Neural Computing and Applications (2023) 35:11833–11846
https://doi.org/10.1007/s00521-023-08324-3

ORIGINAL ARTICLE

Development of a compressed FCN architecture for semantic segmentation using Particle Swarm Optimization

Mohit Agarwal[1] · Suneet K. Gupta[1] · K. K. Biswas[1]

Received: 29 June 2022 / Accepted: 16 January 2023 / Published online: 3 February 2023

Abstract
Researchers have adapted the conventional deep learning classification networks to generate Fully Conventional Networks (FCN) for carrying out accurate semantic segmentation. However, such models are expensive both in terms of storage and inference time and not readily employable on edge devices. In this paper, a compressed version of VGG16-based Fully Convolution Network (FCN) has been developed using Particle Swarm Optimization. It has been shown that the developed model can offer tremendous saving in storage space and also faster inference time, and can be implemented on edge devices. The efficacy of the proposed approach has been tested using potato late blight leaf images from publicly available PlantVillage dataset, street scene image dataset and lungs X-Ray dataset and it has been shown that it approaches the accuracies offered by standard FCN even after 851× compression.

Keywords FCN architecture · Semantic segmentation · Particle Swarm Optimization · Optimization · Compression and acceleration · Disease segmentation

1 Introduction

Semantic segmentation of images has wide range of applications in diverse fields such as road scene segmentation to identify traffic for self-driven cars, CCTV footage segmentation to count the number of people at any instant of time, medical images segmentation to figure out the diseased portion in the presented samples. Recently several convolution networks have been proposed which can segment the different parts of images after training the network using original images and their corresponding segmented ground truth images. Badrinarayan et al. [7] have proposed an encoder decoder architecture which helps in semantic segmentation with high mean IoU accuracy. Ronneberger et al. [44] have also evolved an architecture called U-Net which has a contracting path and corresponding expanding path which are made of pooling and upsampling layers, respectively. This gives the network a U shape. Long et al. [34] have proposed Fully Convolutional Network architectures (FCN) based on AlexNet, VGG16 and GoogleNet for pixel-wise segmentation of the images. Kaymak et al. [29] have used FCN-AlexNet, FCN-8, FCN-16 and FCN-32 in medical domain to segment regions of skin lesion to detect skin cancer. They demonstrated that it achieved better accuracy than other state-of-the-art methods such as UNet and SegNet. It has been shown [9, 21, 42, 45] that these models have high mean Intersection over Union (mIoU) accuracies for segmenting street scenes, medical images, etc.

Generally these models replace the dense layers at the end of the network by convolution layers resulting in huge network sizes. Although they carry out efficient semantic segmentation, due to the huge size their deployment on edge devices having very less memory, such as mobile phones and raspberry pi becomes extremely difficult. Hence in this research a study has been carried out if these models can be compressed without loosing their performance evaluation metric.

Several research works in past few years have tried to utilize various ways to compress CNN models such

✉ Mohit Agarwal
mohit.agarwal@bennett.edu.in

Suneet K. Gupta
suneet.gupta@bennett.edu.in

K. K. Biswas
kbiswas11@gmail.com

[1] Bennett University, Greater Noida 201310, India

singular value decomposition, knowledge transfer, matrix factorization, pruning, etc. [6, 23, 24, 31, 33, 54]. In very recent years evolutionary approaches using genetic algorithm, etc., have also been experimented to compress these CNN models [46, 51, 52]. Beheshti and Johnsson [8] have demonstrated creation of a squeezed version of UNet which needed 12.08 times less memory and also accelerated the inference time by 1.48 times without loosing the performance metric.

Holliday et al. [25] have used an ensemble of FCN8, FCN16, FCN32, FCN-ResNets, FCN-GoogLeNet for semantic segmentation and found that it gives better results than any of the individual models but since its size is very huge, the authors have used it as a teacher model and trained a student model such FCN-ResNet152 or FCN8 using knowledge transfer and found that this helped in compressing the ensemble model with similar accuracy which was more than individual model. Nguyen et al. [41] have used Genetic Algorithm to evolve the best architecture of FCN by adding or dropping convolution layers, pooling layers, and Conditional Random Field-Gated Recurrent Unit (CRF-GRU) layers. This helped the authors to increase the accuracy of semantic segmentation, tested on PASCAL VOC 2012 dataset, by around 9-10%. Calisto and Lai-Yuen [11] have demonstrated that multi-objective evolutionary-based algorithms (MEA) can be used to work out the best ensemble of 2-D and 3-D FCNs for semantic segmentation of 3-D medical images in prostrate segmentation. 2-D FCNs were used for extracting intraslice information and 3-D FCN for interslice information. The multi-objective evolutionary algorithm helps to find best model which maximizes accuracy and minimizes number of parameters in the network.

Chen et al. [13] have proposed the compression of 3D UNet by converting time dimensionality of 3D UNet to frequency domain and reduce the number of parameters. Therefore, 3D Unet needs the lower computational cost as parameters are comparatively less after compression. Authors have tested the compressed model on Brain Tumour Segmentation (BRATS) dataset and demonstrated that compressed 3D UNet has dice score 0.7832, whereas uncompressed 3D UNet has dice score of 0.7974. Moreover, compressed model achieved the speedup ratio of 2x. Mohan et al. [38] have used quantization techniques to compress SqueezeNet for face mask detection due to the COVID-19 outbreak. Authors found that using quantization SqueezeNet can be compressed from 3.84 to 386 KB and accuracy improved from 98.93 to 98.99%. Skandha et al. [48] have shown compression of CNN models with the help of Genetic algorithm for lung disease classification. Authors have demonstrated that on LIDC-IDRI lung dataset, proposed CNN can be reduced by 90.3% in size while maintaining the performance. Yar et al. [53] have also used Differential Evolution for compression of attention based InceptionV3 for Fire images classification.

2 Aim and novelty of the study

In the proposed study, our objective is to compress the FCN architecture so that it can be easily deployed on resource constrained devices such as raspberry-pi, jetson nano, and mobile phones. Such kind of compressed models are extremely required in present times as such devices are very commonly employed in diverse areas such as agriculture, security surveillance, health care, and IoT devices.

Although lots of papers are available in the area of CNN compression, so far no work has been reported regarding compression of FCN architecture. The novelty of proposed work is to employ meta-heuristic-based approach for FCN compression. Among various methods available such as Genetic Algorithm, Differential Evolution, Particle Swarm

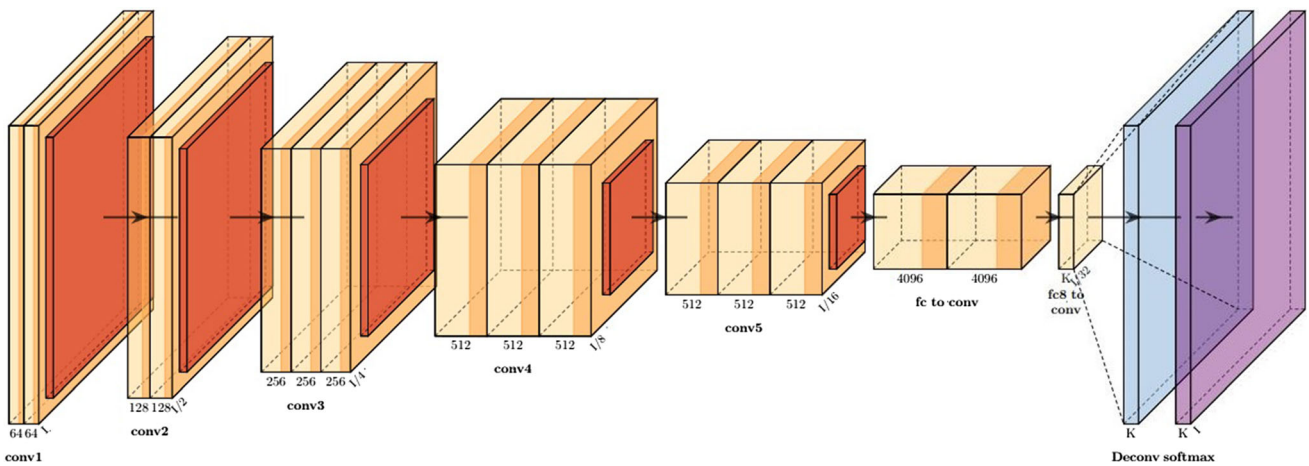

Fig. 1 Architecture of FCN [34]

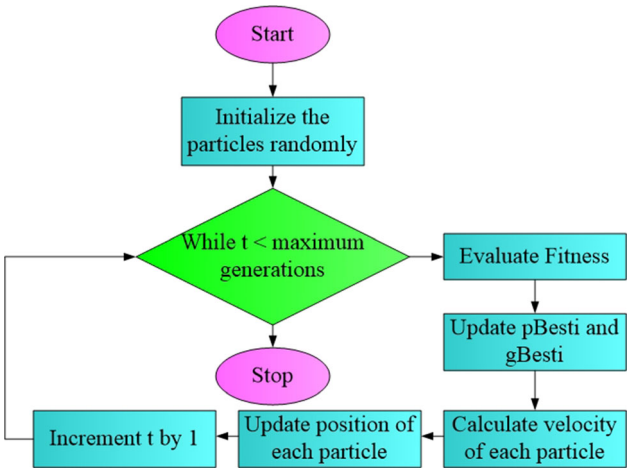

Fig. 2 Particle Swarm Optimization flowchart

Optimization (PSO) and Whale Optimization, the PSO method is reported to be better in terms of high calculation accuracy as well as it carries out global and local search simultaneously.

The novelty of the proposed approach is that compression of the FCN model is carried out without compromising the mean Intersection over Union metric.

The other sections of this paper are organized as follows: The proposed model is explained in Sect. 3. In Sect. 4, there is a discussion on experimental setup and obtained results followed by discussion on real-time deployment of compressed model on mobile device in Sect. 5. The conclusion and future scope is discussed in Sect. 6.

3 Proposed model

VGG16-based FCN is a popular fully convolution layers type deep learning network (refer Fig. 1) which consist of 15 convolution layers. The first two layers are same as first two layers of VGG16 having 64 filters each. This is followed by set of two convolution layers with 128 filters each, set of three convolution layers each having 256 filters, and finally a set of two convolution layers each having 512 filters. Each of these sets have a max-pooling layer in the end. These convolution layers are followed by two fully connected convolution layers with 4096 filters each. At the end an upsampling layer is present with a final softmax layer to give pixel-wise segmented output of same size as input image.

In next subsection a brief introduction to PSO is provided which is being used for compressing FCN models.

3.1 Introduction to PSO

Particle Swarm Optimization (PSO) algorithm developed by Eberhart and Kennedy in 1995 [17] is inspired from natural biology and mimics birds flocking. PSO flowchart is presented in Fig. 2.

In this algorithm given set of particles are randomly assigned initial start points and as algorithm progresses the particles update their position based on their velocity. In each iteration particles update their velocity based on their previous iteration velocity and a component of particles best direction and a component of global best direction. In each iteration fitness value of particle position is calculated and its best and global best is updated. The main equations governing PSO are given in Eqs. (1 and 2).

$$x_{k+1}^{i} = x_{k}^{i} + v_{k+1}^{i} \tag{1}$$

$$v_{k+1}^{i} = w_k \times v_k^i + c_1 \times r_1 \times \left(b_k^i - x_k^i\right) + c_2 \times r_2 \times \left(b_k^g - x_k^i\right) \tag{2}$$

Here x_k^i is the position of the *ith* particle in *kth* iteration. v_k^i is particles velocity, b_k^i is best individual particle position, b_k^g is best swarm position. c_1, c_2 are cognitive and social parameters. r_1, r_2 are random numbers between 0 and 1. w_k is the inertia weight introduced by Shi and Eberhart [47].

Algorithm 1 describes the basic PSO operation.

Algorithm 1 Particle Swarm Optimization

Input : c_1 ▷ Social constant
Input : c_2 ▷ Cognitive constant
Input : Max_iter ▷ Maximum Iterations
Input : Num_particles ▷ Number of particles
Output : b^g_{best} ▷ Best global Particle

procedure PSO(c_1, c_2, Max_iter, Num_particles)
▷ Finds the best particle satisfying fitness criteria
  $x[\,] = \text{INITPARTICLES}()$
  $f^g_{best} \leftarrow 0$ ▷ Set Global best to 0
  $b^g_{best} \leftarrow []$ ▷ Initialize global best particle to null
  for i *in* $range(0, length(x))$ **do**
    $f^i_{best} \leftarrow 0$ ▷ Set each particle best to 0
    $b^i_{best} \leftarrow []$ ▷ Initialize best particle position to null
    $v^i \leftarrow 0$ ▷ Initialize particle velocity to 0
  for k *in* $range(0, Max\_iter)$ **do**
    for i *in* $range(0, Num\_particles)$ **do**
      ▷ Check fitness of each particle
      if $fitness(x[i]) > f^g_{best}$ **then**
        $f^g_{best} \leftarrow fitness(x[i])$
        $b^g_{best} \leftarrow x[i]$
      if $fitness(x[i]) > f^i_{best}$ **then**
        $f^i_{best} \leftarrow fitness(x[i])$
        $b^i_{best} \leftarrow x[i]$
      ▷ Generate 2 random numbers between 0 and 1: r_1, r_2
      $v^i = w_k \times v^i + c_1 \times r_1 \times (b^i_{best} - x[i]) + c_2 \times r_2 \times (b^g_{best} - x[i])$
      $x[i] \leftarrow x[i] + v^i$
      $x[i] \leftarrow \text{ENSURE\_BOUNDS}(x[i])$
  return b^g_{best}

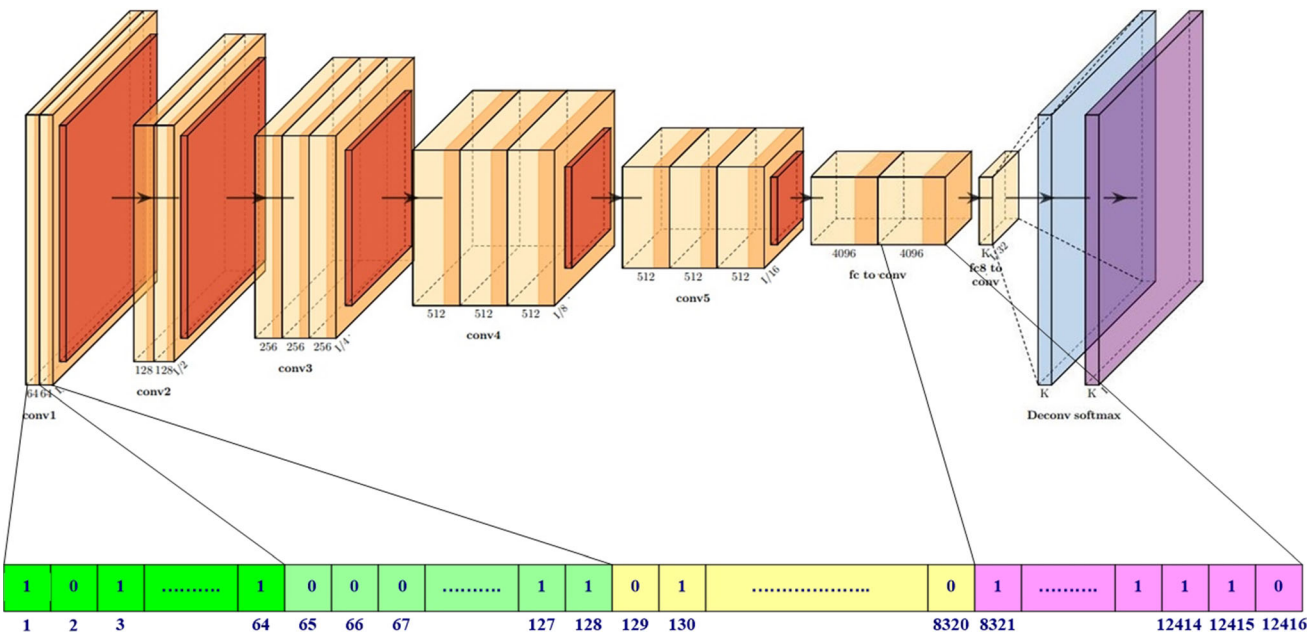

Fig. 3 Particle position corresponding to filters in each layer of FCN

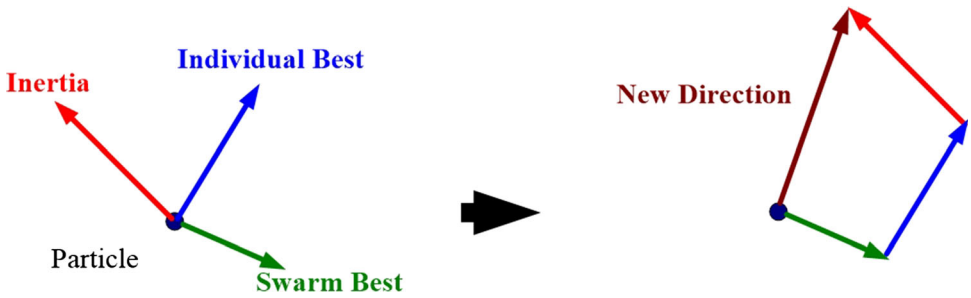

Fig. 4 Calculation of velocity for next iteration using original particle velocity, particle best and swarm best components

x_i	1	0	1	1	0	1	0	0	1	1	.....	1	1	0	1	1	1	1	1	0	1
v_i	0	0	1	1	0	0	0	1	1	0	.....	0	0	1	0	0	0	1	1	1	0
x_best_i	1	1	1	0	0	0	1	1	1	0	.....	1	1	1	1	0	0	1	1	1	0
x_best_g	0	0	1	1	1	1	1	1	0	0	.....	0	0	1	1	0	0	1	0	1	0

c_1	1
c_2	2

$$v\_new = 0.5 \times v\_i + c\_1 \times r\_1 \times (x\_best\_i - x\_i) + c\_2 \times r\_2 \times (x\_best\_g - x\_i)$$

r_1	0.2
r_2	0.6

$$x\_new = x\_i + v\_new$$

v_new	-1.2	0.2	0.5	0.3	1.2	-0.2	1.4	1.9	-0.7	-1.4	.....	-1.2	-1.2	1.9	0	-1.4	-1.4	0.5	-0.7	1.9	-1.4
x_new	-0.2	0.2	1.5	1.3	1.2	0.8	1.4	1.9	0.3	-0.4	.....	-0.2	-0.2	1.9	1	-0.4	-0.4	1.5	0.3	1.9	-0.4
x_new_normalized	0	0	1	1	1	1	1	1	0	0		0	0	1	1	0	0	1	0	1	0

Fig. 5 Calculation of velocity and new particle position based on previous velocity, position, particles best and swarm best positions

3.2 Application of PSO for FCN compression

We propose to use Particle Swarm Optimization (PSO) [17] for FCN model compression by keeping the most dominant neurons and discarding the redundant neurons at each hidden layer.

In our case we assume that a unit vector x represents the nodes/filters at any particular layer. To start with, this is converted into a binary vector with some of the elements being randomly assigned 0 value, suggesting that the corresponding nodes/filters are to be dropped from this layer.

A concatenation of particles vector at each layer of FCN is shown in Fig. 3. The initialization of particles is carried out in algorithm 2. The formation of new particle velocity using original particle velocity (inertial component), particle best and swarm best is shown in Fig. 4. The calculation for new particle position is shown in Fig. 5. After adjusting particle velocities the new particle position results in real values between 0 and 1 for each particle position. Since the aim is to get binary values to decide whether a specific node/filter is to be retained, the values are converted into binary values 0 and 1 by rounding off the real values. This is shown in algorithm 3.

Algorithm 2 Generation of particle position

```
Input : k                                              ▷ Layers in model
Output : A                                             ▷ Matrix of list of vectors
procedure InitParticles(k)                             ▷ Select Initial particle position vectors of PSO
                                                       ▷ Initialize set of initial position vectors to empty list
    A ← {};
                                                       ▷ Generate say 15 particles matching minimum accuracy criteria
    int B[15][n_filters]
    for i ← 0 to 15 do
                ▷ Generate a random vector of length equal to number of filters in k^th layer
        for j ← 0 to n_filters do
            B[i][j] = random()%2                       ▷ random function generates a random positive number
        A ← A ∪ B[i]
    return A
```

Algorithm 3 Ensuring new Particle position is having proper bounds(0-1)

```
Input : xin                                            ▷ Particle Positon vector with real values
Output : xout                                          ▷ Particle position vector having 0-1 values
procedure Ensure_Bounds(xin)                           ▷ Make Particle position in required bounds
                                                       ▷ Initially set output vector to empty list of xin length
    int xout[length(xin)];
    for index in range(0, length(xin)) do
        if xin[index] >= 0.5 then
            xout[index] ← 1
        else
            xout[index] ← 0
    return xout
```

3.3 Selection in PSO for FCN compression

The selection criteria uses a combination of retained neurons fraction γ_j at layer j and mIoU performance metric ζ at any compression step. ζ is calculated using pixel-wise accuracy of test dataset after removal of certain neurons based on the particle vector 0 elements. The fitness function is created as below:

$$max\big(w \times (1-\gamma_j) + (1-w) \times \zeta\big) \text{ subject to } \gamma_j \leq 1 \qquad (3)$$

where w is the weight-age given to achieve a trade-off between dual objective of reducing neurons and maintaining performance metric of pixel-wise accuracy. Experiments were performed by varying value of w to achieve different compression and mIoU values. To maintain the model performance more weight-age was given to second term in fitness function by keeping value of w less than 0.3. The PSO compression steps are repeated unless performance was reduced by more than a predetermined threshold value.

Table 1 Parameters used for PSO Algorithm

S. no.	Parameter	Value
1	Social constant (c_1)	1
2	Cognitive constant (c_2)	2
3	Inertial component weight (w_k)	0.5
4	Number of particles	15
5	Maximum iterations	30

4 Experimental setup and results

NVIDIA DGX v100 machine was utilized to perform FCN training and compression experiments with python programming language. FCN was compressed using PSO algorithm and tested on three datasets namely: street scene images [1], lungs X-Ray dataset [2] and potato late blight leaf images from PlantVillage dataset [26]. Compression was done on FCN8 and FCN16 for street scene images and similarly FCN32 was compressed for leaf and lung images. The various parameters used in PSO algorithm are given in Table 1.

4.1 Compression of FCN8 on street scene images

The street scene images dataset is having 13 classes and after applying FCN8 an accuracy of 80.23% has been achieved with 524971 KB storage space. Moreover, we have applied proposed compression method on FCN8 and achieved compression of 95.52 times by compromising the mIoU $<2\%$. The stepwise compression statistics for FCN8

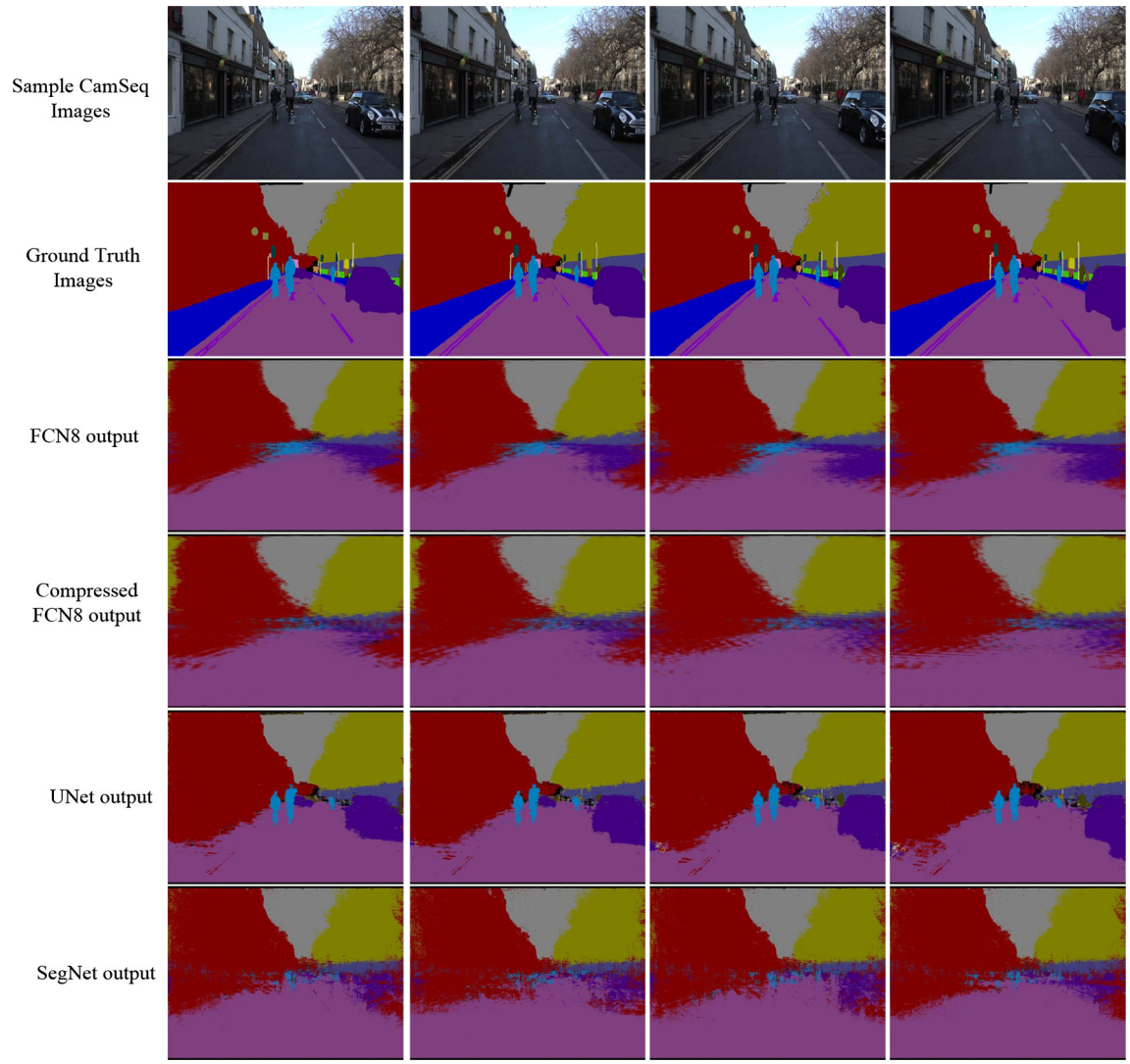

Fig. 6 Sample street scene images, their segmented ground truth images, generated outputs of FCN8, generated output of compressed FCN8, output of UNet and output of SegNet

Table 2 Performance and size statistics for different compression steps of FCN16 on street scene images

Compression step	mIoU (After compression) (%)	Compressed size (KB)	Times compression
1	70.63	171890	3.05×
2	68.90	58869	8.91×
3	69.77	22938	22.83×
4	70.16	9874	53.14×
5	70.12	4538	115.62×

Original FCN16 size: 524726 KB

is presented in Table 3, and predicted outputs are compared with ground truth images through visual representation in Fig. 6. Moreover, performance comparison among FCN8, compressed FCN8, UNet and SegNet is also depicted in Fig. 6. We have also applied proposed method on FCN16 for compression purpose and compression statistics is presented in Table 2.

4.2 Compression of FCN32 on leaf images

There has been considerable interest in plant disease classification using leaf images through deep learning techniques [28, 36, 39]. More recently, researchers have been trying to segment the leaf images in order to work out extent of diseases [19, 27, 30, 32, 55].

In our work, compressed version of FCN32 was applied on images of potato leaves affected by late blight disease, and performance was compared with various non-compressed Deep learning models. Sample leaf images are

Table 3 Performance and size statistics for different compression steps of FCN8 on street scene images

Compression step	mIoU (After compression)(%)	Compressed size (KB)	Times compression
Initial	80.23	524971	1×
1	78.10	173568	3.02×
2	77.98	65097	8.06×
3	78.30	25275	20.77×
4	78.70	11507	45.62×
5	78.75	5554	94.52×

shown in Fig. 7. Ground truth segmented images were created by using HSV transform of images and checking Hue value range for green, yellow and brown colors. The leaf green part was made in ground truth images, disease part which was yellow or brown was made brown and background was made black as shown in second row of Fig. 7. The third row of Fig. 7 shows the results of applying original uncompressed FCN32, the predicted

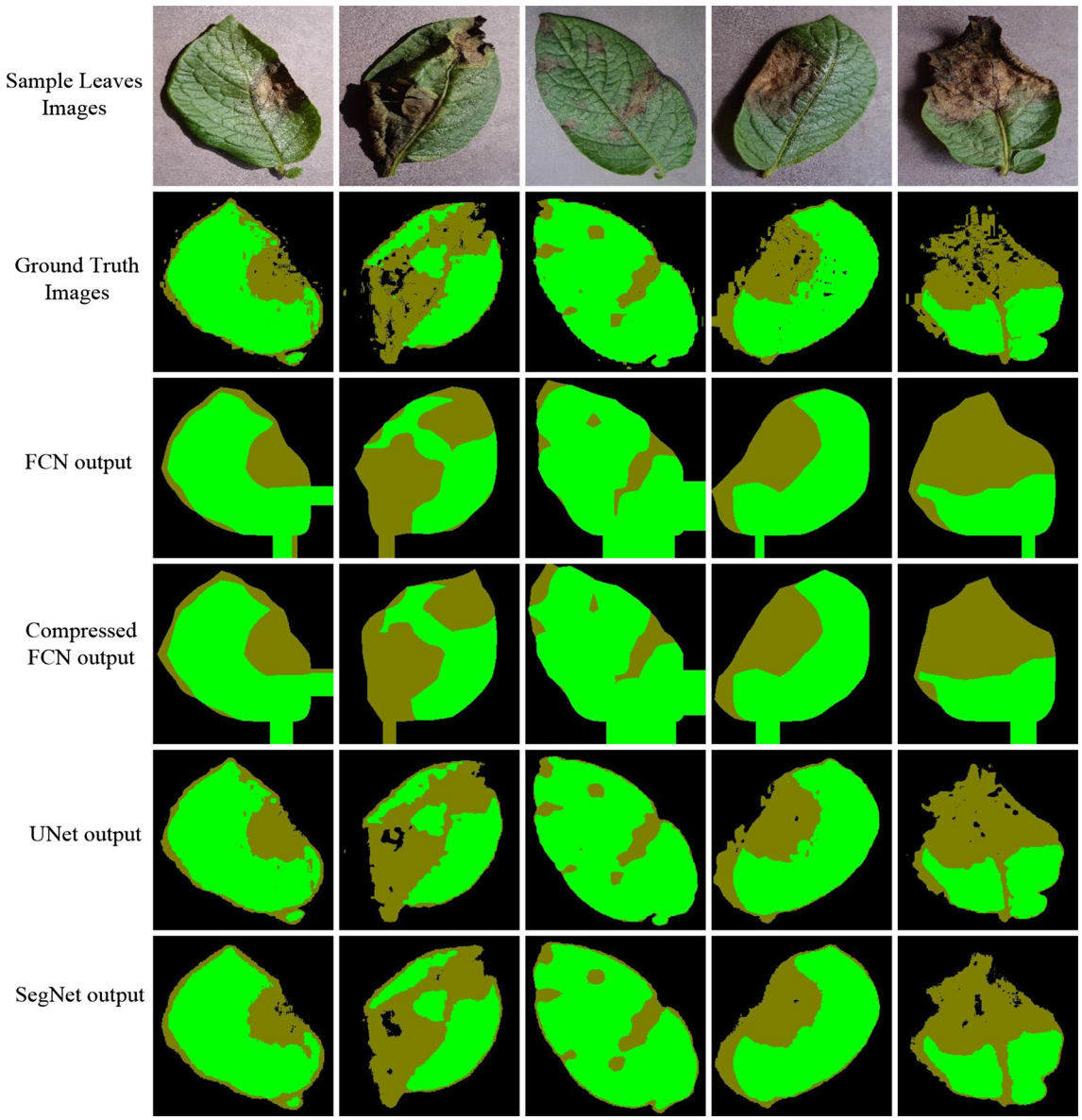

Fig. 7 Sample potato late blight leaf images from PlantVillage dataset, their segmented ground truth images, generated outputs of FCN32, generated output of compressed FCN32 and comparison with UNet and SegNet outputs

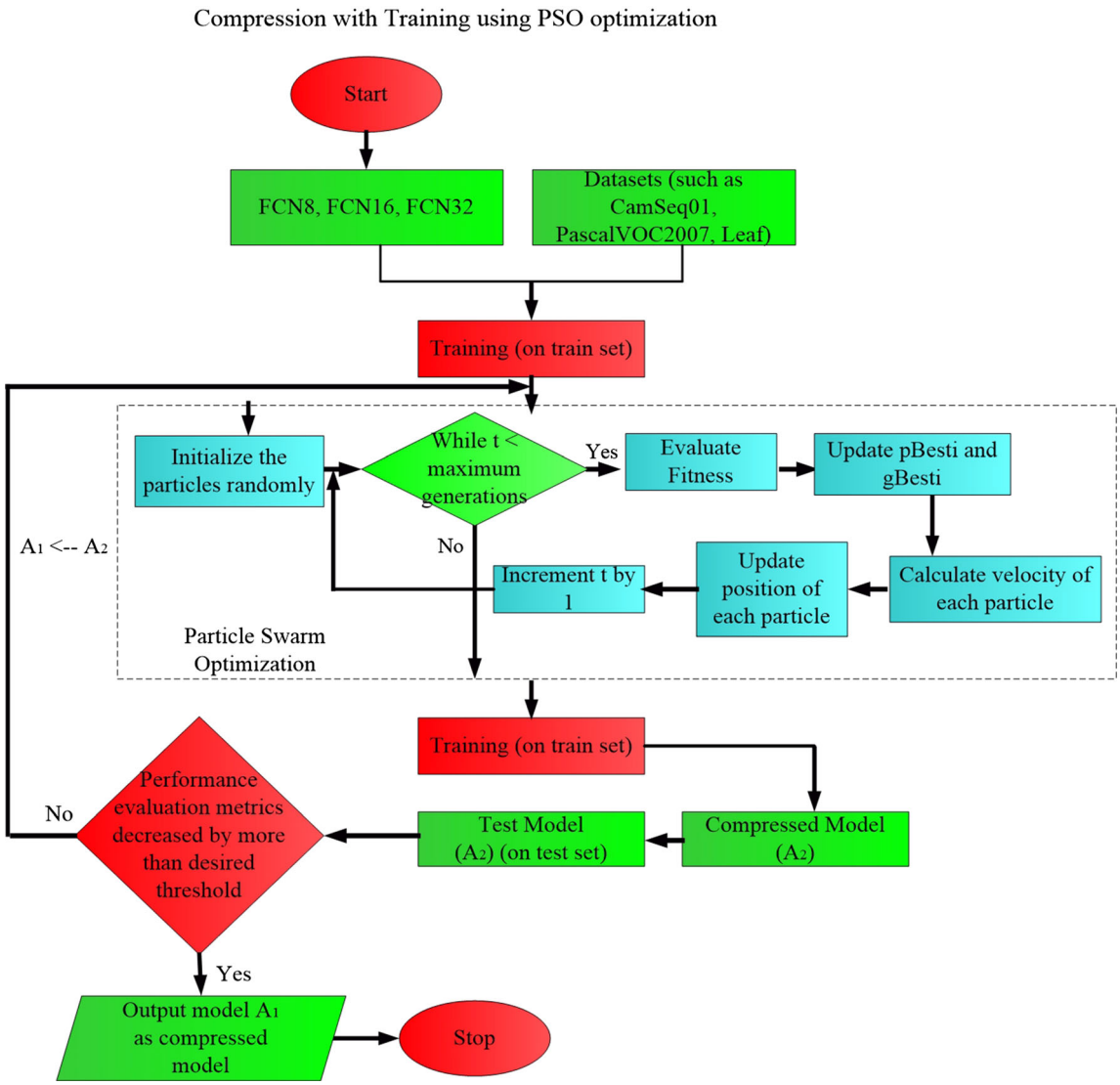

Fig. 8 Flowchart of FCN compression process

Table 4 Performance and size statistics for different compression steps of FCN32 on leaves dataset

Compression step	mIoU (After compression) (%)	Compressed size (KB)	Times compression
Initial	89.17	524566	1×
1	88.10	149520	3.50×
2	87.98	50656	10.35×
3	88.30	20286	25.85×
4	88.70	8900	58.94×
5	88.59	4051	129.49×
6	88.32	2043	256.76×
7	88.64	1064	493.01×
8	88.88	805	651.34×
9	88.70	672	780.60×
10	88.42	616	851.56×

Table 5 Comparison of FCN filter numbers during various compression iterations for leaves dataset

Layer	Filters	Step 1	Step 4	Step 7	Step 10
1	64	53	30	30	30
2	64	48	31	31	31
3	128	86	39	27	27
4	128	92	38	29	29
5	256	171	63	31	31
6	256	175	60	32	32
7	256	159	56	29	29
8	512	287	85	38	29
9	512	304	84	42	32
10	512	316	87	33	22
11	512	313	92	36	27
12	512	307	92	39	31
13	512	309	88	33	25
14	4096	2245	455	142	35
15	4096	2205	450	117	46
Size (KB)	524566	149520	8900	1064	616

output of compressed FCN32 is presented in fourth row. For sake of comparison predicted outputs of UNet and SegNet are presented in fifth and sixth rows. Visual comparison shows that segmentation by compressed FCN32 is as good as by other architectures.

The original FCN32 model needed 524,566 KB storage space and resulted in 89.17% mIoU. It was observed that there was a drastic drop in storage space required as the model went through various PSO compression steps. Figure 8 shows stepwise application of PSO algorithm to compress FCN model iteratively till performance is within desired range.

Table 4 summarizes the achieved mIoU and required storage at each compression step. It is also observed from Table 4 that after 10 steps, size of FCN32 was reduced to 616 KB with hardly any degradation in mIoU. The visual comparison of ground truth and predicted output is depicted in Fig. 7.

The results of FCN32 and compressed FCN32 are similar to other segmentation methods such as UNet and SegNet. As far as comparison of storage space was concerned it was found that UNet needs 121,335 KB and SegNet needs 115,283 KB whereas compressed FCN32 needs only 616 KB and attain very close mIoU performance.

A comparison of number of convolution filters during steps 1, 4, 7, and 10 is provided in Table 5. However, a condition was added to stop compressing a layer further if its filter reduces to less than or equal to 32 to keep enough filters in a layer to pass the features. Moreover, after compression of FCN32 model the number of neurons reduces considerably leading to reduction in mathematical operations needed, which accelerated the inference time by 1.68 times on test dataset.

4.3 Compression of FCN32 on medical dataset of lungs X-rays

The Lungs X-Ray dataset [2] has two classes. On training FCN32 on lungs dataset the storage space needed was 524,550 KB and mIoU was 97.12%. After applying proposed compression method, the size and mIoU in each compression step is presented in Table 6. The predicted outputs are compared with ground truth images through visual representation in Fig. 9. As seen from Fig. 9, even after compressing the model by 808.24 times the segmentation results were similar to original FCN32 model. The results of FCN32 and compressed FCN32 on this dataset are similar to other segmentation methods such as UNet and SegNet.

Table 6 Comparison of performance and size for different compression steps of compressing FCN8 on lungs X-Ray dataset

Compression step	mIoU (After compression) (%)	Compressed size (KB)	Times compression
Initial	97.12	524550	1×
1	96.79	173569	3.02×
2	96.89	63434	8.27×
3	97.05	25943	20.21×
4	97.03	10541	49.76×
5	96.97	4978	105.37×
6	96.85	2343	223.87×
7	96.84	1499	349.93×
8	96.78	999	525.07×
9	96.70	804	652.42×
10	96.85	726	722.52×
11	96.95	649	808.24×

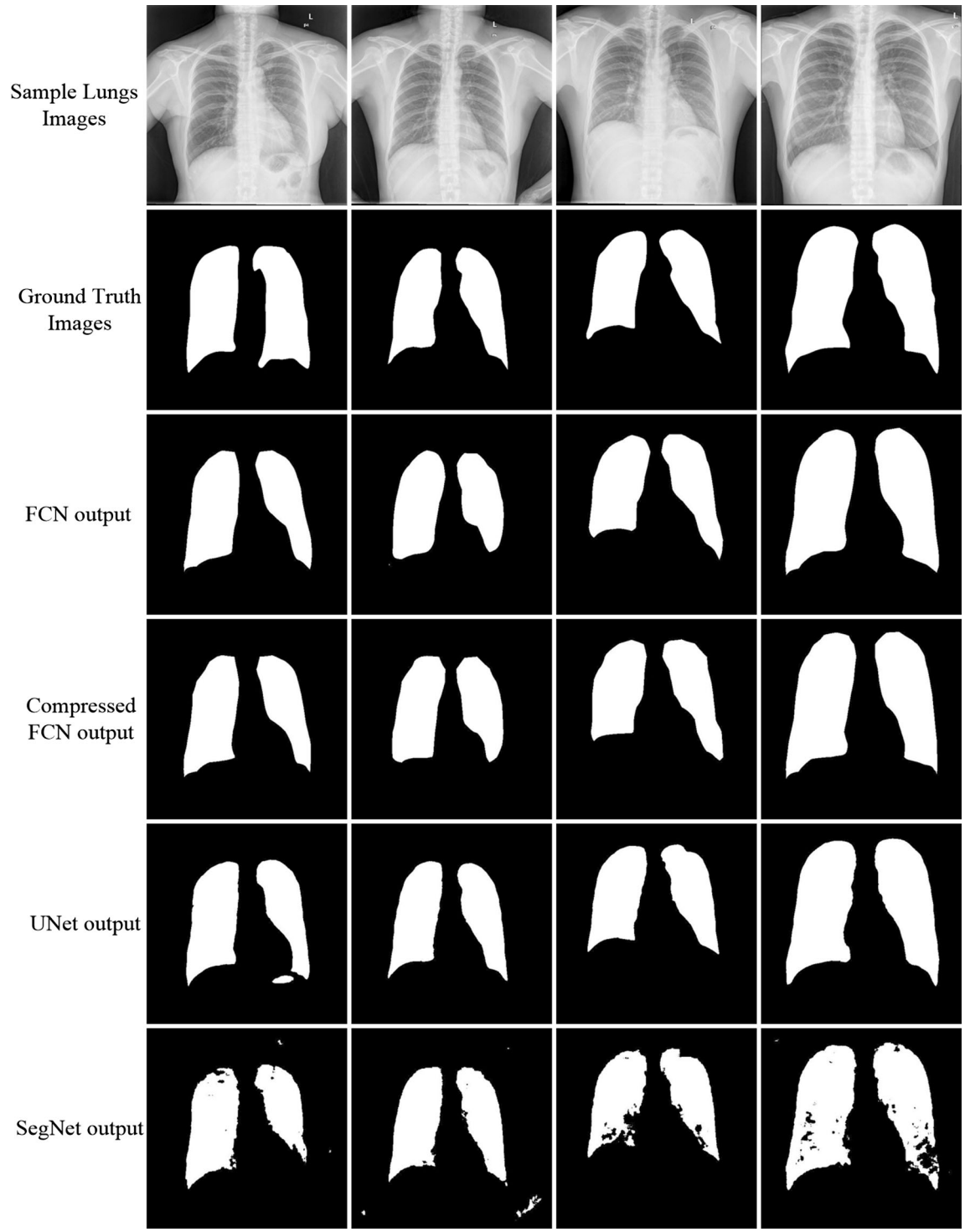

Fig. 9 Sample images of lungs X-Ray dataset, ground truth annotations of same images, predicted outputs of FCN32, predicted output of compressed FCN32 and comparison with UNet and SegNet outputs

5 Actual Implementation

Encouraged by the success of PSO-based model compression of FCN architecture, it was decided to try it out on an edge computing device such as an iPhone. The compressed model was ported on an iPhone to see its performance on the leaf dataset. Figure 10 shows a sample input image, python code and output image as seen on the iPhone.

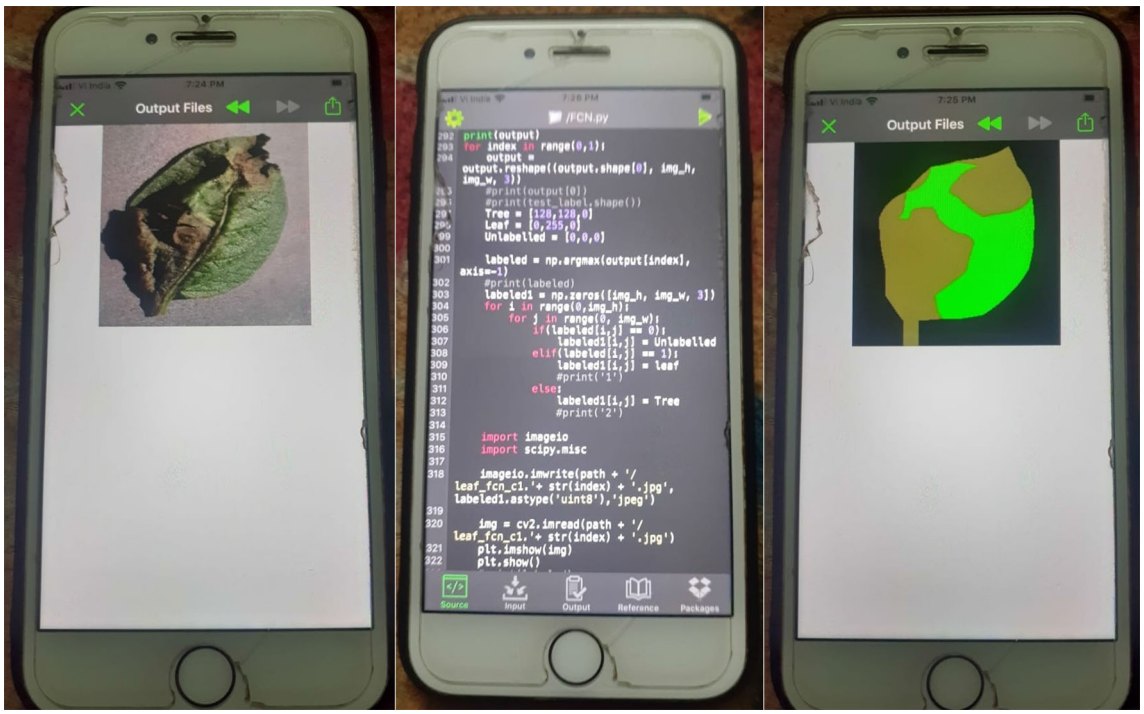

Fig. 10 Sample input image, compressed FCN code and output image on an iPhone

6 Conclusion

This paper has proposed a methodology for compressing FCN architecture using Particle Swarm Optimization, so that it can be deployed on edge devices. The performance of compressed model has been tested on a number of publicly available datasets and results compared with uncompressed models. It has been shown on leaves dataset that less storage of space of around 851.56 times is required along with acceleration of model by 1.68 times. Similarly compression of 94.52 times is achieved on street scene images without loss in performance metric. Compression on medical dataset of lungs images, the achieved compression is 808.24 times with minimal change in mIoU.

This research can be further extended by using different meta-heuristic-based approaches such as Differential Evolution and Whale Optimization to figure out the most suitable method with higher compression but maintaining same level of accuracy. This research can be used to develop IoT-based robotic devices to deploy FCN for automobiles, agricultural or medical needs for semantic segmentation of needed image portions at run time. Finally this research can be extended to other models such as SegNet, UNet, YOLO, VNet, and WNet.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Street scene images dataset (2007) http://mi.eng.cam.ac.uk/research/projects/ VideoRec/CamSeq01/
2. Lungs x-ray dataset (2018) https://www.kaggle.com/eduardomineo/ lung-segmentation-of-rsna-challenge-data/data
3. Agarwal M, Gupta S, Biswas K (2020) A new conv2d model with modified relu activation function for identification of disease type and severity in cucumber plant. Sustain Comput Informat Syst 30:100473
4. Agarwal M, Gupta SK, Biswas K (2020) Development of efficient CNN model for tomato crop disease identification. Sustain Comput Inform Syst 28:100407
5. Alqazzaz S, Sun X, Yang X, Nokes L (2019) Automated brain tumor segmentation on multi-modal MR image using segnet. Comput Vis Media 5(2):209–219
6. Anwar S, Hwang K, Sung W (2017) Structured pruning of deep convolutional neural networks. ACM J Emerg Technol Comput Syst (JETC) 13(3):1–18
7. Badrinarayanan V, Kendall A, Cipolla R (2017) Segnet: a deep

convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell 39(12):2481–2495
8. Beheshti N, Johnsson L (2020) Squeeze u-net: a memory and energy efficient image segmentation network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 364–365
9. Bi L, Feng D, Kim J (2018) Dual-path adversarial learning for fully convolutional network (fcn)-based medical image segmentation. Vis Comput 34(6):1043–1052
10. Brostow GJ, Fauqueur J, Cipolla R (2009) Semantic object classes in video: a high-definition ground truth database. Pattern Recogn Lett 30(2):88–97
11. Calisto MB, Lai-Yuen SK (2020) Adaen-net: an ensemble of adaptive 2D–3D fully convolutional networks for medical image segmentation. Neural Netw 126:76–94
12. Chakraborty UK (2008) Advances in differential evolution. Springer, Heidelberg
13. Chen H, Wang Y, Shu H, Tang Y, Xu C, Shi B, Xu C, Tian Q, Xu C (2020) Frequency domain compact 3d convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1641–1650
14. Chen T, Cai Z, Zhao X, Chen C, Liang X, Zou T, Wang P (2020) Pavement crack detection and recognition using the architecture of segnet. J Ind Inform Integr 18:100144
15. Cheng Y, Wang D, Zhou P, Zhang T (2017) A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282
16. Choudhary T, Mishra V, Goswami A, Sarangapani J (2020) A comprehensive survey on model compression and acceleration. Art Intell Rev 53:5113–5155
17. Eberhart R, Kennedy J (1995) Particle Swarm Optimization. In: Proceedings of the IEEE international conference on neural networks, vol 4, pp 1942–1948
18. Feoktistov V (2006) Differential evolution. Springer, New York
19. Ganesh P, Volle K, Burks T, Mehta S (2019) Deep orange: mask r-CNN based orange detection and segmentation. IFAC-PapersOnLine 52(30):70–75
20. Gong Y, Liu L, Yang M, Bourdev L (2014) Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115
21. Guo D, Zhu L, Lu Y, Yu H, Wang S (2018) Small object sensitive segmentation of urban street scene with spatial adjacency between object classes. IEEE Trans Image Process 28(6):2643–2653
22. Han S, Mao H, Dally WJ (2015) Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149
23. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. Adv Neural Inform Process Syst 28:1135–1143
24. He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE international conference on computer vision, pp 1389–1397
25. Holliday A, Barekatain M, Laurmaa J, Kandaswamy C, Prendinger H (2017) Speedup of deep learning ensembles for semantic segmentation using a model compression technique. Comput Vis Image Underst 164:16–26
26. Hughes D, Salathé M, et al (2015) An open access repository of images on plant health to enable the development of mobile disease diagnostics. arXiv preprint arXiv:1511.08060
27. Islam M, Dinh A, Wahid K, Bhowmik P (2017) Detection of potato diseases using image segmentation and multiclass support vector machine. In: 2017 IEEE 30th Canadian conference on electrical and computer engineering (CCECE), IEEE, pp 1–4
28. Johannes A, Picon A, Alvarez-Gila A, Echazarra J, Rodriguez-Vaamonde S, Navajas AD, Ortiz-Barredo A (2017) Automatic plant disease diagnosis using mobile capture devices, applied on a wheat use case. Comput Electron Agric 138:200–209
29. Kaymak R, Kaymak C, Ucar A (2020) Skin lesion segmentation using fully convolutional networks: a comparative experimental study. Expert Syst Appl 161:113742
30. Lee U, Chang S, Putra GA, Kim H, Kim DH (2018) An automated, high-throughput plant phenotyping system using machine learning-based plant segmentation and image analysis. PLoS One 13(4):e0196615
31. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710
32. Lin K, Gong L, Huang Y, Liu C, Pan J (2019) Deep learning-based segmentation and quantification of cucumber powdery mildew using convolutional neural network. Front Plant Sci 10:155
33. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In: Proceedings of the IEEE international conference on computer vision, pp 2736–2744
34. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
35. Luo JH, Wu J, Lin W (2017) Thinet: a filter level pruning method for deep neural network compression. In: Proceedings of the IEEE international conference on computer vision, pp 5058–5066
36. Ma J, Du K, Zheng F, Zhang L, Gong Z, Sun Z (2018) A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network. Comput Electron Agric 154:18–24
37. Manickam R, Rajan SK, Subramanian C, Xavi A, Eanoch GJ, Yesudhas HR (2020) Person identification with aerial imaginary using segnet based semantic segmentation. Earth Sci Inform 13:1393
38. Mohan P, Paul A.J, Chirania A (2021) A tiny CNN architecture for medical face mask detection for resource-constrained endpoints. In: Innovations in Electrical and Electronic Engineering, Springer, pp 657–670
39. Mohanty SP, Hughes DP, Salathé M (2016) Using deep learning for image-based plant disease detection. Front Plant Sci 7:1419
40. Nguyen HD, Na IS, Kim SH (2019) Hand segmentation and fingertip tracking from depth camera images using deep convolutional neural network and multi-task segnet. arXiv preprint arXiv:1901.03465
41. Nguyen K, Fookes C, Sridharan S (2020) Context from within: Hierarchical context modeling for semantic segmentation. Pattern Recogn 105:107358
42. Ping-Rong C, Hang HM, Sheng-Wei C, Lin JJ (2020) Dsnet: an efficient CNN for road scene segmentation. APSIPA Trans Signal Inform Process 9:e27
43. Rahnamayan S, Tizhoosh HR, Salama MM (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12(1):64–79
44. Ronneberger O, Fischer P, Brox T (2015) U-net: convolutional networks for biomedical image segmentation. In: International conference on Medical image computing and computer-assisted intervention, Springer, pp 234–241
45. Roth HR, Oda H, Zhou X, Shimizu N, Yang Y, Hayashi Y, Oda M, Fujiwara M, Misawa K, Mori K (2018) An application of cascaded 3D fully convolutional networks for medical image segmentation. Comput Med Imag Graph 66:90–99
46. Samala RK, Chan HP, Hadjiiski LM, Helvie MA, Richter C, Cha K (2018) Evolutionary pruning of transfer learned deep convolutional neural network for breast cancer diagnosis in digital breast tomosynthesis. Phys Med Biol 63(9):095005

47. Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: 1998 IEEE international conference on evolutionary computation proceedings. In IEEE world congress on computational intelligence (Cat. No. 98TH8360), IEEE, pp 69–73
48. Skandha SS, Agarwal M, Utkarsh K, Gupta SK, Koppula VK, Suri JS (2022) A novel genetic algorithm-based approach for compression and acceleration of deep learning convolution neural network: an application in computer tomography lung cancer data. Neural Comput Appl 34(23):20915–20937
49. Song S, Lichtenberg SP, Xiao J (2015) Sun rgb-d: a rgb-d scene understanding benchmark suite. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 567–576
50. Storn R, Price K (1997) Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
51. Wang Z, Li F, Shi G, Xie X, Wang F (2020) Network pruning using sparse learning and genetic algorithm. Neurocomputing 404:247–256
52. Yang C, An Z, Li C, Diao B, Xu Y (2019) Multi-objective pruning for CNNS using genetic algorithm. In: International conference on artificial neural networks, Springer, pp 299–305
53. Yar H, Hussain T, Agarwal M, Khan ZA, Gupta SK, Baik SW (2022) Optimized dual fire attention network and medium-scale fire classification benchmark. IEEE Trans Image Process 31:6331–6343
54. Zhang Q, Zhang M, Chen T, Sun Z, Ma Y, Yu B (2019) Recent advances in convolutional neural network acceleration. Neurocomputing 323:37–51
55. Zhou J, Fu X, Zhou S, Zhou J, Ye H, Nguyen HT (2019) Automated segmentation of soybean plants from 3D point cloud using machine learning. Comput Electron Agric 162:143–153