# FPS Killer - A Moving Target Detection Algorithm in FPS Game Based on Yolov3 Frame and Gradient Descent Algorithm

Huang Jiayu*
School of Computing
Jiangsu Vocational College of Commerce
Jiangsu, China
18552636796@163.com

Liu Aofan
School of Computing& Data Science
Xiamen University Malaysia
Sepang, Malaysia
SWE2009510@xmu.edu.my

2024 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) | 979-8-3503-7018-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ASSIC60049.2024.10507956

***Abstract*—With the continuous development of the economic level of the society, the game industry is getting more and more extensive development among young people. As one of the mainstreams in the game industry, FPS games are warmly welcomed by the majority of players. In order to ensure the fairness of the game, anti-cheat technologies have emerged, such as "BattlEye" and "ValveAnti-Cheat". This article applies the current popular computer vision technology to FPS games and proposes a cheating technology that cannot be detected by traditional anti-cheating methods. However, this project will only be used for academic research and discussion, and it is forbidden to use it for commercial sales and esports.**

***Keywords—FPS; YOLOv3; Object Detection; Alpha-Beta Pruning***

## I. INTRODUCTION (*HEADING 1*)

FPS game stands for First Person Shooting Game, and its goal is to simulate as realistic a design experience as possible and make people and make people have an immersive experience on the battle field. Usually, it appears in this way:

In an FPS, no matter what the game is, the main aim is to kill the enemy. This enemy may be another player, or may be computer-controlled bots. But no matter what it is, it takes on a human form.

At the same time, computer vision technology represented by CNN, DNN and RNN has been widely used in machine learning. Whether in face recognition or in the field of motion capture, they have made great process in recent years [1].

The most important thing in FPS game is face recognition and motion capture. If you're standing face to face with the enemy, who can react faster and aim better has an advantage. However, as we played the game, we found it difficult to aim at enemies, especially when they were hiding somewhere or popping out. No matter in what kind of FPS game, you can always find the characters in it. Here is an example of attributes character structure in FPS game.

TABLE I. ATTRIBUTES OF CHARACTERS IN FPS GAME

| | **Attributes** |
|---|---|
| **Position** | Position (X, Y) |
| **Velocity** | Velocity(X): Speed in X axis<br>Velocity(Y): Speed in Y axis<br>Velocity(Z): Speed in Z axis |
| **Acceleration** | Acceleration(X): a in X axis<br>Acceleration(Y): a in Y axis<br>Acceleration(Z): a in Z axis |

Hence, we came up the idea of automatic aiming. In all FPS games, we can locate the enemy's coordinates on the screen through face recognition, and then move the mouse to the corresponding position automatically by calling the Windows Dynamic Linking Library API.

We made an auxiliary tool based on Yolov3 frame, each time when we press a key, the weapon will automatically aim at the head of the enemy. All we need to do is click the mouse, and the gun or something else will hit the enemy automatically.

The motivation of this project is not to use it to cheat in the game, but to combine artificial intelligence with entertainment and improve the anti-cheating techniques in FPS game industry. Therefore, this project will only be used for academic research and discussion, and it is forbidden to use it for commercial sales and esports. Next details what and how CNN and yolov3 will be used in this auto-targeting tool.

## II. METHODOLOGY

This article primarily utilizes two research methodologies: literature research and empirical research. Literature research involves studying historical and contemporary papers to build a scientific understanding of facts through document analysis. On the other hand, empirical research relies on quantitative analysis and data-driven exploration to enhance the precision and scientific credibility of social issue research. [2].

First, we summarized the existing research on CNN and yolo to arrive at a framework suitable for this paper, and finally apply it to FPS games, especially CSGO.

## III. RELATED WORKS

### *A. Introduction to FPS*

According to different playing methods, games can be divided into many types, such as decryption games, shooting games, simulation games, and action games. Shooting games are one of the action games. According to different shooting angles, we can divide them into first-person and third-person shooting games. FPS refers to a shooting game conducted from the player's main perspective, that is, a shooting game in which the player uses his own perspective to experience the immersive experience of the protagonist.

### B. Introduction to CNN

Convolutional Neural Networks (CNNs) have achieved amazing strides in a number of pattern recognition domains during the last ten years [2].Their outstanding performance in large-scale image processing has led to a noticeable increase in their popularity in recent years.

CNNs are feedforward neural networks that perform exceptionally well for image processing tasks by responding within a predefined coverage region surrounding each cell. [3].

An input layer, hidden layers, and an output layer make up these networks. As the convolution kernel traverses the input matrix, the convolution process creates a feature map that is added to the input of following layers.

Artificial neural networks rely heavily on activation functions to understand intricate, nonlinear processes. Once the weighted total of the inputs is determined, they introduce nonlinearity into the network. "Leaky ReLU" is a variation of "ReLU," which is used in the YOLO method because of its effectiveness and variance.

### C. Introduction to Convolutional layers

A CNN's input is a tensor with the following shape: (number of inputs) x (height x breadth x depth) [4]. Convolution: Why does it happen? Assuming the network is given an RGB channel image with dimensions of 48 * 48 pixels in width, height, and depth, it connects the input layer to a single neuron. For this dataset, there ought to be 48*48*3 weight links. The entire procedure appears as Fig. 1.

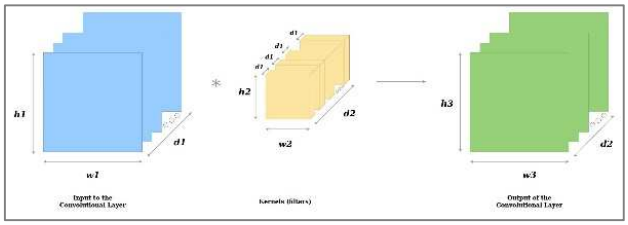


Fig. 1. Introduction to Convolutional Layers

### D. Introduction to Pooling Layer

By combining the outputs from neuron clusters in one layer into a single neuron in the next layer, they reduce the dimensionality of the data. There are two varieties of pooling layers: average pooling and max pooling. The more popular approach is called max pooling, and it entails determining the maximum value at each position. [5].

### E. Introduction to Fully connected layers

The fully connected layer serves as the network's "classifier" in a convolutional neural network. The fully connected layer is responsible for transferring the learned "distributed feature representation" to the labels of the samples, whereas processes such as convolution, pooling, and activation functions map the original input to a feature space in hidden layers. Convolutional operations may be used in the fully connected layer implementation process.

The output of the last pooling or convolutional layer serves as the input for these fully connected layers. In Figure 2, a fully linked layer processes the flattened matrix to identify images.

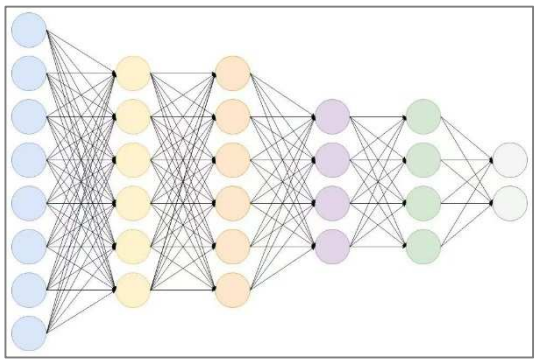

Fig. 2. Introduction to Fully Connected Network

## IV. IMAGE RECOGNITION WITH CNNS

The gradient in mathematics signifies the direction of the steepest change at a specific point.

Gradient descent, as a method, involves these steps::

1) Initially, set θ as a random value or an all-zero vector.

2) Next, adjust the θ value in a way that reduces J(θ) in the direction of decreasing gradient [6].

This process iterates with small subsets of instances from the training set until the average of the objective functions ceases to decrease. [4].

The Gradient reduction process is stated as follows [6]:

$$\frac{\partial}{\partial\theta}J(\theta)=\frac{\partial}{\partial\theta}\frac{1}{2}\sum_{i=1}^{m}(h_\theta(x)-y)^2=(h_\theta(x)-y)x^{(i)}$$

The backpropagation error is the sensitivity of the base of each neuron. It is defined as follows:

$$\frac{\partial E}{\partial b}=\frac{\partial E}{\partial u}\times\frac{\partial u}{\partial b}$$

CNNs receive input for a specific neuron and then scale it using the neuron's delta.

### A. Distinguishing features

Traditional multilayer perceptron models were used in picture recognition in the past, but they ignored the spatial layout of the data. For patterns with spatial proximity, these models inefficiently used neuron connections because they handled neighboring and distant pixels in the same way. Contrarily, Convolutional Neural Networks (CNNs) are a kind of multilayer perceptron variation that differ from MLPs in a few key ways.

3D neuronal volumes come first. Neurons in a CNN layer are composed of three dimensions: depth, breadth, and height. The receptive field previously indicated is coupled to the neuron inside a convolutional layer.

There's a local link, too. A pattern of local connections between neurons in adjacent layers is enforced by CNNs.

Third, there are shared weights. Each filter in a CNN is duplicated throughout the whole visual field. It can cut memory needs by reducing the amounts of free parameters learnt.

Pooling plays a crucial role in CNNs by ideally preserving discriminative information while filtering out irrelevant visual

elements [5]. This process enhances a CNN's ability to remain robust against variations in the positions of visual elements. These four key characteristics—convolution, activation function, weight sharing, and pooling—empower CNNs to achieve superior generalization when encountering various visual challenges.

## V. Evaluation

A popular statistical method in applied machine learning for evaluating model performance is K-fold cross-validation. Its goal is to calculate how well a machine learning model can predict outcomes from unknown data, or what is known as a test set. This method involves partitioning a dataset into k non-overlapping folds or subsets. Each fold takes a turn as a held-out test set, while the remaining folds serve as the training data. This process is iterated k times, with a model fitted and evaluated on each of the k hold-out test sets. The final performance is determined by averaging the evaluation scores across all iterations.

The general procedure is as follows:

a) Randomly shuffling the dataset.

b) Dividing the dataset into k groups or folds.

c) For each unique fold:

Utilize the remaining folds as the training set and treat the folds as the test set. After each iteration, the model is discarded and the evaluation scores are recorded. Fit the model on the training set, then assess its performance on the test set.

d) Summarizing the model's performance using the collected evaluation scores [7].

The pivotal setting in k-fold cross-validation is 'k', defining how many folds a dataset should be divided into. A poorly chosen 'k' might misrepresent the model's effectiveness. Common values include k=3, k=5, and k=10, with k=10 being frequently used in applied machine learning to evaluate models. This choice stems from research findings indicating that k=10 strikes a balance between computational efficiency and minimal bias in assessing model performance. Conducting a sensitivity analysis across various 'k' values can help determine its suitability for our dataset. Lower 'k' values tend to yield a noisy estimate of model performance, while higher values result in a less noisy assessment.

## VI. YOLO used In Our Program

### A. How YOLO used

FPS (First-person shooting) games are technically a subset of action games but have evolved into a distinct genre like RTS games due to their widespread global popularity. In FPS games, the primary objective often revolves around swiftly aiming at and shooting targets, particularly aiming for headshots. The headshot rate serves as direct data that reflects a player's skill level in FPS games..

If we want to use artificial intelligence to achieve fps games, we need to use the YOLO algorithm to quickly locate the target, and then further position the head to complete the headshot and achieve our goal of FPS killer.

### B. How to Implement

Target detection essentially entails receiving input images or videos, analyzing them, and then collecting location-specific data, including the target's upper-left and lower-right corner coordinates, anticipated category, and confidence level.

YOLOv3 detection is divided into two steps:

1. Locating the detected object within the image.

2. Categorizing the detected object, determining what it is.

In other words, once the algorithm identifies what the picture contains, it needs to pinpoint the position of that recognized object and create a frame around it..

Let's first understand how our algorithm works in the follow figure 3.

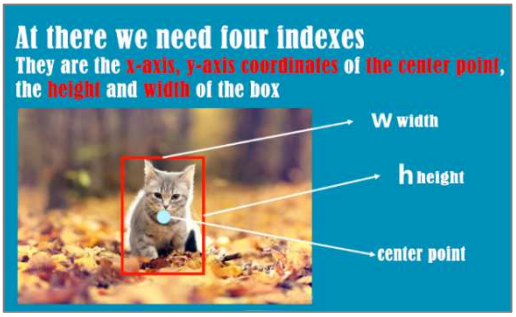


Fig. 3. YOLO Example Step 2 [8]

The red box in the figure 3 and 4 is the bounding box finally obtained by detecting in YOLOv3, and the yellow box in the figure 4.2-3 below is also the bounding box.

The process of YOLOv3 processing pictures is as follows.

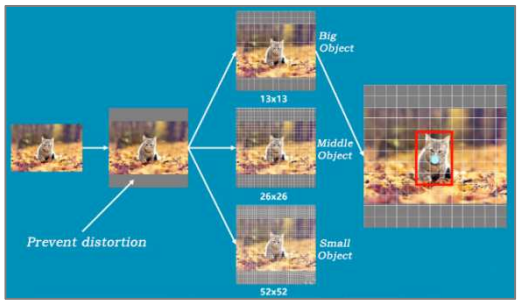


Fig. 4. YOLO Example Step 2 [8]

Initially, an image is fed into YOLO, which subsequently transforms it into a 384x384 grid. To avoid distortion, YOLO adds grey bars. Following this, the image is divided into three separate grid pictures.

$$y1 = 13 \times 13, y2 = 26 \times 26, y3 = 52 \times 52$$

And the detail process will show as follow picture 5 and 6.

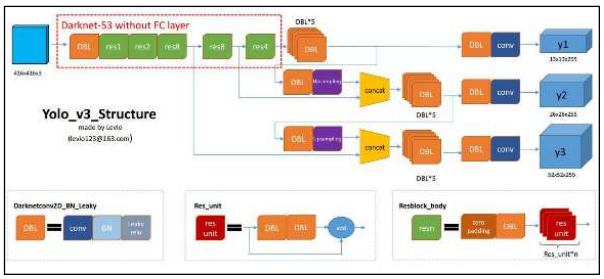


Fig. 5. YOLO Example Step3 [10]

*C. Detailed Process*

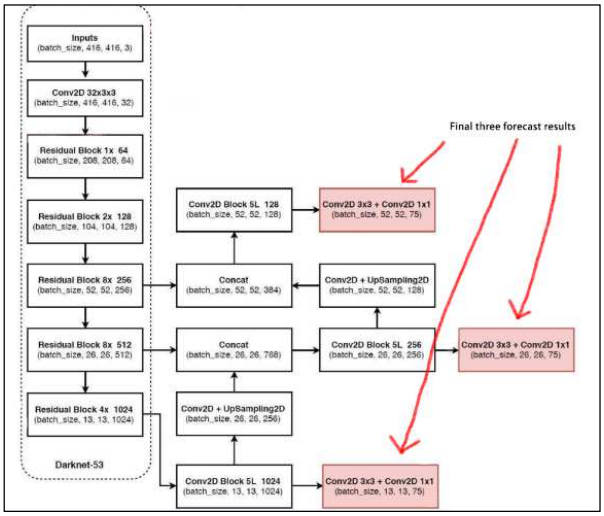


Fig. 6. Detailed process [11]

**Step 1: Obtain prediction results**

1. YOLOv3 identifies multiple feature layers to detect targets. Specifically, three feature layers are extracted from different segments of the backbone feature extraction network, Darknet53. These layers are situated in distinct positions: one in the middle, another in the middle to lower section, and the last one at the bottom. Their respective shapes are (52, 52, 256), (26, 26, 512), and (13, 13, 1024). These three layers are later combined and merged with other feature layers post-upsampling [12].

2. The third feature layer (13,13,1024) undergoes five rounds of convolution processing. Post-processing, a segment is utilized for convolution followed by upsampling, while another segment generates the predicted results (13,13,75). Conv2D operations of 3x3 and 1x1 serve to adjust channels, aligning them to the necessary size for the output [13].

3. Following convolution and upsampling, we acquire a feature layer sized (26, 26, 256). This layer is then combined with another feature layer (26, 26, 512) within the Darknet53 network, resulting in a new shape of (26, 26, 768). Subsequently, five convolution operations are executed, dividing the outcome: one section for convolutional upsampling and the other for generating prediction results sized (26, 26, 75). The Conv2D operations of 3x3 and 1x1 maintain consistent channel adjustments as previously described.

4. After that, splice the feature layer of convolution + upsampling in 3 with the feature layer of shape (52,52,256), and then convolve to obtain the feature layer of shape (52,52,128), and finally Conv2D 3×3 And Conv2D1×1 two convolutions to get (52,52,75) feature layer [14].

There are three red boxes in the last picture. The reason is that some objects are relatively large in the picture, so they are detected by 13×13. If the objects are relatively small in the picture, they will be classified as 52×52 for detection [15].

**Step 2: Decoding the prediction result**

The rationale behind decoding the prediction results lies in the discrepancy between the red box's position and the final predicted box in the image. This decoding is necessary. YOLOv3 operates on the principle of dividing the entire image into grids of 13x13, 26x26, and 52x52, where each grid point in the network is assigned to detect a specific area. The calculation process is depicted in figure (b), where 'b' stands for bounding box [16].

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h} \\
Pr(object) * IOU)(b, Object) &= \sigma(t_o)
\end{aligned}
$$

(cx, cy): These represent the number of grids by which the point's upper left corner varies from the grid's upper left corner.

(pw, ph): Indicate the predetermined box's side length.

(tx, ty): Indicate the target's center point's displacement with respect to the upper left corner of the grid, where the point is situated.

(tw, th): Predict the border's width and height.

σ: Refers to the activation function utilized, specifically LeakyReLU.

**Step 3:** Sorting the anticipated bounding box scores and performing non-maximum suppression screening are the two primary tasks in this phase. First, it removes boxes whose scores are higher than a predetermined cutoff, extracting every kind of box and sorting them according to their scores. Then, it applies non-maximum suppression using the frame position and score. Eventually, the bounding box with the highest probability will be identified. Subsequently, we'll utilize a screenshot from the Rainbow Six game to illustrate this selection process. [17].

First, we can see there are three border, and the score is 0.67, 0.77, 0.92 in incremental order in figure 7.


Fig. 7. Object Recognition step 1

and we will find the largest, obviously it's 0.92 which is with red sign in the figure 4.3-3 below figure 8.

Fig. 8. Object Recognition Step 2

In the last, we can see that the enemy is in the red border and that's the object in figure 9.

Fig. 9. Object Recognition Step 3

## VII. Testing

Now, we will continue a series of tests to prove the feasibility and accuracy of our project. Below we will test under different scenarios and maps.

And after we set many times test for each scenario and found the project are all works well and excellent. So we can say our project can work as our expectation. So for the test we draw two table for them:

TABLE II. Map Testing

| Map | Desert | City |
|---|---|---|
| **Test time** | 100 | 100 |
| **Detection rate** | 97% | 99% |
| **Error rate** | 1% | 0% |

As we can see the project works very well and the detection rate is very high (only sometime there some obstacles or move too fast) and error rate is still very low.

TABLE III. Accuracy and Time in each Map

| Orientation | head-up | Top-down | Bottom-up |
|---|---|---|---|
| **Test time** | 100 | 100 | 100 |
| **Detection rate** | 99% | 100% | 98% |
| **Error rate** | 0% | 0% | 0% |

At there we can see the project works very excellent. At these 300 times test there's so error happened, and the rate is very high.For top-down the enemy is easier to find so the rate can be reach 100% and for the other two cases perform also very well. These experiments can proved our project is fully meet expectations and highly successful.

## VIII. Conclusion

Our project FPS Killer takes YOLOv3 algorithm as the basic framework and applies convolution neural network (CNN) related technology to propose an object movement prediction technology. The project aims to explore the adaptive ability and practical application scenarios of mainstream deep learning models, try to give artificial intelligence projects more possibilities in daily life, and improve the anti-cheating technology in FPS game field [11].

In the whole project, CNN's related concepts and learning model evaluation methods are obtained through reliable existing literature, the calculation data used by YOLO algorithm has been reasonably screened and demonstrated, and the feasibility and stability of the project core code have been verified by many practices.

Simultaneously, we've gained a profound understanding of a wealth of related knowledge from this semester's AI course, which we've successfully put into practice.

As a test and expansion of the learning achievements of AI course, our project has provided background research and preliminary discussion on motion capture and prediction technology. As an example of computer science effectively learning from neuroscience, convolutional neural network has a wide range of applications. Although we generally associate CNN with image processing, in fact CNN can also process many Grid-like Data. Therefore, with the passage of time, the project may be extended to a broader and deeper research field, such as the processing of three-dimension data with time series [6].

As detailed in the report, throughout the process of gathering resources and conducting project research, we faced numerous uncertainties and challenges. However, our team members study hard and work together, continuously trained and improved the project model, and successfully obtained the final research results [18]. For each of us, the practical experience of this project is unforgettable and has important guiding significance.

## Acknowledgment *(Heading 5)*

Special thanks to the IEEE for organizing such a well-run conference and for fostering a culture of innovation and progress. The meticulous efforts of the technical program committee, session chairs, and volunteers are truly appreciated. Their dedication and expertise ensured a smooth and productive conference.

I want to express gratitude to the researchers, collaborators, and mentors who have been there for me during my academic journey. Their guidance and support have been incredibly valuable in preparing me for the dynamic landscape of this field. Additionally, I'd like to acknowledge my loved ones for their consistent love and support. Their encouragement and patience have played a crucial role in my success, both

personally and professionally. Their unwavering support motivates me to aim for excellence continually..

## REFERENCES


[1] A. Liu, M. S. Khatun, H. Liu, and M. H. Miraz, "Lightweight blockchain of things (BCoT) architecture for enhanced security: a literature review," in 2021 International Conference on Computing, Networking, Telecommunications & Engineering Sciences Applications (CoNTESA), IEEE, 2021, pp. 25–30. Accessed: Nov. 09, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9657112/

[2] G. I. Sayed, M. A. Ali, T. Gaber, A. E. Hassanien and V. Snasel, "A hybrid segmentation approach based on Neutrosophic sets and modified watershed: A case of abdominal CT Liver parenchyma," 2015 11th International Computer Engineering Conference (ICENCO), Cairo, 2015, pp. 144-149, doi: 10.1109/ICENCO.2015.7416339.

[3] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-Sign Detection and Classification in the Wild," presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2110–2118. Accessed: Oct. 07, 2023. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Zhu_Traffic-Sign_Detection_and_CVPR_2016_paper.html

[4] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, "Deep convolution neural network for image recognition," *Ecol. Inform.*, vol. 48, pp. 257–268, 2018.

[5] N. Akhtar and U. Ragavendran, "Interpretation of intelligence in CNN-pooling processes: a methodological survey," Neural Comput. Appl., vol. 32, no. 3, pp. 879–898, Feb. 2020, doi: 10.1007/s00521-019-04296-5.

[6] Applebaum, S., Gaber, T., & Ahmed, A. (2021). Signature-based and Machine-Learning-based Web Application Firewalls: A Short Survey. International Conference on Arabic Computational Linguistics.

[7] M. K. Hassan, A. I. El Desouky, S. M. Elghamrawy, and A. M. Sarhan, "A Hybrid Real-time remote monitoring framework with NB-WOA algorithm for patients with chronic diseases," *Future Gener. Comput. Syst.*, vol. 93, pp. 77–95, Apr. 2019, doi: 10.1016/j.future.2018.10.021.

[8] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-face: a real-time face detector," *Vis. Comput.*, vol. 37, no. 4, pp. 805–813, Apr. 2021, doi: 10.1007/s00371-020-01831-7.

[9] W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, "YOLO-face: a real-time face detector," *Vis. Comput.*, vol. 37, no. 4, pp. 805–813, Apr. 2021, doi: 10.1007/s00371-020-01831-7.

[10] "Remote Sensing | Free Full-Text | Improving YOLOv5 with Attention Mechanism for Detecting Boulders from Planetary Images." Accessed: Oct. 07, 2023. [Online]. Available: https://www.mdpi.com/2072-4292/13/18/3776

[11] Jain, S., Tripathy, H. K., Mallik, S., Qin, H., Shaalan, Y., & Shaalan, K. (2023). Autism Detection of MRI Brain Images Using Hybrid Deep CNN With DM-Resnet Classifier. IEEE Access

[12] M. Fazio, A. Celesti, F. G. Márquez, A. Glikson, and M. Villari, "Exploiting the FIWARE cloud platform to develop a remote patient monitoring system | IEEE Conference Publication | IEEE Xplore." Accessed: Nov. 01, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7405526

[13] H. Mara and B. Bogacz, "Breaking the code on broken tablets: The learning challenge for annotated cuneiform script in normalized 2d and 3d datasets," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*, IEEE, 2019, pp. 148–153. Accessed: Nov. 09, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8978050/

[14] A. Angelopoulos, S. Bates, J. Malik, and M. I. Jordan, "Uncertainty Sets for Image Classifiers using Conformal Prediction." arXiv, Sep. 03, 2022. Accessed: Nov. 09, 2023. [Online]. Available: http://arxiv.org/abs/2009.14193

[15] Mst. S. Khatun, A. Liu, and M. H. Miraz, "BCoT-Based Smart Manufacturing: An Enhanced Precise Measurement Management System," in Emerging Technologies in Computing, vol. 463, M. H. Miraz, G. Southall, M. Ali, and A. Ware, Eds., in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 463. , Cham: Springer Nature Switzerland, 2023, pp. 29–53. doi: 10.1007/978-3-031-25161-0_3.

[16] Tahoun, M., Almazroi, A.A., Alqarni, M.A., Gaber, T., Mahmoud, E.E., & Eltoukhy, M.M. (2020). A Grey Wolf-Based Method for Mammographic Mass Classification. Applied Sciences.

[17] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolutional neural network (CNN) for image detection and recognition," in *2018 first international conference on secure cyber computing and communication (ICSCCC)*, IEEE, 2018, pp. 278–282. Accessed: Nov. 09, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8703316/

[18] Y.-H. Kim, I.-K. Lim, J.-W. Lee, and J.-K. Lee, "Sensor Based Real-Time Remote Patient Monitoring System: A Study on Mobile DB Construction of Minimum Network Traffic in Use of HTML5 WebSQL," *Procedia Eng.*, vol. 29, pp. 2382–2387, Jan. 2012, doi: 10.1016/j.proeng.2012.01.319.