Journal of Robotics and Control (JRC)
Volume 5, Issue 2, 2024
ISSN: 2715-5072, DOI: 10.18196/jrc.v5i2.20589

Ovarian Tumors Detection and Classification on Ultrasound Images Using One-stage Convolutional Neural Networks

Van-Hung Le [1], Thi-Loan Pham [2,3*]
[1] Tan Trao University, Vietnam
[2] Information Technology Department, Hai Duong University, Vietnam
[3] Communication Engineering Department, SEEE, Hanoi University of Science and Technology, Vietnam
Email: [2] phamthiloan2011@gmail.com
*Corresponding Author

***Abstract*—Currently, the advent of CNN (Convolutional Neural Network) has brought very convincing results to computer vision problems. One-stage CNNs are a suitable choice for research and development to have an overview of the current results of the process of detecting and classifying OTUM from ovarian ultrasound images. In this paper, we have performed a comprehensive study on one-stage CNNs for the problem of detecting and classifying OTUM on ovarian ultrasound images. The OTUM datasets we tested were two popular OTUM datasets: OTU and USOVA3D. The one-stage CNNs we tested and evaluated belong to the YOLO (You Only Look Once) family (YOLOv5, YOLOv7, YOLOv8 variations, and YOLO-NAS), and the SSD (Single Shot MultiBox Detector) family (VGG16-SSD, Mb1-SSD, Mb1-SSD-Lite, Sq-SSD-Lite, and Mb2-SSD-Lite). The results of detecting OTUM (with or without OTUM on ovarian ultrasound images) are high (with Mb1-SSD of $Acc = 98.90\%, P = 98.58\%, R = 98.9\%$ on "USOVA3D_2D_f r1_80_20" set; with Mb2-SSD-Lite of $Acc = 97.87\%, P = 97.16\%, R = 97.87\%$ on "USOVA3D_2D_f r2_80_20" set). The results of detecting and classifying OTUM into 8 classes are low (the highest is $Acc = 92.04\%, P = 74.81\%, R = 92.04\%$ on the OTU-2D dataset). Regarding computation time, CNNs of the YOLO family have faster computation times than networks of the SSD family. The above results show that the problem of classifying ovarian tumors on ultrasound images still contains many challenges that need to be resolved in the future.**

***Keywords*—Ovarian Tumor Detection, Ovarian Tumors Classification, One-Stage CNNs, YOLO Family, SSD Family**

I. Introduction

A woman's risk of developing OCAN in her lifetime is 1 person in 78 people $\sim$ 1.3% [1]. In particular, a woman's lifetime risk of dying from invasive OCAN is 1 person in 108 people $\sim$0.9%. In particular, the survival rate for OCAN is much lower than for other cancers that affect women. The 5-year relative survival rate for OCAN is 49.7%. Women diagnosed at an early stage - before the cancer has spread have a much higher five-year survival rate than women diagnosed at a later stage. Therefore, early detection and diagnosis of OTUM is very important, which increases the patient's chance of survival.

Nowadays, using ultrasound images to detect and diagnose OTUM has become very popular and helps women not to have to have any surgery. In the new study, Gupta et al. [2] evaluated a method that uses ultrasound imaging to classify ovarian lesions into two types: classic and non-classical. Classical lesions are typically detected as fluid-filled cysts with a very low risk of malignancy. Atypical lesions include those with solid components and blood flow detected on Doppler ultrasound. Classical and non-classical approaches to these ovarian lesions can help doctors diagnose images more quickly. Currently, there is also the IOTA organization [2] that relies on these components to classify the lesions of OTUM [3].

Today, with the strong development of deep learning (DL), especially the birth of Convolutional Neural Networks (CNNs). CNNs have brought very convincing results when solving computer vision problems such as object detection and recognition. The disadvantage of DL is that it requires a large amount of training data and large computational space, often using GPUs to perform calculations [4]. Therefore, building a system to detect and diagnose OTUM that can be applied in hospitals is a very challenging problem, and requires a very large amount of training data. At the same time, the real system for OTUM detection requires low computational space (can be performed on CPU) but still has high results. The problem of detecting and classifying OTUM is the problem that receives the first attention during the application development process.

The problem of detecting objects in images is often solved by one of the two types of CNNs [5], [6]: One-stage (YOLO series/family [7], SSD series/family [8]) or Two-stage (RCNN [9], Fast RCNN [10], Faster RCNN [11], RFCN [12], Mask RCNN [13]), as shown in Fig. 1.

In the studies by [14] and [15], it is shown that the one-stage CNNs have lower accuracy than the two-stage CNNs

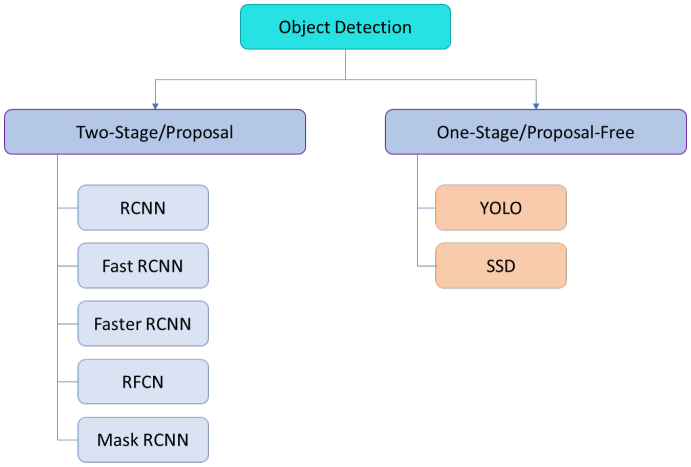


Fig. 1. Some CNNs of one stage and two stages.

for object detection, but the computation time is many times faster than the two-stage CNNs. In a study by Tan Lu et al. [15], the mAP result and processing time of the Faster R-CNN model [11] are 87.69%, and 7fps, respectively, and the mAP result and processing time of the YOLOv3 [16] model are 80.17%, 51fps, respectively. Due to the actual requirements of the system, the OTUM detection and diagnosis system allows the case of "getting it wrong is better than missing it". If an OTUM is not detected, it will lead to the patient not being treated and the tumor will grow quickly, and when discovered it will be at a late stage. When an area of data is mistakenly detected as a tumor, further tests can be performed to determine whether it is an OTUM or not. Therefore, we use one-stage CNNs to test the problem of detecting and classifying OTUM.

Recently, MMOTU [17] with OTU-2D and USOVA3D [18] datasets have been introduced and published. These are two datasets that are widely used to detect, segment, and classify OTUM from ultrasound images. The study of Zhao et al. [17] and Pham et al. [19] tested the segmentation and classification of OTUM on several CNNs such as ResNet [20], U-net [21], PSPnet [22], DANet [23], etc. However, the issue of detecting and classifying OTUM is the first concern in building applications to support the detection and diagnosis of OCAN in hospitals.

In this paper, we conduct a comprehensive study and experiment on the use of one-stage CNNs of the YOLO family (YOLOv5, YOLOv7, YOLOv8, YOLO-NAS) and SSD family (VGG16-SSD, Mb1-SSD, Mb1-SSD-Lite, Sq-SSD-Lite, Mb2-SSD-Lite) families for detecting and classifying OTUM on ovarian ultrasound images from two popular OTUM datasets: OTU-2D and USOVA3D. This is a fundamental study and provides results that serve as a basis for improving and proposing new models for detecting and diagnosing OTUM on ultrasound images to ensure the requirements of a real system can be deployed in hospitals.

The main contributions of our research are as follows:

- We have generalized the background and development of the YOLO and SSD learning one-stage CNNs.
- We have normalized the two databases MMOTU and USOVA3D to prepare for fine-tuning the networks of the YOLO and SSD families.
- We have performed fine-tuning of the model to automatically detect and classify OTUM on ultrasound images from two databases MMOTU and USOVA3D based on one-stage networks of the YOLO and SSD families.
- We have evaluated and compared the results of OTUM detection and classification on ultrasound images from the MMOTU and USOVA3D databases with CNNs of the YOLO and SSD families.
- We analyzed, discussed, and presented challenges when using CNNs for the problem of detecting and classifying OTUM on ultrasound images.

The structure of the paper is organized as follows. In section II, we briefly present some related works on OTUM detection and classification. The background on one-stage CNNs of the YOLO and SSD families is presented in Sec. III. The dataset and experimental results, discussion, and challenges will be presented in section IV. We finally conclude and give some ideas for future works.

II. Related Works

The ovary serves as both a reproductive organ and an endocrine gland with a complex process of formation. Ovaries undergo significant changes in both morphology and function throughout a woman's life, and these changes can lead to irreversible disorders, evolving into pathologies, especially the formation of OTUM shown in Fig. 2. OTUM is one of the most common types of tumors worldwide, accounting for up to 30% of female reproductive system tumors [24].

Making an accurate disease prognosis, in many cases, requires a multidisciplinary approach involving collaboration among medical professionals, especially the participation of experienced experts. In local healthcare facilities, doctors often lack the necessary tools to accurately assess the stage of OCAN or evaluate the risk level of ovarian cysts (benign or malignant). Nowadays, artificial intelligence in general, and machine learning and DL in particular, have made significant breakthroughs in various fields, including healthcare [25], [26], [27], [28], [29], [30], [31].

Recently there has also been a lot of research on using deep learning to build cancer detection and diagnosis systems. There are several fairly complete surveys on the use of DL for lung cancer detection and diagnosis [32], [33]. Studies [34], [35], [36], [37], [38], [39] proposed methods using deep learning techniques for early detection, classification, and segmentation of lung cancer. Some research studies and proposed methods

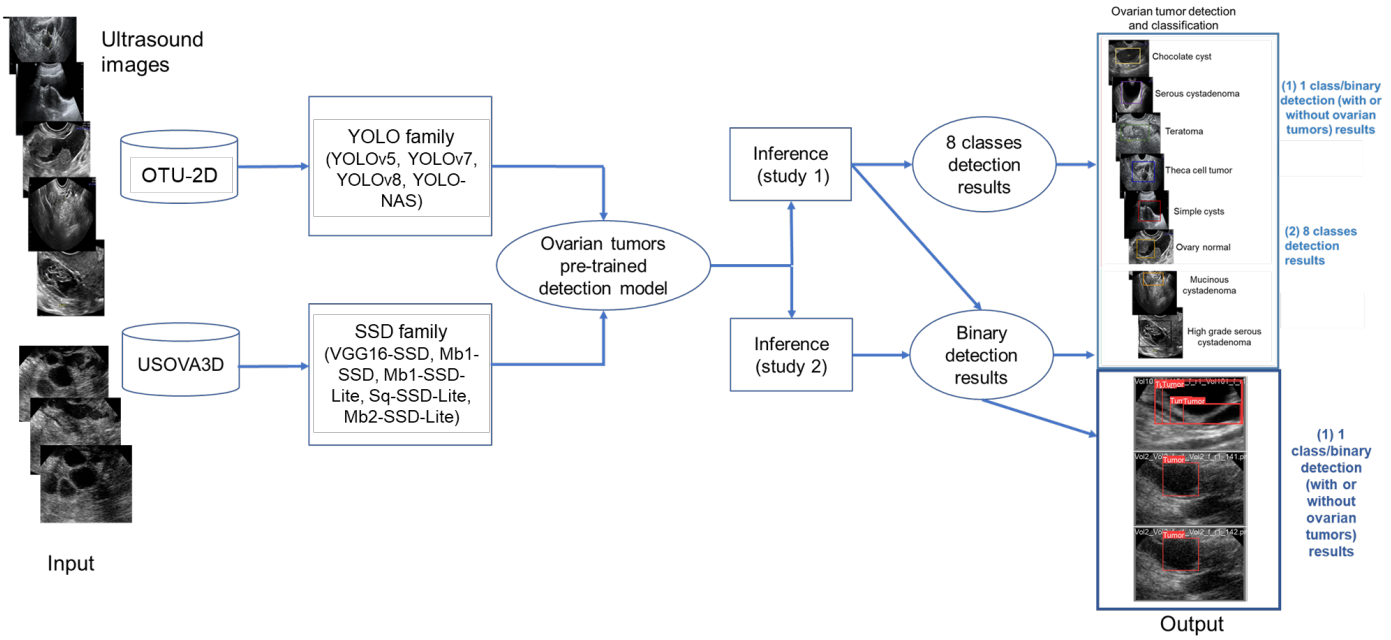


Fig. 2. Research and comparison diagram for detecting and classifying OTUM on ultrasound images.

to detect skin cancer are also presented [40], [41]. Or studies on breast cancer detection and diagnosis using DL [42], [43], [44], [45].

OTUM represents the most common diagnostic challenge for gynecologists. Ultrasonography has become the primary technique for assessing ovarian pathology and distinguishing between malignant and benign OTUM before surgery. In recent years, DL analysis has been widely used in medical image processing. Scientists have proposed many methods to process medical images using deep neural network models, such as CNNs, Fully Convolutional Networks (FCNs), or Recurrent Neural Networks (RNNs), which can segment and assist in diagnosing ovarian diseases from ultrasound images [46]. Recently, there have also been many studies and surveys on detecting and diagnosing ovarian tumors and ovarian cancer based on machine learning and DL [47], [48], [49], [50], [51], [52], [53], [54], [55], [56]. These studies have had very positive results.

In the research of Shih-Tien Hsu et al. [57], they proposed an automatic system that utilizes an ensemble CNN to interpret OTUM ultrasonic images. The system incorporates technologies such as image preprocessing, data augmentation, and ensemble Grad-CAM. For the validation strategy, they repeated the random sampling of the training and validation data ten times to verify model robustness. The means of ACC, SE, and SP of the single network model with optimal performance were 90.51%, 89.02%, and 92%, respectively. The study proposed the ensemble method based on the confidence scores of multiple decision-making models, which achieved the means of ACC, SE, and SP of 92.15%, 91.37%, and 92.92%, respectively. The proposed method increased the means of ACC, SE, and SP of the single network model with optimal performance by 1.64%, 2.35%, and 0.92%, respectively.

In the study by Martinez et al. [58], the authors employed traditional machine learning methods (Linear Discriminant (LD) and Support Vector Machine (SVM)) on a dataset consisting of 384 images from 187 patients to distinguish between benign and malignant OTUM. The results achieved an accuracy with Precision and Recall of approximately 80% and 92%, respectively. However, the study also highlighted limitations in terms of the number of images and patients, which constrained the statistical significance and potential of the reported results.

The authors [59] presented 2-D ultrasound images of a 24-year-old woman's ovarian follicles with multiple cystic areas (antral follicles). The use of convolutional neural networks to segment the cysts and nodules within OTUM has garnered attention in recent research [60].

In the study by Y. Wang and colleagues, the authors provided a visually intuitive comparison table when using advanced machine learning algorithms (e.g., VGNet, CNN with performance $> 93\%$) with traditional models such as Logistic Regression, SVM, or Random Forest (performance $< 80\%$). The adjustment of the VGG-16 model has proven to be effective in OTUM detection, as demonstrated in the research conducted by Sakshi et al. [61].

The term "VGG-16" derives from the "Visual Geometry Group," a group of visual geometry researchers at the Uni-

versity of Oxford who conceived the idea of creating this special 16-layer network and trained it on the ImageNet dataset. The traditional VGG-16 model consists of multiple 3x3 kernel filters to enable the model to learn more complex features by increasing network depth. The convolutional layers in VGG are followed by three fully connected layers, and the authors proposed an adjustment algorithm within the network to save time by not starting from scratch. Modifying the last four layers by training them on OTUM data using complex neural network architecture was performed to achieve a high accuracy rate of 92.11%.

Juebin et al. [62], authors emphasized the significance of detecting and automatically segmenting abnormal regions on ultrasound images for patients with OCAN. In this work, the classical U-net scheme and its multiple variations were used for the automatic segmentation task. The average Pearson correlation was 0.86 (95% CI, 0.83–0.89), 0.87 (95% CI, 0.84–0.90), 0.88 (95% CI, 0.86–0.91), and 0.90 (95% CI, 0.88–0.92) for U-net++, U-net, U-net with Resnet, and CE-Net, respectively.

III. CNNs for OTUM Detection and Classification

Nowadays, CNNs provide very convincing results in the problem of detecting objects in images and they are divided into two types: one-stage and two-stage. One-stage CNNs are networks that can predict the bounding box (BB) of an object right on the grid. Two-stage CNNs use a proposal network to find objects and a second network to fine-tune the proposals and produce a final result that predicts the object's BB [61].

To decide between two-stage and one-stage CNNs for detecting ovarian cysts in ultrasound images, consider the following detailed factors: Size and Characteristics of Ovarian tumor: If ovarian tumors are typically large, and have clear and distinguishable features, one-stage CNNs may be a suitable choice. Models like YOLO or SSD can process images quickly and are suitable for large and easily classifiable objects. If ovarian cysts are typically small, with complex features, two-stage CNNs may yield better results. Models like Faster R-CNN often focus on accurately proposing object regions and have high accuracy in classification. Accuracy and Accuracy Requirements: If accuracy is a decisive factor, and it is crucial to ensure that every ovarian cyst is accurately detected, two-stage CNNs may be a reasonable choice. They often can propose accurate object regions and have high accuracy in the classification process. Next, we will present the YOLO and SSD families for object detection in images.

A. YOLO Families

YOLO (You Only Look Once) [63] is a typical and most used CNN for object detection problems. YOLO has a fairly basic architecture, as presented in Fig. 3, including a base network and extra layers. The development and improvement process of YOLO is shown in Fig. 4, versions of YOLO were announced as YOLOv1 [63] in 2015, YOLOv2/9000 [64] in 2016, YOLOv3 [16] in 2018, YOLOv4 [65] in 2020, YOLOR [66] and YOLOx [67] in 2021, YOLOv5 [68], YOLOv6 [69] and YOLOv7 [46] in 2022, YOLOv8 [70] and YOLO-NAS [71] in 2023. The base network is a Convolution network responsible for extracting image features. Extra layers are the final layers used to analyze features and detect objects. The base network commonly used is Darknet.

The input image is divided into a grid of $S \times S$ cells, also known as cells. Here there is no real image division, but the nature of image division is dividing the output and target into a matrix A of size $S \times S$. If the image, the center of the object is in the $(i, j)^{th}$ cell, then the corresponding output will be in $A[i, j]$.

YOLO's implementation process includes 2 steps:
- The first is a convolution network that extracts image features.
- The second is extra layers (fully connected layers) to analyze and detect objects. Returns output is a matrix A with dimensions ($Shape(A)$) as Eq. (1).

$$Shape(A) = S * S * (5 * B + C) \tag{1}$$

where B is the number of bounding boxes (BBs), each BB consists of five components $(x, y, w, h, confidence\_score)$, (x, y) are the coordinates of the upper left corner of the BB, (w, h) are the width and height of the BB, $confidence\_score$ is the probability that there is an object in that cell or not. Finally, the C element represents the probability distribution about the type of object, that is, the class distribution because this C element is a probability distribution and it needs to be guaranteed according to Eq. (2). Thus, YOLO calculates the BB coordinates, the probability of an object appearing, and the probability distribution to classify the object, and it's all done in one go.

$$\sum_{0}^{c} p_i = 1 \tag{2}$$

A very important next calculation of YOLO is the loss function, YOLO's loss function is divided into 2 parts: $\mathcal{L}_{\text{loc}}$ (localization loss) measures the error of the BB, and $\mathcal{L}_{\text{cls}}$ (confidence loss) measures the error of the probability distribution of classes, as computed in Eq. (3). It can be understood more simply, that $\mathcal{L}_{\text{loc}}$ is the loss function of the predicted BB compared to the actual, that $\mathcal{L}_{\text{cls}}$ is the loss function of the probability distribution where the first sum is the loss of predicting whether there is an object in the cell or not? The second sum is the loss of the probability distribution if there are objects in the cell.

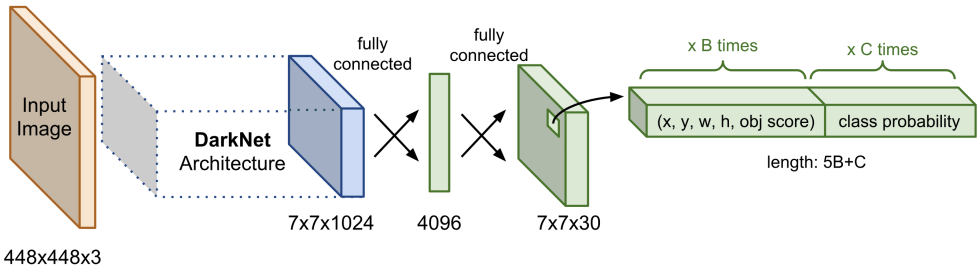


Fig. 3. The general architecture of YOLO [63].

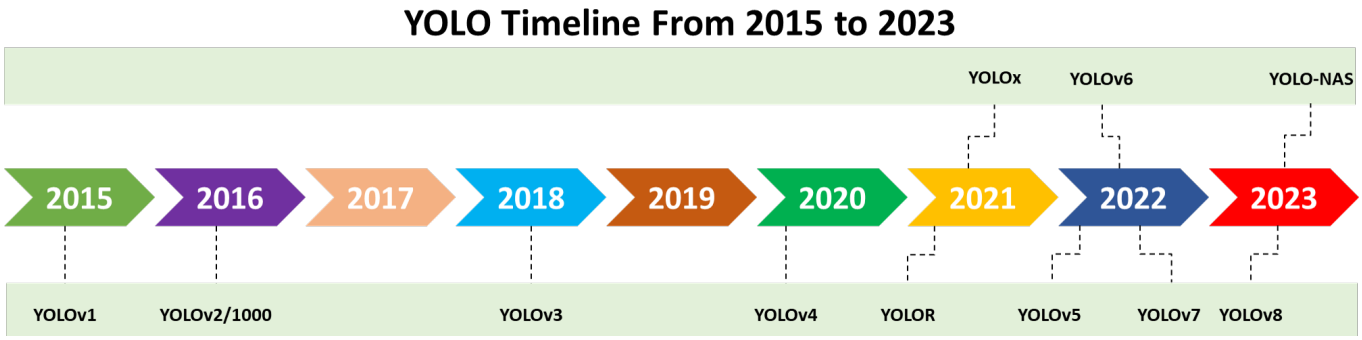


Fig. 4. Time to announce versions of YOLO.

$$
\begin{aligned}
\mathcal{L}_{\text{loc}} &= \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \right. \\
&\quad \left. (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
\mathcal{L}_{\text{cls}} &= \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^{B} (\mathbb{1}_{ij}^{\text{obj}} + \lambda_{\text{noobj}} (1 - \mathbb{1}_{ij}^{\text{obj}}))(C_{ij} - \hat{C}_{ij})^2}_{\text{cell contain object}} + \\
&\quad \underbrace{\sum_{i=0}^{S^2} \sum_{c \in \mathcal{C}} \mathbb{1}_{i}^{\text{obj}} (p_i(c) - \hat{p}_i(c))^2}_{\text{probability distribution classes}} \\
\mathcal{L} &= \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{cls}}
\end{aligned}
\tag{3}
$$

where $\mathbb{1}_{ij}^{\text{obj}}$ is an indicator function with a value 0 or 1 to determine whether the cell contains an object or not. Equals 1 if it contains an object and 0 if it does not. $\mathbb{1}_{ij}^{\text{obj}}$ is the j^{th} BB of the cell i^{th}, whether it is the BB of the predicted object or not. C_{ij} is the cell's confidence score, P(contain object) * IoU (predict BB, ground truth BB). $\hat{C}_{ij}$ is the prediction confidence score. C is the set of all classes. $p_i(c)$ is the conditional probability of whether or not cell i^{th} contains an object of class $c \in C$. $\hat{p}_i(c)$ is the predicted conditional probability.

The next step is to predict the BB. To predict the BB for an object, we rely on a transformation from the anchor box and cell. YOLOv2 [64] and YOLOv3 [16] predict the BB so that it will not deviate too much from the center position. This is an improvement of YOLOv2 and YOLOv3 compared to YOLOv1. This will improve object detection results when many objects are located in the same cell. If the prediction BB can be placed on any part of the image, as in a regional proposal network, model training can become unstable. Given an anchor box of size (p_w, p_h) at a cell located on the feature map with its top left corner being (c_x, c_y), the model predicts 4 parameters (t_x, t_y, t_w, t_h) in the first 2 parameters are the offset compared to the top left corner of the cell and the last 2 parameters are the ratio compared to the anchor box. These parameters will help determine the prediction BB b with center (b_x, b_y) and size (b_w, b_h) through the sigmoid function and exponential function like Eq. (4).

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
\tag{4}
$$

In addition, because the coordinates have been adjusted according to the width and height of the image, the value is always within the threshold $[0, 1]$. Therefore, applying the

sigmoid function helps us limit the coordinates not to exceed these thresholds shown in Fig. 5.

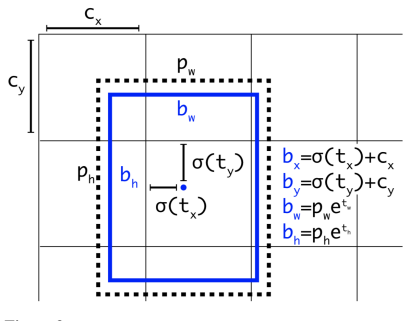


Fig. 5. Estimating BB from anchor box. The outer dashed rectangle is the anchor box with dimensions of (p_w, p_h). The coordinates of a BB will be determined based on both the anchor box and the cell to which it belongs. This helps control the position of the predicted BB somewhere around the position of the cell and the BB without going too far outside these limits. Therefore, the training process will be much more stable than YOLOv1 [64], [16]

Next, we will only introduce and present versions of YOLO from YOLOv5 onwards. ***YOLOv5 [68]***: YOLOv5 focuses on speed and easy use. YOLOv5 model architecture [68] has three important parts: Backbone, Neck, and Head, as illustrated in Fig. 6.

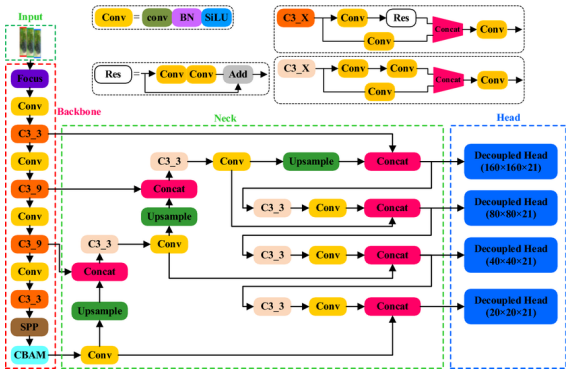


Fig. 6. YOLOv5 architecture [68]

- The backbone aims to extract rich features from the input image. A Cross Stage Partial Network (CSPNet) is used as a model backbone. CSPNets are faster than deeper networks. YOLOv5 improves YOLOv4's CSPResBlock [65] into a new module, little more than a Convolution layer called a C3 module. In YOLOv5 still use CSPDarkNet53 for predicting the original object candidates according to the boxes. In the activation function, YOLOv4 [65] uses Mish or LeakyReLU for the lightweight version, while in YOLOv5, the activation function used is SiLU.
- The Neck constructs feature pyramids. This component helps the model to detect and identify the same objects at different sizes and scales. This leads to better performance with unlearned data. Different types of feature pyramids are used by many models, like FPN, BiFPN, and PANet which is used in YOLOv5. YOLOv5 adopts a module similar to SPP, but twice as fast and calls it SPP-Fast (SPPF). Instead of using parallel MaxPooling as in SPP, YOLOv5 SPPF uses sequential MaxPooling. Furthermore, the kernel size in the MaxPooling of SPPF is all 5 instead of (5,9,13) like the SPP of YOLOv4.
- The Head performs the final detection. It applies anchor boxes on generated features and outputs final vectors containing class probabilities, objectiveness scores, and BBs. YOLOv5 has the same model head as YOLOv3 and YOLOv4.

Other improvements of YOLOv5 compared to YOLOv4 are as follows: (1) About data augmentation: mosaic augmentation, copy-paste augmentation, mixup augmentation; (2) About loss function: add scale factor for Objectness Loss; (3) About anchor boxes: auto anchor using genetic algorithm; about removing grid sensitivity but the formula is different; about exponential moving average weight.

YOLOv7 [72]: YOLOv7 surpasses all known object detectors in both speed and accuracy in the range from 5fps to 160fps and has the highest accuracy at $AP = 0.568$ among all known real-time object detectors with 30fps or higher on GPU V100. The architecture of YOLOv7 is illustrated in Fig 7.

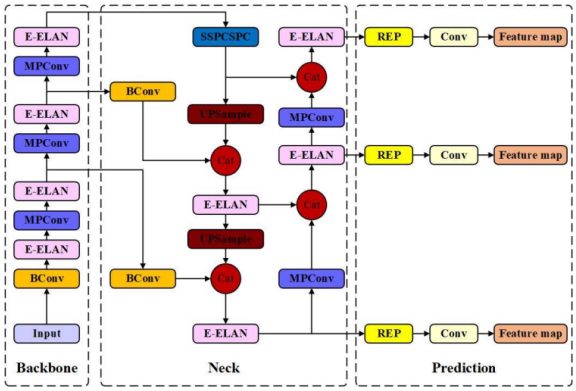


Fig. 7. YOLOv7 architecture [46]

This architecture also includes three main parts.

- The backbone used the ELAN (YOLOv7-p5, YOLOv7-p6) and E-ELAN (YOLOv7-E6E).
- The neck used the SPPCSPC + (CSP-OSA)PANet (YOLOv7-p5, YOLOv7-p6) + RepConv.
- The head used the YOLOR + Auxiliary Head (YOLOv7-p6).

YOLOv7 was announced in 2022 and has been several improvements over previous versions. (1) One of the main improvements is the use of anchor boxes. Anchor boxes are a set of predefined boxes with different aspect ratios that are used to detect objects of different shapes, each anchor box is illustrated in Fig. 5. YOLOv7 uses nine anchor boxes, allowing YOLO to

detect a wider range of object shapes and sizes than previous versions, the process of predicting an object's BB based on anchor boxes is illustrated in Fig. 5, thus helping to reduce the number of wrong identifications. (2) Another important improvement of YOLOv7 is using a new loss function called "focal loss". Previous versions of YOLO used the standard cross-entropy loss function, which is known to be less effective at detecting small objects. Therefore, the results of detecting small-sized objects are not good. In YOLOv7, the focal loss solves this problem by reducing loss weights for well-classified examples and focusing on hard examples such as hard-to-detect objects. YOLOv7 also has different versions like YOLOv7-p5, YOLOv7-p6. Which YOLOv7-p5 group including YOLOv7-tiny, YOLOv7, and YOLOv7x is to use the images with size (640×640) for training. Which YOLOv7-p6 group including YOLOv7-w6, YOLOv7-e6, and YOLOv7-d6 is to use the images with size (1280×1280) for training.

***YOLOv8* [70]**: YOLOv8 is the current latest version of the YOLO family to be published. Like other versions of YOLO, YOLOv8 also includes two main components: Backbone and Head. The architecture of YOLOv8 is shown in Fig. 8. YOLOv8 was developed by Ultralytics, who also created the YOLOv5 model. YOLOv8 includes many changes and improvements in architecture and developer experience compared to YOLOv5. Those changes and improvements are shown as follows:

(1) First, the backbone of YOLOv8 is the same as YOLOv5, it uses the CSPDarknet53 feature extractor. It has some changes like $C2f$ replacing $C3$ to combine high-level features with contextual information to improve detection accuracy. The first 6×6 convolution in the body is converted to the 3×3 convolution. In $C2f$, the output from the bottleneck (which is a combination of two 3×3 transitions with the remaining connections) is combined, as illustrated in Fig. 9, where "f" is the number of features, "e" is the expansion rate and CBS is a block composed of a Conv, a BatchNorm and a SiLU. While in $C3$, only the output from the last bottleneck is used. (2) Second, YOLOv8 has an improvement in using an anchor-free model with a detached head to handle object, classification, and regression tasks independently. This model allows each prediction branch to focus on its task and improves the overall accuracy of the prediction model. The sigmoid function as the activation function for the feature score is used in the output layer of YOLOv8. From this, the probability that the BB contains an object is represented with the softmax function for the class probability, which represents the probability of objects belonging to each possible class. (3) Third, two convolutions (#10 and #14 in the YOLOv5 config) were removed. YOLOv8 provides a semantic segmentation model called the YOLOv8-Seg model to achieve state-of-the-art results on various object detection and semantic segmentation benchmarks while maintaining high speed and efficiency. The loss functions for BB loss and binary cross-entropy for classification loss and to improve the predicted results on small objects. (4) Fourth, the bottleneck in YOLOv8 is still the same as in YOLOv5, except that the kernel size of the first convolution has been changed from 1×1 to 3×3. This change represents a change to the ResNet block, as illustrated in Fig. 10.

Currently, YOLOv8 offers five types of pre-trained models for object detection: YOLOv8n (Nano), YOLOv8s (Small), YOLOv8m (Medium), YOLOv8l (Large), and YOLOv8x (Extra Large). Among them, YOLOv8n is the fastest and smallest, and YOLOv8x is the most accurate but slowest.

YOLO-NAS: YOLO-NAS [71] is developed by Deci.ai company [73] and it offers state-of-the-art (SOTA) performance with superior speed and accuracy performance compared to other versions of YOLO such as YOLOv5, YOLOv6, YOLOv7, and YOLOv8. The NAS is abbreviated from Neural Architecture Search. The architecture of YOLO-NAS is presented in Fig. 11.

YOLO-NAS works on the principle of automatically redesigning a model's architecture to increase its performance when it comes to things like speed, memory usage, and throughput. It typically involves a search space that defines the set of possible architectural choices, such as the number of layers, layer type, kernel size, and interconnection type. The search algorithm then evaluates the different architectures by training and evaluating them on a given task and dataset. Based on these evaluations, the algorithm iteratively explores and refines the architectural space, ultimately returning the space that provides the best performance. In the architecture of YOLO-NAS, there are some new points compared to previous versions of YOLO:

- The use of QSP and QCI blocks [74] combines the advantages of re-parameterization and 8-bit quantization. The blocks above allow for minimal loss function of precision during post-training quantization.
- AutoNAC was used to determine the optimal size and structure of the stages in the backbone and neck parts, including block type, number of blocks, and number of channels in each stage.
- A quantization method that combines selective quantization of certain parts of the model, reducing information loss and balancing latency and accuracy. Standard quantization affects all model layers, often leading to a significant loss of accuracy. Their hybrid method optimizes quantization to maintain accuracy by only quantizing certain layers while leaving other layers untouched. Their layer selection algorithm considers the impact of each layer on accuracy and latency, as well as the impact of converting between 8-bit and 16-bit quantization on overall latency.
- The pre-training mode includes automatically labeled data, self-distillation, and large datasets.
- YOLO-NAS architecture is available under an open-source license. Its pre-trained weights are available for (non-

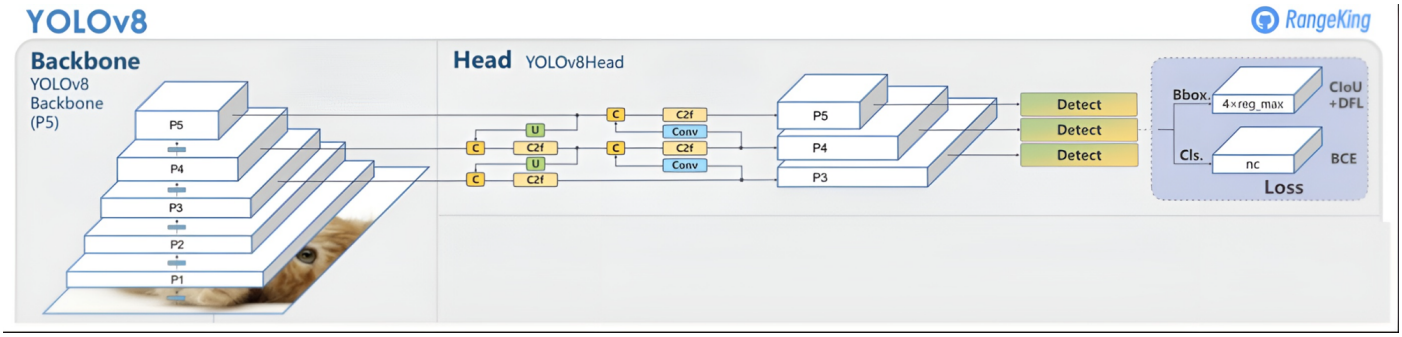


Fig. 8. YOLOv8 architecture [70].

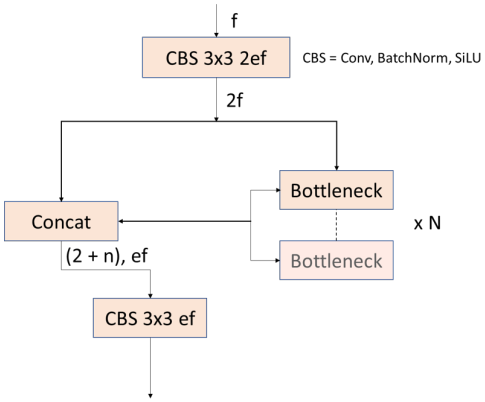


Fig. 9. Illustrating of the $C2f$ module in YOLOv8.

commercial) research use on SuperGradients, Deci's open-source, PyTorch-based, computer vision training library.

B. SSD Families

SSD (Single Shot Multibox Detector) [75]: Like most other object detection architectures, the input to SSD is the BB coordinates of the object (also known as the offsets of the BB) and the label of the object contained in the BB. The special feature that makes the SSD model's speed is that the model uses a single neural network. Its approach is based on object recognition in feature maps (which is a 3D shape output of a deep CNN network after removing the last fully connected layers) with different resolutions $(300 \times 300, 512 \times 512)$. The model will create a grid of squares called grid cells on the feature maps, each cell is called a cell and from the center of each cell determines a set of default boxes for frame prediction. capable of surrounding objects. At prediction time, the neural network will return two values: the probability distribution of the label of the object contained in the BB and a coordinate called the offsets of the BB. The training process is also the process of fine-tuning the label and bounding truth box probabilities to match the ground input values of the model (including labels and BB offsets). Thus, the SSD model will be a combination of two steps:

- Extracting feature maps from the CNN (VGG16 [76], VGG19 [77], ResNet [78], InceptionNet [79], or MobileNet, etc).
- Applying convolutional filters (or kernel filters) to detect objects on feature maps with different resolutions (revolution).

The architecture of the SSD network is shown in Fig. 12.

For example, the **first step** of SSD uses VGG16 as the base network to extract feature maps:

- Inputs are images with dimensions (width $\times$ height $\times$ channels) = $300 \times 300 \times 3$ for SSD300 architecture or $500 \times 500 \times 3$ for SSD500 architecture.

- $Conv5\_3$ Layer: This is the base network that uses the architecture of VGG16 but removes some fully connected layers at the end. The output of this layer is $Conv4\_3$ Layer and is a feature map with dimensions of $38 \times 38 \times 512$.

- $Conv4\_3$ Layer: We can consider $Conv4\_3$ as a feature map with dimensions of $38 \times 38 \times 512$. On this feature map, we will apply 2 main transformations: Apply a convolutional layer like a regular CNN network to obtain the next output layer. Specifically, the convolutional layer has a convolutional kernel of size $3 \times 3 \times 1024$, and the resulting output $Conv6$ has a size of $19 \times 19 \times 1024$. At the same time, in this step, we also apply a classifier as shown in Fig. 13 and rely on a convolutional filter of size 3×3 to detect objects on the feature map. This is a rather complicated process because it must ensure object detection (through BB detection) and object classification. First, we will divide the feature map of size $38 \times 38 \times 512$ into a grid cell of size 38×38 (ignore the depth because we will perform convolution over the entire depth). Then each cell on the grid cell will create 4 default BBs with different aspect ratios. For each default BB, we need to find the following parameters: the probability distribution of the label is a vector. Silk has $n\_classes + 1$ dimension (Note that the number of classes always adds 1 for background). At the same time, we need to add 4 parameters, offsets, to determine the BB of the object in the frame. Therefore, on a default BB there will be $n\_classes + 4$ parameters and on 1 cell there will be

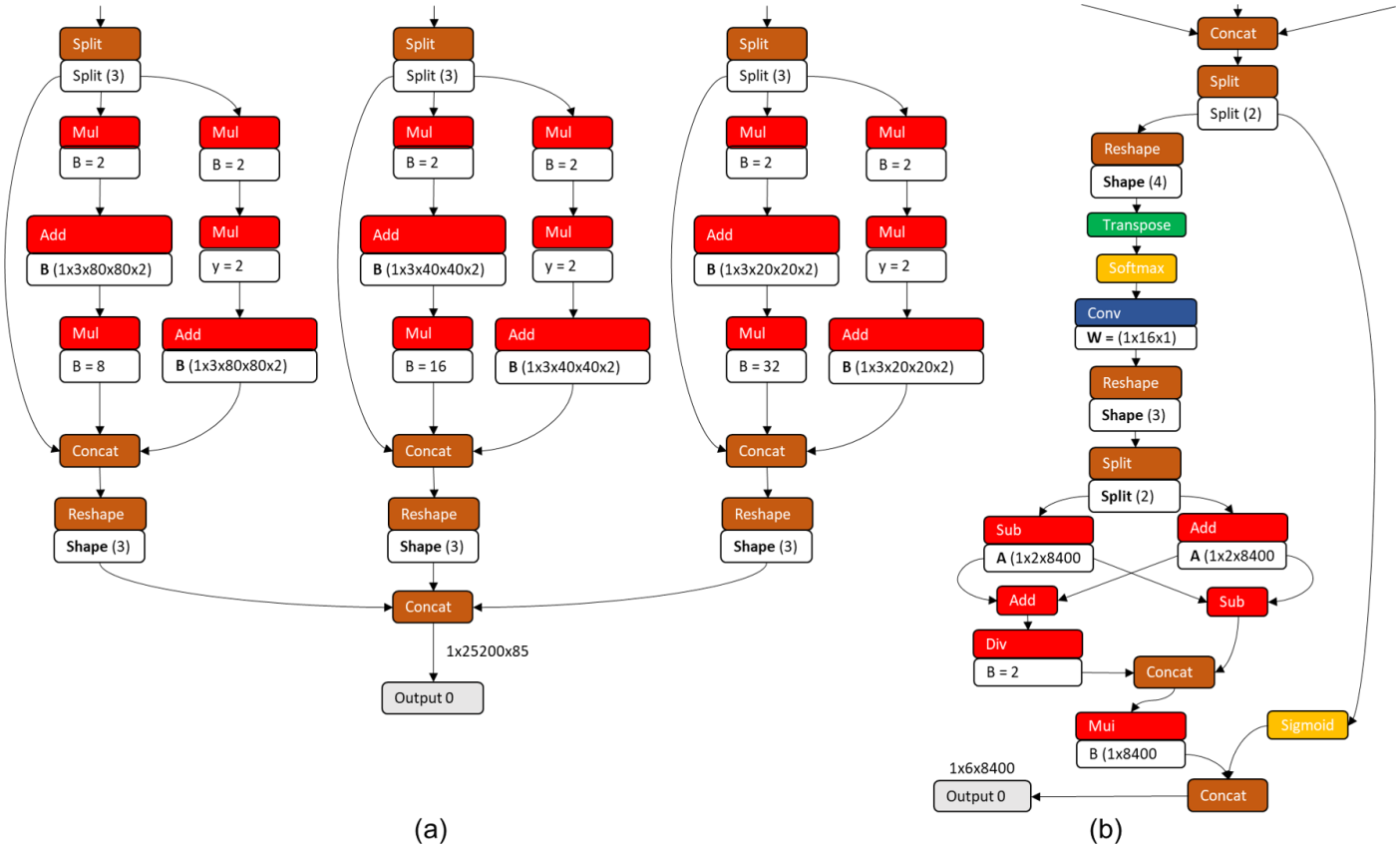


Fig. 10. (a) the detection head of YOLOv5, (b) The detection head for YOLOv8.

$4 * (n\_classes + 4)$ output to predict. Multiply by the number of cells of $Conv4\_3$ to obtain the number of outputs which is a tensor of size $38 \times 38 \times 4 \times (n\_classes + 5)$. In case the background is also considered a label, the tensor has a size of $38 \times 38 \times 4 \times (n\_classes + 4)$. The number of BBs produced is $38 \times 38 \times 4$.

The process of applying a classifier to a feature map is similar to layers $Conv7, Conv8\_2, Conv\_9, Conv10\_2, and Conv11\_2$. The shape of the following layers will depend on the way the convolutional process is applied in the previous layer, the filter kernel size (as in the diagram above, the $kernel\_size$ is always 3×3), and the stride (jump size) of the convolution layer. On each cell in the feature map, we define several 4 or 6 default BBs. Therefore, the number of default boxes produced in the following layers is as follows:

$Conv7 : 19 \times 19 \times 6 = 2166$ boxes (6 boxes/cell)

$Conv8\_2 : 10 \times 10 \times 6 = 600$ boxes (6 boxes/cell)

$Conv9\_2 : 5 \times 5 \times 6 = 150$ boxes (6 boxes/cell)

$Conv10\_2 : 3 \times 3 \times 4 = 36$ boxes (4 boxes/cell)

$Conv11\_2 : 1 \times 1 \times 4 = 4$ boxes (4 boxes/cell)

The total number of boxes in the output will be: $5776 + 2166 + 600 + 150 + 36 + 4 = 8732$

The **second step** is object prediction through a convolutional network. Each feature layer added to the extra feature layers will create a fixed set of outputs y that helps detect objects in the image through the application of convolutional filters. The output size $(with \times height \times chanel)$ at each feature layer size will depend on the kernel filters and is calculated completely similarly to a regular convolutional neural network. We need to associate a set of default BBs with each cell on the feature map. The default boxes will be distributed tiled on the feature map in order from top to bottom and from left to right to calculate convolution, so the position of each default box corresponding to the cell it is associated with is fixed corresponding to an image area on the original image, as shown in Fig. 14.

At each default BB of the feature map, we predict 4 offsets corresponding to a coordinate and its size. The 4 offsets here are understood as a coordinate consisting of 4 parameters (c_x, c_y, w, h) in which (cx, cy) helps determine the center and (w, h) is the length and width of the BB. The second component predicted is the score of the BB corresponding to each class. Note that we will have an additional class $(C + 1)$ to mark

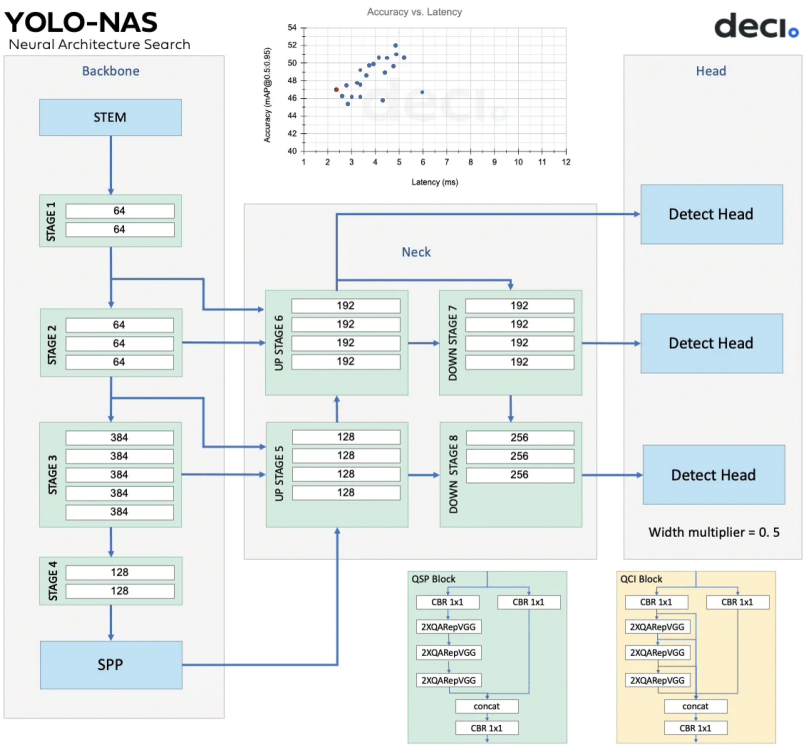


Fig. 11. Architecture of YOLO-NAS [71].

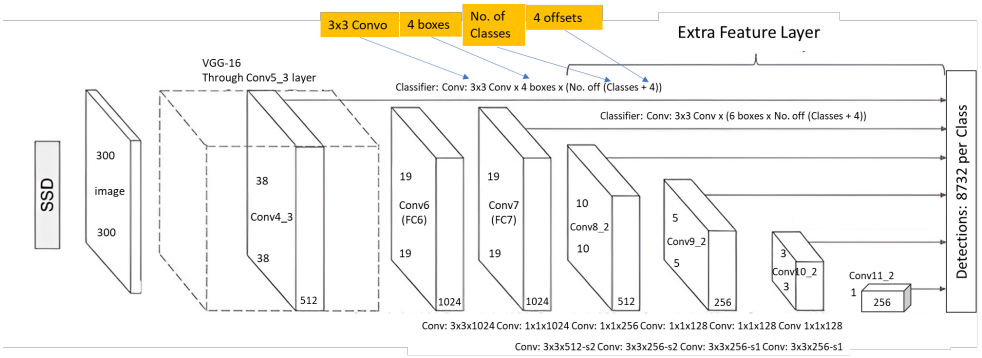
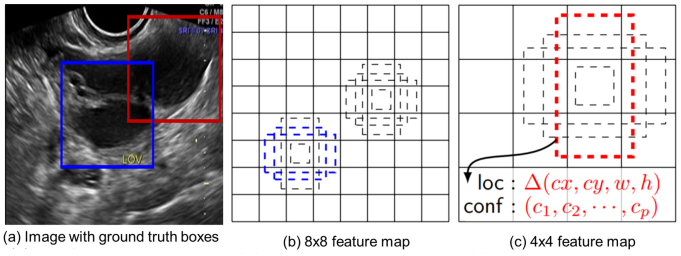


Fig. 12. Architecture of SSD network [75].



(a) Image with ground truth boxes (b) 8x8 feature map (c) 4x4 feature map

Fig. 13. Feature map division method to detect objects with images of different sizes [75].

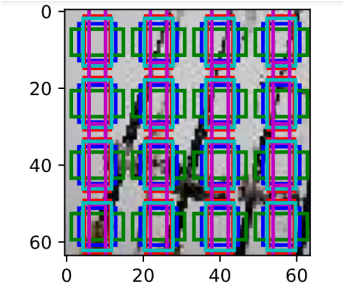


Fig. 14. The position of the default BBs on the original image when applied to a $4x4$ feature map [75].

the case where the default BB has no objects (or falls into the background).

During SSD training, the loss function includes two components: localization loss and confidence loss. Localization loss is a Smooth $L1$ function that measures the error between the parameters of the predicted box (p) and the ground truth box (g). We will need to regress the offsets for the center (x, y) and of the default BB (d) and the height h and the width w, as computed in Eq. (5).

$$
\begin{aligned}
L_{loc}(x,l,g) &= \sum_{i\in \text{ Pos}}^{N} \sum_{m\in\{cx,cy,w,h\}} x_{ij}^{k}\,\text{smooth}_{\text{L1}}\left(l_i^m - \hat{g}_j^m\right) \\
\hat{g}_j^{cx} &= \left(g_j^{cx} - d_i^{cx}\right)/d_i^w \qquad \hat{g}_j^{cy} = \left(g_j^{cy} - d_i^{cy}\right)/d_i^h \\
\hat{g}_j^{w} &= \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)
\end{aligned} \tag{5}
$$

Confidence loss is a loss function calculated based on the label prediction error. For each positive match prediction, we penalize the loss function according to the confidence score of the corresponding labels. For each negative match prediction, we penalize the loss function according to the confidence score of the label "0" which represents the background containing no objects. The confidence loss function is computed as Eq. (6).

$$
\begin{aligned}
L_{\text{conf}}(x,c) &= -\sum_{i\in \text{ Pos}}^{N} x_{ij}^{p}\log\left(\hat{c}_i^p\right) - \sum_{i\in \text{ Neg}}\log\left(\hat{c}_i^0\right) \\
\text{where } \hat{c}_i^p &= \frac{\exp\left(c_i^p\right)}{\sum_p \exp\left(c_i^p\right)}
\end{aligned} \tag{6}
$$

As described above about VGG16-SSD, VGG16 is the backbone for the feature extraction process and building feature maps for the training process of the SSD network. We will present next about Mb1-SSD [80], Mb1-SSD-Lite, Sq-SSD-Lite, and Mb2-SSD-Lite.

MobileNet v1 SSD (Mb1-SSD): In this network, SSD uses MobileNet v1(Mb1) [80] as its backbone. The CNNs such as LeNet [81], AlexNet [82], VGG-16, VGG-19, GoogLeNet [81], and ResNet have a very large number of parameters because these networks perform conventional 2-dimensional convolution, illustrated in Fig. 15. 2-dimensional convolution will normally be computed over the entire depth (channel - c). Therefore, the number of model parameters will increase significantly depending on the depth of the previous layer.

Suppose we have an input of size $(w \times h \times c)$, the usual convolution would need $(k \times k \times c)$ parameter to perform convolution over the entire depth of the layers. Each filter will create an output matrix of size $(w^{'} \times h^{'} \times 1)$. Applying c' different filters will create an output of size $(w^{'} \times h^{'} \times c^{'})$

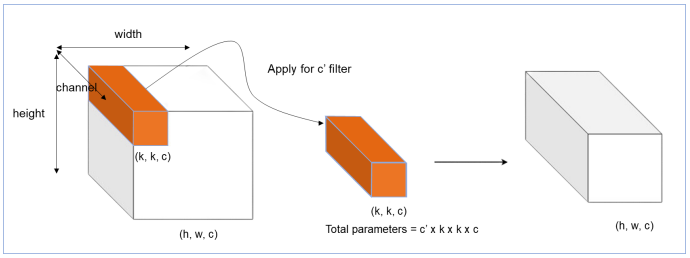


Fig. 15. 2-dimensional (2D) convolution will normally be computed over the entire depth (channel). [80].

(the output matrices when applying convolution on each filter will be concatenated according to depth). Then the number of parameters needed for a normal convolution will be $c^{'} \times k \times k \times c$. Usually, the layers in the first position have a small depth. However, the depth gradually increases to the final layers of the CNN network, and the number of model parameters will increase greatly.

The main idea of MobileNet is that depth-separated convolution will find a way to eliminate the dependence on depth, while convolution still produces an output shape of equivalent size compared to conventional convolution. Specifically, the process will be divided into two sequential steps:

- Depthwise Convolution: the input tensor3D block (w, h, c) is divided into matrix slices according to depth, and performing convolution on each slice as shown in Fig. 16. The output of this step is to obtain an output of size $(h^{'} \times w^{'} \times c^{'})$. Each channel will apply a different filter and does not share parameters. This has three main effects on the model: (1) feature detection: The process of learning and recognizing features will be separated by each filter. If the features on the channels are far apart, using channel-specific filters will be more specialized in detecting the features. For example, if the input is three RGB channels, each channel applies a different, specialized filter. (2) Minimizing the amount of computation: To create a pixel point on the output, normal convolution is needed $(k \times k \times c)$ calculation while separated depth convolution only needs $(k \times k)$ calculations. (3) Reducing the number of parameters: In the depth convolution, we need to use $(k \times k \times c)$ parameters. This is $c^{'}$ times less than regular deep convolution. The results after convolution are concatenated according to depth. Thus, the output obtained is a 3D tensor block with dimensions $(h^{'} \times w^{'} \times c)$.

- Pointwise Convolution: This has the effect of changing the depth of the output of the above step from c to $c^{'}$. We will apply $c^{'}$ filter size $(1 \times 1 \times c)$ so the width and height dimensions do not change, only the depth changes, as illustrated in Fig. 17. The final result we get is an output of size $(h^{'} \times w^{'} \times c^{'})$. The number of parameters to apply in this case is $(c^{'} \times c)$.

To summarize, the improvement of MobileNet compared to other CNNs is shown on two values.

- Number of parameters: To generate a shape with dimen-

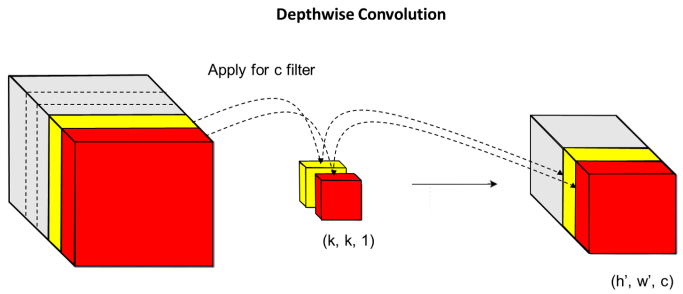


Fig. 16. Performing convolution on each slice [80].

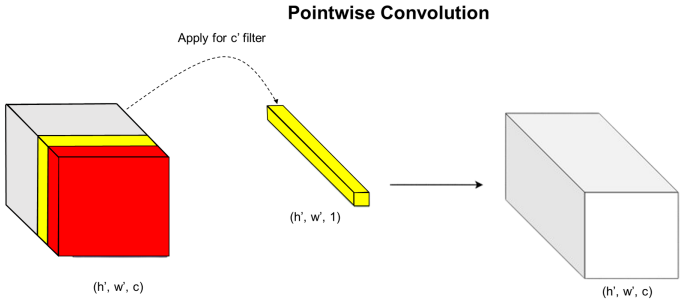


Fig. 17. Computing the point convolution [80].

sions $(h^{'} \times w^{'} \times c^{'})$, the depthwise convolution: $k \times k \times c + c^{'} \times c$, Normally $c^{'}$ will be larger than c because the later the layers, the greater the depth. Therefore the ratio: $\frac{c^{'} \times k \times k \times c}{k \times k \times c + c^{'} \times c} = \frac{c^{'} \times k \times k}{k \times k + c^{'}}$. This ratio will be close to $(k \times k)$ when $c^{'}$ decreases to near $(k \times k)$.

- Number of operations to be performed: To generate an output shape with dimensions $(h^{'} \times w^{'} \times c^{'})$, the depth-separate convolution only has to be performed sequentially on the deep convolution is $(h^{'} \times w^{'} \times c) \times (k \times k)$ multiplication operation, the point convolution is $(h^{'} \times w^{'} \times c) \times (h^{'} \times w^{'})$ multiplication operation.

The ratio of calculations between regular convolution and deep convolution is computed as Eq. (7).

$$\frac{h^{'} \times w^{'} \times c^{'} \times k \times k \times c}{h^{'} \times w^{'} \times c \times k \times k + h^{'} \times w^{'} \times c \times h^{'} \times w^{'}} = \frac{c^{'} \times k \times k}{k \times k + h^{'} \times w^{'}} \quad (7)$$

MobileNet v2 SSD (Mb2-ssd): In this network, SSD uses MobileNet v2 (Mb2) [83] as its backbone. MobileNet V2 has several improvements over MobileNetV1 that give it higher accuracy, fewer parameters, and fewer calculations. MobileNetV2 also uses shortcut connections like the ResNet [20]. Blocks in the previous layer are added directly to the next layer. If we consider the previous layer as ***x***, after going through two-dimensional convolution processing, we get the result ***f(x)***, then the final output is a residual block with value ***x+ f(x)***, as presented in Fig. 18.

However, the shortcut connection in MobileNet V2 is adjusted so that the number of channels (or depth) at the input and output of each residual block is narrowed. That's why they

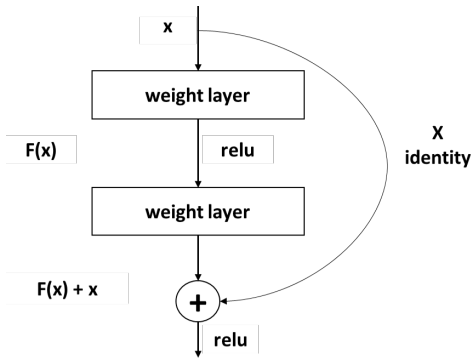


Fig. 18. The residual block of ResNet [20].

are called bottleneck layers (the bottleneck is a term often used in DL to refer to architectures that shrink in size in a certain dimension), as presented in Fig. 19.

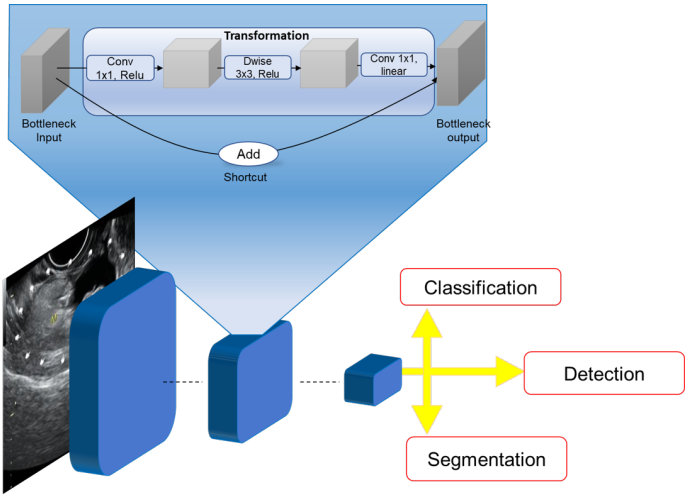


Fig. 19. The the bottleneck block of MobileNet V2 [83].

SqueezeNet SSD (Sq-SSD-Lite): In this network, SSD uses SqueezeNet(Sq) [84] as its backbone. The architecture of SqueezeNet includes 3 strategies, as shown in Fig. 20.

Strategy 1: This strategy reduces the number of 9x parameters by replacing a series of 3×3 filters with 1×1 filters. Typically, a larger 3×3 convolution filter captures the spatial information of pixels that are close together. A 1×1 convolution filter focuses on a single pixel and captures the relationship between its channels instead of neighboring pixels.

Strategy 2: This strategy reduces the number of parameters essentially just by using fewer filters. "Squeeze" layers are convolutional layers made up of only 1×1 filters, and "expand" layers are convolutional layers with a combination of 1×1 and 3×3 filters. By reducing the number of filters in the "squeeze" layer that feeds into the "extend" layer, we are reducing the number of connections into these 3×3 filters, thus reducing the total number of parameters. The authors of this article call

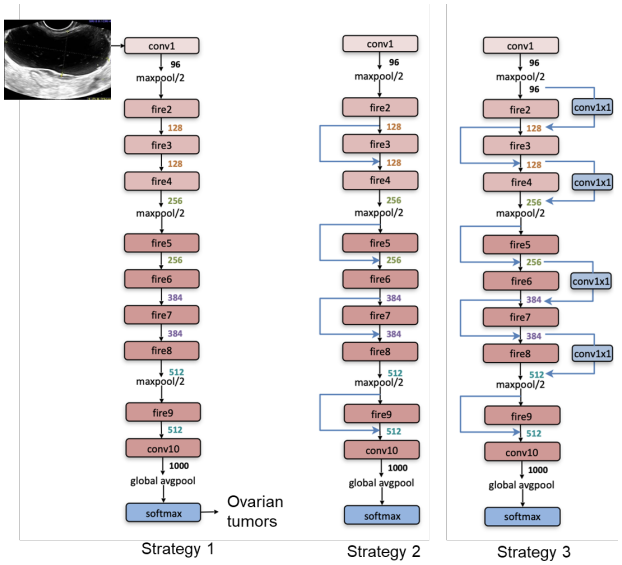


Fig. 20. The architecture of SqueezeNet [84].

this particular architecture the "fire module," and it serves as the basic building block for the SqueezeNet architecture.

Strategy 3: This strategy performs late sampling in the network so that the convolutional layers have large activation maps. Classification accuracy is increased by reducing the stride with later convolutional layers and thus creating larger feature/activation maps later in the network.

Single Shot Detection Lite (SSD-Lite): In this paper, we use SSD-Lite [85] to train models based on feature maps extracted from backbones, the architecture is presented in Fig. 21. This architecture includes three parts: the backbone, the detection neck part, and the detection part. The MBlitenet lightweight backbone is proposed for feature extraction. The CFPN [86](Circle Feature Pyramid Networks) used the level 3-7 feature maps from the backbone network for feature fusion in the detection of the neck part. The detection part used the fused feature maps to get the object class and BB regression.

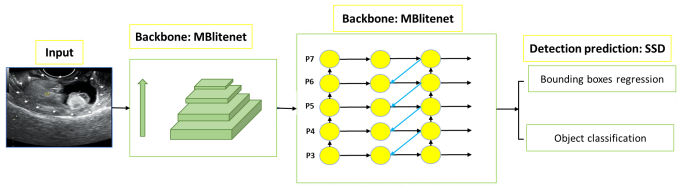


Fig. 21. The architecture of SSD-Lite [85].

IV. Experimental Results

A. OTUM Ultrasound Images Datasets

OTU-2D dataset: The MMOTU image dataset [17] comprises two subsets with two modes: OTU-2D and OTU-CEUS, including 1469 2D ultrasound images and 170 CEUS images. Across both subsets, there are semantic pixel-wise annotations and global category annotations. All images were collected from the Beijing Shijitan Hospital, Capital Medical University. The OTU dataset contains 1639 ovarian ultrasound images collected from 294 patients. The MMOTU image dataset includes eight typical categories of OTUM masses. The samples in each class are imbalanced, mainly due to certain types of tumor masses being more common while others are rare, as detailed in Table I. The OTUM annotations for the MMOTU image dataset are provided at a pixel-level and were created by 27 experts from the Obstetrics and Gynecology department. Each image is initially annotated by one expert and subsequently reviewed by another expert, ensuring the quality of the annotations. During the annotation process, experts reference pathology reports, making the annotations accurate and convincing.

TABLE I. Number of images in OTU-2D dataset

Labels	Classes	Number of images
1	Chocolate cyst	336
2	Serous cystadenoma	219
3	Teratoma	336
4	Theca cell tumor	88
5	Simple cysts	66
6	Ovary normal	267
7	Mucinous cystadenoma	104
8	High grade serous cystadenoma	53

The images in the dataset have varying aspect ratios. In the OTU-2D dataset, the width and height of the images range from 302 to 1135 and 226 to 794 pixels, respectively. In the OTU-CEUS set, the width and height of the images range from 330 to 888 and 218 to 657 pixels, respectively. Before training, input images are resized and randomly cropped to 384×384 pixels. In this paper, we only use the OTU-2D dataset to fine-tune the OTUM detection and classification model. The OTU-2D dataset is divided into two sets, the training set includes 1000 images for model training and the testing set includes 469 images for model testing. The labels of OTUM types are shown in Fig. 22.

The OTU-2D dataset with 8 classes of OTUM is called "OTU-2D_8classes" and is presented in the link ([1]). The characteristics of the tumor classes (as Tab. I) are challenging to distinguish between the classes. Tumors in the classes with labels "1", "2", "3", and "7" are protruding, darker in color compared to the surrounding area, and quite distinct. Tumors in the classes with labels "4", and "5" are no common features among the tumors. Tumors in the classes with labels "6", and "8" are unclear characteristics, pale in color, similar to the surrounding area. The class imbalance occurs when there is a significant difference in the number of samples between different classes. In tumor classification, the classes with labels

[1]https://drive.google.com/drive/folders/1zZAXzgYdrNjXUMgr0GkWxRiN1sbrAC_w?usp=sharing

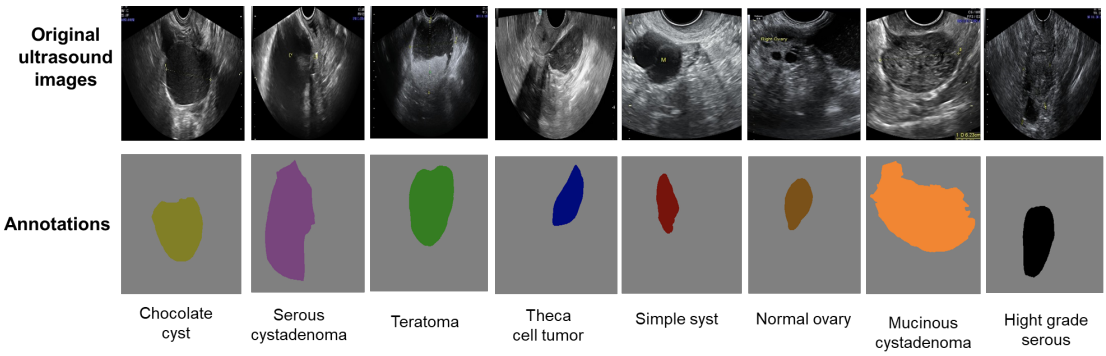


Fig. 22. Illustrating the ultrasound images and eight labels OTUM annotations of OTU-2D dataset MMOTU. The top row is an ultrasound image. The bottom row is the annotation data, each label has a different color.

”2”, ”3”, ”4”, ”7” in the OUT-2D dataset have more instances than another so leading the model to be biased towards the majority class. Meanwhile, ultrasound images in the tumor mass of classes with the label ”7” and label ”8” lacks clear characteristics, appearing faint and blending in color with the surrounding area, making them challenging to differentiate. To prepare data for training the model to detect with or without OTUM on ovarian ultrasound images, we perform data normalization of the annotation ovarian mass data. If a data region has a label of ”1” to ”8” of OTUM labels as shown in Tab. I and Fig. 22, then that region is labeled ”1”. This set is called ”OTU-2D_1class” and is presented in the link ([2]).

USOVA3D dataset: In 2020, Potočnik et al. [18] developed and introduced the USOVA3D ultrasound image dataset. OTUM masses were evaluated and labeled by two independent experts. The USOVA3D ovarian ultrasound image dataset was constructed by a group of gynecologists and ultrasound experts from UKC along with researchers from UM FERI. Medical experts assessed the images and provided annotations for OTUM masses. This dataset comprises 35 volumes of OTUM masses from women that have been annotated. Each entry in USOVA3D dataset includes a 3D ultrasound image and corresponding annotations of OTUM masses. Entries in the dataset are linked to five files, each as follows: One file contains the 3D image, two files contain the segmentation of the ovary, and two files contain the segmentation of the OTUM mass, performed by two independent evaluators. All data files are in VTK data format.

All entries in the USOVA3D dataset are divided into two datasets: a training set and a testing set. The training set includes 16 volumes, with each entry containing the original 3D image and annotations for both the ovary and OTUM masses from both evaluators. In contrast, the testing set consists of 19 volumes, where only the original 3D images are provided, as presented in Fig. 23. The dataset’s separation and design, without ground truth data for the testing set, allow for algorithm evaluation using the USOVA3D web service. From the various volumes, we can extract 2D ultrasound images for training, and testing, and the OTUM annotation is illustrated in Fig. 24.

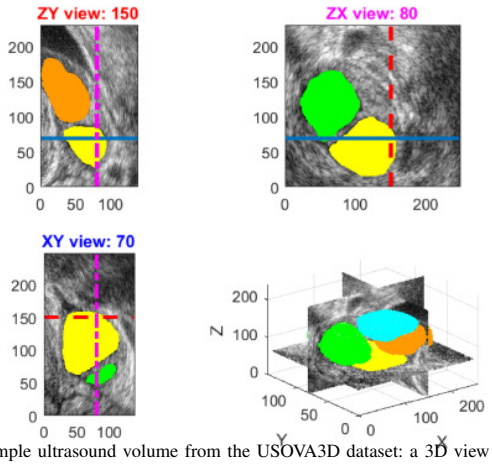


Fig. 23. Sample ultrasound volume from the USOVA3D dataset: a 3D view of selected cross-sections through volume with annotated follicles (bottom-right) and 2D views of annotated follicles superimposed on selected cross-sections (from top-left to bottom-left) [18].

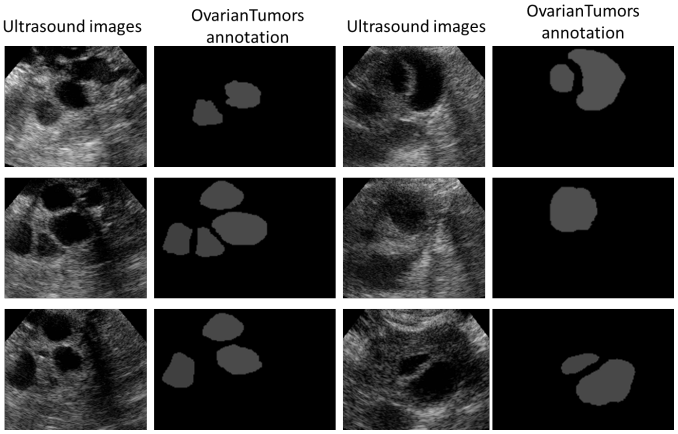


Fig. 24. Illustration of ovarian ultrasound images and annotation data of OTUM in the USOVA3D dataset obtained in 2D space.

Based on the annotation data of two experts, the annotation data of the first expert is called ”USOVA3D_2D_f_r1”, and

[2]https://drive.google.com/drive/folders/1MVcfb84cUZZkEQa30ETgzoCp9P5\\H8KCA?usp=sharing

is presented in the link ([3]), the annotation data of the second expert is called "USOVA3D_2D_f_r2", and is presented in the link ([4]). To prepare for training and testing the model, we divide the data into a training set, validation set, and testing set with the following ratios: 70% training, 15% validation, 15% testing (called the "USOVA3D_2D_f_r1_70_15_15", "USOVA3D_2D_f_r2_70_15_15"); 80% training, 20% testing (called the "USOVA3D_2D_f_r1_80_20", "USOVA3D_2D_f_r2_80_20"). In USOVA3D dataset, ultrasound image data has some challenges. (1) Limited data: ultrasound data is often quite limited and expensive to gather. This can pose challenges in training deep learning models with high accuracy. (2) Low resolution: ultrasound images typically have lower resolution compared to images captured using conventional imaging methods. This can increase difficulties in extracting crucial information from the images. (3) Noise issues: ultrasound images often contain high levels of noise, especially when collected from hard-to-reach areas. The model needs to effectively operate in noisy conditions to ensure accuracy. (4) Uniformity across hospitals: the data from multiple hospitals may exhibit low uniformity. Models need to efficiently work across various types of equipment and diverse datasets.

B. Fine-Tuning OTUM Detection and Classification Model

YOLO families allow fine-tuning of OTUM detection models on custom datasets. As shown in Fig. 24, we perform fine-tuning of the OTUM detection and classification model on two datasets: OTU-2D and USOVA3D.

On the OTU-2D dataset, we normalized to the size of 640×480 pixels, and the BB is normalized in the format of the COCO2017 dataset [87], and the labels of OTUM on the images. We perform fine-tuning YOLOv8n, YOLOv8s, YOLOv8m, and YOLOv8l for two types of output predictions (study 1): (a) with or without OTUM on the image; (b) detection and classification of 8 types of OTUM. The implementation process is shown in Fig. 27. In this paper, we use the source code of YOLOv5 (in link ([5])), YOLOv7 (in link ([6])), YOLOv8 (in link ([7])), YOLO-NAS (in link ([8])). In this paper, we also perform fine-tuning of the OTUM detection and classification model with CNNs of the SSD (VGG16-SSD, Mb1-SSD, Mb1-SSD-Lite, Sq-SSD-Lite, Mb2-SSD-Lite) families, the source code is shown in the link ([9]). These CNNs also use original data with the same structure as COCO format, but the original data is formatted in a "*.json*" file, the data format source code is also saved with the dataset.

On the USOVA3D dataset, we normalized to the size of 180×120 pixels. USOVA3D data is also standardized according to the COCO2017 format. In this paper, we only perform fine-tuning of the model to detect with or without ovarian masses on ovarian ultrasound images (study 2).

Before performing the training and evaluation of the right-hand detection model on the image, we normalize the BB annotation (as illustrated in Fig. 25) of the hand to YOLO's BB format with the COCO2017 dataset [87], as presented in Eq. (8).

$$\begin{aligned} a &= \frac{\frac{x\_max+x\_min}{2}}{w\_b}; \quad b = \frac{\frac{y\_max+y\_min}{2}}{h\_b} \\ c &= \frac{w\_b}{w\_im}; \quad d = \frac{h\_b}{h\_im} \end{aligned} \tag{8}$$

where (x_max, y_max) are the coordinates of the top right corner of the hand BB on the image, (x_max, y_max) is the coordinates of the bottom left corner of the hand BB on the image, w_b is the width size of the BB of the hand on the image, h_b is the height size of the hand BB on the image, w_im is the width size of the image, h_im is the height size of the image.

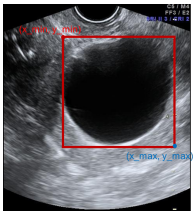


Fig. 25. Illustration of the BB annotation of the OTUM.

The data structure of the BB in the format of COCO2017 used for training and evaluation has the form (l, a, b, c, d), where l is the label of the object. In this paper, we deployed the fine-tuning and testing models on a server with NVIDIA GeForce RTX 2080 Ti, 12GB GPU. The programs were written in the Python language ($\geq$3.7 version) with the support of the CUDA 11.2/cuDNN 8.1.0 libraries. In addition, there are several libraries such as OpenCV, matplotlib, mmcls$\geq$0.20.1, numpy, packaging, prettytable, PyTorch 1.5+, etc.

C. Evaluation Matrix

To evaluate OTUM detection and classification, we employ common metrics such as Recall(R), Precision(P), Accuracy(Acc), and IoU.

- **Recall**:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{9}$$

[3]https://drive.google.com/drive/folders/1lK2OFzMHbqbjVL_GcwcGx_3_Lmy_m334?usp=sharing

[4]https://drive.google.com/drive/folders/1Kc-geRaAfhKrLz28yhWfxzSKb7weVkDT?usp=sharing

[5]https://github.com/ultralytics/yolov5

[6]https://github.com/WongKinYiu/yolov7

[7]https://github.com/ultralytics/ultralytics

[8]https://github.com/naseemap47/YOLO-NAS

[9]https://github.com/tensorturtle/mobilenet-ssd-training

- **Precision**:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{10}$$

- **Accuracy**:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{11}$$

where TP (True Positive) is the correct label prediction, FP (False Positive) is a mislabeled prediction, TN (True Negative) is the correct prediction of the negative label, and FN (False Negative) is a false negative label prediction.

- **IoU**: Intersection Over Union (IoU) is a number that quantifies the degree of overlap between two boxes. In the case of object detection and segmentation, IoU evaluates the overlap of the ground truth and prediction region.

The threshold of IoU we use is 0.5. In the process of fine-tuning the model to detect ovarian tumors on ultrasound images, we performed it many times with different numbers of epochs. When the number of epochs is 300, the trained model is no longer better when evaluating the validation set and the evaluation result on the validation set is the highest. Therefore, we choose the number of epochs being 300 to limit all CNNs tested and compared in this paper.

D. Results, Discussions, and Challenges

As compared in [88], and [89], Faster R-CNN (two-stage network) has more accurate mAP results than YOLOv3 and SSD. However, the computation time of Faster R-CNN is much slower than YOLOv3 and SSD. Due to the computational time requirements of an ovarian tumor detection system, it needs to be the processing time in real-time. In this paper, we only compare single-stage networks (YOLO family and SSD family). The results of detecting with or without OTUM in ovarian ultrasound images (USOVA3D_2D_f_r1_70_15_15) are shown in Table II. The results of detecting with or without OTUM on ovarian ultrasound images (USOVA3D_2D_f_r2_70_15_15) are shown in Table III. In this paper, we compared the detection results of YOLOv5, YOLOv7, YOLOv8, YOLO-NAS, VGG16-SSD, Mb1-SSD, Mb1-SSD-Lite, Sq-SSD-Lite, Mb2-SSD-Lite with or without ovarian tumors with MU Net [90], the results are shown in Table II and Table III.

The results of detecting with or without OTUM on ovarian ultrasound images in "USOVA3D_2D_f_r1_70_15_15" and "USOVA3D_2D_f_r2_70_15_15" sets of MU Net [90] are the highest ($Acc = 98.4\%, P = 98.3\%, R = 63.8\%$). The results of CNNs belonging to the SSD family are superior to those of the YOLO family, MU Net [90] is improved with the combination of MobileNetV2 and U-Net for ovarian tumor detection, so its results are higher than the networks belonging to the YOLO family and SSD family. However, the result of $R = 63.8\%$ of MU Net is very low. This problem shows that the falsely detected result of MU Net [90] was 36.2%, which is a very high rate of mistakenly detecting ovarian tumors on ultrasound images. While the results of detecting the with or without ovarian tumors on ultrasound images of Mb1-SSD-Lite are high ($Acc = 96.\%50, P = 95.70\%, R = 96.52\%$ in Table II) and ($Acc = 97.\%69, P = 97.11\%, R = 97.69\%$ in Table III).

The results of detecting with or without OTUM on ovarian ultrasound images (USOVA3D_2D_f_r1_80_20) are shown in Table IV. In Table IV, the highest result in detecting OTUM is Mb1-SSD ($Acc = 98.9\%, P = 98.58\%, R = 98.9\%$). The results show that when the SSD network uses the standard model, it gives the highest results and is higher than when using the lightweight model. The CNNs belonging to the YOLO family with YOLOv8l have the highest results ($P = 97.22\%, R = 75.48\%$). The results of detecting with or without OTUM on ovarian ultrasound images (USOVA3D_2D_f_r2_80_20) are shown in Table V. In Table V, the highest result in detecting OTUM is YOLOv8l ($P = 99.34\%, R = 81.27\%$). The results of detecting and classifying 8 classes of OTUM of the OTU-2D dataset are presented in Table VI. In Table VI, the best result based on the Acc measure is Sq-SSD-Lite model with ($Acc = 92.04\%$), the best result based on the P measure is YOLOv5 model with ($P = 82.91\%$), the best result based on the R measure is Sq-SSD-Lite model with ($R = 92.04\%$).

Fig. 26 shows the confusion matrix of OTUM detection and classification with 8 classes of OTUM of the OTU-2D dataset when using the fine-tuned YOLOv5 model for prediction. Fig. 26 also shows the prediction result of the 3^{rd} label (Teratoma) having the highest result of 81%, and the 5^{th} label (Simple syst) having the lowest result of 53%. As Fig. 22, the geometric structure of OTUM on ultrasound and ground-truth images is the simplest geometric structure. The 5^{th} label is detected and wrong classified with the 2^{nd} label (Serous cystadenoma) and the background. Another label that also has a high rate of wrong classification is the 4^{th} label, which is often wrongly classified with the 5^{th} label.

Fig. 27 illustrates the ground truth and predictive results for 8 classes of OTUM and the BB of OTUM on the OTU-2D dataset. The top (four rows of images above) is the ground truth BB and tumor label data, and the bottom (four rows of images below) is the BB and label prediction result of OTUM.

At the top, the first image is named *"1002.jpg"*, labeled *"5"*, and the OTUM is represented by a green BB. Prediction results at the bottom, the first image is named *"1002 .jpg"*, the OTUM label prediction is *"6"*, and the confidence score is 0.3, represented by a lighter blue BB labeled *"5"*. This is a case where the label of an OTUM was wrong predicted.

In the second image of row 1 at the top, the image is named *"548.jpg"*, the label of the OTUM is *"1"* and is represented by a BB with the IndianRed color. The prediction and classification results are shown in the second image row 1 at the bottom, the image is named *"548.jpg"*, the prediction result is labeled *"1"*,

TABLE II. OTUM DETECTION RESULTS ON THE USOVA3D_2D_F_R1_70_15_15 SET (300 EPOCHS)

Models/ Metrics	VGG16-SSD	Mb1-SSD	Mb1-SSD-Lite	Sq-SSD-Lite	Mb2-SSD-Lite	YOLO v5	YOLO v7	YOLO v8m	YOLO v8s	YOLO v8n	YOLO v8l	YOLO -NAS-m	MU Net
Acc(%)	81.44	95.60	96.50	95.07	94.49	-	-	-	-	-	-	-	**98.4**
P(%)	66.67	94.87	95.70	91.75	93.69	95.50	93.20	96.22	94.33	95.49	97.48	51.29	**98.3**
R(%)	81.44	95.60	**96.52**	95.07	94.49	66.80	62.11	73.10	64.10	71.58	75.24	58.04	63.8

TABLE III. OTUM DETECTION RESULTS ON THE USOVA3D_2D_F_R2_70_15_15 SET (300 EPOCHS)

Models/ Metrics	VGG16-SSD	Mb1-SSD	Mb1-SSD-Lite	Sq-SSD-Lite	Mb2-SSD-Lite	YOLO v5	YOLO v7	YOLO v8m	YOLO v8s	YOLO v8n	YOLO v8l	YOLO -NAS-m	MU Net
Acc(%)	84.73	97.41	97.69	95.39	97.12	-	-	-	-	-	-	-	**98.4**
P(%)	33.10	96.59	97.11	92.98	95.91	97.10	94.50	98.55	95.78	96.34	98.72	55.27	**98.3**
R(%)	84.73	97.41	97.69	95.39	97.12	72.70	64.25	68.40	66.32	68.29	73.85	62.79	63.8

TABLE IV. OTUM DETECTION RESULTS ON THE USOVA3D_2D_F_R1_80_20 SET (300 EPOCHS)

Models/ Metrics	VGG16-SSD	Mb1 -SSD	Mb1-SSD-Lite	Sq-SSD-Lite	Mb2-SSD-Lite	YOLO v5	YOLO v7	YOLO v8m	YOLO v8s	YOLO v8n	YOLO v8l	YOLO -NAS-m
Acc(%)	80.62	**98.90**	98.24	96.70	98.46	-	-	-	-	-	-	-
P(%)	28.41	**98.58**	97.83	95.19	97.92	96.50	92.46	96.73	94.37	95.44	97.22	52.99
R(%)	80.62	**98.90**	98.24	96.70	98.46	67.20	65.34	72.19	67.29	68.17	75.48	59.35

TABLE V. OTUM DETECTION RESULTS ON THE USOVA3D_2D_F_R2_80_20 SET (300 EPOCHS)

Models/ Metrics	VGG16-SSD	Mb1 -SSD	Mb1-SSD-Lite	Sq-SSD-Lite	Mb2-SSD-Lite	YOLO v5	YOLO v7	YOLO v8m	YOLO v8s	YOLO v8n	YOLO v8l	YOLO -NAS-m
Acc(%)	79.74	96.80	97.44	95.95	97.87	-	-	-	-	-	-	-
P(%)	33.19	95.44	97.01	94.31	97.16	98.50	94.23	98.44	97.66	96.74	99.34	54.46
R(%)	79.74	96.80	97.44	95.95	97.87	73.30	69.47	75.24	74.55	72.94	81.27	59.76

TABLE VI. THE RESULTS OF DETECTING AND CLASSIFYING 8 OTUM CLASSES OF THE OTU-2D DATASET (300 EPOCHS)

Labels	Model Metrics	VGG16 -SSD	Mb1 -SSD	Mb1-SSD -lite	Sq-SSD -Lite	Mb2 -SSD-Lite	YOLO v5	YOLO v7	YOLO v7-w6	YOLO v7-d6	YOLO v7-e6	YOLO v8m	YOLO v8s	YOLO v8n	YOLO v8l	YOLO -NAS-m
1	**Acc (%)**	80.30	93.94	98.48	100.00	100.00	-	-	-	-	-	-	-	-	-	-
	P (%)	45.20	90.91	87.50	77.78	85.71	73.30	56.70	58.20	55.40	52.50	69.20	73.50	74.60	80.00	-
	R (%)	80.30	93.94	98.48	100.00	100.00	70.00	76.10	74.50	74.50	80.00	71.80	68.20	70.90	62.70	-
2	**Acc (%)**	75.00	98.15	93.52	100.00	92.59	-	-	-	-	-	-	-	-	-	-
	P (%)	32.45	95.34	66.67	82.70	81.35	82.50	44.40	42.10	36.70	39.30	63.70	67.40	74.70	72.00	-
	R (%)	75.00	98.15	93.52	100.00	92.59	74.20	86.40	84.80	93.90	86.40	71.20	63.60	71.20	66.20	-
3	**Acc (%)**	38.71	83.87	90.32	93.55	80.65	-	-	-	-	-	-	-	-	-	-
	P (%)	22.46	66.67	80.00	71.43	67.25	88.20	61.30	62.40	53.80	41.00	85.20	77.70	76.40	85.80	-
	R (%)	38.71	83.87	90.32	93.55	80.65	75.90	82.30	77.00	80.60	89.80	79.80	78.70	81.50	72.20	-
4	**Acc (%)**	5.26	89.47	89.47	89.47	84.21	-	-	-	-	-	-	-	-	-	-
	P (%)	2.36	83.33	54.55	75.00	70.00	81.50	48.60	37.60	38.00	26.80	72.50	54.50	71.90	75.30	-
	R (%)	5.26	89.47	89.47	89.47	84.21	58.10	33.50	19.40	51.60	35.50	59.50	51.60	58.10	58.90	-
5	**Acc (%)**	8.05	83.91	83.91	85.06	77.01	-	-	-	-	-	-	-	-	-	-
	P (%)	4.56	50.00	52.46	55.67	52.90	90.00	98.70	0.00	15.40	11.50	66.50	26.60	69.90	39.00	-
	R (%)	8.05	83.91	83.91	85.06	77.01	68.40	10.50	0.00	36.80	15.80	52.20	47.40	52.60	53.80	-
6	**Acc (%)**	93.94	90.91	87.88	96.97	96.97	-	-	-	-	-	-	-	-	-	-
	P (%)	78.23	70.00	75.00	90.60	71.43	79.10	39.10	51.70	54.10	47.00	89.40	68.70	71.20	77.60	-
	R (%)	93.94	90.91	87.88	96.97	96.97	60.90	63.20	74.70	80.50	81.60	69.00	64.40	68.00	59.80	-
7	**Acc (%)**	73.33	80.00	93.33	100.00	86.67	-	-	-	-	-	-	-	-	-	-
	P (%)	62.45	75.00	78.25	92.50	68.70	86.80	27.90	23.80	21.00	17.70	68.80	45.70	78.00	58.90	-
	R (%)	73.33	80.00	93.33	100.00	86.67	66.70	48.50	54.50	81.80	66.70	63.30	56.20	57.60	57.60	-
8	**Acc (%)**	2.56	52.40	62.40	71.29	65.34	-	-	-	-	-	-	-	-	-	-
	P (%)	1.59	32.70	45.70	52.80	49.80	81.90	39.10	33.20	26.90	23.10	51.80	22.50	36.30	43.20	-
	R (%)	2.56	52.40	62.40	71.29	65.34	46.70	20.00	26.70	13.30	30.10	60.00	46.70	46.70	46.70	-
Average	**Acc (%)**	47.14	84.08	87.41	**92.04**	85.43	-	-	-	-	-	-	-	-	-	-
	P (%)	31.16	70.49	67.52	74.81	68.39	**82.91**	51.98	38.63	37.66	32.36	70.89	54.58	69.13	66.48	43.67
	R (%)	47.14	84.08	87.41	**92.04**	85.43	65.11	52.56	51.45	64.13	60.74	65.85	59.60	63.33	59.74	77.97

the confidence score is 0.5, and is represented by a BB equals BB with color IndianRed. This result is the result of the correct classification of OTUM.

Based on the results in Table II-VI, there are many cases of undetected OTUM. This proves that many OTUM have not been detected and classified properly. This is very dangerous in detecting and diagnosing OCAN. For example, a woman has an OTUM but it is not detected through medical examination, and this person will not receive early treatment for the tumor, which will lead to the tumor becoming larger and metastasizing to organs, and others in the body When detected, the disease is in a late stage and very difficult to treat. This reduces the patient's chance of survival. In medical examination and treatment, especially when detecting OTUM, it is often better to accept "a mistake than a miss", thereby increasing the ability to detect early and promptly treat OTUM.

Currently, there is a CNN model with very high accuracy and recall results, nearly 100% for the problem of detecting tumors on ovarian ultrasound images. However, the results of classifying 8 types of OTUM in Tab. VI are very poor. This

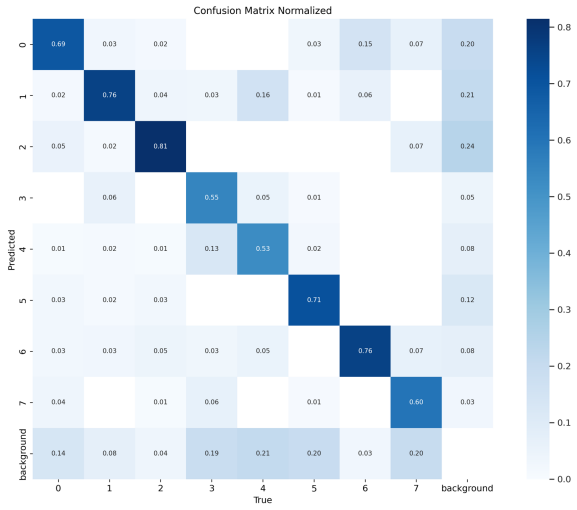


Fig. 26. Illustrate the confusion matrix normalized for detection and classification of 8 classes of OTUM of the OTU-2D using the fine-tuned YOLOv5 model.

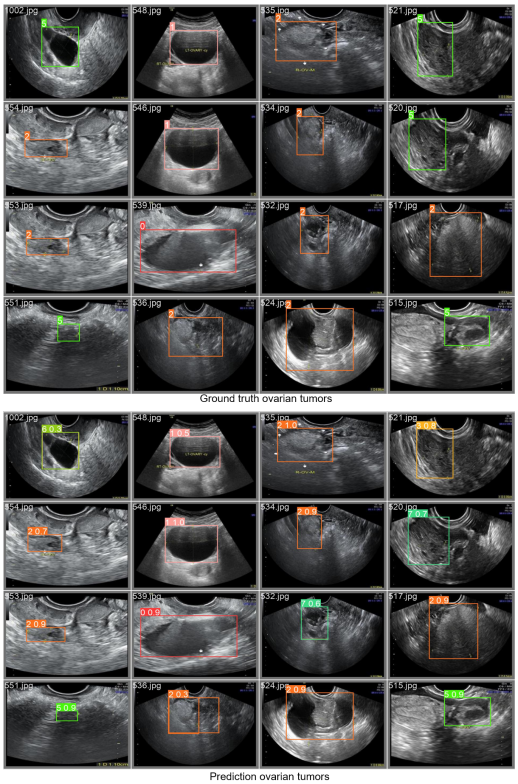


Fig. 27. Illustrating the ground truth data and prediction of 8 classes of OTUM by YOLOv5. The top is the ground truth data about the label and BB of the OTUM area, the bottom is the prediction result of the label, confidence score, and BB of the OTUM.

happens because the YOLO and SSD networks mainly perform well on the object detection problem but do not perform well on the object classification problem. The process of object classification here is to detect ovarian tumors labeled as one of 8 tumor types in the OTU-2D dataset. Therefore, in the future,

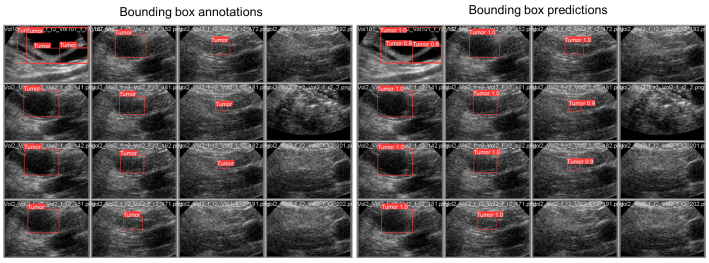


Fig. 28. Illustrating the ground truth data and prediction of OTUM by Mb1-SSD. The left is the ground truth data about the label and BB of the OTUM area, the right is the prediction result of the label, confidence score, and BB of the OTUM.

it is necessary to improve the problem of classifying tumors. During the process of conducting this research, we realized some challenges in building the following support system for detecting and diagnosing OTUM. (1) the resolution of ovarian ultrasound images is very low, such as in the USOVA3D dataset, images are only about 120pixels in size; (2) the problem of labeling tumor data is very difficult, as this problem often requires expert work and consumes a lot of time; (3) confidentiality issues and the difficulty of collecting ovarian ultrasound images; (4) requirements for a hospital support system need to be implemented on computers with low computing space, especially running only on CPUs.

About the performance of YOLO and SSD. YOLO: Renowned for high processing speed, and suitable for real-time processing applications, SSD is also fast, but typically slower than YOLO. About the accuracy, YOLO generally has good accuracy, especially for larger objects and SSD achieves high accuracy as well, but may be weaker than YOLO for smaller objects. About handling small objects: YOLO may struggle with detecting small objects, but SSD usually performs better in detecting small objects. By handling overlapping objects: YOLO may have issues with overlapping objects as it primarily relies on grid cells for predictions. SSD is Better at handling overlapping objects by using multiple default bounding boxes for predictions. By generalization and training capability: YOLO exhibits good generalization and is easily trainable on large datasets. SSD also has high generalization capabilities and can be trained on various types of data.

V. Conclusions and Future Works

One-stage CNNs have shown outstanding ability in solving computer vision problems. Although they have normal accuracy, their speed is many times faster than two-stage CNNs. To meet the requirements of building a system to help detect and diagnose OTUM on ovarian ultrasound images, we tested a series of CNNs from the YOLO and SSD families for the problem of detecting and classifying OTUM on ultrasound images of two typical ovarian ultrasound image databases: OTU and USOVA3D. The results were evaluated in terms of precision, accuracy, and recall, with the issue of detecting OTUM (with or

without OTUM on ovarian ultrasound images), the results were very high (the highest is $Acc = 98.9\%, P = 98.58\%, R = 98.8\%$ of Mb1-SSD on the USOVA3D_2D_f_r1_80_20 set, highest is $Acc = 97.87\%, P = 97.16\%, R = 97.87\%$ of Mb2-SSD-Lite on the USOVA3D_2D_f_r2_80_20 set). However, with the problem of tumor classification (8 OTUM classes on the OTU dataset), the results are low (highest is only $Acc = 92.04, 74.81\%, R = 92.81$ and YOLO and SSD results are both less than 85%). In the future, we will continue to research to extract good features for classifying OTUM.

ACKNOWLEDGMENT

This research is funded by Tan Trao University in Tuyen Quang province, Vietnam.

REFERENCES

[1] A. C. Society, "Key Statistics for Ovarian Cancer," https://www.cancer.org/cancer/types/ovarian-cancer/about/key-statistics.html, 2023, [Online; accessed 19-September-2023].

[2] A. Gupta, P. Jha, T. M. Baran, K. E. Maturen, K. Patel-Lippmann, H. M. Zafar, A. Kamaya, N. Antil, L. Barroilhet, and E. Sadowski, "Ovarian cancer detection in average-risk women: classic-versus nonclassic-appearing adnexal lesions at us," *Radiology*, vol. 303, no. 3, pp. 603–610, 2022, doi: 10.1148/radiol.212338.

[3] E. Meys, J. Kaijser, R. Kruitwagen, B. Slangen, B. Van Calster, B. Aertgeerts, J. Verbakel, D. Timmerman, and T. Van Gorp, "Subjective assessment versus ultrasound models to diagnose ovarian cancer: A systematic review and meta-analysis," *European Journal of Cancer*, vol. 58, pp. 17–29, 2016, doi: 10.1016/j.ejca.2016.01.007.

[4] A. Voulodimos, N. Doulamis, A. Doulamis, E. Protopapadakis *et al.*, "Deep learning for computer vision: A brief review," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–13, 2018, doi: 10.1155/2018/7068349.

[5] A. Lohia, K. D. Kadam, R. R. Joshi, and A. M. Bongale, "Bibliometric analysis of one-stage and two-stage object detection," *Libr. Philos. Pract*, vol. 4910, p. 34, 2021.

[6] M. C. García, J. T. Mateo, P. L. Benítez, and J. G. Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," *Remote Sensing*, vol. 13, no. 1, p. 89, 2020, doi: 10.3390/rs13010089.

[7] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using yolo: Challenges, architectural successors, datasets and applications," *multimedia Tools and Applications*, vol. 82, no. 6, pp. 9243–9275, 2023, doi: 10.1007/s11042-022-13644-y.

[8] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision–ECCV 2016*, vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[9] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580-587, 2014, doi: 10.1109/CVPR.2014.81.

[10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[11] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017, doi: 10.1109/TPAMI.2016.2577031.

[12] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," *Advances in neural information processing systems*, vol. 29, 2016.

[13] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[14] M. G. Naftali, J. S. Sulistyawan, and K. Julian, "Comparison of object detection algorithms for street-level objects," *Computer Vision and Pattern Recognition*, 2022, doi: 10.48550/arXiv.2208.11315.

[15] L. Tan, T. Huangfu, L. Wu, and W. Chen, "Comparison of yolo v3, faster r-cnn, and ssd for real-time pill identification," *Research Square*, 2021, doi: 10.21203/rs.3.rs-668895/v1.

[16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv*, 2018.

[17] Q. Zhao, S. Lyu, W. Bai, L. Cai, B. Liu, M. Wu, X. Sang, M. Yang, and L. Chen, "A multi-modality ovarian tumor ultrasound image dataset for unsupervised cross-domain semantic segmentation," *Computer Vision and Pattern Recognition*, 2022, doi: 10.48550/arXiv.2207.06799.

[18] B. Potočnik, J. Munda, M. Reljič, K. Rakić, J. Knez, V. Vlaisavljević, G. Sedej, B. Cigale, A. Holobar, and D. Zazula, "Public database for validation of follicle detection algorithms on 3d ultrasound images of ovaries," *Computer Methods and Programs in Biomedicine*, vol. 196, p. 105621, 2020, doi: 10.1016/j.cmpb.2020.105621.

[19] T.-L. Pham, V.-H. Le, T.-H. Tran, and D. H. Vu, "Comprehensive study on semantic segmentation of ovarian tumors from ultrasound images," in *Conference on Information Technology and its Applications*, pp. 262–273, 2023.

[20] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016, doi: 10.1109/CVPR.2016.90.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference*, vol. 9351, pp. 234–241, 2015, doi: 10.1007/978-3-319-24574-4_28.

[22] H. Zhao, J. Shi, X. Qi, X. Wang and J. Jia, "Pyramid Scene Parsing Network," *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6230-6239, 2017, doi: 10.1109/CVPR.2017.660.

[23] J. Fu et al., "Dual Attention Network for Scene Segmentation," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3141-3149, 2019, doi: 10.1109/CVPR.2019.00326.

[24] Z. Momenimovahed, A. Tiznobaik, S. Taheri, and H. Salehiniya, "Ovarian cancer in the world: Epidemiology and risk factors," *International Journal of Women's Health*, vol. 11, pp. 287–299, 2019, doi: 10.2147/IJWH.S197604.

[25] J. M. Puaschunder, "The Potential for Artificial Intelligence in Healthcare," *SSRN Electronic Journal*, vol. 6, no. 2, pp. 94–98, 2020, doi: 10.2139/ssrn.3525037.

[26] J. Bajwa, U. Munir, A. Nori, and B. Williams, "Artificial intelligence in healthcare: transforming the practice of medicine," *Future Healthcare Journal*, vol. 8, no. 2, 2021, doi: 10.7861/fhj.2021-0095.

[27] V. Sudha and T. R. Ganeshbabu, "A convolutional neural network classifier VGG-19 architecture for lesion detection and grading in diabetic retinopathy based on deep learning," *Computers, Materials and Continua*, vol. 66, no. 1, pp. 827–842, 2021, doi: 10.32604/cmc.2020.012008.

[28] A. Karagoz, D. Alis, M. E. Seker, G. Zeybel, M. Yergin, I. Oksuz, and E. Karaarslan, "Anatomically guided self-adapting deep neural network for clinically significant prostate cancer detection on bi-parametric MRI: a multi-center study," *Insights into Imaging*, vol. 14, no. 1, 2023, doi: 10.1186/s13244-023-01439-0.

[29] S. Jin, G. Liu, and Q. Bai, "Deep Learning in COVID-19 Diagnosis, Prognosis and Treatment Selection," *Mathematics*, vol. 11, no. 6, pp. 1–17, 2023, doi: 10.3390/math11061279.

[30] M. Mokoatle, V. Marivate, D. Mapiye, R. Bornman, and V. M. Hayes, "A review and comparative study of cancer detection using machine learning: SBERT and SimCSE application," *BMC Bioinformatics*, vol. 24, no. 1, pp. 1–25, 2023, doi: 10.1186/s12859-023-05235-x.

[31] A. Ibrahim, H. K. Mohamed, A. Maher, and B. Zhang, "A Survey on Human Cancer Categorization Based on Deep Learning," *Frontiers in Artificial Intelligence*, vol. 5, 2022, doi: 10.3389/frai.2022.884749.

[32] S. H. Hosseini, R. Monsefi, and S. Shadroo, "Deep learning applications for lung cancer diagnosis: A systematic review," *Multimedia Tools and Applications*, 2023, doi: 10.1007/s11042-023-16046-w.

[33] M. A. Thanoon, M. A. Zulkifley, M. A. A. Mohd Zainuri, and S. R. Abdani, "A Review of Deep Learning Techniques for Lung Cancer

Screening and Diagnosis Based on CT Images," *Diagnostics*, vol. 13, no. 16, pp. 1–27, 2023, doi: 10.3390/diagnostics13162617.

[34] S. Wankhade and V. S., "A novel hybrid deep learning method for early detection of lung cancer using neural networks," *Healthcare Analytics*, vol. 3, no. May, p. 100195, 2023, doi: 10.1016/j.health.2023.100195.

[35] V. Rajasekar, M. P. Vaishnnave, S. Premkumar, V. Sarveshwaran, and V. Rangaraaj, "Lung cancer disease prediction with CT scan and histopathological images feature analysis using deep learning techniques," *Results in Engineering*, vol. 18, 2022, p. 101111, 2023, doi: 10.1016/j.rineng.2023.101111.

[36] L. Wang, "Deep Learning Techniques to Diagnose Lung Cancer," *Cancers*, vol. 14, no. 22, pp. 1–24, 2022, doi: 10.3390/cancers14225569.

[37] T. I. Mohamed, O. N. Oyelade, and A. E. Ezugwu, utomatic detection and classification of lung cancer CT scans based on deep learning and ebola optimization search algorithm, *Plos One*, vol. 18, no. 8, 2023, doi: 10.1371/journal.pone.0285796.

[38] N. Kalaivani, N. Manimaran, S. Sophia, and D. D. Devi, "Deep Learning Based Lung Cancer Detection and Classification," *IOP Conference Series: Materials Science and Engineering*, vol. 994, no. 1, 2020, doi: 10.1088/1757-899X/994/1/012026.

[39] A. Shimazaki, D. Ueda, A. Choppin, A. Yamamoto, T. Honjo, Y. Shimahara, and Y. Miki, "Deep learning-based algorithm for lung cancer detection on chest radiographs using the segmentation method," *Scientific Reports*, vol. 12, no. 1, pp. 1–10, 2022, doi: 10.1038/s41598-021-04667-w.

[40] J. Vineeth, S. Hemanth, C. V. Rao, N. Pavankumar, H. Jayanna and C. Janardhan, "Skin cancer detection using deep learning," *2022 Fourth International Conference on Cognitive Computing and Information Processing (CCIP)*, pp. 1-6, 2022, doi: 10.1109/CCIP57447.2022.10058685.

[41] M. Hajiarbabi, "Skin cancer detection using multi-scale deep learning and transfer learning," *Journal of Medical Artificial Intelligence*, vol. 6, pp. 1–9, 2023, doi: 10.21037/jmai-23-67.

[42] D. A. Ragab, M. Sharkas, S. Marshall, and J. Ren, "Breast cancer detection using deep convolutional neural networks and support vector machines," *PeerJ*, vol. 2019, no. 1, pp. 1–23, 2019, doi: 10.7717/peerj.6201.

[43] I. Buyung, A. Q. Munir, and P. Wanda, "Effective Breast Cancer Detection using Novel Deep Learning Algorithm," *JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer)*, vol. 8, no. 2, pp. 98–104, 2023, doi: 10.33480/jitk.v8i2.4077.

[44] M. Kord Tamandani, "Breast Cancer Detection Using Deep Multilayer Neural Networks," *Journal of Epigenetics*, vol. 3, no. 1, pp. 27–34, 2022, doi: 10.5220/0007386206550660.

[45] B. S. Abunasser, M. R. J. AL-Hiealy, I. S. Zaqout, and S. S. Abu-Naser, "Convolution Neural Network for Breast Cancer Detection and Classification Using Deep Learning," *Asian Pacific Journal of Cancer Prevention*, vol. 24, no. 2, pp. 531–544, 2023, doi: 10.31557/APJCP.2023.24.2.531.

[46] M. Wu, G. Cui, S. Lv, L. Chen, Z. Tian, M. Yang, and W. Bai, "Deep convolutional neural networks for multiple histologic types of ovarian tumors classification in ultrasound images," *Frontiers in Oncology*, vol. 13, p. 1154200, 2023, doi: 10.3389/fonc.2023.1154200.

[47] M. T. Hira, M. A. Razzaque, and M. Sarker, "Ovarian Cancer Data Analysis using Deep Learning: A Systematic Review from the Perspectives of Key Features of Data Analysis and AI Assurance," *ArXiv*, 2023, doi: 10.48550/arXiv.2311.11932.

[48] M. Akazawa and K. Hashimoto, "Artificial intelligence in ovarian cancer diagnosis," *Anticancer Research*, vol. 40, no. 8, pp. 4795–4800, 2020, doi: 10.21873/anticanres.14482.

[49] B. Ziyambe, A. Yahya, T. Mushiri, M. U. Tariq, Q. Abbas, M. Babar, M. Albathan, M. Asim, A. Hussain, and S. Jabbar, "A Deep Learning Framework for the Prediction and Diagnosis of Ovarian Cancer in Pre- and Post-Menopausal Women," *Diagnostics*, vol. 13, no. 10, pp. 1–15, 2023, doi: 10.3390/diagnostics13101703.

[50] H. Wang, C. Liu, Z. Zhao, C. Zhang, X. Wang, H. Li, H. Wu, X. Liu, C. Li, L. Qi, and W. Ma, "Application of Deep Convolutional Neural Networks for Discriminating Benign, Borderline, and Malignant Serous Ovarian Tumors From Ultrasound Images," *Frontiers in Oncology*, vol. 11, no. 20, pp. 1–9, 2021, doi: 10.3389/fonc.2021.770683.

[51] H. Heartlin Maria, A. Maria Jossy, and S. Malarvizhi, "A Machine Learning approach for classification of ovarian tumours," *Journal of Physics: Conference Series*, vol. 2335, no. 1, 2022, doi: 10.1088/1742-6596/2335/1/012018.

[52] D. Sengupta, S. N. Ali, A. Bhattacharya, J. Mustafi, A. Mukhopadhyay, and K. Sengupta, "A deep hybrid learning pipeline for accurate diagnosis of ovarian cancer based on nuclear morphology," *PLoS ONE*, vol. 17, no. 1, pp. 1–20, 2022, doi: 10.1371/journal.pone.0261181.

[53] D. Schwartz, T. W. Sawyer, N. Thurston, J. Barton, and G. Ditzler, "Ovarian cancer detection using optical coherence tomography and convolutional neural networks," *Neural Computing and Applications*, vol. 34, no. 11, pp. 8977–8987, 2022, doi: 10.1007/s00521-022-06920-3.

[54] L. Y. Guo, A. H. Wu, Y. X. Wang, L. P. Zhang, H. Chai, and X. F. Liang, "Deep learning-based ovarian cancer subtypes identification using multi-omics data," *BioData Mining*, vol. 13, no. 1, pp. 1–12, 2020, doi: 10.1186/s13040-020-00222-x.

[55] Y. T. Jan, P. S. Tsai, W. H. Huang, L. Y. Chou, S. C. Huang, J. Z. Wang, P. H. Lu, D. C. Lin, C. S. Yen, J. P. Teng, G. S. Mok, C. T. Shih, and T. H. Wu, "Machine learning combined with radiomics and deep learning features extracted from CT images: a novel AI model to distinguish benign from malignant ovarian tumors," *Insights into Imaging*, vol. 14, no. 1, 2023, doi: 10.1186/s13244-023-01412-x.

[56] Y. Jung, T. Kim, M. R. Han, S. Kim, G. Kim, S. Lee, and Y. J. Choi, "Ovarian tumor diagnosis using deep convolutional neural networks and a denoising convolutional autoencoder," *Scientific Reports*, vol. 12, no. 1, pp. 1–10, 2022, doi: 10.1038/s41598-022-20653-2.

[57] S.-T. Hsu, Y.-J. Su, C.-H. Hung, M.-J. Chen, C.-H. Lu, and C.-E. Kuo, "Automatic ovarian tumors recognition system based on ensemble convolutional neural network with ultrasound imaging," *BMC Medical Informatics and Decision Making*, vol. 22, no. 1, p. 298, 2022, doi: 10.1186/s12911-022-02047-6.

[58] J. Martínez-Más, A. Bueno-Crespo, S. Khazendar, M. Remezal-Solano, J.-P. Martínez-Cendán, S. Jassim, H. Du, H. Al Assam, T. Bourne, and D. Timmerman, "Evaluation of machine learning methods with fourier transform features for classifying ovarian tumors based on ultrasound images," *PLoS One*, vol. 14, no. 7, 2019, doi: 10.1371/journal.pone.0219388.

[59] W. Froyman, D. Van Schoubroeck, and D. Timmerman, "Automated follicle count using three-dimensional ultrasound in polycystic ovarian morphology," *Ultrasound in Obstetrics and Gynecology*, vol. 51, no. 1, 2018, doi: 10.1002/uog.18896.

[60] X. Liang, J. Fang, H. Li, X. Yang, D. Ni, F. Zeng, and Z. Chen, "Cr-unet-based ultrasonic follicle monitoring to reduce diameter variability and generate area automatically as a novel biomarker for follicular maturity," *Ultrasound in Medicine and Biology*, vol. 46, no. 11, pp. 3125–3134, 2020, doi: 10.1016/j.ultrasmedbio.2020.07.020.

[61] M. ArathiBoyanapalli *et al.*, "A study of preprocessing techniques and features for ovarian cancer using ultrasound images," *European Journal of Molecular and Clinical Medicine*, vol. 7, no. 10, pp. 293–303, 2020.

[62] J. Jin, H. Zhu, J. Zhang, Y. Ai, J. Zhang, Y. Teng, C. Xie, and X. Jin, "Multiple u-net-based automatic segmentations and radiomics feature stability on ultrasound images for patients with ovarian cancer," *Frontiers in Oncology*, vol. 10, p. 614201, 2021, doi:10.3389/fonc.2020.614201.

[63] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788, 2016, doi: 10.1109/CVPR.2016.91.

[64] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517-6525, 2017, doi: 10.1109/CVPR.2017.690.

[65] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv*, 2020.

[66] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "You Only Learn One Representation: Unified Network for Multiple Tasks," *ArXiv*, pp. 1–11, 2021.

[67] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *ArXiv*, pp. 1–7, 2021.

[68] Y. Zhu, S. Li, W. Du, and Y. Du, "Identification of table grapes in the natural environment based on an improved yolov5 and localization of picking points," *Precision Agriculture*, vol. 24, no. 4, pp. 1–22, 2023, doi: 10.1007/s11119-023-09992-w.

[69] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei,

and X. Wei, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *ArXiv*, 2022, doi: 10.48550/arXiv.2209.02976.

[70] D. Reis, J. Kupec, J. Hong, and A. Daoudi, "Real-Time Flying Object Detection with YOLOv8," *Computer Vision and Pattern Recognition*, 2023, doi: 10.48550/arXiv.2305.09972.

[71] YOLO-NAS, "A Next-Generation, Object Detection Foundational Model generated by Deci's Neural Architecture Search Technology," https://github.com/Deci-AI/supe-gradients/blob/master/YOLONAS.md, 2023, [Online; accessed 19-September-2023].

[72] C. -Y. Wang, A. Bochkovskiy and H. -Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 7464-7475, 2023, doi: 10.1109/CVPR52729.2023.00721.

[73] deci.ai, "he Deep Learning Platform Production-Grade Performance, Faster," https://deci.ai/, 2023, [Online; accessed 19-September-2023].

[74] X. Chu, L. L. Bo, and Z. Meituan, "Make repvgg greater again: A quantization-aware approach," *ArXiv*, 2022, doi: 10.48550/arXiv.2212.01593.

[75] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *European Conference on Computer Vision*, vol. 9905, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[76] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015.

[77] V. Sudha and T. R. Ganeshbabu, "A convolutional neural network classifier VGG-19 architecture for lesion detection and grading in diabetic retinopathy based on deep learning," *Computers, Materials and Continua*, vol. 66, no. 1, pp. 827–842, 2021, doi: 10.32604/cmc.2020.012008.

[78] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778, 2016, doi: 10.1109/CVPR.2016.90.

[79] C. Szegedy et al., "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1-9, 2015, doi: 10.1109/CVPR.2015.7298594.

[80] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv*, 2017.

[81] Simon Haykin; Bart Kosko, "GradientBased Learning Applied to Document Recognition," in *Intelligent Signal Processing*, pp.306-351, 2001, doi: 10.1109/9780470544976.ch9.

[82] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Communications of the ACM*, vol. 60, no, 6, pp. 1–1432 2017, doi: 10.1145/3065386.

[83] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. -C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition,* pp. 4510-4520, 2018, doi: 10.1109/CVPR.2018.00474.

[84] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size," *ArXiv*, pp. 1–13, 2016.

[85] S. Guo, Y. Liu, Y. Ni, and W. Ni, "Lightweight SSD: Real-time lightweight single shot detector for mobile devices," in *VISIGRAPP 2021 - Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*, vol. 5, pp. 25–35, 2021, doi: 10.5220/0010188000250035.

[86] Y. Quan, D. Zhang, L. Zhang and J. Tang, "Centralized Feature Pyramid for Object Detection," in *IEEE Transactions on Image Processing*, vol. 32, pp. 4341-4354, 2023, doi: 10.1109/TIP.2023.3297408.

[87] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, vol. 8693, pp. 740–755, 2015, doi: 10.1007/978-3-319-10602-1_48.

[88] A. Adel, K. Anis, A. Mohanned, S. Abdulrahman, and B. Bilel, "Aerial images processing for car detection using convolutional neural networks: Comparison between faster r-cnn and yolov3," *electronics*, vol. 10, no. 7, 2021, doi: 10.3390/electronics10070820.

[89] L. Rahmawati, S. Rustad, A. Marjuni, M. A. Soeleman, P. Nurtantio Andono and Pujiono, "Foggy-Based Object Detection In Video Using Faster R-CNN, YOLOv3, and SSD," *2023 International Seminar on Application for Technology of Information and Communication (iSemantic),* pp. 412-416, 2023, doi: 10.1109/iSemantic59612.2023.10295340.

[90] D. Saha, D. A. Mandal, and R. Ghosh, "MU Net: Ovarian Follicle Segmentation Using Modified U-Net Architecture," *International Journal of Engineering and Advanced Technology*, vol. 11, no. 4, pp. 30–35, 2022, doi: 10.35940/ijeat.D3419.0411422.