Tabriz Journal of Electrical Engineering (TJEE), vol. 55, no. 1, 2025 Serial no. 111
DOI: 10.22034/tjee.2024.58239.4711

# Real-time Detection of Capsule Endoscopy Using YOLO: A Comparative Study with GPDNET

Shokoufeh Hatami, Sina Behnam, Reza Shamsaee*

Faculty of Computer Engineering and Information Technology, Sadjad University, Mashhad, Iran.
E-mail addresses
*r_shamsaee@sadjad.ac.ir

Received:02/09/2023, Revised:19/11/2023, Accepted:19/06/2024.

**Abstract**
Capsule endoscopy (CE) technology is rapidly advancing due to its easy usability, long battery life, and exceptional image quality. The more resolution of CE images, the more time is needed to spend on the detection of a desired content in them. To address the issue, a new approach is presented in this paper using the popular YOLO v5 neural network architecture to detect the location and label of lesions in two public CE contents. A GPD neural network based on AlexNet is used as a rival classifier. The primary goal of this research is to reduce diagnostic time while maintaining accuracy. The results show a 6% increase in detection accuracy over the well-tailored rival. This is a significant achievement that could have a positive impact on the diagnosis and treatment of various medical domains. Interestingly, YOLO shows 58% more time-efficiency with an average prediction time of 5.39 milliseconds per frame. The scalability of YOLO is also analysed over Kvasir. The results indicate, while a workload is increased in 324 magnitudes, its accuracy is boosted more than 1% and it is only slower a 6.95 times graceful degradation, proving YOLO's real-time applicability. Implementations and supplementary data are available on GitHub.

**Keywords**
Gastroenterology, Capsule endoscopy, YOLO, GPD.

## 1. Introduction

Gastroenterology is covering vast diseases in many organs, that just one of them is stomach disease, which eventually leads to stomach cancer, and it is the second most common cause of death in the world [1].
Capsule endoscopy (CE) is a leading technology in the field that has proven its functionality in polyp detection for some related organs [1] [2]. While the CE usage is increasing, reducing the overall time of diagnostic is more concerned [3].
Although different methods can be used, deep learning and neural networks have been developed to increase the accuracy and reduce the response time. Many deep architectures such as AlexNet [4], SqueezeNet [5], Resnet [6] etc. have been applied, but there is no consensus on a single algorithm type.
As an off-the-shelf software package, YOLO is a CNN that is the most widely used algorithm for object detection and classification. For the first time in 2015, Redmon et al [7] introduced the first version of it. The main benefit of YOLO is its impressive response time while the classification accuracy is intact or reduced within a tolerable range. That makes YOLO a favourable technique for real-time jobs in video and image processing. In addition, there are many versions around its main architecture [8].
In this paper, we present how YOLO v5 [7] is adapted to detect, location and the type (classify) of any lesions on public access CE contents. The main concern of this research was to reduce the overall time of diagnostic while the accuracy remains intact. However, in some test cases our results show significantly improve in its accuracy measures. The CE contents that serves us is located in [9].
The article is structured as follows: Section 2 illustrates material and method descriptions, Section 3 presents results of the new method, Section 4 covers our discussion about the whole process.

### *1.1. Previous works*

Many technologies are applying to shorten reading CE images such as QuickView [10], Omni Mode [11] and Express View (EV) [12] by omitting similar images. EV as a best choice compresses CE contents about 95% so its processed frame rate is boosting and 70 minutes of initial

image sequence are compressed to 13 minutes of content. But a focused human operator for a couple of minutes is still needed.

Detaction techniques of the type and location of any lesions have made significant progress, during the past twenty years. They fall into one of two categories regarding the time of conducted research: before 2014 or after 2014. The former, that is known as the “traditional object detection”, and the later which is bloomed by outperforming “deep learning” techniques. The timeline of the development of object detection methods is shown in Fig. 1.

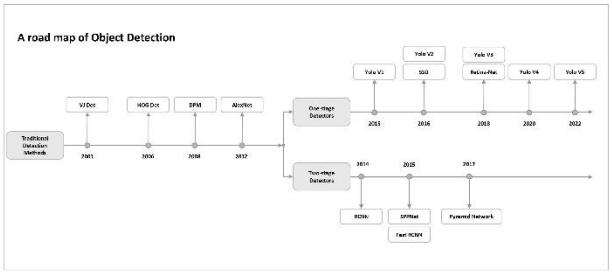


Fig. 1. The timeline of the development of object detection methods

In Fig. 1 the time line of object detection development can be seen. Significant detectors in this the time line are VJ Det. [13], HOG Det. [14], DPM [15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20], SSD [21], Pyramid Networks [22], Retina-Net [23]. As our research is the usage of YOLO over CE contents, and it is a deep learning-based algorithm, so we try to pin point its architecture and specifications regarding to the other rival algorithms.

### *1.2. Deep learning-based detection*

Deep learning approach, have made significant progress in the field of medicine, diagnosis and classification of diseases since the first usages of neural network algorithm that was introduced by Szegedy et al [24].

A typical neural network consists of some different layers, which will be reviewed below [25] for each definition of the layer some usage examples are referenced:

I-Convolutional Layer: In a neural network architecture, the most important part is the convolution layer, which consists of many filters (kernels) and they receive the input image. Using them, convolutions of image are created that each of them is referred to as a feature map [2] [26].

II-Pooling Layer: This layer has the role of reducing feature maps, which produces smaller feature maps, regarding a pooling operation over a region of a filter [26] [27].

III-Activation Function: All activation functions perform input to output mapping [26].

IV-Fully Connected Layer: In this layer, each neuron is connected to all the neurons of the previous layer, and this is why it is called fully connected, and usually the layer is placed at the end of the neural network [28].

**V**-Loss Function: It computes dissimilarity between ground truth of the input and the predicted label of a neural network. There are many loss functions. Using them, weights of a neural network are adjusted. So, two parameters are needed for the approximation of the error: the actual labels, and the predicted output of the neural network [29].

Two different CNN architectures for image classification that play a major role in this paper is illustrated in Fig. 3 and 4. It seems that deep learning techniques are shared many similarities together, at first glance, but they are categorized based on their architectures that made them suitable for some usages. In general, they are divided into two following types [30]:

- Two-stage detection
- One-stage detection

### *1.3. Two-stage detection*

For this type of neural networks, some object proposals are extracted first (for example by a deformable templates) and then each one is rescaled by selective search and fed into a CNN model. At the end, a complex classifier (such as: SVM) is used for detection of the object categories [30].

The architecture is a well-tailored approach for a specific problem. Due to this adaptation, boosting in many measures such as prediction accuracy and response time is expected. Some more examples of it can be found in [16], [5] and [2]. They are examples of the method. In a brief review [16] uses CNN with high capacity and selective search to detect objects and then the features are independently extracted for classification, while [5] presents a CNN architecture that has 50× fewer parameters than AlexNet. In addition [2] performs detection and classification using AlexNet [4] and SqueezeNet [5] architectures and using hyper-parameters such as L2 Regularization, step decay, and Gaussian Noise.

AlexNet was the first major CNN model that used GPUs for training and AlexNet showed that using ReLU nonlinearity, deep CNNs could be trained much faster than using the saturating activation functions like tanh or sigmoid. However, it also has significant disadvantages such as requires a lot of memory and AlexNet is prone to overfitting, which can be a problem when training on smaller datasets.

This neural network consists of 11 layers, which are composed of 5 convolutional layers, 3 fully connected layers, and 3 pooling layers [31].

YOLO is a real-time object detection algorithm that is extremely fast and can process images at 45 frames per second but its struggles with small objects within the image.

### *1.4. One-stage detection*

The method is unification of the following two stages together. They are namely the bounding box regression, and the object classification. Regardless of any region proposals, regression of bounding boxes is performed.

As examples of this category, SSD (2016) [21], RetinaNet(2017) [23], and YoloV4(2020) [32] can be pointed out. The SSD Method is a fast-one-time object detector for multiple classes that uses multi-scale convolutional bounding box outputs attached to multiple feature maps at the top of the network. The RetinaNet Method is a one-stage object detection model that utilizes a focal loss function that reshaping the standard cross

entropy loss such that it down-weights the loss assigned to well-classified examples. You Only Look Once (YOLO) was the first one-stage detector in deep learning era that is presented first by [20] and revised many times [7]. YOLO algorithm is a very fast one compared to the other methods. This network divides the image into regions and predicts bounding boxes and probabilities for each region simultaneously. The architecture of the network is shown in Fig. 3.

### *1.5. YOLO architecture in general*

YOLO (you only look once) is an algorithm that is presented first in [20] for real-time object detection. YOLO implementation has been evolved during the time for various purposes and platforms. They are referred under the name of YOLO versions [7] [8].

The algorithm uses only one forward propagation. YOLO useful in various fields, including diagnosing diseases, cars, animals, etc. The following are the most important reasons for the popularity of this algorithm:

**Speed**: The reason for the high speed of the algorithm is that it omits complex pipeline and detects frame as a regression problem [20].

**Accuracy**: As the algorithm considers the entire image at once, instead of a specific region it has minimal background errors [20].

YOLO algorithm uses the following two techniques to detect objects:

**1-Bounding box regression**: The bounding box is a rectangular shape that defines the location of the object in any input image or frame. Every bounding box in the image consists of the following attributes: Width (i.e. bw), Height (i.e. bh), Class (for example: lesion types, person, car, traffic light, etc.), Bounding box centre (i.e. bx, by)

**2-Intersection Over Union (IOU)**: IOU is a term used to describe the extent to which two boxes overlap. The greater the overlap area, the greater the IOU. In the object recognition process, the overlap between the predicted box and the correct box of the object is calculated. The concept is illustrated in Fig. 2.

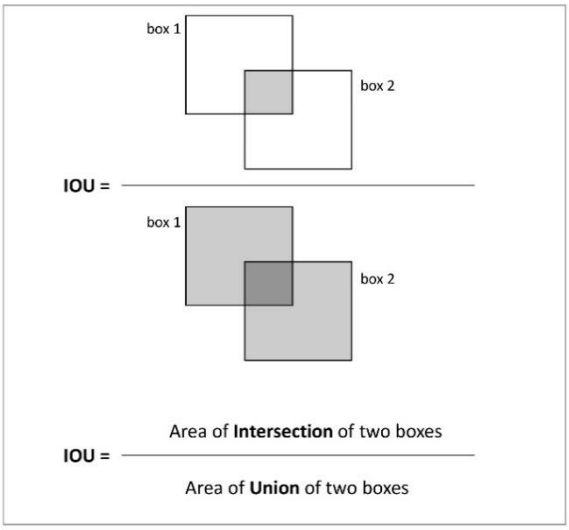


Fig. 2. Intersection Over Union (IOU) in YOLO

Yolo architecture is shown in Fig. 3, It consists of three parts: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head. The data are first input to CSPDarknet for feature extraction, and then fed into PANet for feature fusion. Finally, Yolo Layer outputs detection results (class, score, location, size).

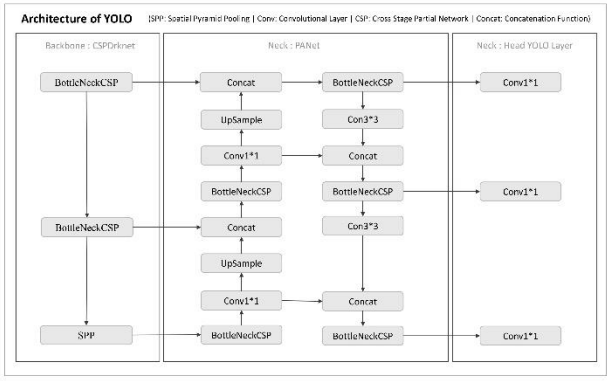


Fig. 3. It shows the architecture of YOLO and its three different parts.

Fig. 4 shows the architecture of GPDNet [2]. Its basic network is Alex's CIFAR-10 [33]. In order to reduce its parameters and improve its accuracy, some fire modules from SqueezeNet [7] has been applied instead of multiple convolution layers.

As it is fully explained in [2], the fire module is a single convolution layer with a single 1x1 filter fed into an extended layer that has a combination of 1x1 and 3x3 convolutional filters. Using implementation of this idea, [2] achieves a 50X reduction in model size.

To avoid the vanishing gradient problem, GPD has been used RELU as its activation function. In addition, for reducing overfitting L2 regularization has been applied to penalizing big weights that is also known as weight decay. Learning Rate (LR) optimization methods reduces the learning rate per epoch to increase the model's accuracy. Finally, for GPD data augmentation, a zero-centred Gaussian noise before dense layer is applied.

As mentioned above, following methods have been used for tuning GPD's hyper-parameters:

- L2 regularization
- Learning Rate (LR) optimization
- Gaussian noise
- RELU function [34]

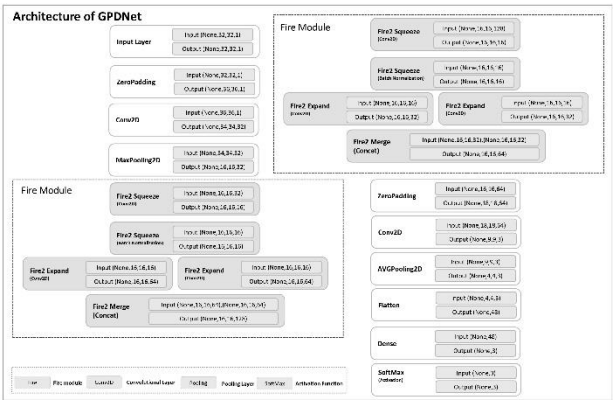


Fig. 4. The architecture of GPDNet [2] is illustrated in here.

## 2. Method

Using YOLO and GPD algorithms, numerical models of the training data are learned. the learned model of YOLO

is optimized by TensorRT. Then, all samples of a test dataset are presented to each algorithm separately. Asking the label of any unseen sample from any algorithm, a prediction process is performed.

In the first subsection, we present the definition of training parameters and our recommended settings. The next subsection is dedicated to the specifications of datasets. To compare the outputs, two perspectives are considered: model quality, and time. The corresponding subsections define them and present the results for YOLO and GPD. Also, scalability of YOLO has a separated subsection.

In order to have a fare comparison between the two methods, all the experiments and training tasks are conducted over Google Colaboratory framework with the same following conditions for the both rivals. OS: Win10; GPU: Tesla k80; CPU: Intel(R) Core (TM) i7-7700 K CPU @ 4.20 GHz. The experimental platform was developed based on Python programming language. The implementations of these two algorithms, and the datasets, pre-processed images, and some related supplementary data are accessible via the following public access of GitHub link [9].

### *2.1. Training parameters*

We applied YOLO v5 algorithm in order to classify any gastric lesions in this study. Such as any other algorithms in the field of machine learning, some hyper-parameters should be set up first for the algorithm. To understand some important of these parameters the following formulas are helpful.

Regarding ($\boldsymbol{f}$), as a neural network that contains ($N$) parameters $\boldsymbol{\theta} = (\theta_1, \dots, \theta_N)$ and maps its input ($\boldsymbol{x}$) to output ($\hat{y}$) formula (1) is formed:

$$\hat{y} = f(x;\theta) \tag{1}$$

Supposing a desired targeted output (y), a loss function is defined such as formula (2):

$$L(y;\hat{y}) \tag{2}$$

The loss function measures dissimilarity between (y) and ($\hat{y}$). It can be selected among a set of different functions such as: mean absolute error (MAE), mean squared error (MSE), binary cross entropy, and etc. [35].

To avoid updating of the parameters ($\boldsymbol{\theta}$) repeatedly, a batch of (m) inputs is considered. Thus, another loss function is calculated regarding to the batch size (m) of inputs, via formula (3):

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} L(y_i;\hat{y}_i) \tag{3}$$

From here on, we mentioned it as the loss function. Finding the best network parameters, a minimization problem is considered over the loss function. So, its gradient function is calculated by (4):

$$\nabla J(\theta) = \left(\frac{\partial J(\theta)}{\partial \theta_1},\dots,\frac{\partial J(\theta)}{\partial \theta_N}\right) \tag{4}$$

Updating the network parameters, different algorithms can be selected. Stochastic Gradient Descent with momentum (SGDM) [36] is the algorithm that serves us here. Its updating process is described by the set of formula (5)

$$\begin{aligned} v_{t+1} &= \rho v_t - \nabla J(\theta_t) \\ \theta_{t+1} &= \theta_t - \alpha_k v_{t+1} \end{aligned} \tag{5}$$

In (5), (t) is the step of operations, the current network parameters is ($\boldsymbol{\theta}_t$), its next updated parameters is ($\boldsymbol{\theta}_{t+1}$). ($v_t$) is the current momentum value and ($\alpha_k$) is the current learning rate. So, for their first round, constant values are used. ($\rho$) is the friction constant. Suppose them as quantities from [0,1] interval. Updating the current learning rate ($\alpha_k$), different method can be applied; in this study LR liner is used [36].

The values of the most important parameters for the YOLO algorithm, are presented in Table I. Those parameters that are not pointed out in the table remain intact as their default values suggest for YOLO v5.

**Table I.** It shows our setting for YOLO hyper-parameters in brief.

| # | Parameter | Value/State |
|---|---|---|
| 1 | Initial learning rate | 0.01 |
| 2 | Final One Cycle LR learning rate(lr0*lrf) | 0.1 |
| 3 | SGD momentum | 0.937 |
| 4 | Optimizer weight decay | 0.0005 |
| 5 | Batch size | 1025 |
| 6 | Epochs | 100 |
| 7 | Drop out | Not Used |
| 8 | Optimizer | SGD |

### *2.2. Dataset description*

Two different public access image datasets of the gastric precancerous disease are applied to this experiment. The first one is from Sir Run Shaw Hospital that is originally published in [33] and accessible in [9]; hear after we call it Dataset1. The second one is Kvasir [37]. The type of each image in the both datasets are labelled by expert doctors. The total images in Dataset1 are increased by cropping, rotation and other image pre-processing methods to facilitate machine learning. Table II shows the above information in brief.

The original size of the images in Dataset1 is 560x475 pixels. Considering previous researches [2] around GPD (the rival of YOLO), it performs classification with a good accuracy for Dataset1, but for localized and downsized images. Moreover, issues such as run-time limitation of Google Colaboratory, and GPD non-small training time, makes us to reduce images into 32x32 in a set of experiments.

To analyses YOLO scalability, Kvasir dataset is applied. As it consists of the images with different resolution from 720x576 up to 1920x1072.

The images in the Kvasir dataset have been pre-processed. However, to enable comparison with dataset 1, all of its images are resized to 576x576 and three incommon classes are just selected: Erosion, Polyps, and Ulcer from all the classes in the Kvasir dataset. The number of samples within each class is 1000. Thus, the total number of samples is 3000, and the sample increasing pre-process is not performed.

Figure 5 presents a selection of sample images from the Kvasir database, classified into three distinct categories: Erosion, Polyp, and Ulcer.


Fig. 5. Shows some samples of Kvasir dataset: (a) Polyp class, (b) Ulcer class, and (c) Erosion class

**Table II.** There are three labels for any given image on Dataset 1 that the first column shows. The second column presents the initial number of images within any labels. The third one illustrates a pre-processed number of images.

| Labels | No. of Original images | No. of Pre-Processed images |
|---|---|---|
| Erosion | 388 | 1209 |
| Polyps | 467 | 1216 |
| Ulcer | 467 | 1242 |
| **Total** | **1331** | **3667** |

*2.3. Train and test datasets*

To divide the datasets into tests and trains randomly, we use k-fold (k=5) cross-validation in Scikit-learn [38]. The basic idea behind k-fold cross-validation is to divide the data into k subsets of equal size. Then, one of the k subsets is used as the test set and the remaining k-1 subsets are used as the training set.

To do so, the training dataset percentage is set to 70 percent, while the testing and validation together is 30 percent. All images are accessible via [9]. The datasets comprise three classes, which the number of samples in tests, trains, and validations are described in the following tables. Table III describes the situation for Dataset1.

**Table III.** The specifications of train and test datasets are described below for Dataset1.

| Labels | Train | Test | Validation |
|---|---|---|---|
| Erosion | 889 | 176 | 144 |
| Polyps | 868 | 189 | 159 |
| Ulcer | 891 | 186 | 165 |
| **Total** | **2648** | **551** | **468** |

Table IV explains the distribution of samples for Kvasir within the different class labels regarding its training, testing, and validation. There are more than three classes in Kvasir, the same class labels of Database1 are selected. The total number of samples for each class label is 1000.

**Table IV.** The train and test datasets specifications are illustrated in below for Kvasir dataset.

| Labels | Train | Test | Validation |
|---|---|---|---|
| Erosion | 700 | 150 | 150 |
| Polyps | 700 | 150 | 150 |
| Ulcer | 700 | 150 | 150 |
| **Total** | **2100** | **450** | **450** |

## 3. Results

*3.1. Model quality*

Using many different quantitative measures, the quality of learned models, by the two rival algorithms can be evaluated. The most popular one is the predicted accuracy that is calculated by the following equation:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

In formula 6, TP, TN, FP, and FN are true positive, true negative, false positive and false negative of a classified label respectively. To define them, a confusion matrix is needed that illustrates the results of any algorithm over a test dataset. Table V, presents the confusion matrix for the GPD model, while Table VI is belonging to YOLO results. The both tables describes results for Dataset1.

**Table V.** The confusion matrix of the GPD model is presented on Dataset1 [2]. The first column from the left shows their true labels, while other columns show classified results regarding their labels.

| True Labels | Erosion | Polyps | Ulcer |
|---|---|---|---|
| Erosion | 167 | 8 | 1 |
| Polyps | 3 | 159 | 27 |
| Ulcer | 3 | 41 | 142 |

The datasets comprise three number of classes that are namely: Erosion, Polyp, and Ulcer. Therefore, a classification task over the data, leads into three separate accuracy measures for each class and in average.

The GPD method, is just a classifier. Therefore, its results are three different accuracy measures as there are three class labels. But, YOLO is a detector algorithm [7], so it tentatively looks for any possible object location first, then that image portion is assigned to a class label. In this way, the detection process makes the result to be resided in four classes domain. In other words, a non-detected label can be outputted in addition to the previous classes. Table VII, illustrates the accuracy of each class separately, that can be calculated out of Table V and VI.

Generally, a classification task in a higher dimension, results in a fainter quality of the model, as its FP and FN cover larger class labels. Due to this issue, we were not expecting to have a good model quality out of YOLO at all. Looking to Table VII, the best result among the three primary classes (Erosion, Polyp, and Ulcer) of the dataset belongs to the GPD method over Erosion label. Meanwhile, the best average accuracy over three classes is belong to YOLO.

**Table VI.** The YOLO confusion matrix on the same Dataset1 [2]. The raw results of the classification task are described in the table. The first column from the left illustrates their true labels. The last column shows those images that are not classified; the other columns present the classified results regarding their labels.

| True Labels | Erosion | Polyps | Ulcer | None |
|---|---|---|---|---|
| Erosion | 168 | 3 | 4 | 0 |
| Polyps | 6 | 170 | 4 | 3 |
| Ulcer | 5 | 5 | 175 | 1 |
| None | 0 | 0 | 0 | 0 |

**Table VII.** It shows the achieved accuracy within each class label from the two rival algorithms for Dataset1. The table covers four classes, as the dataset contains three primary classes (Erosion, Polyp, and Ulcer) plus a non- detected class (Non) to its results. The average of accuracies in the three primary classes are calculate as the last row of the table.

| Labels | GPD | YOLO |
|---|---|---|
| Erosion | **0.9727** | 0.9669 |
| Polyps | 0.8566 | 0.9613 |
| Ulcer | 0.8693 | 0.9650 |
| None | - | 0.9926 |
| **Avg. (Three Classes)** | 0.8995 | **0.9644** |

Considering average accuracies for three classes, the GPD model reaches 89.95% and YOLO achieves 96.44%. It means more than 6% improvement. Fortunately, despite of our initial expectations, the model quality of YOLO in sense of average accuracy is better than what GPD achieved.

*3.2. Time*

There are two separate time factors for a neural network algorithm: training time, and prediction time. The former issue refers to the consumed time that takes an algorithm achieves its model. The latter one describes the amount of time that is used by any algorithm to judge about an input image.

The training time is not our main concern, as it impacts the algorithms just once in their training phase. In contrast, a slight improvement in the prediction time, leads into a more efficient and responsive system. Directly talking, the prediction time for an algorithm is important factor and the shorter one is the better.

To be real time, for any incoming frame, the prediction time should be in the scale of milliseconds. Additionally, the time factor is not a fix value for any incoming images. To be meaningful, the total prediction time or the average of it, should be calculated over a set of frames

The total prediction time of the GPD algorithm is 7.09 seconds, while it is 2.97 seconds, for YOLO. Comparing together, YOLO shows 58 % increase in time efficiency. Which it means, GPD takes twice as much time as YOLO to predict an incoming image.

Considering, the total 551 images of the test set, and the total prediction time of any algorithm, the average prediction time can be calculated. Thus, GPD average prediction time is 12.86 milliseconds, while this factor for YOLO is 5.39 milliseconds. Although the average prediction time is impressive, in our experience for the content of Dataset1, YOLO achieved 45 frames per second (FPS).

To explore the research further, the YOLO method is also compared with the SqueezeNet which is based on the idea of the GPD method. As a real-time object detector with high accuracy is required, SqueezeNet is not mentioned compared to YOLO.

*3.3. YOLO scalability*

To study scalability of YOLO over Gastroenterology data, it should be understood how its performance varies with the image size. Investigating the effects of an escalation in the size of input images, Kvasir [37] public access dataset is used. The analysis is performed based on the same perspectives that are model quality, and time.

The model quality and time are explored for the images of Kvasir that are normalized to 576x576. Reminding Dataset1, that its contents are pre-processed to 32x32, the amount of pixels for each image of Kvasir presents 324 times larger workload than the other dataset. Table VIII reports the confusion matrix of YOLO for Kvasir. Compare to the results of YOLO for 32x32 in Dataset1 (Table VI), it appears that scaling up the inputs makes YOLO to be able to detect the labels better. Focusing on not-detected labels (None), there is no sample in Kvasir to make YOLO stray in classification.

**Table VIII.** The confusion matrix of the YOLO model is presented. The results of the classification task on gastric precancerous disease of Kvasir are described in the table.

| True Labels | Erosion | Polyps | Ulcer | None |
|---|---|---|---|---|
| Erosion | 150 | 0 | 0 | 0 |
| Polyps | 1 | 144 | 5 | 0 |
| Ulcer | 1 | 7 | 142 | 0 |
| None | 0 | 0 | 0 | 0 |

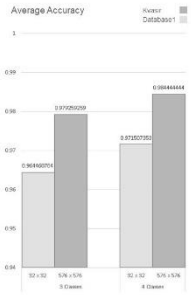


Fig. 6. The YOLO average accuracy (i.e. model qualities), for the different input sizes. The light gray bars are related to images with a resolution of 32x32 from Database1, while the darker bars belong to Kvasir and their contents are resized to 576x576. The left cluster of bars belongs to the results of three labels of the classification, and the right ones show the results of four labels.

Fig. 6. Presents model qualities. Light-grey bars illustrate the results for Dataset1 while its input images are 32x32. Dark-grey ones, show the results for Kvasir and the input images are 576x576. Fig. 6. shows average accuracies of YOLO based on the different input contents. As it mentioned before in the text, the average accuracy can be calculated for 3-classes (i.e. Erosion, Polyp, and Ulcer) or 4-classes version (i.e. Erosion, Polyp, Ulcer and None). The results for 3-classes and 4-classes are arranged in two different clusters of bars. The more resolution of inputs, the better model qualities are achieved.

Comparing the average prediction time of YOLO in milliseconds shows that the result of 576x576 is 6.95 times worse than that of 32x32. Linearly speaking, it is a graceful degradation, while YOLO experiences 324 more significant workloads.

## 4. Conclusion

This article details the modifications made to a YOLO v5 algorithm to accurately detect lesions in publicly available CE content with shorter diagnostic times. The study resulted in a 6% improvement in detection accuracy compared to its rival, and a 58% increase in time efficiency, with YOLO processing at 5.39 milliseconds per frame for Dataset1. The scalability of YOLO was also examined on the Kvasir dataset, with its larger workload and normalized image sizes. Despite the increased demand in 324 magnitudes, the YOLO algorithm only experienced a 6.95 times graceful degradation, achieving an average prediction time of 37.5 milliseconds and real-time domain processing. Higher input resolutions resulted in better accuracy, with supporting data and implementation available on GitHub [9].

The research proposes the use of the YOLO method, which is a real-time object detector, instead of classification methods. The YOLO method can identify polyps that do not belong to any specific category and this allows specialist physicians to examine them with greater accuracy and provide better care to their patients.

The method can detect polyps instantly and with higher accuracy, which can speed up the disease diagnosis process and have a significant impact on the treatment of diseases and provide better care to patients.

## 5. References


[1] C. Hamashima, “Update version of the Japanese Guidelines for Gastric Cancer Screening,” *Jpn J Clin Oncol.,* vol. 48, no. 7, pp. 673-683, 2018.

[2] S. Hatami, R. Shamsaee and M. Olyaei, “Detection and classification of gastric precancerous diseases using deep learning,” in *IEEE 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS) (pp. 1-5)*, 2020.

[3] S. Piccirelli, A. Mussetto, A. Bellumat, R. Cannizzaro, M. Pennazio, A. Pezzoli, A. Bizzotto, N. Fusetti, F. Valiante, C. Hassan, S. Pecere, A. Koulaouzidis and C. Spada, “New Generation Express View: An Artificial Intelligence Software Effectively Reduces Capsule Endoscopy Reading Times,” *Diagnostics,* vol. 12, no. 8, p. 1783, 2022.

[4] M. Alom, T. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. Nasrin, B. Van Esesn, A. Awwal and V. Asari, “The history began from alexnet: A comprehensive survey on deep learning approaches,” *arXiv preprint,* p. arXiv:1803.01164, 2018.

[5] F. Iandola, S. Han, M. Moskewicz, K. Ashraf, W. Dally and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size,” *arXiv preprint,* p. arXiv:1602.07360, 2016.

[6] S. Targ, D. Almeida and K. Lyman, “Resnet in resnet: Generalizing residual architectures,” *arXiv preprint,* p. arXiv:1603.08029, 2016.

[7] P. Jiang, D. Ergu, F. Liu, Y. Cai and B. Ma, “A Review of Yolo algorithm developments,” *Procedia Computer Science,* vol. 199, pp. 1066-1073, 2022.

[8] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in IEEE conference on computer vision and pattern recognition (CVPR) (pp. 6517-6525), 2017.

[9] sina-behnam. [Online]. Available: https://github.com/sina-behnam/GPD_Classify.

[10] J. Saurin, M. Lapalus, F. Cholet, P. D'Halluin, B. Filoche, M. Gaudric, S. Sacher-Huvelin, C. Savalle, M. Frederic, P. Lamarre and E. Ben Soussan, “Can we shorten the small-bowel capsule reading time with the "Quick-view" image detection system?,” *Dig Liver Dis.,* vol. 44, no. 6, pp. 477-481, 2012.

[11] S. Beg, E. Wronska, I. Araujo, B. González Suárez, E. Ivanova, E. Fedorov, L. Aabakken, U. Seitz, J. Rey, J. Saurin, R. Tari, T. Card and K. Ragunath, “Use of rapid reading software to reduce capsule endoscopy reading times while maintaining accuracy,” *Gastrointest Endosc.,* vol. 91, no. 6, pp. 1322-1327, 2020.

[12] C. Gomes, R. Pinho, A. Ponte, A. Rodrigues, M. Sousa, J. Silva, E. Afecto and J. Carvalho, “Evaluation of the sensitivity of the Express View function in the Mirocam® capsule endoscopy software,” *Scand J Gastroenterol,* vol. 55, no. 3, pp. 371-375, 2020.

[13] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR*, 2001.

[14] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005.

[15] P. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, “Object Detection with Discriminatively Trained Part-Based Models,” *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 9, pp. 1627-1645, 2010.

[16] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich feature hierarchies for accurate object

detection and semantic segmentation," in *IEEE conference on computer vision and pattern recognition (pp. 580-587)*, 2014.

[17] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *arXiv preprint,* p. arXiv:1406.4729, 2014.

[18] R. Girshick, "Fast R-CNN," *arXiv preprint,* p. arXiv:1504.08083, 2015.

[19] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv preprint,* p. arXiv:1506.01497, 2016.

[20] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv preprint,* p. arXiv:1506.02640, 2016.

[21] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu and A. Berg, "Ssd: Single shot multibox detector," in *Springer European conference on computer vision (pp. 21-37)*, 2016.

[22] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," *arXiv preprint,* p. arXiv:1612.03144, 2016.

[23] T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal loss for dense object detection," in *IEEE international conference on computer vision (pp. 2980-*, 2017.

[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, "Going deeper with convolutions," in *IEEE conference on computer vision and pattern recognition (pp. 1-9)*, 2015.

[25] L. Alzubaidi, J. Zhang, A. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. Fadhel, M. Al-Amidie and L. Farhan, "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions," *Journal of big Data,* vol. 8, no. 1, pp. 1-74, 2021.

[26] X. Zhang, F. Chen, T. Yu, J. An, Z. Huang, J. Liu, W. Hu, L. Wang, H. Duan and J. Si, "Real-time gastric polyp detection using convolutional neural networks," *PloS one,* vol. 14, no. 3, p. p.e0214133, 2019.

[27] C. Bailer, T. Habtegebrial and D. Stricker, "Fast feature extraction with CNNs with pooling layers," *arXiv preprint,* p. arXiv:1805.03096, 2018.

[28] Q. Zhang and D. Liang, "Visualization of fully connected layer weights in deep learning CT reconstruction," *arXiv preprint,* p. arXiv:2002.06788, 2020.

[29] A. Akbari, M. Awais, M. Bashar and J. Kittler, "A Theoretical Insight Into the Effect of Loss Function for Deep Semantic-Preserving Learning," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 34, no. 1, pp. 119-133, 2023.

[30] Z. Zou, Z. Shi, Y. Guo and J. Ye, "Object detection in 20 years: A survey," *arXiv preprint,* p. arXiv:1905.05055, 2019.

[۳۱] بهبود تشخیص نفوذ به شبکه "جواد حمید زاده ,مونا زنده دل اینترنت اشیاء با استفاده از یادگیری عمیق و الگوریتم بهینه سازی مجله مهندسی برق دانشگاه تبریز، جلد ۵۳، شماره "،میگوی آشوبی ۲، صفحات ۱۲۷-۱۳۸، ۱۴۰۲.

[32] A. Bochkovskiy, C. Wang and H. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint,* p. arXiv:2004.10934, 2020.

[33] X. Zhang, W. Hu, F. Chen, J. Liu, Y. Yang, L. Wang, H. Duan and J. Si, "Gastric precancerous diseases classification using CNN with a concise model," *PLoS One,* vol. 12, no. 9, p. e0185508, 2017.

[34] M. VASOUJOUYBARI, E. Ataie and M. Bastam, "An MLP-based Deep Learning Approach for Detecting DDoS Attacks," *Tabriz Journal of Electrical Engineering (TJEE),* vol. 52, pp. 195-204, 2022.

[35] J. Qi, J. Du, S. Siniscalchi, X. Ma and C. Lee, "On Mean Absolute Error for Deep Neural Network Based," *arXiv,* p. arXiv:2008.07281, 2020.

[36] J. Chen, C. Wolfe, Z. Li and A. Kyrillidis, "Demon: Improved Neural Network Training with Momentum Decay," *arXiv,* p. arXiv:1910.04952, 2021.

[37] Kvasir. [Online]. Available: https://datasets.simula.no/downloads/kvasir/kvasir-dataset-v2.zip.

[38] Scikit-learn. [Online]. Available: https://scikit-learn.org/stable/.