be able to outperform SegNet and SUIMNet. The higher training time for the network must be accounted for, in that the although U-Net was visually quite good, it took much longer to train than that of SegNet and SUIMNet.

As described above, SegNet could not be trained for as many epochs as necessary, as it was simply too costly in terms of time. Thus, the results of our implementation of SegNet was significantly limited by the amount of time available for training. Beyond time-cost, there were significant memory limitations in the available hardware. As a result, measures were taken to optimize memory usage at the expense of computation time. For instance, gradients were only computed every 20 batches and batch size was limited to 2 sets of images and masks. This greatly increased the needed time to train the network.

Similarly, because of the complexity of SUIMNet, there were significant limitations imposed by the amount of memory available on the available hardware. Tesla V100 GPUs were used to train the network, however, because of the sheer number of trainable parameters of SUIMNet, several memory saving measures were taken. The above method of decreasing batch size and calculating gradients every 4 batches was taken, however the intermediate outputs of the model were checkpointed using the PyTorch package for Python [8]. These methods effectively increased the batch size while training, leading to ineffective training relative to the process described in [1]. To preserve a semblance of similarity to the methods put forward in the original paper detailing SUIMNet, the model was trained for 50 epochs, rather than more.

Beyond the shortcomings of our implmentaions, image segmentation is a long standing image processing task and research continues. More techniques exist and show promising results. Specifically, a method of high promise is that of the Mask R-CNN model [9]. This model architecture builds upon the architecture of Faster R-CNN [10]. Both apply a fully convolutional neural network to construct region proposals for object detection, however Mask R-CNN goes further to attempt to segment the proposed regions of the detected objects.

In addition, a relatively new architecture, You Only Look Once (YOLO) [11], applies a similar downsampling technique as the studied methods but for the purpose of object detection. Given that Mask R-CNN is based upon an object detection algorithm, the YOLO architecture could be applied to segment images. Notably, the newer iterations of YOLO have approached near real-time object detection, lending credibility to the idea that an image segmentation network built on top of YOLO may show promise for real-time image segmentation.

V. Conclusion

The results of the paper evaluating image segmentation techniques by Islam et al could not be reproduced. This is likely not due to the shortcomings of their methods, but rather due to the authors' inexperience in Deep Learning methods. The ending loss of SegNet was 1.386, the ending loss of U-Net was, and the ending loss of SUIMNet was 1.599. SegNet preformed the best, likely due to a more complex architecture than the other methods

applied. However, SUIMNet could be trained faster, likely due to a less complex architecture. This shows promise for all methods in different contexts, in that SegNet will likely outperform SUIMNet in accuracy, but when given enough training time and data, SUIMNet may outperform SegNet in delay for segmenting images in real-time. While the implementation of these networks may not have succeeded, there exist several promising methods for the task of underwater image segmentation.

REFERENCES

[1] Islam, Md Jahidul, et al. "Semantic Segmentation of Underwater Imagery: Dataset and Benchmark." 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020, https://doi.org/10.1109/iros45743.2020.9340821.

[2] Ronneberger, Olaf, et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." Lecture Notes in Computer Science, 2015, pp. 234–241., https://doi.org/10.1007/978-3-319-24574-428.

[3] Eraqi, Hesham M., et al. "Efficient Occupancy Grid Mapping and Camera-Lidar Fusion for Conditional Imitation Learning Driving." 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), 2020, https://doi.org/10.1109/itsc45102.2020.9294222.

[4] S. Mentasti and M. Matteucci, "Multi-layer occupancy grid mapping for autonomous vehicles navigation," 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), 2019, pp. 1-6, doi: 10.23919/EETA.2019.8804556

[5] Badrinarayanan, Vijay, et al. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 12, 2017, pp. 2481–2495., https://doi.org/10.1109/tpami.2016.2644615.

[6] He, Kaiming, et al. "Deep Residual Learning for Image Recognition." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, https://doi.org/10.1109/cvpr.2016.90.

[7] J. Canny, "A Computational Approach to Edge Detection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679-698, Nov. 1986, doi: 10.1109/TPAMI.1986.4767851.

[8] Mishra, Pradeepta. "Introduction to Pytorch, Tensors, and Tensor Operations." PyTorch Recipes, 2019, pp. 1–27., https://doi.org/10.1007/978-1-4842-4258-21.

[9] He, Kaiming, et al. "Mask R-CNN." 2017 IEEE International Conference on Computer Vision (ICCV), 2017, https://doi.org/10.1109/iccv.2017.322.

[10] Ren, Shaoqing, et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, 2017, pp. 1137–1149., https://doi.org/10.1109/tpami.2016.2577031.

[11] Dahirou, Zoubaydat, and Mao Zheng. "Motion Detection and Object Detection: Yolo (You Only Look Once)." 2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC), 2021, https://doi.org/10.1109/icnisc54316.2021.00053.