Proficient: Achieving Progressive Object Detection over a Lossless Network using Fragmented DCT Coefficients

Emad Felemban, Saleh Basalamah, Adil Shaikh and Atif Nasser
eafelemban@uqu.edu.sa, smbasalamah@uqu.edu.sa adilamjad@gmail.com, anahmed@uqu.edu.sa
Umm Al-Qura University, Makkah, Saudi Arbia

Abstract
In this work, we focused on reducing the amount of image data to be sent by extracting and progressively sending prominent image features to high-performance computing systems taking into consideration the right amount of image data required by object identification application. We demonstrate that with our technique called **Pro**gressive Object Detection over a Lossless Network using **F**ragmented DCT Coeff**icient**s (Proficient), object identification applications can detect objects with at least 70% combined confidence level by using less than half of the image data.

Keywords:
DCT, Image Recognition, Object Recognition, DCT Coefficients, Wireless Networks

1. Introduction

On a system level, there are two main approaches to deploying image recognition applications. The cutting-edge and advanced way is to use Edge Computing concepts to perform all image processing techniques on the edge side (Camera) and make it smarter and AI capable. The second classic way is to push all media data to be processed into a centralized location. Image processing is performed at a centralized location with high-performance processing machines. There are contradictory comparison results between the two approaches, and each has its pros and cons. Taking the communication overhead for example, it is obvious that the centralized approach has a huge communication overhead compared to the edge approach due to the massive amount of data that needs to be transported from cameras to the centralized processing location.

In this paper, we are introducing the "Progressive Object Detection using **F**ragmented DCT Coeff**icient**s (Proficient) "mechanism to help mitigate the communication overhead problem for centralized image processing systems. Using the proposed method, a 50% reduction of communication overhead cost is achieved without affecting the recognizability of the taken image at the centralized processing location. In other words, Proficient was able to achieve 50% cost of transmitting images to the central location while maintaining 70% of the image quality and the objects to be detected from the image.

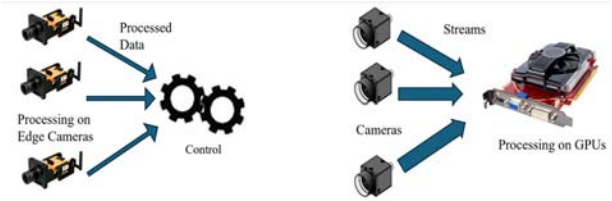

Figure 1: Two Approaches to Implement Object Detection Application. Left Camera Nodes are Equipped with AI models to Perform Image Recognition on the Edge, Right Camera Nodes Transmit Full Data to a Centralized Location for Image Processing

The proposed Proficient mechanism works as follows (Figure 2).

1. Encoding the image into a format that is suitable for progressive image processing applications.
2. Fragmenting and sending the encoded image in a way that critical image data is sent sooner than less critical image data.
3. Receiving and defragmenting the image data fragments at the high-performance image processing unit (HP-IPU).
4. Decoding the received image data, progressively.
5. Providing object detection quality feedback to the fragmentation module. If a certain quality is achieved, no additional data is required for the image.

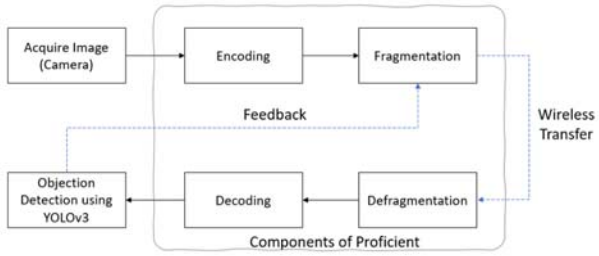

Figure 2: Proficient Basic Mechanism

More details will be given in Section II, Figure 3.

Manuscript received April 5, 2024
Manuscript revised April 20, 2024
https://doi.org/10.22937/IJCSNS.2024.24.4.6

2. Methodology

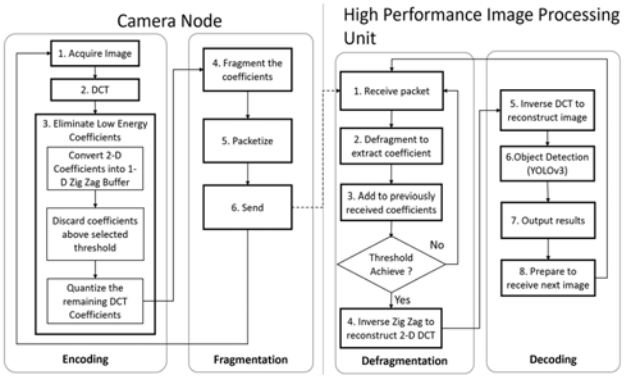

Figure 3: Detailed Diagram Explaining Proficient Working Mechanism

a. Flow at Camera Node

1. After an image is captured at the camera node, it will be encoded with discrete cosine transform (DCT) to achieve a high level of image compression. DCT expresses signals in terms of a sum of cosine functions oscillating at different frequencies, both in horizontal and vertical directions as a result a 2-D matrix of DCT coefficients is generated for each image captured.
2. Since this mechanism is a lossy mechanism, DCT coefficients below a certain threshold will be eliminated. The effect of elimination of these coefficients is very minimal to the image quality. However, dropping these coefficients reduces the amount of required transmitted data to the receiver. Network traffic is reduced as a result.
3. DCT produces coefficients in floating point format. Quantization is carried out to convert them from floating point to fixed point numbers that can easily fit in network packets. Quantization is also lossy. It further reduces the size of data to be transmitted.

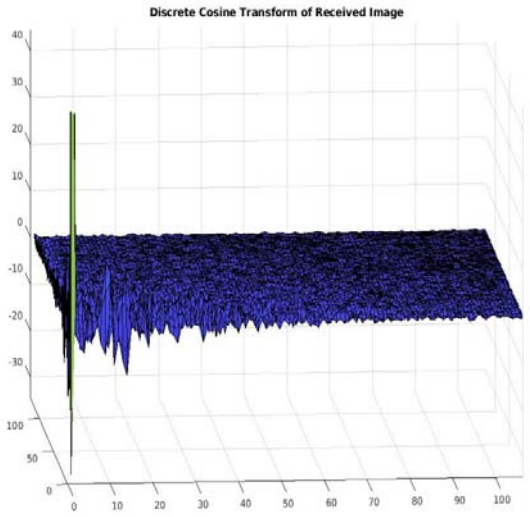

Figure 4: Received Image 2D DCT coefficient Matrix

After DCT and quantization, Figure 4 depicts a certain pattern in which DCT coefficients are arranged. The coefficient of frequencies near the origin has higher energy. Whereas, as the distance of coefficients of frequencies from the origin increases, their energy becomes less and lesser. Higher energy coefficients contribute to more variations in the image. Lower frequency coefficients contribute to lesser variations in the image. Hence, coefficients near the origin are more crucial for the image quality than the ones further away from the origin.

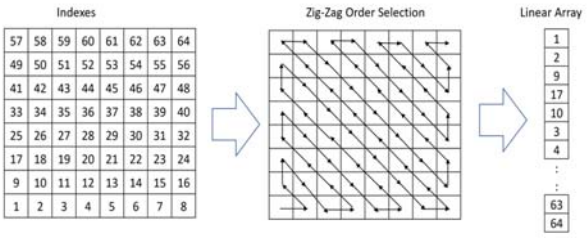

Figure 5: Zig-Zag selection of 2-D Coefficients for Linear Array

4. To transmit the 2-D coefficient matrix, the sender converts it into a linear array ensuring that higher energy coefficients are selected first. This can be done by selecting 2-D coefficients in a zig-zag manner as shown in Figure 5. As shown in the figure, the left box shows the indexes of the 2-D coefficients. The center box shows the zig-zag order in which coefficients would be selected. Hence, from the 2-D matrix on the left of the figure, we would get a linear array depicted on the right side of the figure. This step completes the encoding stage of the image, depicted in the leftmost box in Figure 3.

5. The next stage consists of breaking up the linear array of coefficients into fragments that fit inside a network packet. The last step of the previous stage, the zig-zag selection of coefficients, arranged the coefficients in a linear array starting with a coefficient of frequencies with the highest power followed by coefficients of lesser power and so on. Hence, the first fragment of the linear array would contain coefficients of frequencies with the highest powers. The second fragment will contain coefficients of frequencies with lesser power and so on.
6. At this stage, Proficient packetizes each fragment of the linear array and transmits them. As a result, the first packet to be transmitted will contain information that would contribute to the maximum quality of the received image. The second packet would contain the second-highest contribution to the quality of the received image and so on. This stage of Proficient keeps on sending further fragments until the preset number of packets has been sent. After this, the camera node restarts work on the next image starting at the first step.

b. Flow at Processing Unit

1. Packets containing the image's coefficient are received sequentially.
2. From each packet, the extracted coefficients are used to reconstruct the linear array. The sender inserts a fragment number in each packet so that the receiver can know where to place the coefficients in the placeholder linear array.
3. Hence, using this fragment number, the receiver defragments the received packet and places the coefficients into the linear array. After this, it is checked if the preset threshold number of packets has been received or not. If the threshold has been achieved, step 4 is executed. Otherwise, the next packet is awaited in step 1.
4. After the threshold number of packets has been received, the linear array is converted into a 2-D DCT matrix using a reversed zig-zag operation.
5. Inverse DCT is carried out to reconstruct the received image from the received DCT of the image.
6. The reconstructed image is passed to an object recognition algorithm for object detection. Objection detection consists of identifying several objects in the image as well as their respective object identification confidence levels. As the number of received packets is above a selected threshold, they contribute to recreating a minimal-quality image that is required for detection by an object recognition algorithm. The selection of this threshold will be discussed in the experimentation section.
7. The application is notified about the results and the process restarts from step 1.

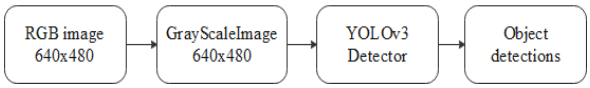

Figure 6: Object detection methodology

3. Experiments

To evaluate the effectiveness of Proficient, our setup included a combination of tools. We used the YOLOv3 architecture for image recognition. We selected 100 different images from the Microsoft COCO dataset [1]. We used a Linux Machine powered by an NVIDIA TitanXP GPU to detect objects using YOLOv3. These images contain different types of objects ranging from 1 to 14. Figure 6 shows the methodology used in detecting objects from input images. The input image is 640 x 480 with an RGB scale. The images are converted into grayscale before applying the image detection algorithm. We mark all the objects detected by YOLOv3 as the ground truth of that image.

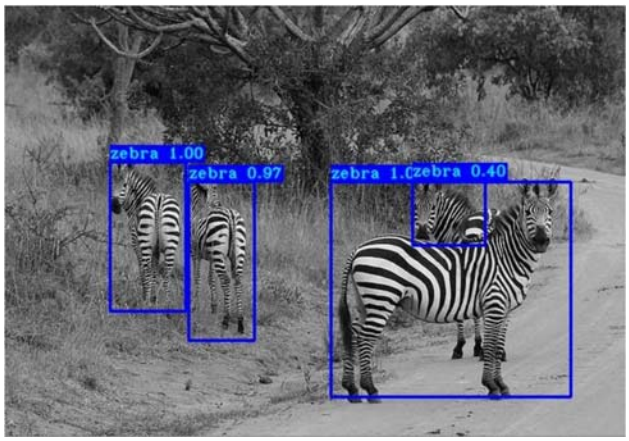

Figure 7: Sample Output of YOLOv3 on a random image

We define the combined confidence level by summing the confidence level of each detected object divided by the total number of objects. This is shown in Equation 1. For simplicity, this paper mentions combined confidence level as a percentage without units.

$$CCL = \frac{\sum_{k=1}^{N}(C_k)}{N} \quad (1)$$

Where,

CCL is combined confidence level,
N is the number of objects in the image,
C_k is the confidence level of object *k*.

For the image shown in Figure 7, YOLOv3 detected 4 zebras: two with a confidence level of 100 %, one with a confidence level of 97 %, and one with a confidence level of 40 %. Hence, using equation (1) for combined confidence level (*CCL)*, we can say that the combined confidence level of YOLOv3 is: $\frac{(1 + 1 + 0.97 + 0.4) \times 100}{4} = 84.25$ %.

We subjected each image to YOLOv3 to find ground truth. After finding the ground truth for all the grayscale images, we sent all images using Proficient.

The third step carried out by the sender depicted in Figure M3 of Proficient eliminates low energy coefficients. It is expected that this reduction in data to be transmitted is achieved by this step. This paper focuses on evaluating and improving this step. We want to eliminate as many low energy coefficients as we can without compromising the results of object detection. Therefore, through experimentation, we want to find a generalized threshold of image data required to be sent to HP-IPUs that can be used to detect objects with a certain predefined combined confidence level.

The resolution of each image was 1280x720 grayscale with one byte per pixel, However, each DCT coefficient calculated was larger. Some precision was lost when each DCT coefficient was encoded in two bytes due to truncation of result bits introducing quantization error. This loss is unavoidable, but acceptable in lossy techniques such as DCT. The truncation should be small enough so that it causes little loss in quality when inverse DCT is taken in the latter step of Proficient. The optimal number of bits required to represent each DCT coefficient is out-of-scope in this paper and will be addressed in future research. As mentioned before, we used two bytes to store each DCT coefficient.

Note that to find this threshold, packets containing coefficients were sent to the HP-IPU over a lossless network. With this, we explain how ground truth is established and the minimum number of packets (threshold) is selected.

c. Finding Ground Truth

In the first set of experiments, we first converted the original 100 test images to grayscale with a resolution of 1280x720 and directly passed them to YOLOv3 without passing them through Proficient. The goal of this stage was to establish the ground truth for each image.

Figure 8 in the next section shows the results of the ground truth experiments. The bins on the x-axis represent the number of objects in an image. As the 100 images were chosen randomly, the number of detectable objects in them was not known. The idea is to entirely rely on YOLOv3 for detection with and without Proficient.

The number at the top of the bar at the bin represents the number of images in the dataset that had that many objects. For example, there is no bar for the bin of 0 objects. Therefore, none of the images were empty. The bar of the bin with 1 object has 7 at the top of the bar. This means that there were 7 images in the dataset that contained 1 object. Similarly, the bar representing 2 objects has the number 14 on its top. This means that there were 14 images in the dataset that had 2 objects each. In the same way, we can see that the dataset included only 3 images that had 16 objects in them. As the images were selected at random from the database, the number of objects in each image was not controlled in this experiment.

d. Finding Coefficients Threshold

To find the number of coefficients required to detect all objects with a 70% combined confidence level, a modification was made to the Proficient receiver end as shown in Figure M3. Feedback is sent from the image detection module to the packetizer module so that we can log the number of packets required for the threshold.

At the processing unit, a linear array is created and filled with zeros. When a packet is received, the following steps are performed:

- A packet is received from the sender.
- Coefficients are extracted from the received packet so that they can be placed in the linear array at the position defined by fragment number.
- The received coefficients are added to the linear array.
- The inverse zigzag is performed on the linear array to reconstruct the 2-D DCT. Note that this step is carried out after each packet is received because we want to find the threshold of packets that gives us a 70 % combined confidence level at step 6 of the HP-IPU with the number of objects equal to ground truth at step 7 of the HP-IPU. This contrasts with the original Proficient implementation discussed in the previous section where we wait for the threshold number of packets to be received.
- Inverse DCT has been applied to the received coefficient so far.

- The reconstructed image is then fed to YOLOv3 to detect the number of objects in it. YOLOv3 detects objects along with some confidence level.
- If the combined confidence level of the detections in the reconstructed image crosses 70 %, the number of objects detected by YOLOv3 is compared to the ground truth established in the first set of experiments.

o If the detected number of objects is achieved with a 70 % combined confidence level, feedback is sent to the packetizer module of the camera node so that it logs the number of packets that were sent along with the PSNR of the reconstructed image (Step 7 of the sender). PSNR is calculated and logged to see its effect due to the number of packets received. Note that the number of packets that it had sent is proportional to the threshold we are looking for. This feedback also indicates that further packets would most likely not impact the results of object identification. When this feedback is received by the sender, it stops sending further fragments of the linear array to the receiver, because more fragments would not contribute enough to improve the image quality. Moreover, the feedback from the receiver signifies that it has successfully identified objects in the received image. This also signals the sender to start processing the next image.

o If the detected number of objects is not the same, the next packet from the packetizer module is awaited. Once received, step 1 onwards is executed. This process was carried out for all 100 images. For each image, two statistics were recorded. The number of packets required to achieve ground truth with a 70 % combined confidence level and the PSNR level at that threshold.

4. Results Explanation

Figure E1 shows the results of the experiments carried out. As discussed earlier, the x-axis bins represent the number of objects in the image, whereas the number on top of each bar represents the number of images of that object count used in the experiment. The height of the bar represents the number of packets required to detect all objects with a minimum of 70 % combined confidence level.

To packetize the entire 2-D DCT array, 480 packets of 1000 bytes each, were required. As shown in Figure 7, for the 7 images that contained one object, an average of 5.86 packets were required. The average PSNR for the 7 images was 24.8. For the 14 images that contained 2 objects, an average of 13.28 packets were required. The average PSNR was 22.6. These statistics are shown for each set of images in Figure 7.

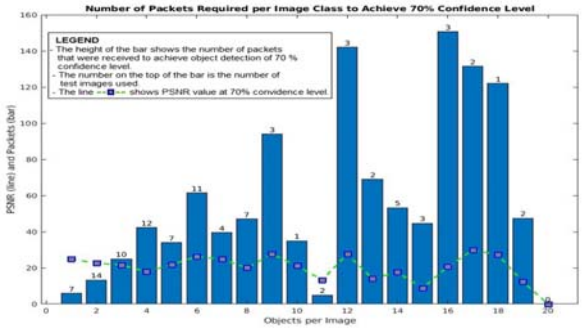

Figure 8: Stage 1 of Proficient Experiments

The figure also shows that as the number of objects in an image increase, a higher number of packets are required to achieve full object detection with a minimum of 70 % combined confidence level. The figure also shows that the PSNR of received images has no relation to the number of objects in the image. On average, a PSNR over 22 has shown good detection results with 70 % confidence.

As mentioned before, the total number of packets containing DCT coefficients per image was 480. Using experiments, we wanted to identify the threshold of packets required to detect objects efficiently. For one object per image, we saw that 5.86 packets were required. This means that only 6 packets out of 480 were enough to detect the object in the images with a 70 % combined confidence level. Similarly, 14 packets per image for 2-object images were enough, and so on. From Figure 8, it is seen that a maximum of 156 packets were required to detect all 17 objects in each of the 3 images used. Therefore, with 156/480 = 32.25% of the coefficients, we were able to detect all objects in the 100 random images.

With the expectation that less than 50 % of the coefficients would be enough to identify all objects in all images, we wanted to analyze the effect of each set of coefficients on the final results. Hence, taking the above variation into account, we want to fine-tune the threshold. For this, we carried out another set of experiments explained in the next sub-section.

5. Analysis of the number of Coefficients Required to Detect Objects with Combined Confidence of 70 %

One image was selected at random to see the results of its reconstruction and detection from packet 110 to packet 200. Note that the number of packets is analogous to the number of coefficients of the image's DCT. Figure 9 shows

the results of an experiment carried out on a randomly selected image.

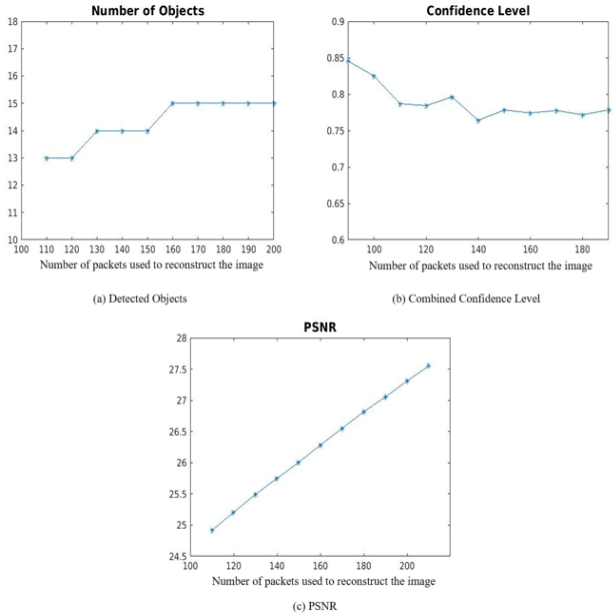

Figure 9: Experiment Results: Understanding the relation of packets required for image reconstruction with Combined Confidence Level and PSNR

The first Figure 9 (a) shows the number of objects detected by YOLOv3 from images reconstructed from 110 packets, 120 packets, 130 packets, and up to 200 packets. Figure 9 (b) shows the combined confidence level output of YOLOv3 for the objects detected per image. Finally, Figure 9 (c) shows the PSNR of reconstructed images.

Number of Packets (Total 480)	100	110	120	130	140	150	160	170	180	190	200
Corresponding Threshold	21%	23%	25%	27%	29%	31%	33%	35%	38%	40%	42%

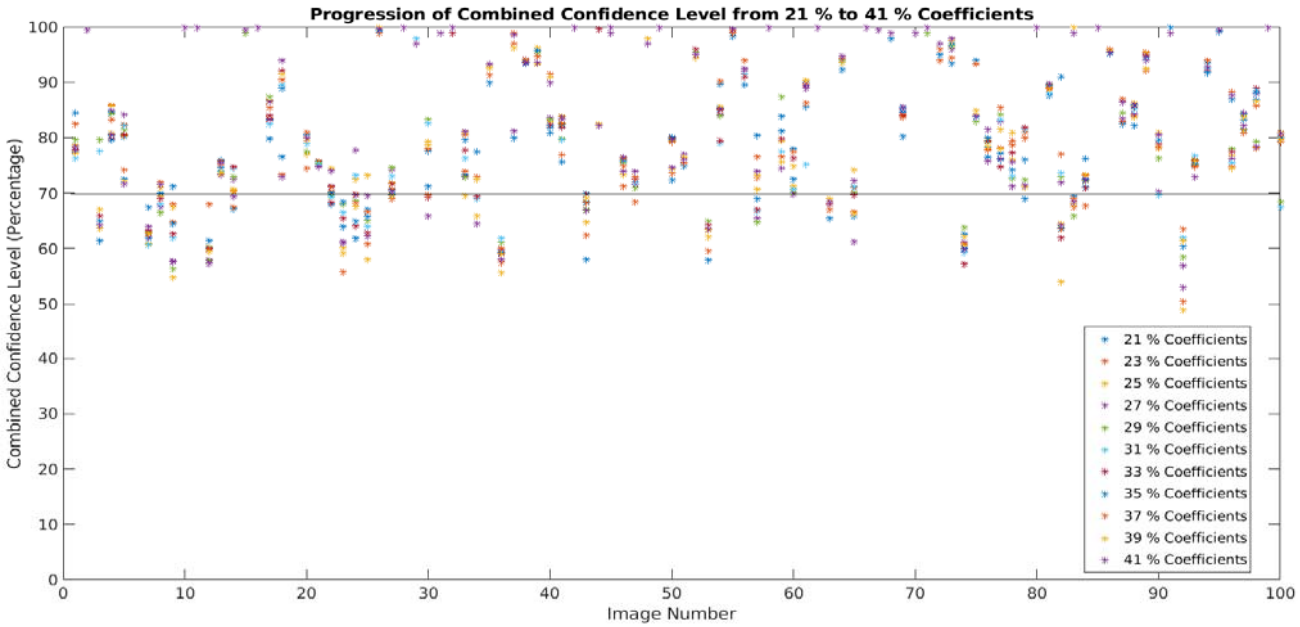

Figure 10: Progression of Combined Confidence Level from 21 % to 41 % Coefficients

It is seen that when the image reconstructed with 110 packets is passed to YOLOv3, 13 objects were detected with a combined confidence level of 85 %. Note that the ground truth for this image was 15 objects. The PSNR of the same image was 24.8.

With the image reconstructed with 120 packets, still 13 objects were detected but the combined confidence level decreased. Analyzing the individual objects in the selected image, it was seen that with 120 packets, some objects previously detected with a certain confidence were not detected at all, whereas some new objects were detected with a lower confidence level. Hence, this caused a decrease in the combined confidence level.

With the image reconstructed with 130 packets, the object count increased to 14 with a slight change in the combined confidence level. In this case, one of the previously detected and then undetected objects was detected again. The increase in PSNR was almost constant in this case as well.

As the number of packets increased to 160, the number of objects detected became the same as ground truth. The combined confidence level of all objects detected became 77 %. The PSNR gradually increased to 26.3.

Above 160, the number of objects detected remains the same. The combined confidence level of all the objects in the image also remains around 77 % whereas the PSNR increases. Hence, for this image 160 out of 480 packets was enough to detect all objects with more than 70 % combined confidence level.

The next set of experiments was carried out to check the accuracy of Proficient with the following set of thresholds:

Figure 10 shows how the combined confidence level fluctuates as we increase the threshold of coefficients from 21 % to 41 %. As explained in the previous sub-section, the combined confidence level may decrease as the number of coefficients increases, until we have enough coefficients to identify all objects with respect to ground truth. The above figure shows that as the number of coefficients reached 41% almost all objects in all images had been identified with a combined confidence level above 70 %, with the exception of images 36, 54, 75, and 93. Therefore, we can confidently conclude that with a max of 41 % coefficients, Proficient can detect all objects with a combined confidence level of 70 % in 96 out of 100 randomly selected test images.

6. Related work

Many people in the literature proposed methods to compress the images before transmission and apply certain AI algorithms to analyze the compressed images. They compare the accuracy of object detection with and without compression techniques. A similar approach is presented by Lin et al. [2] for real-time application data compression. They proposed a hybrid approach for image compression in which they used the regional information of the image as well as the temporal information. This approach reduces the significant number of temporal redundancies.

Zhou et al. [3] proposed an image compression technique for unified object detection. Their approach reduces the extra computational overhead of a convolutional network. The method merges the image sub-sampling and object detection in one step as compared to the conventional approach of object detection in which both act as a separate module. Choi et al. [4] used quantization of data and encoded it using a PNG encoder. Using this approach, they compressed the data before uploading it to the cloud. They used VOC2007 and VOC2012 datasets for model training. The results show the impact of lossless compression (after the Q-layer) on accuracy.

Dodge et al. [5] studied the image classification performance with and without image compression. They compare the performance of deep learning architectures on image classification after applying five different types of image distortions (i) additive Gaussian noise, (ii) blur via convolution using a Gaussian kernel, (iii) contrast reduction via blending with a uniform gray image with a varying blending factor, (iv) JPEG compression at different quality levels (reflected by the Q parameter), and (v) JPEG2000 compression with a different target peak signal to noise ratio (PSNR). The experiments show that the compression impacts the performance of classification, but the model accuracy did not decrease a lot.

Goodfellow et al. [6] used deep CNN models to investigate object detection by influencing lossy image compression. The authors described methods for generating additive, seemingly random, noise with low amplitude, causing models to classify modified images wrongly. Gandor et al. [7] investigate the influence of JPEG compression on the performance of CNNs for object detection. They analyze the performance by varying the compression levels. The plethora dataset that contains 5000 images having 80 different objects is used in the experimentation. For object detection, they investigated nine different deep neural models. The results show that the precision remained constant regardless of the compression

quality. The effect of compression is on the detection of small objects.

Choi et al. [8] developed a bit allocation and rate control method to improve the object detection of state-of-the-art object detector YOLO9000 [9]. They used the bit allocations after the initial convolutional layers of the YOLO model to detect the objects more accurately. The results show the 7%-bit savings by using the proposed strategy and also it accurately detects and classifies the objects. Table 1. Shows the summary of related work with the key findings.

Table 1: Summary of Key Findings from Literature

Author	Proposed System	Compression Technique	Data set	Model Investigated	Comparison Metrics	Application
Lin et al. [2]	Segmentation algorithm (SPEC), Lossless coding method	Compound Image Compression	Computer screen images (webpages, wallpapers, and characters)	-	PSNR (JPEG, JPEG-2000, SPEC)	Real-time applications
Zhou et al. [3]	Compressive Convolutional Network (CCN) for efficient object detection and image compression	Compressive Convolutional Network	BSD100 [100 images] VOC (2007+2012) [21530 images] and COCO [200000 images with 80 classes]	YOLOv2	PSNR, Coherence Measurement	Embedded Systems
Choi et al. [4]	DEEP FEATURE COMPRESSION FOR COLLABORATIVE OBJECT DETECTION	quantization of data	VOC2007 and VOC2012 (16,551 Images with 20 Classes)	YOLO9000	Map	Smart Cities
Dodge et al. [5]	Evaluation of Image Quality Affects on Image Classification	Gaussian noise, Blur, Contrast Reduction, JPEG Compression, JPEG2000 Compression	ImageNet2012 (1000 Classes)	AlexNet GoogleNet	Top 1 Accuracy, Top 5 Accuracy	Surveillance
Gandor et al. [7]	Impact of Image Compression on Object Detection Using Deep Learning	JPEG compression	plethora dataset (5000 Images with 80 Classes)	RetinaNet. ResNet50, ResNet101, ResNetXt	AP and mAp	Digital photography and document archiving Video Surveillance
Choi et al. [8]	High-efficiency compression for object detection	bit allocation and rate control strategy	PASCAL VOC 2007 dataset which has 9963 images	YOLO9000	Map	surveillance and visual analytics

7. References

[1] Microsoft COCO Dataset. Online: https://cocodataset.org/ Date Last Visited: March 23, 2024

[2] T. Lin and Pengwei Hao, "Compound image compression for real-time computer screen image transmission," in *IEEE Transactions on Image Processing*, vol. 14, no. 8, pp. 993-1005, Aug. 2005, doi: 10.1109/TIP.2005.849776.

[3] Zhou, X., Xu, L., Liu, S., Lin, Y., Zhang, L., & Zhuo, C. (2019). An Efficient Compressive Convolutional Network for Unified Object Detection and Image Compression. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 5949-5956. https://doi.org/10.1609/aaai.v33i01.33015949

[4] Choi, Hyomin and Ivan V. Bajić. "Deep Feature Compression for Collaborative Object Detection." *2018 25th IEEE International Conference on Image Processing (ICIP)* (2018): 3743-3747.

[5] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," 2016 Eighth International Conference on Quality of Multimedia Experience (QoMEX), 2016, pp. 1-6, doi: 10.1109/QoMEX.2016.7498955.

[6] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy: Explaining and Harnessing Adversarial Examples. ICLR (Poster) 2015

[7] Gandor, T.; Nalepa, J. First Gradually, Then Suddenly: Understanding the Impact of Image Compression on Object Detection Using Deep Learning. *Sensors* 2022, *22*, 1104. https://doi.org/10.3390/s22031104

[8] Choi, Hyomin and Ivan V. Bajić. “High Efficiency Compression for Object Detection.” 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (2018): 1792-1796.

[9] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” in IEEE Conf. Computer Vision and Pattern Recognition, Jul. 2017.

Emad Felemban, PhD Received his PhD and MSc from Ohio State University and BSc from King Fahd University of Petroleum and Minerals He is a full professor in Umm Al-Qura University in Saudi Arabia. He specializes in Wireless Networks and IoT. His research interests include Wireless Communication in challenging environments such as underwater. He has more than 7 patents registered in USPTO and more than 40 papers in reputed journals.

Saleh Basalamah is a Professor of Computer Engineering in Umm Al-Qura University, he graduated with a PhD in Bioengineering from Imperial College London in 2005. He became the dean of the College of Computing and information systems between 2009 and 2012. He co-founded GIS Technology Innovation Center (GISTIC) in Umm Al-Qura University in 2013 and was the deputy director of the center until 2016. He also co-founded 3 technology spinoffs from GISTIC which were incubated by Wadi Makkah. He works on several research areas including computer vision, spatial systems, and multimedia. He published many journal papers, conferences, and patents. He's a senior member of IEEE and a member of ACM. Contact him at smbasalamah@uqu.edu.sa.

Adil Amjad, holds a double Masters Degree from RWTH Aachen, Germany and University of Trento, Italy. His research interests include Industrial Edge, Industrial IoT and Smart Cities IoT. He is currently working in Siemens Digital Industries as Principal Software Engineer

Atif Naseer obtained his PhD degree from the University of Malaga Spain, MS degree from NUST Pakistan, and BS degree from UET Taxila Pakistan. He is an academician by profession and is currently serving as a lecturer and researcher at Umm-al-Qura University, Makkah, Kingdom of Saudi Arabia. He has been attached to academia for over 12 years and has served as permanent faculty in many prestigious universities. His areas of specialization are Software Engineering, Wireless Sensor Networks, Big Data Analysis, AI, and Machine learning. Contact him at Umm-al-Qura University, Makkah Saudi Arabia; anahmed@uqu.edu.sa.