scientific reports

Check for updates

OPEN

Optimizing FCN for devices with limited resources using quantization and sparsity enhancement

Muhammad Faizan-Khan[1]✉, Nisar Ali[2], Raja Hashim Ali[3], Areej Alasiry[4], Mehrez Marzougui[4], Shabbab Ali Algamdi[5] & Yunyoung Nam[6]✉

This study addresses the optimization of fully convolutional networks (FCNs) for deployment on resource-limited devices in real-time scenarios. While prior research has extensively applied quantization techniques to architectures like VGG-16, there is limited exploration of comprehensive layer-wise quantization specifically within the FCN-8 architecture. To fill this gap, we propose an innovative approach utilizing full-layer quantization with an L_2 error minimization algorithm, accompanied by sensitivity analysis to optimize fixed-point representation of network weights. Our results demonstrate that this method significantly enhances sparsity, achieving up to 40%, while preserving performance, yielding an impressive 89.3% pixel accuracy under extreme quantization conditions. The findings highlight the efficacy of full-layer quantization and retraining in simultaneously reducing network complexity and maintaining accuracy in both image classification and semantic segmentation tasks.

Keywords Deep learning, Fixed-point quantization, Fully convolutional network, Sensitivity analysis

Deep learning models with multiple layers have revolutionized computer vision, boosting performance in various applications, such as music generation and semantic segmentation[1]. This success stems from abundant labelled data, new network architectures, and powerful GPUs[2]. By effectively approximating complex functions, these models, particularly Convolutional Neural Networks (CNNs), excel in machine learning tasks like classification[3,4], segmentation[5], and regression[6]. CNNs overcome the limitations of earlier models by introducing convolution and pooling layers, tackling overfitting and spatial ratio issues in 2D data[7]. These layers preserve spatial information and make the network resistant to translations, rotations, and scaling[7]. Advances like transposed convolutional layers, skip connections, and variable-sized inputs/outputs have led to Fully Convolutional Networks (FCNs) that excel in semantic segmentation, assigning class labels to each pixel in an image[8]. While CNNs have achieved good results in semantic segmentation tasks[9], FCNs have demonstrated the best performance in pixel-wise classification for semantic segmentation when trained end-to-end[5].

Although FCNs and other deep learning models have demonstrated effectiveness in semantic segmentation, their application on resource-constrained devices is limited[10]. Neural networks tend to have excessive parameters, and deep learning models have significant redundancies, leading to inefficient use of memory and computational resources[11]. The floating-point arithmetic operations used in artificial neural networks also tend to be more expensive than integer and fixed-point operations, making deep learning-based applications difficult to deploy on mobile devices[10].

Quantization techniques

The optimization of storage and energy utilization in deep learning models, enabling their deployment on resource-constrained devices, has been a topic of ongoing research. Several strategies based on fixed-point

[1]Departament d'Enginyeria Electrònica, Elèctrica i Automàtica, Universitat Rovira i Virgili, Tarragona, Spain. [2]Faculty of Engineering and Applied Science, University of Regina, Regina S4S0A2, Saskatchewan, Canada. [3]University of Europe for Applied Sciences, Potsdam 14469, Germany. [4]College of Computer Science, King Khalid University, Abha 61413, Saudi Arabia. [5]Department of Software Engineering, College of Computer Science and Engineering, Prince Sattam bin Abdulaziz University, Al Kharj, Saudi Arabia. [6]Department of Computer Science and Engineering, College of Engineering, Soonchunhyang University, Asan, South Korea. ✉email: muhammadfaizan.khan@urv.cat; ynam@sch.ac.kr

approximations have been proposed to mitigate the computational requirements of deep learning models. These include the 8-bit fixed-point method[12], low-precision fixed-point arithmetic for inference[13,14], low-rank approximation of network parameters with accuracy trade-off[15], training the network with binary weights[16], and the L_2 error minimization approach[17,18]. Another group of popular quantization techniques involves codebook-based methods[19], such as the sparse network with random behaviour and ternary weights[20] or the HashedNet approach based on hashing[21]. These fixed-point approximations and codebook-based methods reduce the complexity of deep learning models through compression and quantization. Additional solutions include the consolidation of multiple models into a single model[22] and the combination of pruning and quantization techniques[23].

Recent literature has explored various directions to enhance the effectiveness of quantization and binarization. For instance, LoRA-based quantization techniques for large language models have focused on preserving information retention during fine-tuning[24], while methods like QuantSR have applied low-bit quantization for efficient super-resolution in vision tasks[25]. Generative approaches for data-free quantization have also emerged to improve flexibility and accuracy under strict constraints[26]. On the binarization front, distribution-sensitive training strategies have been shown to improve representational capacity in binary neural networks[27], and binarization has been successfully employed for efficient video matting in low-resource environments[28]. These works demonstrate the broad applicability of quantization and binarization across domains, and our study builds on this foundation by adapting low-bit quantization to dense semantic segmentation with memory awareness and retraining strategies.

Gap analysis

While quantization and compression methods effectively reduce the size of deep learning models, they also result in a decline in the accuracy of inferences. Furthermore, current fixed-point optimization techniques, such as those proposed by Anwar et al.[17] and Shin et al.[18], only focus on quantizing convolution and pooling layers. Quantization of the complete FCN has been performed by Xu et al.[29], including both convolution and transposed convolution, up-convolution, and de-convolution layers. However, to the best of our knowledge, there is currently no technique that has quantized all layers of the FCN-8 architecture based on VGG-16. Thus, the reduction of the size of the network while maintaining accuracy through the quantization of other layers in the FCN-8 architecture of VGG-16 remains an unexplored area of research.

Our contribution

This study investigates the idea of quantizing each layer of an FCN-8 using the same network configuration as Shelhamer et al.[30] for reducing the computational cost of inference via the L_2 error minimization approach. The network is first quantized, then a layer-wise sensitivity analysis is performed to find the optimal value of quantization for each layer in the network, and an updated retraining algorithm is then deployed to quantize weights to recover loss in accuracy even with low precision. In addition, this study examines the impact of using parallel architecture on training time with and without using a GPU. It also compares our method's findings to those of other quantization techniques. The results show that, across all layers, the L_2 error minimization technique retains the best ratio between network performance and network sparsity size with fewer bits than both the embedded fixed-point algorithm and Lloyd's method. Figure 1 illustrates the overall summary of the experiments performed in the current work.

Table 1 illustrates the summary of the research work performed in the field of quantization of FCNs and highlights the contributions and novelty of the proposed approach.

Materials and methods

The dataset used, the architecture of FCN deployed, an overview of the modified L_2 error minimization technique used, and the comparison methodology are mentioned in detail below.

Dataset

FCN8 was trained on the PASCAL VOC 2012 dataset and validated on the PASCAL VOC 2011 dataset[31]. The PASCAL VOC 2012 dataset contains 20 classes with 9630 labelled images and is used as a standard benchmark dataset for object detection and segmentation[32]. A few sample images from the VOC 2012 dataset, their labels, and their pixel-wise classification are shown in Appendix A. The PASCAL VOC 2011 dataset consists of 736 non-intersecting labelled images and was used for validation.

Architecture of fully convolutional network

For this study, the same network architecture (FCN-8) as that proposed by Shelhamer et al.[30] and illustrated in Fig. ure 2 was deployed so that the results from the proposed method are comparable with other quantization techniques. FCN-8 architecture adopts the pixel-wise loss function, and the stochastic gradient descent approach is used in the current study to optimize the weights and biases of the network.

For quick learning, a VGG pre-trained model is picked for initializing the encoder sub-network. The VGG network uses fewer trainable parameters per layer due to the introduction of a small filter of size 3×3. A fixed learning rate is used during training. Table 2 shows the network configuration to train the FCN for performing segmentation on the PASCAL VOC dataset.

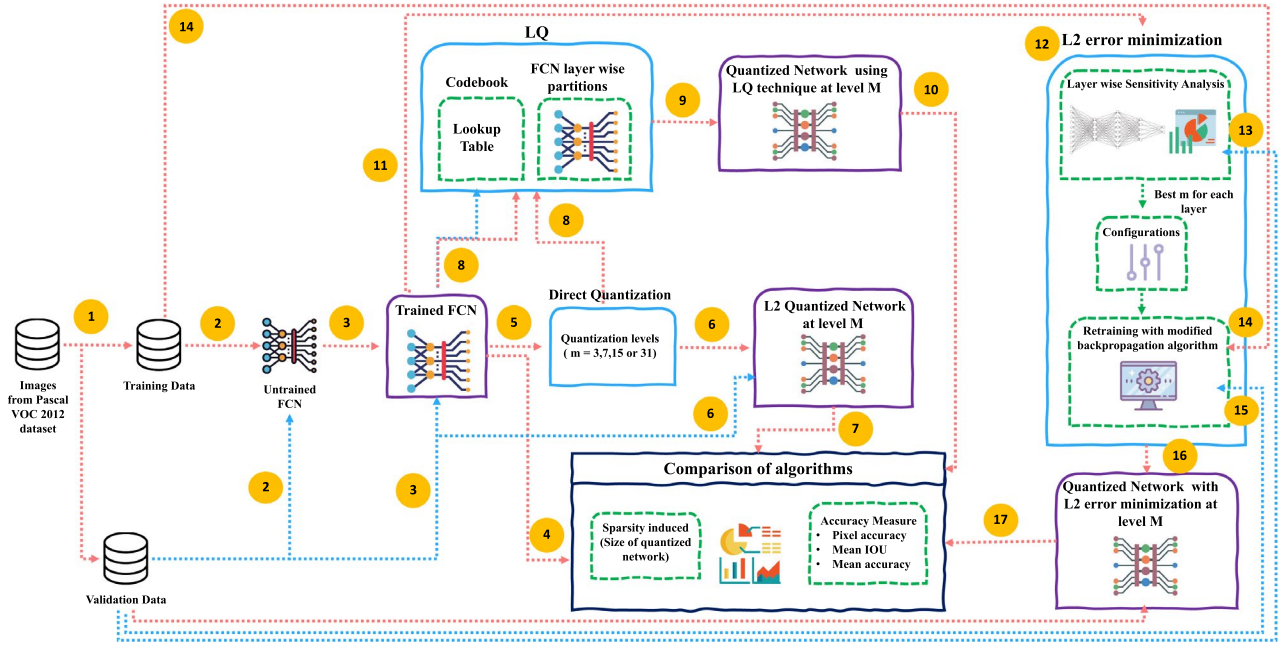

Fig. 1. Figure showing the flowchart proposed for FCN-8 quantization and the comparison pipeline followed (for quantization techniques, i.e., direct quantization, Lloyd's quantizer, and L_2 error minimization) in the current study based on pixel accuracy, mean IOU, and mean accuracy.

Paper name	FCNs used	L_2 error minim.	Applied on				Signal quantized	Dataset used	No. of bits	Layerwise sens. analysis	Sem. segm.
			Conv. layer	Skip layer	Trans. layer	Fully conn. layer					
Vanhoucke et al.[26]						✓	✓				
Courbariaux et al.[27]			✓			✓		MNIST, SVHN, CIFAR-10	10		
Gupta et al.[28]			✓			✓		MNIST, CIFAR-10	12		
Denton et al.[29]			✓			✓		ImageNet 2012			
Lin et al.[30]			✓			✓		MNIST, SVHN, CIFAR-10			
Hwang et al.[31]		✓				✓		MNIST	2,3,4,5		
Anwar et al.[32]		✓	✓			✓		MNIST	2,3,4,5		
Shin et al. [33]		✓				✓		TIMIT Corpus, MNIST	2,3,4,5	✓	
Gong et al.[34]			✓			✓		ImageNet 2012			
Xu et al. [39]	✓		✓	✓	✓			MICCAI Gland 2015	7		✓
Hubara et al. [46]			✓			✓	✓	MNIST, SVHN, CIFAR-10, ImageNet 2012, Penn-tree bank	4		
Lin et al. [47]			✓			✓		CIFAR-10	1,2,3,4		
Cai et al. [48]			✓			✓		Microsoft COCO, ImageNet 2012	8, 32	✓	×
Proposed Approach	✓	✓	✓	✓	✓			PASCAL VOC 2012	2,3,4,5	✓	✓

Table 1. Literature review table showing various contributions in the quantization of networks.

Fixed-point quantization techniques

Embedded fixed-point algorithm

An example of direct quantization, the same quantization level M, is set for all the layers in the embedded fixed-point quantization algorithm. Various M quantization levels are opted for, where each level represents a certain number of bits, i.e., M=3 for 2 bits, M=7 for 3 bits, M=15 for 4 bits, and M=31 for 5 bits, respectively.

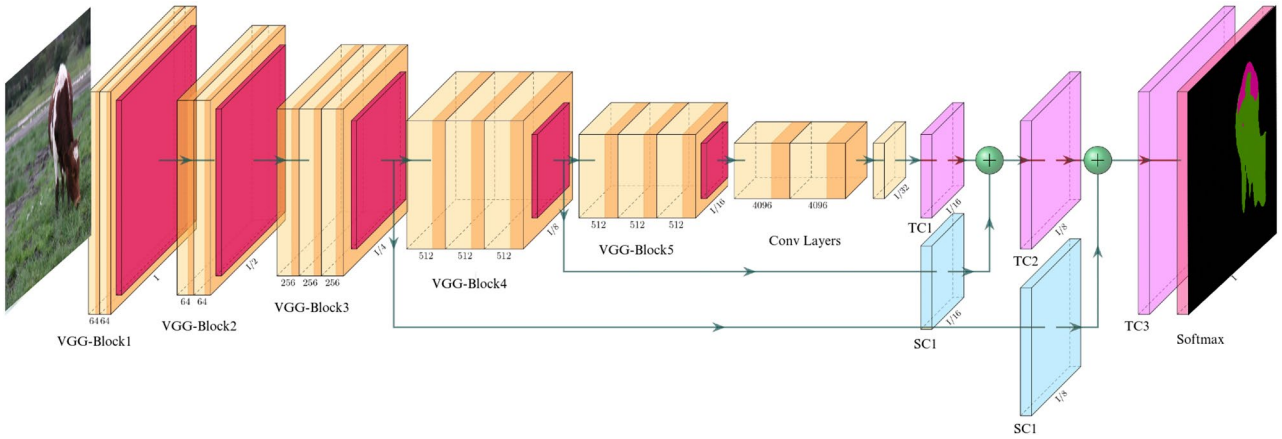

Fig. 2. Figure showing the architecture of FCN-8. The figure displays the different layers and the number of units in each layer for the FCN-8. Each box indicates a certain layer in the architecture, where the colour of the box indicates the type of layer present, while the number of units in the layer is written underneath it. Light yellow colour represents the convolution layer, red colour is for the pooling layer, light turquoise colour indicates the transpose convolution layer, pink depicts the softmax activation (output layer), and lavender colour shows the skip connection layer.

Network configuration	
Epochs	50
Learning rate	0.0001
Mini batch size	20
Optimizer	SGD
Momentum	0.9
Weight decay	0.0002
L_2 Regularization	None
Samples in training set	8498
Samples in validation set	786

Table 2. Configuration table showing the network configuration of FCN used in this study. The table shows the various configuration settings used for FCN-8.

Lloyd's quantizer

Lloyd's quantizer[33] (LQ) is a codebook-based quantization technique that uses partitions (different adjacent and non-overlapping ranges of values that belong to a set of real numbers) and a codebook (a lookup table) to determine the quantization of input values. Hence, by design, LQ transforms the input weights into a codebook entry, which results in retaining the network performance while significantly decreasing the network size.

Our proposed approach

Quantization mechanism

The weights of neural networks are represented in floating-point precision. Still, a fixed-point representation of network parameters greatly reduces the cost of each operation at the cost of precision and accuracy. In this study, we quantize the weights of the network to convert continuous data into discrete points for fixed-point representation using the L_2 error minimization technique, as inspired by several studies[17,18] for network quantization. The network is first divided into signals, weights, and biases. The weights and biases are translated to fixed-points, but the signals are kept in floating-point precision. Direct quantization is used first, followed by layer-wise sensitivity analysis, and finally, retraining is performed using a modified back-propagation algorithm.

Layer-wise sensitivity analysis

A layer-wise sensitivity analysis was performed for the convolutional layer and transposed convolution layer to achieve maximum sparsity while maintaining the same accuracy, where weights in one layer were quantized with a given M quantization level and the weights of remaining layers were used with floating-points to maintain high precision. This process was repeated for each layer in the network, and the effective quantization level for that layer was calculated. Table 3 shows the six different quantization levels (M) configurations used in this analysis. After conducting the sensitivity analysis, FCN layers were grouped layer-by-layer according to their range and sensitivity, and an optimum value of M for each layer was calculated.

Config	Quantization level (M) for encoder layers							Quantization level (M) for decoder layers					
Name	C1	C2	C3 - C10	C11	C12	C13	C14	C15 - C16	TC1	SC1	TC2	SC2	TC3
E33-D33-SC0	3	3	3	3	3	3	3	3	3	0	3	0	3
E77-D77-SC0	7	7	7	7	7	7	7	7	7	0	7	0	7
E73-D33-SC0	7	7	7	3	3	3	3	3	3	0	3	0	3
E37-D33-SC0	3	3	7	7	3	7	3	3	3	0	3	0	3
E33-D33-SC3	3	3	3	3	3	3	3	3	3	3	3	3	3
E77-D77-SC7	7	7	7	7	7	7	7	7	7	7	7	7	7

Table 3. Configurations for Quantization. The table shows the different configurations applied on each layer of the floating-point network with $PA = 91.01\%$,$MIU = 62.2\%$,$MA = 78.1\%$ accuracy (FCN-8 used for image segmentation). C indicates the convolution layer, TC represents the transpose convolution layer, and SC is used for the skip connections layer.

Retraining the quantized network

A modified back-propagation algorithm was used for retraining the network to adjust the quantized weights. The modified back-propagation algorithm is given in Appendix D. The results from the L_2 error minimization technique applied to all layers are compared with the embedded fixed-point algorithm and with Lloyd's quantizer approach in this study.

Experiment settings and implementation details

To test the efficacy of applying quantization to FCNs, several simulations were performed. The Nvidia GeForce GTX 1060 GPU was used to train and test the models, and experiments were run in the MatConvNet simulation environment. The quantization procedure for L_2 error minimization was implemented in CUDA, and a MEX function was used to link the CUDA code to MATLAB.

Comparison metrics

Pixel accuracy (PA), the mean intersection of the union (mean IoU), and the mean accuracy (MA) as percentages were used to determine the best configuration for L_2 error minimization and for comparing our method with other quantization techniques. PA is the percentage of pixels of the image that are correctly classified. IoU is the field of overlap divided by the area of the union between the predicted label and the ground-truth label. For multi-class segmentation, Mean IoU calculates the percent intersection over the union of each class and then determines the average for all classes. On the other hand, MA takes the percentage of average accuracy of all classes.

Let p_{ij} be the number of pixels of class *i* that are predicted to belong to class *j*, t_i be the total number of pixels in class *i*, then *PA*, is given in Eq. 1.

$$PA = \sum_i p_{ij} / \sum_i t_i \tag{1}$$

Let p_{ij} be the numbers of pixels of class *i* that are predicted to belong to class *j* and p_{cl} be the number of pixels for different classes, then Mean IoU, *MeanIoU*, is given in Eq. 2.

$$MeanIoU = (1/p_{cl}) \sum_i \frac{p_{ij}}{(t_i + \sum_j p_{ji} + p_{ij})} \tag{2}$$

The MA, is given in Eq. 3

$$MA = (1/p_{cl}) \sum_i p_{ij}/t_i \tag{3}$$

The comparison for the performance of the L_2 error minimization technique, the embedded fixed-point algorithm, and the Lloyd quantizer (LQ) was made using mean IoU, PA, and MA. Since it was impossible to plot a graph for every layer, averaging the results from the encoder and decoder groups was applied by evaluating the accuracy metrics.

Hybrid quantized network

The accuracy for all configurations for the retrained quantized network was calculated using the hybrid quantized network. In the hybrid quantized network, skip connection layers (SC1 and SC2) were quantized directly, while the remaining layers (C1-16 and TC1-3) were quantized using the layer-wise sensitivity analysis.

Modified backpropagation algorithm for L_2 error minimization

The error signal, represented by δ_i, in i^{th} iteration of the original back-propagation algorithm is shown in Eq. 4.

$$\delta_i = -\frac{\partial E}{\partial net_i} \tag{4}$$

where δ_i is the error signal in i^{th} iteration, E is the output error and net_i is the sum of all the input values that belong to i.

The net_i and the output of a neuron $y_j^{(q)}$ in the updated feedforward pass are defined, respectively, in Eqs.s 5 and 6.

$$net_i = \sum_{j \in A_i} w_{i,j}^{(q)} y_j^{(q)} \tag{5}$$

$$y_j^{(q)} = R_i\left(\phi_i\left(net_i\right)\right) \tag{6}$$

where $w_{i,j}^{(q)}$ are the quantized weights, $y_j^{(q)}$ is the quantized signal, A_i is the neuron i that are near to input layer in the network, $R(.)$ is the signal quantizer, and $\phi(.)$ is the activation function.

The delta in the updated back-propagation pass is given in Eq. 7.

$$\delta_j = \phi_j'\left(\mathrm{net_i}\right) \sum_{i \in p_i} \delta_j w_{i,j}^{(q)} \tag{7}$$

where ϕ_j' is the change in activation function, $w_{i,j}^{(q)}$ is the quantized weight, δ_j is the error in neuron j, and p_i is the unit i that are near to end layer in the network.

Now the error gradient $\frac{\partial E}{\partial w_{i,j}}$ is calculated as shown in Eq. 8.

$$\frac{\partial E}{\partial w_{i,j}} = -\delta_i y_j^{(q)} \tag{8}$$

where $\partial w_{i,j}$ is the change in weight, ∂E is the change in error, and $y_j^{(q)}$ is the quantized signal.

The weight update $w_{ij,new}$ can be calculated as shown in Eq. 9.

$$w_{ij,new} = w_{ij} - \alpha \left\langle \frac{\partial E}{\partial w_{i,j}} \right\rangle \tag{9}$$

Then the weight update during an iteration q is denoted by $w_{ij,new}^{q}$ and given in Eq. 10.

$$w_{ij,new}^{q} = Q_{ij}\left(w_{ij,new}\right) \tag{10}$$

where α is the learning rate, and $w_{i,j}$ is the floating-point weight.

Results

To investigate the impact of GPU processing on time, we first developed results for FCN without quantization and then used a parallel quantization algorithm. After that, layer-wise sensitivity analysis is performed, which is then evaluated using standard assessment metrics such as mean IoU, PA, and MA. We compared the results of our approach with other popular quantization techniques on network sparsity, mean IoU, PA, and MA after determining the best configuration for L_2 error minimization.

Accuracy of FCN without quantization

The network is first trained with the stochastic gradient descent and error-back propagation algorithms. For comparison purposes, the same dataset was used in training as that used by Shelhamer et al.[30]. On the PASCAL VOC 2011 dataset, the model took about 34 hours to train on the GPU, with a PA of 91.01%, a mean IoU of 62.2%, and a MA of 78.1%.

Accuracy of FCN with direct quantization

After standard training, the network is directly quantized where the quantization is embedded in the retraining procedure. The layer-wise weights and the zero weights before and after quantization in the network are displayed as raw tables in Appendix B. Table 4 demonstrates the results of direct quantization on the trained FCN network for the 6 quantization configurations shown in Table 3.

Layer-wise sensitivity analysis

Mean intersection over union

Figure 3 Panel A) shows the Mean IoU after performing a sensitivity analysis on the weights of the encoder layers at various levels. At level $M = 3$, the weights of all encoder layers showed more sensitivity to quantization, especially C1, C7, C11, and C13. On the other hand, at level $M = 7$, weights of all the encoder layers exhibited low sensitivity to quantization except C1.

FCN without quantization			FCN with quantized networks						
Percentage accuracy (%)			Config #	Percentage accuracy with direct quantization (%)			Percentage accuracy with retrained quantization (%)		
PA	Mean IOU	PA		PA	Mean IOU	MA	PA	Mean IOU	MA
91	62	78	E33-D33-SC0	75	4	5	83	46	58
			E77-D77-SC0	76	7	8	90	59	77
			E73-D33-SC0	75	4	5	88	53	74
			E37-D33-SC0	75	4	5	89	56	75
			E33-D33-SC3	75	4	5	75	4	10
			E77-D77-SC7	76	7	8	76	4	5

Table 4. Performance of FCN without quantization (base case) and with direct and retrained quantizations. The table shows the performance of FCN in terms of percentage accuracy (PA, mean intersection over union (mean IoU), and MA). Notice the severe degradation in performance with direct quantization of the network, while the performance is almost retained by the retrained network in the first four configurations. A retrained network with configuration E77-D77-SC0 produces the best performance.

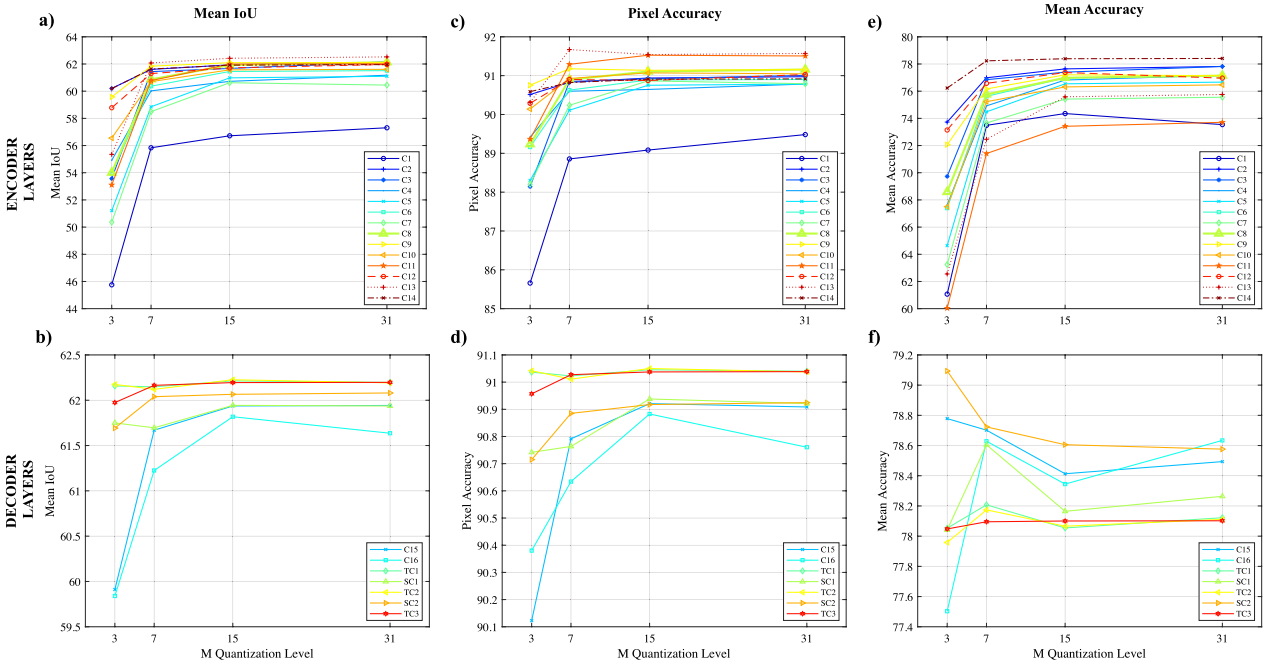

Fig. 3. Figure showing the sensitivity analysis of the retrieved quantized network for different quantization levels. Panels (**a**), (**c**), and (**e**) show mean IoU, PA, and MA values for encoder layers (C1–C14), while panels (**b**), (**d**), and (**f**) display the same values for decoder layers (C15-16, SC1-2 and TC1-3). All evaluation methods show that each layer is sensitive to the number of bits in the quantizer, where more bits mean higher accuracy.

Figure 3 Panel B) shows the Mean IoU after performing a sensitivity analysis on the weights of the decoder layers. At level $M = 3$, the weights of all decoder layers show low sensitivity to quantization, while at level $M = 7$ they showed comparably improved results.

Pixel accuracy

Figure 3 Panel C) shows the PA after performing a sensitivity analysis on the weights of the encoder layers. At level $M = 3$, the weights of all layers show more sensitivity to quantization, especially layers C1, C3, C7, and C5. On the other hand, at $M = 7$ weights of almost all layers show less sensitivity to quantization except the C1 layer. However, C13, C11, and C9 interestingly give more accurate results than the ones obtained from the floating-point network.

Figure 3 Panel D) shows the PA after performing sensitivity analysis on the weights of the decoder layer, which shows that at level $M = 3$, all layers show less sensitivity to quantization, and at level $M = 7$, all layers show higher accuracy than at $M = 3$.

Mean accuracy

Figure 3 Panel E) shows the MA after performing sensitivity analysis on the weights of the encoder layers, where it was found that at level $M = 3$, C2 and C14 layers give comparably better results, but all other layers show higher sensitivity to quantization, mainly C11, C1, C13, and C5. On the other hand, at level $M = 7$, the accuracy of C14 is the same as that for a floating-point network. However, all layers, except C11, C1, and C13, show lower sensitivity to quantization than $M = 3$.

Figure 3 Panel F) shows the MA after performing a sensitivity analysis on the weights of the decoder layers. At level $M = 3$, layers SC2 and C15 give more accurate results than the floating-point network. However, the weights of other layers show lower sensitivity to quantization.

Retrained quantized network

After the layer-wise sensitivity analysis, the network layers were retrained one by one based on the configuration in Table 2. All the networks took nine days for training.

Table 4 shows the results for the retrained quantized network, where the first four configurations show significant improvement in network performance, while the last two configurations show the same network performance as that of the direct quantization network. Therefore, the two skip connection layers, SC1 and SC2, are very sensitive when we apply any *M* level to quantize these layers. In addition, configuration E77-D77-SC0 shows nearly the same accuracy as the floating-point network on the test set.

Figure 4 shows five sample images taken from the PASCAL VOC 2012 dataset, used as a test dataset, along with the resulting pixel labels for these images from the unquantized FCN-8 network, the hybrid quantized network, and a few selected retrained quantized networks. Figure 5 presents qualitative segmentation results on a diverse set of challenging examples. These include scenes with occlusions, fine-grained object boundaries, and visually similar classes such as chair vs. sofa or person vs. bike. Each image is shown with its ground truth, the prediction from the floating-point FCN8 model, and outputs from four different quantized configurations. This comparison highlights how different quantization strategies perform in complex scenarios, illustrating the trade-offs between model compression and segmentation accuracy.

In terms of sparsity produced by different configurations for retrained networks with L_2 error minimization, configuration E73-D33-SC0 induces the highest network sparsity of 45.2% followed by configuration E37-D33-SC0 (network sparsity of 40.8%) and then configuration E77-D77-SC0 (network sparsity of 27.9%). The rest of the quantization configurations mentioned in Table 3 produced lower sparsity than the configuration E77-D77-SC0.

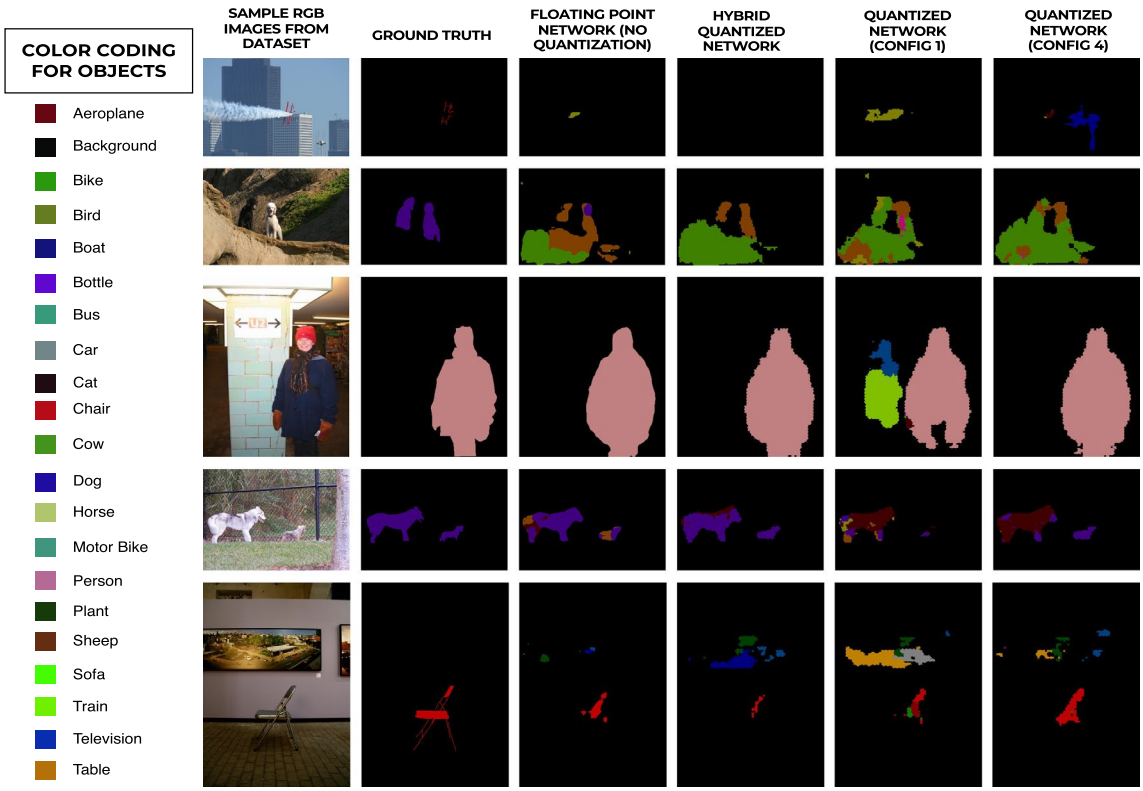

Fig. 4. Image showing some sample images present in the dataset, their pixel-wise labels, and resulting pixel labels from the floating-point network, hybrid quantized network, and two configurations of quantized networks. The legend displays the colour and class (name) of the object to be identified in the image. Five sample images containing aeroplanes, dogs, person, and chairs are shown along with their classification. The data and the pixel labels (ground truth) are taken from the PASCAL VOC 2012 dataset.

Fig. 5. Qualitative segmentation outputs on challenging examples. Sixteen sample images with their pixel-wise classification results are shown for a variety of visually complex or ambiguous scenes. Each row illustrates the segmentation produced by the unquantized (floating-point) FCN8 network and by four different quantized configurations (Config 1–4), alongside the ground truth. These examples include thin structures, occlusions, or inter-class confusion (e.g., person vs. bike, sofa vs. chair), and are selected to demonstrate performance in difficult cases, as requested by the reviewer..

The network's parameter distribution before and after quantization provides valuable information about each layer's sparsity. Figure 6 displays the fraction of network weights present in each layer, along with the percent of layer-wise sparsity achieved for the layer and its M level for configuration E37-D33-SC0.

Sparsity induced by the best performing configuration

The best-performing configuration is defined as the configuration that produces the most sparsity while minimizing the loss of accuracy. As illustrated in Table 4, we find that the configuration E77-D77-SC0 produced the best ratio. The proposed solution achieves an average network sparsity of 45 percent. For some layers, extreme sparsity is induced by L_2 error minimization technique, e.g., for two of three transpose convolution layers, more than 95% sparsity is induced. On the other hand, a minimum of 30% sparsity is induced for the remaining layers, with the lowest sparsity of 30.07% for the skip connection layer SC1.

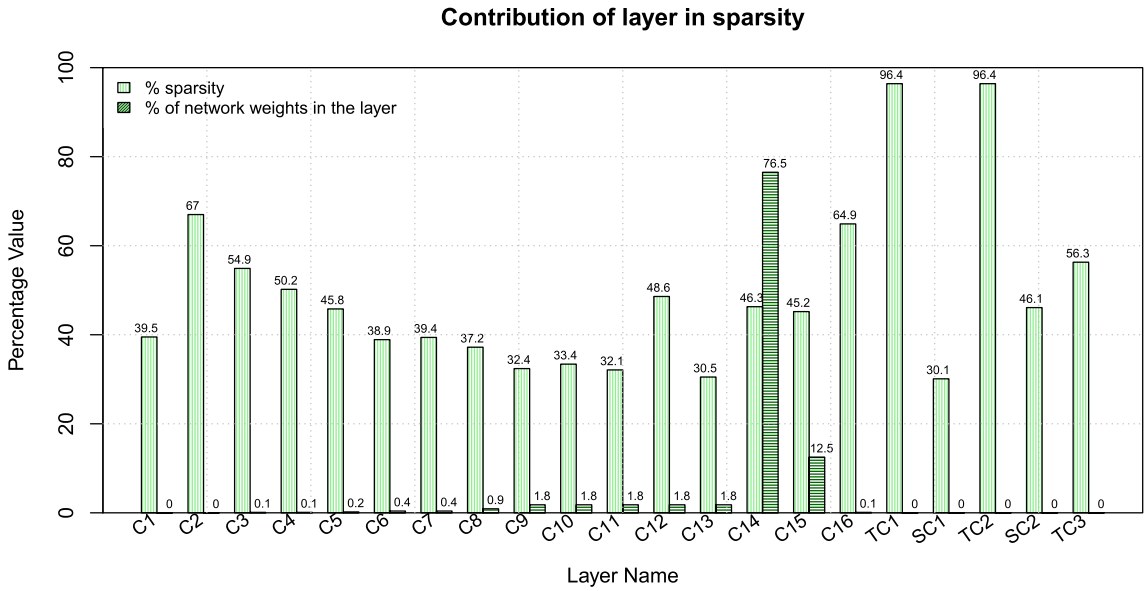

Fig. 6. Bar graph showing the sparsity induced by each layer. The figure shows a bar chart showing the effect of quantization on the overall sparsity of the network. The blue bars show the quantization level as defined for Configuration E37-D33-SC0, the green bars show the percentage sparsity induced by the configuration in the retrained network, and the gray bars indicate the percentage of weights contained in the layer. Notice that the maximum sparsity (green bars) is induced in TC1 and TC2 (greater than 95%), while the greatest number of weights are contained in the C14 and C15 layers (approximately 79%). Many layers contain close to 0% weights, hence, any configuration for those layers does not affect the overall sparsity of the network.

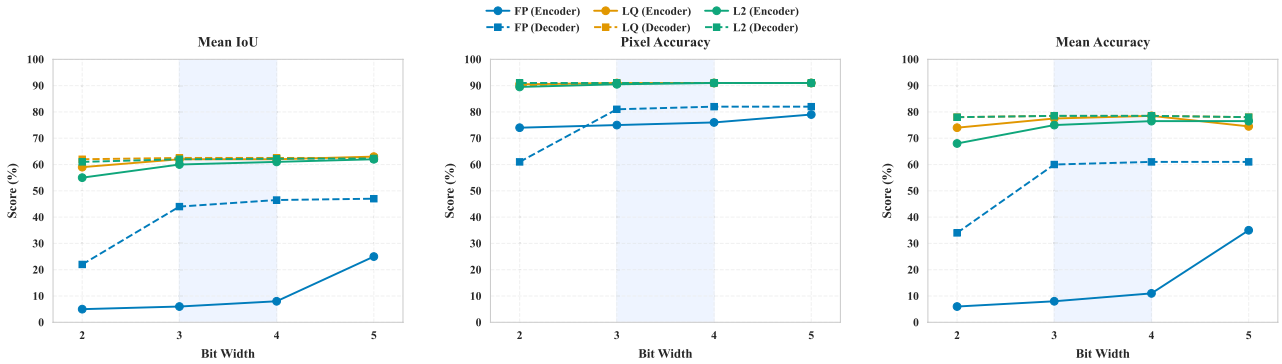

Fig. 7. Performance comparison of three quantization techniques–Fixed-Point (FP), Lloyd's Quantizer (LQ), and L_2 Error Minimization–across encoder and decoder layers. The three panels show mean IoU, PA, and MA as a function of bit width. Quantizers are colour-coded for clarity, and encoder/decoder results are visually grouped. Best-performing regions are highlighted to emphasize trends. The L_2 and LQ methods consistently outperform FP across all metrics and bit widths, demonstrating greater robustness to quantization.

Comparison with other quantization techniques

After applying sensitivity analysis using Fixed-Point (FP), Lloyd's Quantizer (LQ), and L_2 error minimization techniques, the performance of each method is illustrated in Fig. 7. The three panels display results for mean IoU, PA, and MA across encoder and decoder layers. The Y-axis in each panel represents the corresponding evaluation metric, while the X-axis indicates the bit width used for quantization. Quantizers are colour-coded, and encoder/decoder results are grouped to facilitate comparison. The figure highlights that both LQ and L_2 consistently outperform FP, with reduced sensitivity to bit width variations.

In terms of reduction in the size of the network by the quantization technique, Fig. 8 compares the memory footprint and pixel-level accuracy of three quantization strategies–full precision (FP), linear quantization (LQ), and the proposed L_2-based quantization–at 2-bit and 3-bit weight configurations. The original network size before quantization is 457 MB. FP achieves the smallest model size but at the cost of significantly lower accuracy. In contrast, the L_2 method achieves accuracy on par with LQ while requiring less memory in both quantization settings. For instance, at 2-bit precision, L_2 reduces memory usage by approximately 8MB compared to LQ, with both achieving 90.25% accuracy. Similarly, at 3-bit precision, L_2 attains 91% accuracy while using 2 MB

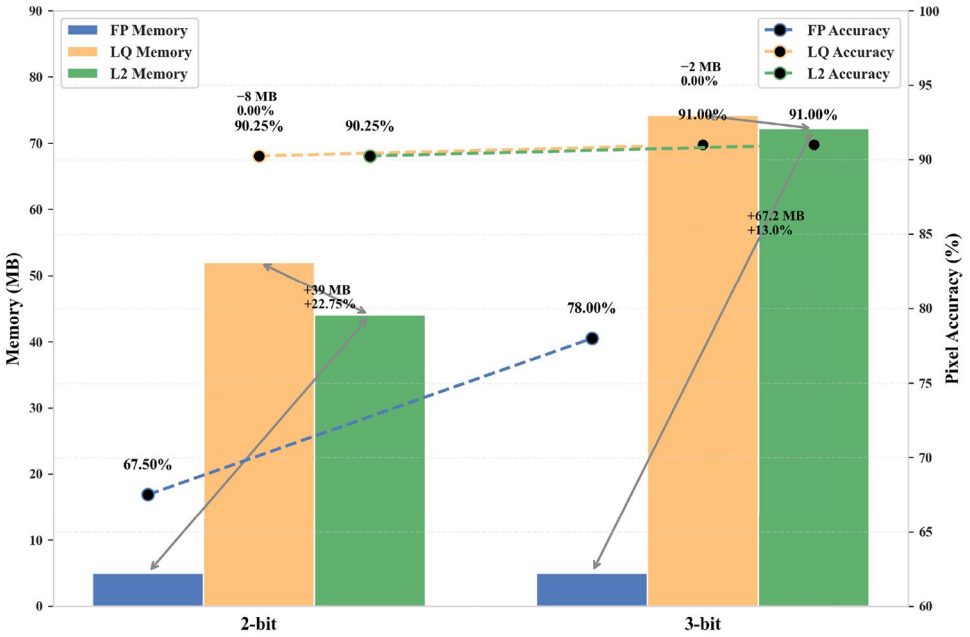

Fig. 8. Memory-accuracy trade-off comparison of Full Precision (FP), Linear Quantization (LQ), and the proposed L_2 Quantization under 2-bit and 3-bit configurations. Bar plots represent model size in megabytes (MB), while dashed lines with solid black markers indicate pixel-level accuracy (%). Annotated bidirectional arrows emphasize L_2's effectiveness: it achieves the same accuracy as LQ with reduced memory usage and significantly outperforms FP in accuracy with a moderate memory increase. These results highlight L_2's suitability for resource-constrained environments where both memory efficiency and predictive performance are critical.

less memory than LQ. Compared to FP, L_2 offers considerable accuracy gains–22.75% and 13% at 2-bit and 3-bit levels respectively–with only a moderate increase in memory usage. These results highlight the efficiency of L_2 quantization as a compelling alternative for memory-constrained deployments, offering a favourable trade-off between compactness and predictive performance.

Discussion

In this study, various quantization techniques for fully convolutional networks for semantic segmentation were explored and compared for performance and induced sparsity. FCNs are hard to deploy on resource-limited devices because enormous computational power and large memory storage are required. FCN has millions of trainable parameters and contains diverse types of layers such as convolution, transposed convolution, and skip connections. The general solution proposed in the literature to this challenge is to perform quantization using various techniques, e.g., embedded fixed-point quantization and Lloyd's Quantizer, to induce sparsity at the cost of network performance. However, these techniques fail to find the right balance and result in either little sparsity or severe degradation of network performance, or both.

The solution proposed in this study for quantizing FCNs is to perform layer-wise sensitivity analysis to find the optimal quantization level for each layer. An updated retraining algorithm with error backpropagation was employed to recover the network loss during training. A direct relationship between quantization level and the performance of the encoder layers is observed due to the higher sensitivity of these layers to quantization, i.e., the higher the quantization level of the encoder part, the better the performance of the network (shown in Fig. 3). Therefore, selecting the right quantization level of these layers is significant to finding the right balance between network performance and induced sparsity.

Since the kernel of a skip connection layer is initialized with zeros, quantizing the skip connection layer with any level of M produces the same accuracy as that for direct quantization. Hence, instead of quantizing the FCN-8 network using the layer-wise sensitivity analysis, hybrid quantization is used, where the skip connection layers (SC1 and SC2) are quantized directly, and the remaining layers are quantized using the layer-wise sensitivity analysis. The skip connection layers (SC1 and SC2) in the FCN-8 architecture were quantized directly without performing sensitivity analysis. This decision was due to their zero-initialization, which inherently results in minimal initial contribution to the output distributions of the network. Consequently, sensitivity analysis would yield negligible insights for these layers, as their quantization does not substantially impact overall accuracy or stability. This rationale aligns with findings in existing literature, which indicate that zero-initialized layers or connections typically show robustness against quantization-induced variations, particularly within image-based deep learning architectures[34]. Thus, sensitivity analysis was reserved for layers whose quantization significantly affects network performance.

Although our experiments were conducted on a single NVIDIA GTX 1060 GPU, the core quantization methods proposed in this work are intended to improve deployability on memory- and compute-constrained platforms. The reduction in model size and fixed-point representation is expected to translate well to mobile or embedded devices, such as ARM-based SoCs or edge accelerators. Future work will extend this study by evaluating inference performance and energy efficiency on such platforms to assess real-world portability.

The proposed research work that uses a fixed-point optimization approach to quantize the fully convolutional network with 2 or 3-bit precision and then fine-tune the quantized weights with retraining. Although this approach is similar to the work of Anwar et al.[17] and Shin et al.[18], there are a few notable differences in terms of approach with these studies. Anwar et al.[17] and Shin et al.[18] only addressed the quantization of convolution and pooling layers, while we have addressed the quantization of all layers, implemented using a GPU processor. Other notable differences between these techniques are the initialization of the quantization step size with the averaging method, deployment of sensitivity analysis to find the optimal value of quantization level for each layer in the network, and retraining of the network by embedding quantization during the retraining process. In addition, we compared the L_2 quantization approach with other quantization methods, such as the embedded fixed-point algorithm and Lloyd's method. Note that experiments done in this study and those done by Shelhamer et al.[30] used the same dataset (PASCAL VOC 2011 dataset) for validation so that the results obtained by both techniques are comparable.

Furthermore, the front and end layers have fewer parameters than the middle layers; e.g., the C14 layer contains the highest number of parameters (more than 100 million) and the highest percentage of weights (approximately 76%). The overall sparsity induced by the L_2 error minimization technique is 45% as most of the network weights are associated with layers C14 and C15, accounting for approximately 89% of the total weights in the network, and these layers show a sparsity percentage of approximately 46%. The L_2 error minimization technique has already been applied on convolutional layers constituting approximately 95% of network weights in FCN-8, and hence the contribution of this study (quantization of other layers in the decoder, constituting 5% of network weights) is not observable for FCN-8. Yet, other network architectures with most network weights in the decoder (e.g., 3D FCNs with 2,037,080 weights in skip connection layers and 93,789 weights in transpose convolution layers versus 269,770 weights in the convolutional layers[35], exist and will be the prime beneficiaries of quantization in terms of induced sparsity in the decoder.

In terms of accuracy, LQ is the most accurate method, slightly better than or equal to L_2 error minimization technique in all encoder and decoder layers for all three accuracy evaluation metrics, as depicted in Fig. 7. However, the difference in performance between these two methods and with the fixed-point method is significant, as there is a noticeable decrease in network performance for all bits and accuracy metrics after performing fixed-point quantization on a network. So, while significant sparsity is observed for FP in Fig. 8, it comes at the cost of accuracy, which means that FP is not a useful quantization technique. Instead, while there is little difference between L_2 and LQ in terms of network performance, L_2 induces significantly higher sparsity than LQ. For example, in the case of 3-bit quantization, L_2 reduces the number of weights in the network by 27.2%while LQ induces only 0.01% sparsity in the network. Hence, L_2 is a quantization technique that retains the best mix between network performance and sparsity induced by the quantization technique.

In terms of sensitivity analysis (Fig. 7), FP is seen to be the most sensitive to bit size as the increase in bit size leads to higher accuracy, and a decrease in bit size quickly degrades the network performance in all three accuracy measures. On the other hand, LQ appears to be insensitive to changes in bit size, while L_2 shows little sensitivity to changes in bit size as well. In addition, L_2 shows no effect on accuracy w.r.t. change in bit size in the decoder layers, while a relatively more observable change is seen for the encoder layers, especially for PA and MA.

While visual comparisons of L_2 and LQ quantized outputs are not included in this work, our quantitative findings indicate that L_2 maintains accuracy slightly better than LQ, particularly in low-bit scenarios. This behaviour is likely linked to L_2's objective of minimizing total reconstruction error, making it less susceptible to performance degradation under quantization. In contrast, LQ relies on uniform scaling, which may introduce greater variance in performance depending on layer-wise weight distributions. Although both methods are effective, L_2 appears to offer enhanced robustness under tighter resource constraints. Further analysis on generalization across datasets and robustness to noise is planned as future work.

The best result of quantization with retraining achieves a sparsity of 40% at the cost of approximately 10% reduction in accuracy as measured by the mean intersection-over-union (62 pre-quantization vs. 56 post-quantization), which is the standard score for the task of semantic segmentation. Although at first, it may not be a worthwhile sacrifice in accuracy, as the current work proposes a reduction in the total size of memory for VGG-16 by 6.35x and other studies with more extreme compression, which combine quantization with pruning and other techniques, reduce the total size in memory of the backbone architecture by 49x[36]. However, both works have a significant difference, and their results are not directly comparable with the current study. The current study mainly consists of applying quantization and fixed-point representation of convolutional network parameters to the particular case of FCN-8s and focuses on applying both post-training quantization (PTQ on a trained network) and quantization-aware training (QAT during the training process). However, the work done by Han et al.[36] performs post-training quantization (PTQ) only and applies its results only on the first fifteen layers of FCN-8 based on VGG19 architecture. Alternatively, the ZeroQ method proposed by Cai et al.[37] adopts existing quantization methods (both QAT and PTQ) with minor modifications and also performs layer-wise sensitivity analysis, albeit on the Microsoft COCO dataset and ImageNet. However, the performance of ZeroQ is not evaluated for semantic segmentation and is thus not comparable to the current study.

Note that the experiments designed for this study measure the quantities of interest for accuracy, parameter sparsity, and memory usage and are in line with the standards of most work on quantization as established by[17,18]. The study compared multiple types of quantizers, such as fixed-point, Lloyd's, and L_2 error minimization

techniques. In the literature, most studies have focused on parameter quantization, but a few recent studies have proposed quantization of both the parameters and activation functions simultaneously, as otherwise conversions between floating and fixed-points are necessary[38]. Hence, one of the avenues to explore in the future is to see the effect of quantization techniques on both the network parameters and the activation functions.

Another point to ponder is the choice of the pilot network (FCN-8) studied in this pilot study. Since the backbone architecture of these networks, VGG-16, has more than 100 million parameters, the effect of quantization and fixed-point representation is most visible with such a large number of weights. However, note that this is a pilot study for measuring the effect of quantization on VGG-16, and the results from the current study encourage the application of these techniques on more recent deep-learning-based architectures such as FaceNet, ResNet, and GoogleNet. Also, while VGG-16 has more than 100 million parameters, contemporary networks like ResNet-50 have only approximately 20 million parameters, and an argument can be made that switching the backbone alone without quantization will save more memory.

Although this study focuses on the FCN-8 architecture, which is derived from VGG, the proposed quantization and sparsity methods are not restricted to this architectural family. The fixed-point quantization strategy, along with L_2-based retraining, is designed to operate independently of specific network topologies. As such, these techniques can be extended to deeper or more modern architectures, such as ResNet or lightweight models like MobileNet, with appropriate adjustments during calibration and retraining. Future work will aim to evaluate the effectiveness of this approach across a broader range of network designs to validate its scalability and generalizability.

Hence, one of the proposed future works on continuing this line of research is to complement the techniques applied in this study with ResNet and see if it further reduces the memory needed. In addition, it will be interesting to investigate if the results of L_2 error minimization and other quantization techniques also hold on the residual connections layer introduced in these recent architectures.

One limitation of the current study is the retraining overhead associated with restoring accuracy after aggressive quantization. In our setup, full retraining of the FCN-8 model required several days of GPU time, largely due to the dense output space and high-resolution inputs typical of semantic segmentation tasks. To address this challenge, future work will explore more efficient strategies such as selectively fine-tuning high-impact layers, progressive precision lowering, and knowledge distillation techniques. These approaches hold promise for reducing retraining time and making the quantization pipeline more practical for frequent model updates and real-time deployments.

Conclusion

This study explored ways to shrink and speed up FCNs, specifically for FCN-8 architecture based on VGG-16, for real-time applications on resource-limited devices. Three quantization techniques were tested in this study. Embedded fixed-point quantization induced a high sparsity but compromised accuracy. Lloyd's quantizer method maintained high accuracy but induced little or no sparsity. On the other hand, our proposed L_2 error minimization approach showed the best balance between accuracy and induced sparsity, leading to faster inference with little degradation in inference accuracy on resource-limited devices. Hence, our study shows that L_2 has the best trade-off between accuracy and speed. In the future, our work can be extended by applying the L_2 error minimization technique on recent architectures like FaceNet, ResNet, GoogleNet, etc. Furthermore, exploring methods other than L_2 error minimization technique, e.g., combining the pruning with the quantization technique or quantizing both weights and activation may further compress the network.

Data availability

The datasets used and/or analyzed during the current study are available from the corresponding authors on reasonable request.

Received: 10 April 2025; Accepted: 11 June 2025
Published online: 04 August 2025

References

1. Mueed, I., Arshad, U. & Ali, R. H. Revolutionizing campus exploration with gikilens: A deep learning-powered object detection app, in *2023 18th International Conference on Emerging Technologies (ICET)*. IEEE, pp. 315–320, (2023).
2. Xiang, H. et al. Deep learning for image inpainting: A survey. *Pattern Recogn.* **134**, 109046 (2023).
3. Ali, N., Ansari, S., Halim, Z., Ali, R. H., Khan, M. F. & Khan, M. Breast cancer classification and proof of key artificial neural network terminologies, in *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)*, pp. 1–6, (2019).
4. Ali, N., Mohammed, A., Bais, A., Sangha, J. S., Ruan, Y. & Cuthbert, R. D. Lodgenet: an automated framework for precise detection and classification of wheat lodging severity levels in precision farming, *Frontiers in Plant Science*, vol. 14–2023, 2023. [Online]. Available: https://doi.org/10.3389/fpls.2023.1255961.
5. Ali, N., Ijaz, A. Z., Ali, R. H., Abideen, Z. U. & Bais, A. Scene parsing using fully convolutional network for semantic segmentation, in *2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, pp. 180–185, (2023).
6. Ali, N., Mohammed, A., Bais, A., Berraies, S., Ruan, Y., Cuthbert, R. D. & Sangha, J. S. Field-scale precision: Predicting grain yield of diverse wheat breeding lines using high-throughput uav multispectral imaging, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 17, pp. 11 419–11 433, (2024).
7. Pathan, R. K. *et al.* Sign language recognition using the fusion of image and hand landmarks through multi-headed convolutional neural network. *Sci. Rep.* **13**(1), 16975 (2023).
8. Alfred Daniel, J., Chandru Vignesh, C., Muthu, B. A., Senthil Kumar, R., Sivaparthipan, C. & Marin, C. E. M. Fully convolutional neural networks for lidar–camera fusion for pedestrian detection in autonomous vehicle, *Multimed. Tools Appl.* pp. 1–24, (2023).
9. Qureshi, I. *et al.* Medical image segmentation using deep semantic-based methods: A review of techniques, applications and emerging trends. *Infor. Fusion* **90**, 316–352 (2023).

10. Kamath, V. & Renuka, A. Deep learning based object detection for resource constrained devices-systematic review, future trends and challenges ahead, *Neurocomputing*, (2023).
11. Denil, M., Shakibi, B., Dinh, L., Ranzato, M. & De Freitas, N. Predicting parameters in deep learning, in *Advances in Neural Information Processing Systems*, pp. 2148–2156, (2013).
12. Vanhoucke, V., Senior, A. & Mao, M. Z. Improving the speed of neural networks on cpus, in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, (2011).
13. David, J., Courbariaux, M. & Bengio, Y. Training deep neural networks with low precision multiplications, *Comput. Sci.* (2014).
14. Gupta, S., Agrawal, A., Gopalakrishnan, K. & Narayanan, P. Deep learning with limited numerical precision, in *International Conference on Machine Learning*, pp. 1737–1746, (2015).
15. Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y. & Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation, in *Advances in Neural Information Processing Systems*, pp. 1269–1277, (2014).
16. Lin, Z., Courbariaux, M., Memisevic, R. & Bengio, Y. Neural networks with few multiplications, *arXiv preprint*arXiv:1510.03009, (2015).
17. Anwar, S., Hwang, K. & Sung, W. Fixed point optimization of deep convolutional neural networks for object recognition, in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 1131–1135, (2015).
18. Shin, S., Boo, Y. & Sung, W. Fixed-point optimization of deep neural networks with adaptive step size retraining, in. IEEE International conference on acoustics, speech and signal processing (ICASSP). *IEEE***2017**, 1203–1207 (2017).
19. Gong, Y., Liu, L., Yang, M. & Bourdev, L. Compressing deep convolutional networks using vector quantization, *arXiv preprint*arXiv:1412.6115, (2014).
20. Arora, S., Bhaskara, A., Ge, R. & Ma, T. Provable bounds for learning some deep representations, in *International Conference on Machine Learning*, pp. 584–592, (2014).
21. Chen, W., Wilson, J., Tyree, S., Weinberger, K. & Chen, Y. Compressing neural networks with the hashing trick, in *International conference on machine learning*, pp. 2285–2294, (2015).
22. Hinton, G., Vinyals, O. & Dean, J. istilling the knowledge in a neural network, *arXiv preprint*arXiv:1503.02531, (2015).
23. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. & Rabinovich, A. Doing deeper with convolutions, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
24. Qin, H., Ma, X., Zheng, X., Li, X., Zhang, Y., Liu, S., Luo, J., Liu, X. & Magno, M. Accurate lora-finetuning quantization of llms via information retention, *arXiv preprint*arXiv:2402.05445, (2024).
25. Qin, H. *et al*. Quantsr: Accurate low-bit quantization for efficient image super-resolution. *Adv. Neural Infor. Process. Syst.* **36**, 56838–56848 (2023).
26. Qin, H. *et al*. Diverse sample generation: Pushing the limit of generative data-free quantization. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**(10), 11689–11706 (2023).
27. Qin, H. *et al*. Distribution-sensitive information retention for accurate binary neural network. *Int. J. Comput. Vision* **131**(1), 26–47 (2023).
28. Qin, H. *et al*. Bimatting: Efficient video matting via binarization. *Adv. Neural Infor. Process. Syst.* **36**, 43307–43321 (2023).
29. Xu, X., Lu, Q., Yang, L., Hu, S., Chen, D., Hu, Y. & Shi, Y. Quantization of fully convolutional networks for accurate biomedical image segmentation, in *Proceedings of the IEEE Conference on Computer Vision And Pattern Recognition*, 2018, pp. 8300–8308.
30. Shelhamer, E., Long, J. & Darrell, T. Fully convolutional networks for semantic segmentation. *IEEE Ann. Hist. Comput.* **04**, 640–651 (2017).
31. Everingham, M., Van Gool, L., Williams, C. K., Winn, J. & Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision* **88**(2), 303–338 (2010).
32. Hariharan, B., Arbeláez, P., Bourdev, L., Maji, S. & Malik, J. Semantic contours from inverse detectors, in *2011 International Conference on Computer Vision*. IEEE, pp. 991–998, (2011).
33. Lloyd, S. Least squares quantization in pcm. *IEEE Trans. Inf. Theory* **28**(2), 129–137 (1982).
34. Umirzakova, S., Muksimova, S., Mardieva, S., Sultanov Baxtiyarovich, M. & Cho, Y.-I. Mira-cap: Memory-integrated retrieval-augmented captioning for state-of-the-art image and video captioning. *Sensors* **24**(24), 8013 (2024).
35. Roth, H. R. *et al*. An application of cascaded 3d fully convolutional networks for medical image segmentation. *Comput. Med. Imaging Graph.* **66**, 90–99 (2018).
36. Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, *arXiv preprint*arXiv:1510.00149, (2015).
37. Cai, Y., Yao, Z., Dong, Z., Gholami, A., Mahoney, M. W. & Keutzer, K. Zeroq: A novel zero shot quantization framework, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 13169–13178.
38. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R. & Bengio, Y. Quantized neural networks: Training neural networks with low precision weights and activations. *J. Mach. Learn. Res.* **18**(1), 6869–6898 (2017).

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2023-00218176) and the Soonchunhyang University Research Fund. The authors also extend their appreciation to the Deanship of Research and Graduate Studies at King Khalid University for funding this work through the Large Research Project under grant number RGP2/471/46.

Author contributions

M.F.K. and N.A. contributed to the investigation, methodology, validation, and writing of the original draft. R.H.A. was responsible for the project administration, resource provision, supervision, and writing—review & editing. A.A. and M.M. contributed to funding acquisition and manuscript review & editing. S.A.A. assisted with review & editing. Y.N. provided funding, supervision, and contributed to the review & editing of the manuscript. All authors reviewed and approved the final manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version of this article (https://doi.org/10.1038/s41598-025-06848-3) contains supplementary material, which is available to authorized users.

Correspondence and requests for materials should be addressed to M.F.-K. or Y.N.