Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079831

ARTICLE

DSSeg-FLHA: A Decentralized Secure Self-Adapting Image Segmentation Framework Using Federated Learning and Hybrid Architectures

Rifat Sarker Aoyon[1], Fahmid Al Farid[2,3], Ismail Hossain[4], Mahe Zabin[5], Sarina Mansor[2,*] and Jia Uddin[6,*]

[1]Department of Computer Engineering, Chosun University, Gwangju, Republic of Korea
[2]Centre for Image and Vision Computing (CIVC), COE for Artificial Intelligence, Faculty of Artificial Intelligence and Engineering (FAIE), Multimedia University, Cyberjaya, Malaysia
[3]Faculty of Computer Science and Informatics, Berlin School of Business and Innovation, Berlin, Germany
[4]Department of Computer Science, University of Denver, Denver, CO, USA
[5]Human and Digital Interface Department, JW Kim College of Future Studies, Woosong University, Daejeon, Republic of Korea
[6]AI and Big Data Department, Woosong University, Daejeon, Republic of Korea

*Corresponding Authors: Sarina Mansor. Email: sarina.mansor@mmu.edu.my; Jia Uddin. Email: jia.uddin@wsu.ac.kr

Received: 29 January 2026; Accepted: 14 April 2026; Published: 15 June 2026

ABSTRACT: This research introduces an innovative lightweight image segmentation framework where models of hybrid architectures work together to predict the output and also have self-adapting ability, along with maintaining data privacy. In this framework, data is distributed and trained in a decentralized way using different deep learning architectures. That is how the advantages of all these models will be integrated into the system. Each trained model makes its own prediction, and the final output is determined through cooperation among these models. Here, the confidence-level and pixel-wise voting majority algorithms will be utilized for the co-operation-based output prediction system. Due to the efficient setup of the operations of these two algorithms, each input will get its accurate output. Additionally, the federated learning-based self-adapting feature facilitated the proposed framework for advancing its performance consistently by interacting with the inputs. Here, UNet, SegNet and FCNN models have been trained and integrated into the prediction framework. Here, the Oxford-IIT pet dataset was used. And all the data of this dataset is distributed among these three models. The framework's effectiveness was measured using metrics like average pixel accuracy, IoU, F1 score, precision, and recall, which resulted in scores of 89.26%, 71.48%, 81.29%, 83.96% and 81.16%, respectively. Another notable feature of this proposed framework is allocating comparatively fewer computational resources and taking less time. To validate these claims, the proposed system is compared with three other state-of-the-art models, and the proposed system delivered superior performance among all.

KEYWORDS: Image segmentation; decentralized; federated learning; UNet; SegNet; FCNN; data privacy; hybrid architecture; self-adapting

1 Introduction

Deep learning-based image segmentation is playing a crucial role nowadays to assist human beings by reducing the human workload and dependencies [1]. In image segmentation, an image is divided into multiple segments or regions, each corresponding to distinct features or objects within the image [2]. Due to this reason, it is feasible to get a more detailed, pixel-level understanding of an image, unlike many

other image processing methods. And to execute image segmentation tasks appropriately, deep learning-based methods outperform classical image segmentation methods like thresholding-based methods or edge-based methods. However, these classical techniques are well-suited to simpler images, smaller datasets, or computational constraints while being inappropriate to deploy in complex tasks that involve intricate, non-linear relationships between pixels. In such cases, deep learning is being widely used due to its ability to fetch complex patterns of the image [3]. As a result, deep learning-based methods are being widely used currently.

Nowadays, numerous deep-learning-based methods are used for image segmentation tasks. UNet, SegNet, FCNN, etc., are some of the most popular image segmentation methods in this decade. Long et al. have presented another image segmentation method using Fully Convolutional Network (FCN) [4]. The author makes the performance of FCN faster by processing the entire image in a single forward pass, unlike traditional patch-by-patch methods. Here, the authors reduced the computational demands but failed to maintain the confidentiality of data.

That is why we have unveiled an innovative framework that erases the need to manage powerful computers alongside privacy-preserving and self-adapting capability. To fulfil this motive, we decided to use ensemble methods where the entire data of the dataset will be trained separately instead of together and merge all these trained models afterwards to the prediction system. Here, it is not like another prediction system where output for an input is predicted from a single trained model. Instead, when any input is delivered to the prediction system, each trained model will predict the confidence level of the output. And based on the confidence level, the participants for the ultimate output prediction will be determined. In the proposed framework, we do not rely on just pixel-wise majority voting [5]. Because, in that case, there are some issues that can arise. When an input is given to the prediction system, if just one model of the system is familiar with this data and the rest are not, then the familiar model may provide accurate output, and the others may not deliver correct output. And in such a situation, incorrect output gets most of the votes from all models. To illustrate this with a real-life example: Imagine seven people are asked, “What is the capital of France?” Two of them know that Paris is the correct answer, while one person says London, and the other four say New York. If we simply go with majority voting, New York would be chosen, which is clearly wrong. However, if we first filter out participants based on their knowledge level, we would end up with just the two people who know the correct answer. When they both vote, the answer would be Paris, and Paris would be selected as the correct answer. Such a concept is followed in the proposed system, where the first model will be shortlisted based on the confidence level of their prediction. Afterwards, majority voting will be conducted among these shortlisted models. That is how the correct answer will be selected via collaboration among the models. The confidence-based shortlisting system minimizes the influence of models with poor knowledge and ensures the dominance of the most knowledgeable models to contribute to the final decision. By only considering confident models, the algorithm can optimize computational resources and make the process faster. In this framework, confidence-level filtering is applied before the pixel-wise majority voting process. This preliminary filtering step removes predictions with low confidence that may introduce uncertainty or noise. As a result, majority voting is performed only among models that have already shown reliable confidence scores. Since low-confidence models are excluded, the chance of conflict between confidence filtering and majority voting becomes small. In practice, this approach allows the voting process to occur only among a few reliable models, which helps produce more stable and trustworthy final predictions.

For an image input, the trained model which generates the most ultimate pixels, this input and the ultimate output will be saved into the database of this certain model architecture. After getting sufficient data in the database of this certain model, then the training will occur separately with these collected data using this specific architecture. Following the training process, this trained model will be aggregated with the same existing architecture of this prediction system using a federated learning algorithm. Such an architecture selection method leads to better scalability, faster adaptation, and ultimately, higher performance. Each deep learning has its own characteristics and strengths that make it better suited for specific types of segmentation tasks. When the data is directed to the model that performs best on it, the system leverages the architecture's strengths, optimizing its learning process. That is how the proposed framework will advance its performance continuously by interacting with input. We named this advancement feature "Self-Adapting". Overall, by following the proposed algorithm, it will be feasible to train a model with a low-powered machine alongside privacy-preserving and self-adapting ability. Besides, another unique task can be executed by the proposed method. Here, it is not required to have a model of the same architecture in the prediction system. Here, three different architecture models are used at the prediction system. And three models are UNet, SegNet and FCNN. In most of the ensemble learning frameworks, the same type of architecture is used for the prediction system [6]. But all the output is not in the same category, and each architecture has different uniqueness. For some cases, UNet might perform better; in others, SegNet or FCNN might be more accurate. This flexibility allows the models to work together, providing the best possible output for each situation. The proposed framework is considered lightweight. Unlike conventional approaches, where all data are trained together in a centralized system, the proposed method trains the data separately on different machines. This distributed training reduces the computational load on a single machine. In addition, the framework uses heterogeneous neural networks such as U-Net, SegNet, and Fully Convolutional Network. Instead of using the same model for all types of data, different models can handle different data characteristics. Here, it is possible to use simpler models to process simpler data and avoid unnecessary use of complex models. That is how the overall computational cost can be reduced.

1.1 Research Objective

The major objective of this research is to implement a framework that simultaneously ensures confidentiality of users' data and improves its performance by consistently interacting with the users and models of different architectures work together to make predictions for the users' input. Besides, in our proposed framework, there is the feasibility to integrate new types of datasets anytime without retraining the entire model from scratch. Instead, you just need to train the data of the new variety. To fulfil these objectives, all the data will be trained separately with three different types of model architecture: UNet, SegNet and FCNN. Following that, all trained models are added to the proposed framework. Here, instead of data, the trained model will be sent to the framework. So, here now we will not access the entire data. Because here each piece of data will be trained separately in a decentralized way. That is how the data will stay secure. The ultimate output for an input will be predicted mainly in two phases. At first, each trained model will predict the output. Along with the output, the confidence of the output will also be calculated for each model. If any model's confidence score is not satisfactory, then this model will not be able to qualify for the ultimate pixels prediction. All the models with satisfactory confidence scores will be selected to participate in pixel-wise voting majority. Eventually, the final output will be selected based on a pixel-wise voting majority. Afterwards, the ultimate output and input will be saved in the database. When the amount of data is sufficient, then these data will be trained and merged with the existing models via the federated learning algorithm. In federated learning, instead of the actual data, the training weight is sent to the central place, which is called the global model. Thus, data security will be ensured via federated learning. Even when new varieties of data arrive,

it will also be feasible to include them in the system without retraining from scratch. That is how we have invented a novel framework of image segmentation using federated learning and multi-architecture models integration with privacy-preserving and self-learning ability.

1.2 Problem Statement

Even after having lots of advantages, some challenges can still arise in the case of deep learning [7]. Deep learning is based on complex neural networks. Therefore, it is required to manage high computational demand to train the model via deep learning. Another important issue is the requirement of a vast amount of data that may not be collected instantly. Along with these drawbacks, there are some instances where people will not be interested in sharing their personal data with anyone [8]. To mitigate these issues, this research has been conducted where there is no need to manage powerful machines. Besides, it is not urgently needed to collect a large amount of data, while the amount of data will be consistently increased via interacting with the users. Moreover, new types of data can be easily integrated into the system without retraining the entire system. Unlike typical methods, the data will be trained decentrally, and newly added data are included in the system using federated learning. So, here such powerful machines are not needed that are needed in centralized training. Because here, instead of training all data together, the entire data will be divided into a few segments, and each segment will be trained separately. In this case, the requirement for less data is a less powerful machine. That is how these challenges are wiped out via the proposed framework.

1.3 Contributions

Motivated by the earlier work, Huang et al. have invented a unique federated learning algorithm where a local model improves its performance during training by getting help from other models [9]. Unlike a typical federated learning model, the low-accuracy model cannot affect the performance of the global model. Here, each local model gets efficient training by collaborating with the others. By getting knowledge from the above-mentioned research, this image segmentation research is conducted where models of different architectures will work together and be trained in a decentralized manner to ensure data security and model scalability. Below, several major contributions of this research have been provided:

- This paper presents a decentralized training framework involving multiple heterogeneous deep learning models. Each model is trained independently on distributed data and collaboratively contributes to the final prediction. This decentralized approach ensures data security. Besides, this framework enables each model to leverage its strengths while mitigating its limitations.
- Unlike centralized training methods that require aggregating all data, the proposed decentralized approach enables parallel and independent training. Thus, training time is reduced significantly. Experimental results show that the proposed method requires only 24.66 min for training, which is considerably lower than centralized state-of-the-art models. Additionally, the framework is energy-efficient, consuming only 0.0068 kWh during training, which is substantially less than other approaches.
- The framework introduces a unique collaborative prediction mechanism. Here, only models with high confidence are selected for the voting process. Initially, models are shortlisted based on their confidence levels. Following that, only the qualified models participate in majority voting. So, the prediction system achieves an average pixel accuracy of 89.26%, an average IoU of 71.48%, and average F1-score, precision, and recall of 81.29%, 83.96%, and 81.16%, respectively.

- The proposed framework offers flexible scalability, where no additional computations or complex aggregation mechanisms are needed to integrate newly trained models. Any trained model can be seamlessly added to the prediction system. Experimental results demonstrate that the framework with three models outperforms configurations with two or one model in all evaluation metrics. This is justified that performance consistently improves with the integration of additional models.
- Another significant advantage of the framework is its automated self-adaptive capability. The system is capable of continuously learning by interacting with the inputs. These learnings can be aggregated to the system using the federated learning algorithm. This enables continuous data growth, automatic retraining, and seamless model integration, ensuring long-term adaptability and robustness.

1.4 Outline

The Section 2 outlines the foundation of this research paper. On the other hand, the Section 3 provides the explanations of the steps involved in constructing and implementing the framework. And the Section 4 has presented the effectiveness of the proposed framework. Finally, the Section 5 summarises the findings of this paper.

2 Background Study

In this section, an overview of key components has been provided. The methods that were used to fulfil this research, along with the dataset description, are explained here.

2.1 Federated Learning

Federated learning is a framework where training the deep learning algorithms is decentralized to ensure data privacy [10]. Unlike typical deep learning training approaches, the entire dataset will not be trained centrally in the federated learning framework. Instead, all the data is divided into several segments, and each segment is trained separately. Following that, instead of training data, the training weight of each model is passed to a central place. And the training weight is averaged afterwards. In the model where data is trained, it is called the local model. And the central place where all the training weights gather and an average of these weights is calculated is called the global model. After calculating the average training weight, the accuracy and loss of the global model are measured. If accuracy and loss are not satisfactory, then this process will be done once again. And this loop will continue until getting satisfactory results. Here, each loop is called a communication round. Along with ensuring the data security, it is bandwidth efficient. Fig. 1 shows the overview of federated learning.

2.2 UNet

UNet is a convolutional neural network (CNN)-based U-shaped architecture that is designed for image segmentation [11]. UNet consists mainly of two parts. These are an encoder and a decoder. Both the encoder and decoder have equal amounts of blocks. Each encoder block connects with its corresponding decoder block via a skip connection. Such a connection is required to preserve high-resolution spatial details from the encoder. Because, in the deeper encoder block, the block loses the fine details of the previous blocks. Therefore, skip connections are required. Deeper encoder layers capture semantic information. So, the deep layer can fetch what objects are and their relationships. But the image quality decreases. That means spatial details like edges and small shapes get blurred or lost. And the deepest layer of UNet is called the bottleneck. This part captures the most abstract, semantic features of the image. And upsampling is beginning from this level. Unlike the encoder level, the fine details of the image become richer from the lower to the upper level. Because of skip connections, each decoder block receives the fine details from the corresponding encoder

block and combines these with the semantic information from the deeper layers. As a result, the network can produce outputs that are both semantically meaningful and rich in spatial detail. The top level gets the actual image size of the input and produces the final detailed segmentation of the original image.

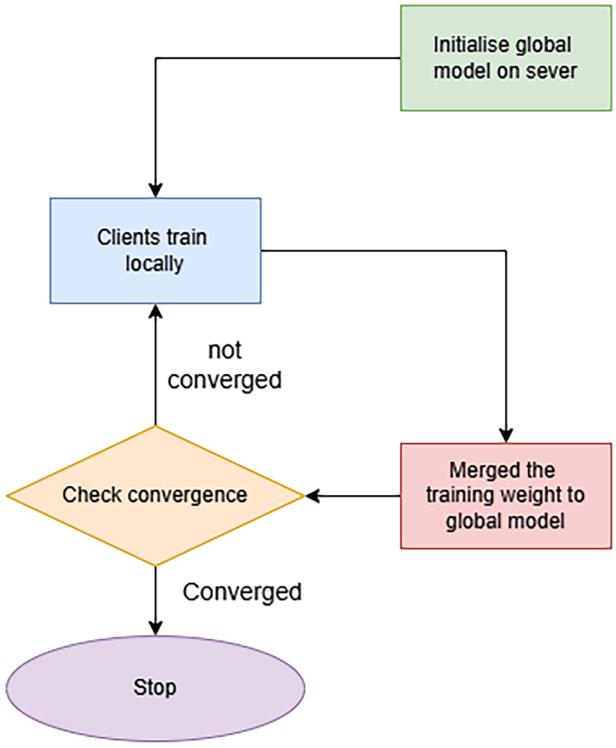

Figure 1: Overview of federated learning.

Each encoder block applies two convolutions with activation, as shown in (1).

$$F^{(l)} = \sigma\left(W_2^{(l)} * \sigma\left(W_1^{(l)} * F^{(l-1)} + b_1^{(l)}\right) + b_2^{(l)}\right) \tag{1}$$

$F^{(l-1)}$ denotes the input feature map to the l-th convolution, $W^{(l)}$ and $b^{(l)}$ represent convolution weights and bias, $*$ denotes the convolution operation, σ is the activation function (ReLU), and $F^{(l)}$ is the output feature map. Using Eq. (2), the encoder feature map is down sampled using max pooling to reduce the image size while preserving the most important features. If the encoder has L blocks, it can be expressed as shown in Eq. (3). In this equation, $F^{(l-1)}$ represents the input feature map to the l-th encoder block, and $F^{(l)}$ is the resulting output feature map after applying the operations defined in the encoder block.

$$F_{\text{pooled}}^{(l)} = \text{MaxPool}\left(F^{(l)}\right) \tag{2}$$

$$F^{(l)} = \text{EncoderBlock}^{(l)}\left(F^{(l-1)}\right), \quad l = 1, 2, \ldots, L \tag{3}$$

$$F^{(L)} = \sigma\left(W_2^{(L)} * \sigma\left(W_1^{(L)} * F^{(L-1)} + b_1^{(L)}\right) + b_2^{(L)}\right) \tag{4}$$

Eq. (4) indicates the bottleneck stage of the UNet architecture. The input feature map $F^{(L-1)}$ is passed through two convolution layers. Here, weights are $W_1^{(L)}$ and $W_2^{(L)}$ and biases are $b_1^{(L)}$ and $b_2^{(L)}$. And the activation function is σ (ReLU). The output feature map $F^{(L)}$ contains the most important and high-level information about the image at a lower spatial resolution and is used as the starting point for the decoder.

$$F_{\text{cat}} = \text{Concat}\left(F_{\text{up}}, F_{\text{skip}}\right) \tag{5}$$

$$F_{\text{dec}} = \sigma\left(W_2 * \sigma\left(W_1 * F_{\text{cat}} + b_1\right) + b_2\right) \tag{6}$$

Eq. (5) indicates the skip connection operation in the decoder. Here the upsampled feature map F_{up} from the previous decoder layer is concatenated with the corresponding encoder feature map F_{skip}.

Eq. (6) indicates the feature refinement process in the decoder block. The concatenated feature map F_{cat} is passed through two convolution layers. ReLU activation is applied after each convolution using weights W_1 and W_2 and biases b_1 and b_2. The F_{dec} is a refined feature map that has improved spatial accuracy.

$$Y = W_{\text{out}} * \sigma\left(W_2 * \sigma\left(W_1 * \text{Concat}(\text{Upsample}(F_{\text{up}}), F_{\text{skip}}) + b_1\right) + b_2\right) + b_{\text{out}} \tag{7}$$

Eq. (7) represents the final output generation of the UNet architecture. The final segmentation map Y is obtained by applying a 1×1 convolution with weights W_{out} and bias b_{out} to the refined decoder feature map, producing pixel-wise class predictions.

2.3 SegNet

SegNet is a deep learning architecture that is built for semantic image segmentation [12]. This entire deep learning architecture consists of two major components, which are the encoder and decoder. The encoder part is made based on convolutional layers of VGG16 that play a role in extracting high-level features from the input image. Like UNet, the segment also performs upsampling in the decoder part. However, the upsampling operation is non-learnable in SegNet, while learning upsampling occurred in UNet. Moreover, there is no skip connection available in the segment that makes the process faster and more memory efficient but may lack fine details.

$$f_l = \sigma(W_l * p_{l-1} + b_l), \quad l = 1, 2, \ldots, L \tag{8}$$

The Eq. (8) represents the convolution operation in the encoder, and the output f_l is a higher-level feature map that captures semantic information from the input image. At the l^{th} layer, the input feature map p_{l-1} is convolved with learnable filters W_l, added with bias b_l, and passed through the ReLU activation function $\sigma(\cdot)$. The output f_l is a higher-level feature map that captures semantic information from the input image.

$$[p_l, I_l] = \text{MaxPool}(f_l) \tag{9}$$

The Eq. (9) describes the downsampling step. Max-pooling reduces the spatial resolution of the feature map f_l to produce p_l, while I_l stores the indices of the maximum values.

For each pooling region $R_{i,j}$, the pooled value and its corresponding index are computed as Eqs. (10) and (11).

$$p_l(i, j) = \max_{(u,v)\in R_{i,j}} f_l(u, v) \tag{10}$$

$$I_l(i,j) = \arg \max_{(u,v)\in R_{i,j}} f_l(u,v) \tag{11}$$

Here, $R_{i,j}$ represents the local pooling window centred at position (i,j), f_l is the feature map from the l^{th} convolutional layer, p_l is the downsampled output, and I_l contains the locations of the maximum activations within each pooling window. Using the stored pooling indices I_l, the decoder upsamples the feature map p_{l+1} to obtain $\hat{f}_l$, as shown in Eq. (12).

$$\hat{f}_l = \text{Upsample}(p_{l+1}, I_l) \tag{12}$$

Each value $p_{l+1}(i,j)$ is placed at the location $I_l(i,j)$, while all other positions (u,v) in $\hat{f}_l(u,v)$ are set to zero, as formally defined in Eq. (13).

$$\hat{f}_l(u,v) = \begin{cases} p_{l+1}(i,j), & \text{if } (u,v) = I_l(i,j) \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

After upsampling, a learnable convolution is applied to the sparse feature map $\hat{f}_l$ using decoder weights $\tilde{W}_l$ and bias $\tilde{b}_l$. The output $\tilde{f}_l$ is a dense feature map at decoder layer l, and $\sigma(\cdot)$ denotes the ReLU activation function, as defined in Eq. (14).

$$\tilde{f}_l = \sigma(\tilde{W}_l * \hat{f}_l + \tilde{b}_l) \tag{14}$$

After the last decoder layer, a 1×1 convolution with weights $W_{k,c}^{(\text{out})}$ and bias $b_k^{(\text{out})}$ is applied to the decoder output $\tilde{f}_1$. This gives the class score map $Y(x,y,k)$ for each pixel (x,y) and class k, as shown in Eq. (15). The class scores are then changed into probabilities $P(x,y,k)$ using the softmax function over all K classes, as described in Eq. (16). Finally, the predicted class label $\hat{Y}(x,y)$ for each pixel is chosen as the class with the highest probability, following Eq. (17).

$$Y(x,y,k) = \sum_{c=1}^{C_L} W_{k,c}^{(\text{out})} \tilde{f}_1(x,y,c) + b_k^{(\text{out})} \tag{15}$$

$$P(x,y,k) = \frac{e^{Y(x,y,k)}}{\sum_{k'=1}^{K} e^{Y(x,y,k')}} \tag{16}$$

$$\hat{Y}(x,y) = \arg\max_k P(x,y,k) \tag{17}$$

2.4 Fully Convolutional Neural Network

A fully convolutional neural network (FCNN) is a type of deep learning algorithm that is used for many image processing tasks [13]. Unlike typical CNN methods, it does not utilize a fully connected layer at the end [14]. Because here, instead of classifying the image, FCNN works to fetch out what class or object each pixel in the image belongs to. FCNN is flexible to any size of the input image and also has the feasibility to recover the spatial resolution of the image after downsampling. Moreover, the spatial relationship between pixels is also maintained in FCNN, which plays a significant role in doing the image segmentation task more efficiently.

Eq. (18) shows how a convolutional layer works in FCNN. $F^{(l-1)}$ is the input feature map, and $F^{(l)}_{i,j,k}$ is the output at pixel (i, j) in channel k. The layer uses weights $W^{(l)}_{m,n,c,k}$ and bias $b^{(l)}_k$, applies a nonlinear function $\sigma(\cdot)$, and sums over the filter size and all input channels. This extracts important features while keeping the image structure.

$$F^{(l)}_{i,j,k} = \sigma\left(\sum_{c=1}^{C_{l-1}} \sum_{m=-M}^{M} \sum_{n=-N}^{N} W^{(l)}_{m,n,c,k} \cdot F^{(l-1)}_{i+m,\, j+n,\, c} + b^{(l)}_k\right) \tag{18}$$

After extracting features with the convolutional layer (Eq. (18)), Eq. (19) presents the operation of the pooling layer. This layer reduces the spatial size of the feature maps. Here, $F^{(l-1)}$ is the input feature map, and $F^{(l)}_{i,j,k}$ is the pooled output at pixel (i, j) in channel k. The layer takes the maximum value within a local window Ω with a stride s.

$$F^{(l)}_{i,j,k} = \max_{(p,q)\in\Omega} F^{(l-1)}_{s\cdot i+p,\, s\cdot j+q,\, k} \tag{19}$$

After the pooling operation, a transposed convolution layer restores the spatial resolution of feature maps following Eq. (20). Here, $F^{(l-1)}$ is the input feature map, $F^{(l)}$ is the upsampled output, $W^{(l)}$ are the learnable filter weights, and $b^{(l)}$ is the bias term. The transposed convolution enables the network to reconstruct higher-resolution features.

$$F^{(l)} = F^{(l-1)} *_t W^{(l)} + b^{(l)} \tag{20}$$

After upsampling with transposed convolution, Eq. (20), a 1×1 convolution layer, Eq. (21), maps each pixel to class scores. Here, the feature at pixel (i, j) in each channel is multiplied by the weights and added to the bias for each class. This produces an output tensor, Eq. (22), with height H, width W, and K channels, where K is the number of classes. Finally, the Softmax function, Eq. (23), converts these class scores into probabilities for each pixel.

$$Y_{i,j,k} = \sum_{c=1}^{C'} W^{(1\times1)}_{c,k}\, F_{i,j,c} + b_k \tag{21}$$

$$Y \in \mathbb{R}^{H\times W\times K} \tag{22}$$

$$P(y_{i,j} = k \mid X) = \frac{e^{Y_{i,j,k}}}{\sum_{k'=1}^{K} e^{Y_{i,j,k'}}} \tag{23}$$

2.5 Dataset Description

The Oxford IIIT pet dataset is one renowned dataset across the globe and is widely used for many image processing tasks like image segmentation, image classification and so on [15]. It consists of 7349 images with 37 varieties of breeds. Among 37 breeds, 25 are dogs, and the remaining ones are all cats. For each breed, around 200 images are allocated, where 100 images are assigned for training and validation, and the remaining 100 images are used for testing purposes. Each image has a detailed segmentation map of the pet. To collect data for this dataset, several social websites are used, and some rules and conditions are followed to select ultimate images for the dataset. Here, mainly based on image quality and appropriate cat position, the images were selected.

3 Methodology

The methodology section presents the complete design and implementation of the proposed collaborative segmentation framework, and the overview of this methodology is visually presented in Fig. 2. In this research, we collected data initially and preprocessed it afterwards. Following that, the data was trained. After that, we made the prediction system and self-learning ability, respectively.

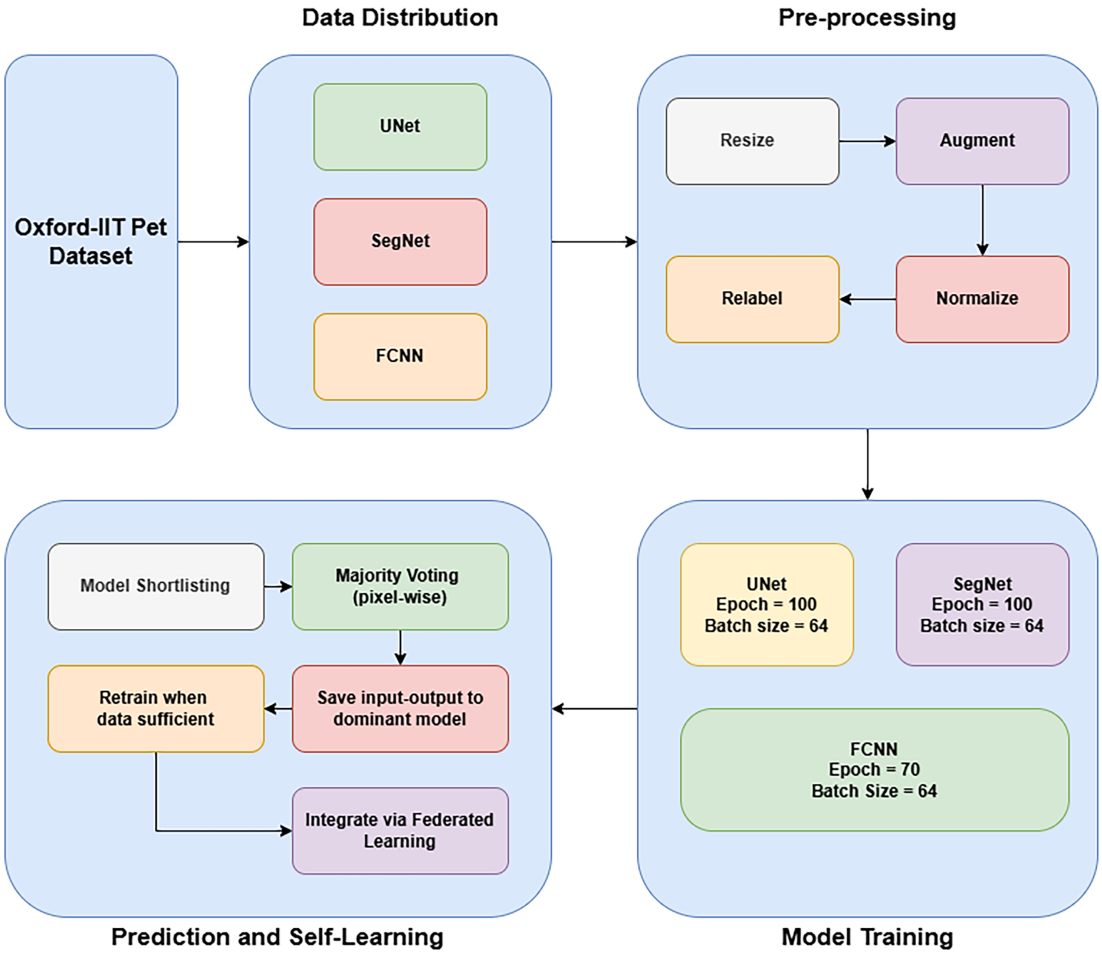

Figure 2: Overview of the proposed methodology.

3.1 Dataset Collection and Distribution

For our research, we used the Oxford IIT pet dataset. In Fig. 3, three sample inputs of this dataset and their corresponding masks are visually presented. Unlike typical methods of training, the dataset was distributed into three segments for three types of models. Here, each model did not get the data equally. Because, in the proposed approach, the framework will improve its performance by continuously interacting with the inputs. Here, it is not possible that every segment will always have an equal amount of data. Therefore, to measure the effectiveness of the performance in a real case, the data was distributed unequally. For UNet architecture, the 2000 data was allocated. These 2000 data are the data of the first 2000 data of the training of the Oxford IIT pet dataset. In the Oxford IIT pet dataset, there are a total of 3680 images assigned for training. In our research, the initial 2000 data of this dataset is provided to train via UNet. The remaining 1680 data were delivered to SegNet. However, for the FCNN architecture, 2000 data points were

used for training, and these were gathered from the testing set. And the initial 2000 images of the testing set are assigned to train the FCNN model.

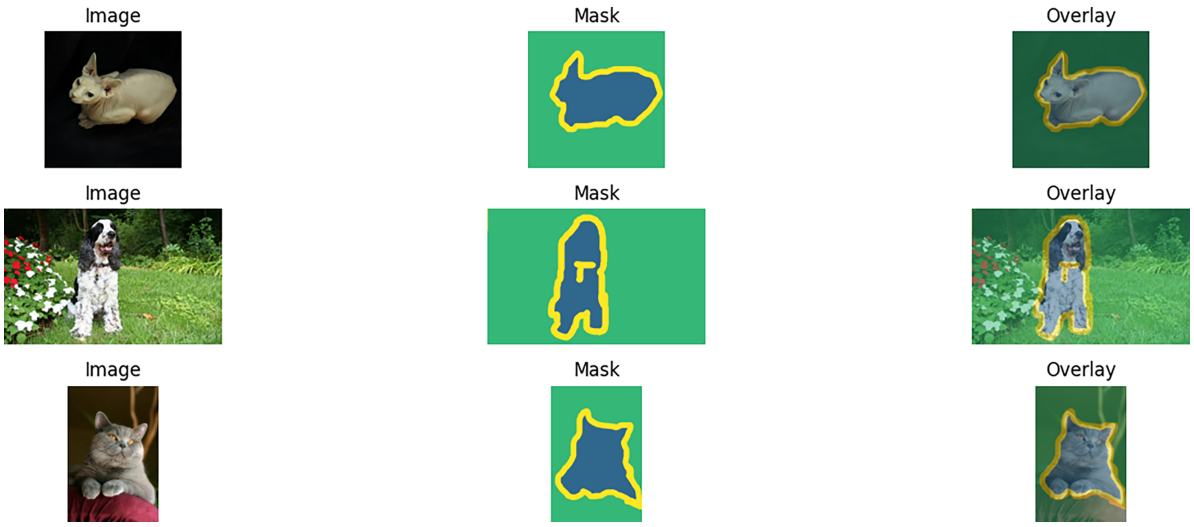

Figure 3: Sample data of Oxford-IIT pet dataset.

For testing purposes, we used a testing set for the cases of UNet and SegNet. On the other hand, all the training datasets, along with the remaining unallocated data for testing, were assigned for FCNN testing purposes.

3.2 Data Pre-Processing

After collecting and distributing data, we have preprocessed the assigned data [16]. At first, all the images were resized into 128 × 128 pixels using nearest-neighbour interpolation. Here, the training images were augmented to increase the diversity, which is quite important for generalizing. Afterwards, each image was normalized by scaling pixel values to the range [0, 1]. Because, usually, raw images are stored as [0,255], which creates various issues, including unstable training, less efficient optimization, and difficulty in achieving balanced weight updates during learning. In such cases, normalization plays a significant role in wiping out such troubles during training and ensures numerical stability, speeds up training, and improves model performance. Following that, the segmentation mask labels were shifted from the original set 1, 2, 3 to 0, 1, 2 to match the requirements of the loss function.

The preprocessing of images and masks can be represented mathematically as:

$$\tilde{X}, \tilde{Y} = \text{Preprocess}(X, Y) = \left\{ \underbrace{\text{Augment}(\text{Resize}_{128\times128}(X))}_{\text{Image Processing}}, \underbrace{Y - 1}_{\text{Mask Label Shift}} \right\}, \quad X_{\text{norm}} = \frac{\tilde{X}}{255}$$

Explanation of terms:

- X: Original images
- Y: Original segmentation masks $\{1, 2, 3\}$
- $\text{Resize}_{128\times128}(\cdot)$: Resizing images to 128×128
- $\text{Augment}(\cdot)$: Data augmentation (rotation, flipping, etc.)
- X_{norm}: Normalized images in $[0, 1]$
- $Y - 1$: Shift mask labels from $\{1, 2, 3\}$ to $\{0, 1, 2\}$

3.3 Model Training

In the Table 1, the overall common configurations of the three models are given. In this table, it is visible that each model uses a uniform batch size of 64. However, the step per epoch is different due to the imbalance in data size. But the same optimizer, learning rate and loss function are used here, and these values are Adam, 1e–3 and Sparse Categorical Cross-Entropy, respectively. On the other hand, Unet and SegNet require 100 epochs, and 70 epochs were needed for FCNN.

Table 1: Training setup comparison.

Training Setup	UNet	SegNet	FCNN
Batch Size	64	64	64
Steps per Epoch	31	26	31
Optimizer	Adam [17]	Adam	Adam
Epochs	100 [18]	100	70
Learning Rate	1e–3	1e–3	1e–3
Loss Function	Sparse Categorical Cross-Entropy [19]	Sparse Categorical Cross-Entropy	Sparse Categorical Cross-Entropy

Additionally, Table 2 provides an overall summary of the UNet model architecture. On the other hand, Tables 3–5 deliver summaries of Unet architecture much more deeply.

Table 2: UNet model summary.

Component	Value
Input Size	128 × 128 × 3
Output Size	128 × 128 × 3
Depth	4 encoder + bottleneck + 4 decoder
Base Filters	64
Max Filters	1024
Output Activation	Softmax [20]
Loss	Sparse categorical crossentropy
Optimizer	Adam

Table 3: UNet encoder architecture.

Stage	Description	Output Shape	Filters	Operations
Input	Input image	128 × 128 × 3	–	–
Encoder Block 1	Conv2D × 2 → ReLU	128 × 128 × 64	64	conv2d_block
↓ MaxPool + Dropout		64 × 64 × 64	–	pooling
Encoder Block 2	Conv2D × 2 → ReLU	64 × 64 × 128	128	conv2d_block
↓ MaxPool + Dropout		32 × 32 × 128	–	pooling
Encoder Block 3	Conv2D × 2 → ReLU	32 × 32 × 256	256	conv2d_block
↓ MaxPool + Dropout		16 × 16 × 256	–	pooling
Encoder Block 4	Conv2D × 2 → ReLU	16 × 16 × 512	512	conv2d_block

(Continued)

Table 3 (continued)

Stage	Description	Output Shape	Filters	Operations
↓ MaxPool + Dropout		8 × 8 × 512	–	pooling
Bottleneck	Conv2D × 2 (1024)	8 × 8 × 1024	1024	bottleneck

Table 4: UNet decoder architecture.

Stage	Input	Output Shape	Filters	Operation
Decoder 1	Bottleneck + Skip from Encoder 4	16 × 16 × 512	512	Conv2DTranspose → Concat → Conv
Decoder 2	Previous + Skip from Encoder 3	32 × 32 × 256	256	Conv2DTranspose → Concat → Conv
Decoder 3	Previous + Skip from Encoder 2	64 × 64 × 128	128	Conv2DTranspose → Concat → Conv
Decoder 4	Previous + Skip from Encoder 1	128 × 128 × 64	64	Conv2DTranspose → Concat → Conv

Table 5: UNet output layer.

Final Layer	Output Shape	Activation	Classes
Conv2D (1 × 1)	128 × 128 × 3	softmax	3

From Table 3, it is visible that the output shape is consistently reducing in the deeper layer, which indicates that spatial resolution is decreasing. However, feature channels are increasing. At the first stage, the output shape was 128×128×3. Encoder Block 1 applies two convolutional layers with ReLU activation to produce 64 feature maps at the same resolution. After extracting features using convolution layers, max-pooling reduces the spatial resolution from 128×128 to 64×64. This downsampling makes the feature representation more compact and computationally efficient while preserving the most important information. On the other hand, dropout removes some activation functions to avoid overfitting. Encoder 2 repeats this same process. It increases the filters to 128 and downsamples the feature maps to 32 × 32 × 128. Following that, encoder blocks 3 and 4 compute the same task. Eventually, in the bottleneck phase, spatial resolution becomes the lowest, and the number of feature channels becomes the highest, which are 8×8 and 1024, respectively.

In Table 4, the overview of the decoder operations is given. It is visible that the opposite tasks of the encoder are executed in the decoder. In the decoder, spatial resolution is increasing, but feature channels are decreasing. The bottleneck layer is connected with the deepest and first layer of the decoder. When the bottleneck output arrives in the decoder 1 stage, the feature map is upsampled and becomes 16×16 with 512 channels. Then it is combined with features from the matching encoder layer through skip connections. That is how the model preserves the important details that were lost during downsampling. These processes continue until decoder 4, and the image size increases step by step from 16 × 16 to 128 × 128, while the number of feature channels decreases from 512 to 64.

After completing the decoding tasks, the final output layer converts the decoded features into a segmentation map. Here, the 1 × 1 convolution is used to map the 64-channel feature representation at each pixel to 3 output channels. Here, each channel represents one segmentation class. Besides, the softmax activation function is present here to calculate the class for each pixel. The Table 5 represents the output layer of the UNet.

Table 6 represents the encoder of the SegNet architecture. Here, each encoder block contains two convolution layers with a 3 × 3 filter and ReLU activation. Also, a max-pooling layer is available in each encoder block. Here, two convolution layers are applied to learn important features such as edges, textures, and object parts. On the other hand, the max-pooling layer reduces the spatial size by half to help the network focus on high-level features and reduce computation. After the final stage of encoding, the spatial resolution decreases from 128 × 128 to 8 × 8, and the number of feature channels increases from 64 to 512.

Table 6: SegNet encoder architecture.

Block	Input Size	Layers per Block	Filters	Downsampling	Output Size
Input	128 × 128 × 3	–	–	–	128 × 128 × 3
E1	128 × 128 × 3	2× Conv (3 × 3) + ReLU	64	MaxPool 2 × 2	64 × 64 × 64
E2	64 × 64 × 64	2× Conv (3 × 3) + ReLU	128	MaxPool 2 × 2	32 × 32 × 128
E3	32 × 32 × 128	2× Conv (3 × 3) + ReLU	256	MaxPool 2 × 2	16 × 16 × 256
E4	16 × 16 × 256	2× Conv (3 × 3) + ReLU	512	MaxPool 2 × 2	8 × 8 × 512

Table 7 presents the decoder architecture of SegNet. Like the encoder, the decoder also contains four blocks. And each block upsamples the feature map by a factor of 2 to increase the spatial resolution. Besides, two convolution layers with ReLU activation work to refine the upsampled features. In the decoder, the spatial resolution increases step-by-step from 8 × 8 back to 128 × 128. On the other hand, the number of filters decreases from 512 to 64.

Table 7: SegNet decoder architecture

Block	Layers per Block	Filters	Output Resolution
D4	UpSample + 2× Conv (3 × 3) + ReLU	512	16 × 16
D3	UpSample + 2× Conv (3 × 3) + ReLU	256	32 × 32
D2	UpSample + 2× Conv (3 × 3) + ReLU	128	64 × 64
D1	UpSample + 2× Conv (3 × 3) + ReLU	64	128 × 128

In Table 8, the summary of the final output layer of SegNet is visible. This layer converts the feature maps into 3 output channels.

Table 8: SegNet output layer.

Layer	Operation	Filters	Activation	Output Size
Output	Conv 1 × 1	3	Softmax	128 × 128 × 3

Table 9 discloses the encoder segment of the FCNN model. Here, the encoder block consists of three Conv2D + ReLU layers. After each Conv2D + ReLU layer, a MaxPooling2D layer is visible. Conv2D + ReLU extracts meaningful features from the image. Here, each conv2D layer applies a 3 × 3 filter to capture these local patterns. And ReLU applies a non-linear activation, $\mathrm{ReLU}(x) = \max(0, x)$. On the other hand, MaxPooling2D reduces the image and preserves the strongest features. That is why the feature map size has been reduced from 128 × 128 × 3 to 16 × 16 × 256 in the encoder. In the bottleneck layer, the feature map size remains the same, but the spatial resolution remains 16 × 16, and the depth increases from 256 to 512.

The decoder segment consists of three Conv2DTranspose + ReLU, and each works to restore the spatial resolution lost during encoding. Here, the feature map upsamples consistently and increases by 2, and the feature channel is reduced by two. That is, after going through the three Conv2DTranspose + ReLU, the feature map is converted from 16 × 16 to 128 × 128 and the channel depth from 512 to 64. And the final output layer converts the 64 feature maps into 3 class probability maps.

Table 9: FCNN architecture summary.

Layer Type	Filters	Kernel/Stride	Output Shape
Input	–	–	128 × 128 × 3
Conv2D + ReLU	64	3 × 3	128 × 128 × 64
MaxPooling2D	–	2 × 2	64 × 64 × 64
Conv2D + ReLU	128	3 × 3	64 × 64 × 128
MaxPooling2D	–	2 × 2	32 × 32 × 128
Conv2D + ReLU	256	3 × 3	32 × 32 × 256
MaxPooling2D	–	2 × 2	16 × 16 × 256
Bottleneck – Conv2D + ReLU	512	3 × 3	16 × 16 × 512
Conv2DTranspose + ReLU	256	3 × 3/stride 2	32 × 32 × 256
Conv2DTranspose + ReLU	128	3 × 3/stride 2	64 × 64 × 128
Conv2DTranspose + ReLU	64	3 × 3/stride 2	128 × 128 × 64
Conv2D (Softmax Output)	3	1 × 1	128 × 128 × 3

3.4 Prediction System and Self Learning

3.4.1 Model Shortlisting

Unlike typical ways of output prediction, the proposed framework does not deliver a response from a trained model. Instead, a unique approach is followed to make our novel framework. Here, in the proposed framework, multiple trained models with different architectures participate in the output prediction. Initially, for an input, an output will be predicted from all the trained models alongside the confidence score. Here, the model's confidence score discloses how confident it is in its output. And the confidence score will be calculated using the formula of Eq. (24).

$$\gamma_{i,j}^{(m)} = \max_{c\in\{1,2,\ldots,C\}} p_{i,j,c}^{(m)} \quad \text{where} \quad p_{i,j,c}^{(m)} = \Pr\left(y_{i,j} = c \mid X; \theta^{(m)}\right) \tag{24}$$

In this formulation, $\gamma_{i,j}^{(m)}$ represents the confidence score of model m for the pixel located at (i, j). The term $p_{i,j,c}^{(m)}$ denotes the predicted probability that the pixel (i, j) belongs to class c, given the input image X and the parameters of model m, represented as $\theta^{(m)}$. Here, C indicates the total number of semantic classes considered in the segmentation task.

These model outputs are then sorted in descending order of confidence. To make sure only reliable predictions are used, we apply a confidence gap threshold $\delta = 0.10$. The model with the highest confidence is always included, and any other model is added only if its confidence is very close (within 0.10) to the top model. For example, if the top model has a confidence of 0.92 and another model has 0.87, the difference is 0.05 (<0.10), so both are included. But if a model has 0.75, the difference from 0.92 is 0.17 (>0.10), so it is excluded. Then the shortlisted models will be forwarded for the majority voting task.

Here, the confidence threshold value is $\delta = 0.10$. Because the prediction of models within this range value can attach the prediction, which is highly relevant. This value provides a balanced trade-off between prediction reliability and model participation. If the threshold is set too low, predictions may be highly uncertain. Such can introduce noise and reduce segmentation accuracy. Conversely, setting the threshold too high may exclude useful predictions from certain models. It reduces the diversity of architectural perspectives. To fix such issues, the ensemble framework is designed. The value $\delta = 0.10$ allows the framework to filter out highly uncertain predictions while still retaining sufficient model contributions for collaborative decision-making.

3.4.2 Majority Voting

After shortlisting, each shortlisted model will vote on the output for each pixel. The output with the most votes is assigned as the final label for that pixel. If there is a tie, then the model which has the highest confidence score, its output will be the final output for this pixel. That is how the output of all pixels will be determined by following this process. After predicting the ultimate output, the input and the ultimate predicted output will be saved in the database. But in the proposed framework, each model has its own dataset. So, in which dataset the ultimate output will be saved is determined by the contributions of the models. The model which contributes most of the pixels for the ultimate output–this data will be saved in this model's dataset.

$$Y^*_{i,j} = \begin{cases} \arg\max_c \ \text{count}\left(\left\{ Y^{(m)}_{i,j} = c \mid m \in Selected_{i,j} \right\}\right), & \text{if unique majority} \\ \arg\max_c \ \sum_{m \in Selected_{i,j},\ Y^{(m)}_{i,j}=c} \gamma^{(m)}_{i,j}, & \text{if tie} \end{cases} \tag{25}$$

The Eq. (25) represents the process of majority voting. Here, $Y^*_{i,j}$ is the final assigned class for pixel (i, j). For each pixel (i, j), let $Y^{(m)}_{i,j}$ be the predicted class from model m, and let $\gamma^{(m)}_{i,j}$ denote the confidence score of model m for that pixel. Furthermore, let $Selected_{i,j} \subseteq \{1, 2, \ldots, M\}$ represent the set of models shortlisted for pixel (i, j) based on their confidence scores. The Algorithm 1 presents the step-by-step process of the proposed prediction system.

Algorithm 1: Prediction system

Input: Predicted probabilities $P^{(m)}_{i,j,c}$ from each model m, confidence gap Δ
Output: Final segmentation mask $Y[i, j]$, model contribution counts $Count[m]$
for *each pixel* (i, j) **do**
  for *each model* m **do**
    $PredClass[m] \leftarrow \arg\max_c P^{(m)}_{i,j,c}$;
    $Conf[m] \leftarrow \max_c P^{(m)}_{i,j,c}$;
  Construct list $\mathcal{L} = \{(Conf[m], PredClass[m], m)\}$;
  Sort $\mathcal{L}$ in descending order of confidence;
  Initialize $Selected$ with the top-ranked model;
  for *remaining models in* $\mathcal{L}$ **do**
    if *confidence difference* $< \Delta$ **then**
      Add model to $Selected$;
    else
      break;

(Continued)

Algorithm 1 (continued)

Assign $Y[i, j]$ to the class with the majority vote in $Selected$;
if *a tie occurs* **then**
Select the class with the highest summed confidence;
Update $Count[m]$ for contributing model(s);
return Y *and* $Count[m]$

3.4.3 Self-Adapting

In this framework, for each image segmentation prediction, the model that predicts the largest number of pixels for the image is considered the main contributor. The input image and its final predicted output are saved in this model's database. Once enough new data is collected, a new model with the same architecture is created, and the newly added data is assigned to this new model for training.

Three model architectures are used in the system: UNet, SegNet, and FCNN. For example, when the UNet model predicts the most pixels and accumulates enough data, a new UNet model is created. The newly collected data becomes the training data for this new UNet model. This new UNet is then aggregated with the existing UNet model using the federated learning algorithm. At this point, there are two local UNet models: the existing model and the newly assigned model. During federated training, only the newly assigned UNet model is trained over the communication rounds, while the existing UNet model remains frozen.

The same process applies to SegNet and FCNN. This approach allows the system to continuously integrate new learning through the federated learning method described in Algorithm 2.

Algorithm 2: Self-adapting via federated learning

Input: Frozen global model M_1, trainable local model M_2, pseudo-labeled dataset $D = \{(x_i, y_i)\}$, learning rate η, local epochs E, federated rounds R, aggregation weight α
Output: Federated model M_{fed}
for *round* $r = 1$ **to** R **do**
Local Training:
Train M_2 on D for E epochs using Adam(η) optimizer and sparse categorical cross-entropy loss
Layer-wise Safe Aggregation:
for *each pair of layers* (L_1, L_2) *in* (M_1, M_2) **do**
$W_1 \leftarrow L_1$.get_weights(), $W_2 \leftarrow L_2$.get_weights()
if W_1 *and* W_2 *exist and have same number of tensors* **then**
for *each pair of tensors* (w_1, w_2) *in* (W_1, W_2) **do**
if $shape(w_1) == shape(w_2)$ **then**
$w_{\text{new}} \leftarrow \alpha \cdot w_1 + (1 - \alpha) \cdot w_2$
else
Skip this layer
Update L_2 with compatible w_{new}
Assign federated model: $M_{\text{fed}} \longleftarrow M_2$
return M_{fed}

By redistributing newly generated training samples to the most relevant architecture, the framework encourages architecture-level specialization. Different segmentation architectures exhibit different inductive biases and feature extraction behaviours. As a result, certain models naturally perform better on particular visual structures, object boundaries, or spatial patterns. Assigning new samples to the model that performed

best on those patterns allows the framework to reinforce its strengths rather than uniformly retraining all models on the same data. Additionally, such redistribution reduces unnecessary retraining of less relevant models. Thus, it is possible to improve computational efficiency and scalability. Instead of retraining all the models whenever new data appears, only the most suitable architecture receives additional learning signals. This selective update mechanism is particularly important in decentralized environments where computational resources may be limited. As a result, the segmentation performance improves gradually in the long run.

4 Result and Analysis

The proposed framework of image segmentation has exposed a tremendous method of image segmentation, unlike other approaches. After building the desired framework, it is required to justify its effectiveness. As a result, the performance of the system is evaluated not only for testing. Alongside, the training results were also considered [21]. As in the proposed framework, the entire data was divided into three segments, and each segment was trained separately with three different types of architectures. That is the reason; here the training and validation performances of these three trained models are presented in Table 10. Moreover, these results are visually presented in Fig. 4. From the Table 10, it is visible that all these three models deliver more than 90% training accuracy with a low amount of loss. Besides, validation performance is also fantastic. Additionally, the difference between training and validation performance is very low. So, it can be claimed that there are no issues like overfitting and underfitting available here.

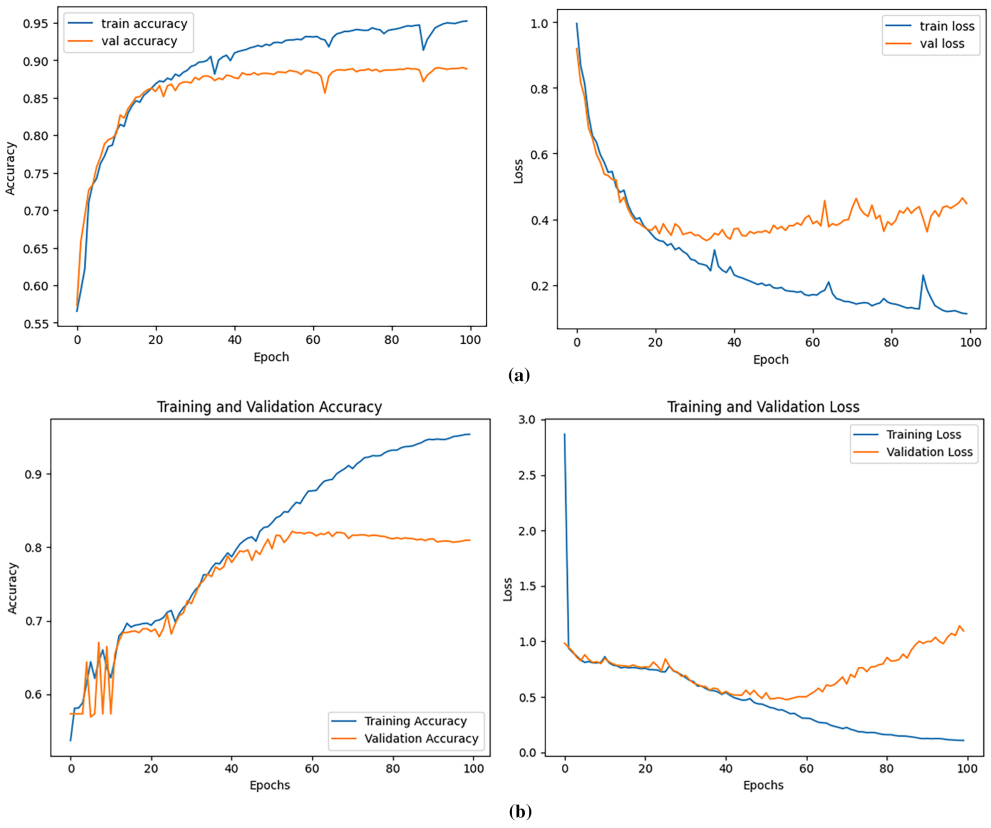

Figure 4: (Continued)

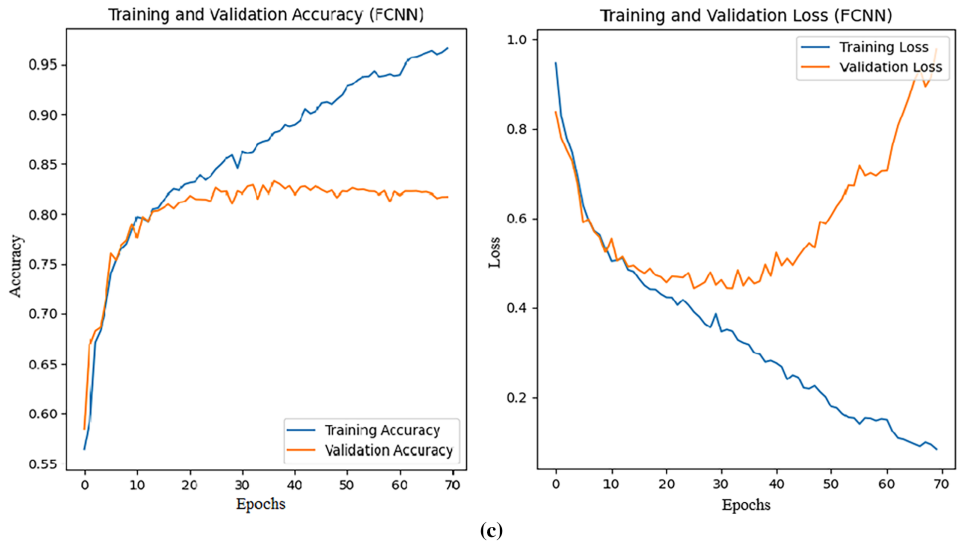

Figure 4: Training and validation performance of all models: (**a**) UNet, (**b**) SegNet, (**c**) FCNN.

Table 10: Training and validation performance.

Model	Epochs	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
UNet	100	0.9520	0.1130	0.8884	0.4483
SegNet	100	0.9540	0.1081	0.8098	1.0955
FCNN	70	0.9649	0.0862	0.8175	0.9776

Like training and validation, testing performance results are also quite important to know the effectiveness of this proposed framework. The Table 11 represents the testing result. Here, it is visible that the performance with one model is the worst, and the performance with the highest number of models is the best. It indicates that the performance is consistently improving by increasing the number of models. Here, the three models with different architecture are applied. Each model has different types of uniqueness. When these three models are merged together in a single framework, it is possible to use the facilities of all these three models. Therefore, the single model performance is poor. But in the case of three models, performance is excellent. Because in this proposed framework, all three models will work together to fetch the best output. Such a collaborative output prediction framework can provide phenomenal performance. The average pixel accuracy with three models is 89.26%, which is far better than a solo prediction system. This average pixel accuracy ensures the appropriateness of classified pixels. Additionally, average IoU, F1 score, precision, and recall values are 71.48%, 81.29%, 83.96%, and 81.16%, respectively [22,23]. The IoU of 71.48% reflects strong overlap between predicted and ground-truth regions. In addition, the F1 score, precision, and recall indicate a good balance between false positives and false negatives. Eventually, based on the results of Table 11, it is justified that combining different model architectures helps the system to produce more accurate and reliable segmentation results. Table 11 shows that the performance consistently improves as more models are added. The single-model configuration produces the lowest performance among all cases, while the configuration with three models achieves the best overall results. To further validate the effectiveness of the proposed method, the experiments are not limited to a single dataset. In addition to the Oxford-IIIT Pet dataset, we also evaluate the framework on the Breast Ultrasound Images(BUSI) dataset [24]. The results on the BUSI dataset also demonstrate strong performance, indicating that the proposed method can achieve

reliable segmentation results across different datasets. The performance improvement with multiple models is mainly due to architectural diversity. UNet preserves spatial details through skip connections, SegNet provides efficient encoding–decoding, and FCNN maintains spatial consistency. Their combination allows the framework to reduce errors and improve segmentation accuracy. Additionally, the confidence threshold further influences this improvement. Confidence-based filtering eliminates unreliable predictions. Only high-confidence models are allowed to vote. This makes the system more robust, especially in complex or unfamiliar cases.

Table 11: Testing performance.

Model	Average Pixel Accuracy	Average IoU	Average F1 Score	Average Precision	Average Recall
Single Model (Oxford-IIIT Pet)	79.12%	53.83%	65.85%	68.09%	66.92%
2 Models (Oxford-IIIT Pet)	87.55%	67.86%	78.38%	82.30%	78.40%
3 Models (Oxford-IIIT Pet)	89.26%	71.48%	81.29%	83.96%	81.16%
3 Models (BUSI)	96.53%	74.54%	82.59%	78.55%	94.88%

In Fig. 5, the tSNE of the proposed framework with three different types of scenarios is visible [25]. Here, it is visible that the t-SNE with three models performs best in terms of generalization. On the other hand, the t-SNE with two models has shown excellent generalization ability. Still, the performance with three models is comparatively better. And the t-SNE with one model performs worst. In the case of just one model, it is not required to follow the proposed framework. In such a situation, the conventional prediction approach is applied. Therefore, performance is not as good as the other two situations. In the rest of the scenes, two models and three models are integrated into the system. And these models work together to predict output by our proposed collaboration-based prediction system. That is when the model number grows; then it increases the possibility to output accurately.

Along with presenting the quantitative performance, Fig. 6 presents the qualitative performance. This figure represents the sample of five outputs for five inputs. From this figure, it is visible that the predicted outputs through our proposed framework are quite accurate. Unlike conventional ways, the proposed prediction framework delivers outputs by doing collaboration among all models. At first, models are shortlisted for majority voting based on confidence level. After that, the ultimate output is predicted using the majority vote among the shortlisted models. Table 12 contains the shortlisted models and confidence scores of the outputs of Fig. 6. From input 1 to input 3, all the models were selected for majority voting. Because, in input 1, the highest confidence score is 0.9501, and the confidence levels for all the remaining models are above $(0.9501 - 10) = 0.8501$. Therefore, all models participated in majority voting for input 1. The same thing happens in input 2 and input 3. But in input 4, the highest confidence score is 0.9527, and model 3's confidence score is below $(0.9527 - 10) = 0.8527$. That is why Model 3 was not qualified for majority voting. The same case is visible for input 5.

Comparison Results with State-of-Art Models

One of the major advantages of this proposed framework is reducing the training time and resource efficiency. As the model is trained decentralised in the proposed framework, it is supposed to take less training time than conventional methods. In addition, this proposed system consumes lower energy. To justify this statement, the proposed method is compared with three state-of-the-art methods named UNet, SegNet and FCNN. As our proposed framework, all training data, along with the initial 2000 testing data was used; these three models were also trained with these data with 30 epochs. And the performance of these models will be compared with the UNet of the initial 2000 training. Because, in the proposed framework, UNet was trained via the initial 2000 data, and it demands the highest energy. So, the remaining two models

did not need such power that was required for UNet. Therefore, if we calculate the performance of only UNet, then we will get the total training time and the highest amount of consumed energy. And all the methods were run on the same platform, Google Colab, with a Tesla T4 GPU.

As shown in Table 13, the proposed method requires at most 24.66 min, while other methods run significantly longer than this amount. The proposed framework takes lower training time compared to centralized approaches. The reduced training time is mainly due to decentralized training, where smaller data partitions are processed independently. This avoids the overhead of centralized large-scale training and enables more efficient resource utilization. After the proposed method, the FCNN ranked second. This model took 47.49 min, which is extremely higher than the proposed method. Under the column of energy consumption, the cumulative energy consumption of CPU, GPU and RAM is given. And in this case, the proposed method outperforms the other three models. This proposed system performs far better than other methods. Eventually, it is justified that our proposed method is not only time-consuming but also resource and energy-efficient.

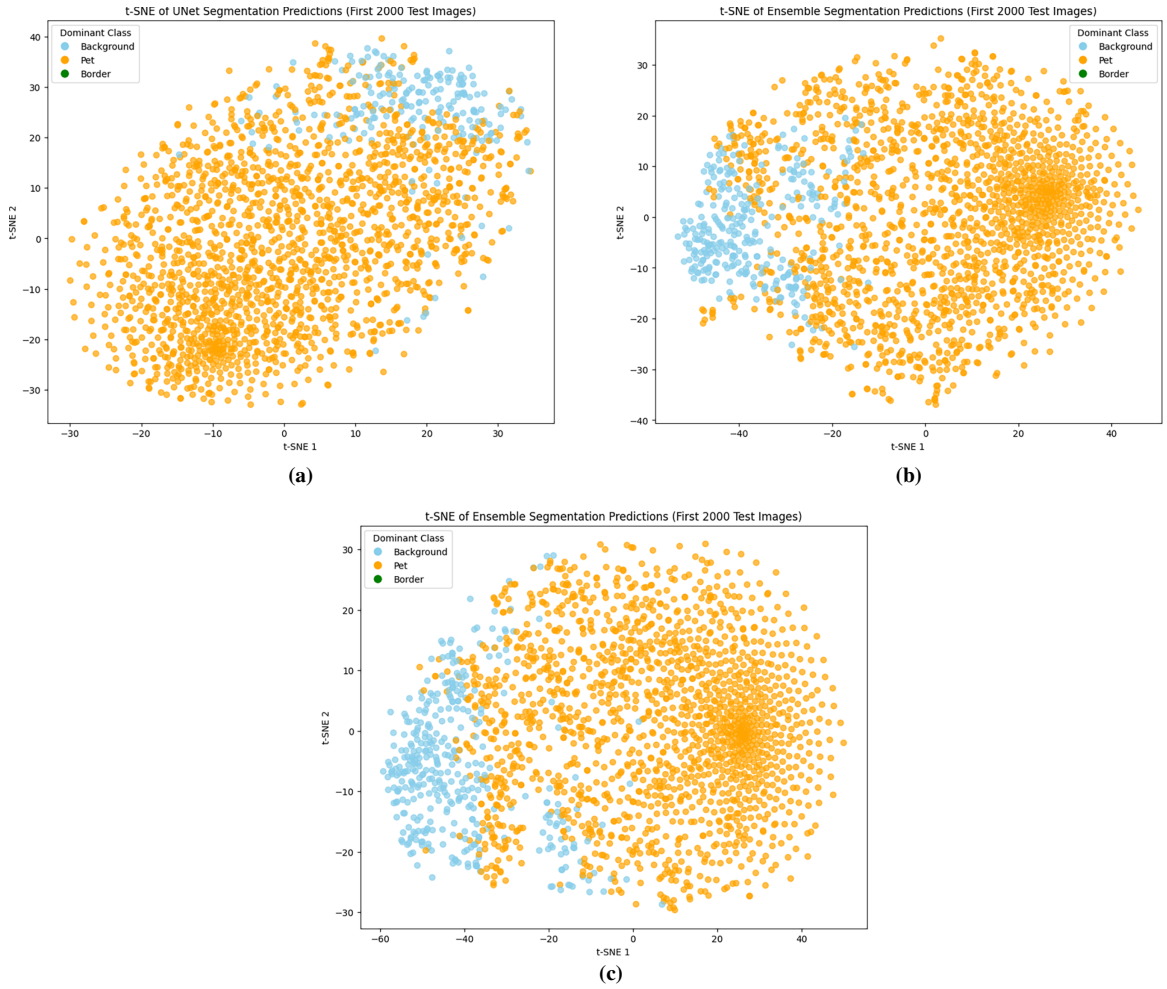

Figure 5: t-SNE analysis of model configurations. (**a**) Single model, (**b**) Two models, (**c**) Three models.

Figure 6: Sample output predictions: (**a**) Input 1, (**b**) Input 2, (**c**) Input 3, (**d**) Input 4, (**e**) Input 5.

Table 12: Confidence scores and shortlisted models for each input.

Input	Confidence Score			Shortlisted Models
	Model 1	Model 2	Model 3	
Input 1	0.9347	0.9501	0.8572	Model 1, Model 2, Model 3
Input 2	0.9363	0.9483	0.9085	Model 1, Model 2, Model 3
Input 3	0.9463	0.9595	0.9171	Model 1, Model 2, Model 3
Input 4	0.9527	0.9331	0.6970	Model 1, Model 2
Input 5	0.9499	0.9524	0.6970	Model 1, Model 2

Table 13: Comparison results with state-of-art models (After 30 epochs).

Method	Training Time	Energy Consumption
Proposed Method	24.66 min	0.0068 kWh
UNet	66.44 min	0.0619 kWh
SegNet	47.55 min	0.0326 kWh
FCNN	47.49 min	0.0250 kWh

5 Conclusion

This paper has presented a method where three different types of model named UNet, SegNet and FCNN, are trained decentrally and predict accurately via a collaborative prediction system. Such a decentralized training method makes this system secure, scalable and resource efficient. By using different deep learning architectures, the framework combines the strengths of each model into a single system. The collaborative prediction system helps in selecting the best output using confidence-score-based model selection and majority voting. In addition, the framework can learn from user input over time using federated learning.

In the future, we have a plan to develop it by reducing the time complexity of the prediction system without compromising the performance. Besides, we will work to improve the accuracy of this framework without collecting huge amounts of data by making the collaboration system stronger, along with including a large language model. In that collaboration system, all trained models will use their experience and ideas from LLM to predict almost accurate results in unfamiliar data. These predicted outputs will then be used to continuously improve the performance of the system through a lightweight and less complex self-learning integration process.

Acknowledgement: None.

Funding Statement: This research was funded by Multimedia University, Cyberjaya, Selangor, Malaysia (Grant Number: PostDoc(MMUI/240029)).

Author Contributions: Conceptualization, Rifat Sarker Aoyon; methodology, Rifat Sarker Aoyon and Jia Uddin; software, Rifat Sarker Aoyon; validation, Sarina Mansor, Jia Uddin and Mahe Zabin; formal analysis, Mahe Zabin and Jia Uddin; investigation, Ismail Hossain and Jia Uddin; resources, Rifat Sarker Aoyon; data curation, Ismail Hossain; writing—original draft preparation, Fahmid Al Farid and Jia Uddin; writing—review and editing, Fahmid Al Farid, Mahe Zabin and Jia Uddin; visualization, Rifat Sarker Aoyon and Ismail Hossain; supervision, Jia Uddin and Sarina Mansor; funding acquisition, Sarina Mansor. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The dataset of this research available at https://www.kaggle.com/datasets/tanlikesmath/the-oxfordiiit-pet-dataset.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Minaee S, Boykov Y, Porikli F, Plaza A, Kehtarnavaz N, Terzopoulos D. Image segmentation using deep learning: a survey. IEEE Trans Pattern Anal Mach Intell. 2021;44(7):3523–42. doi:10.1109/tpami.2021.3059968.
2. Yu Y, Wang C, Fu Q, Kou R, Huang F, Yang B, et al. Techniques and challenges of image segmentation: a review. Electronics. 2023;12(5):1199. doi:10.3390/electronics12051199.
3. Archana R, Jeevaraj PE. Deep learning models for digital image processing: A review. Artif Intell Rev. 2024;57(1):11. doi:10.1007/s10462-023-10631-z.
4. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. Piscataway, NJ, USA: IEEE; 2015. p. 3431–40.
5. Luo H, U K, Zhao W. Multi-focus image fusion through pixel-wise voting andmorphology. Multimed Tools Appl. 2023;82(1):899–925. doi:10.1007/s11042-022-13218-y.
6. Ganaie MA, Hu M, Malik AK, Tanveer M, Suganthan PN. Ensemble deep learning: A review. Eng Appl Artif Intell. 2022;115:105151. doi:10.1016/j.engappai.2022.105151.
7. Talaei Khoei T, Ould Slimane H, Kaabouch N. Deep learning: Systematic review, models, challenges, and research directions. Neural Comput Appl. 2023;35(31):23103–24. doi:10.1007/s00521-023-08957-4.
8. Yang P, Xiong N, Ren J. Data security and privacy protection for cloud storage: A survey. IEEE Access. 2020;8:131723–40. doi:10.1109/access.2020.3009876.
9. Huang W, Ye M, Du B. Learn from others and be yourself in heterogeneous federated learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. Piscataway, NJ, USA: IEEE; 2022. p. 10143–53.
10. McMahan B, Moore E, Ramage D, Hampson S, y Arcas BA. Communication-efficient learning of deep networks from decentralized data. In: Artificial intelligence and statistics. London, UK: PMLR; 2017. p. 1273–82.
11. Ronneberger O, Fischer P, Brox T. U-net: convolutional networks for biomedical image segmentation. In: Proceedings of the international conference on medical image computing and computer-assisted intervention. Cham, Switzerland: Springer International Publishing; 2015. p. 234–41.
12. Badrinarayanan V, Kendall A, Cipolla R. Segnet: a deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans Pattern Anal Mach Intell. 2017;39(12):2481–95.
13. Sun X, Liu T, Liu C, Dong W. FCNN: simple neural networks for complex code tasks. J King Saud Univ-Comput Inf Sci. 2024;36(2):101970.
14. O'shea K, Nash R. An introduction to convolutional neural networks. arXiv:1511.08458. 2015.
15. Parkhi OM, Vedaldi A, Zisserman A, Jawahar CV. Cats and dogs. In: Proceedings of the 2012 IEEE conference on computer vision and pattern recognition. Piscataway, NJ, USA: IEEE; 2012. p. 3498–505.
16. Petrou MM, Petrou C. Image processing: The fundamentals. Hoboken, NJ, USA: John Wiley & Sons; 2010.
17. Ogundokun RO, Maskeliunas R, Misra S, Damaševičius R. Improved CNN based on batch normalization and adam optimizer. In: Proceedings of the international conference on computational science and its applications. Cham, Switzerland: Springer International Publishing; 2022. p. 593–604.
18. Komatsuzaki A. One epoch is all you need. arXiv:1906.06669. 2019.
19. Bishop CM, Nasrabadi NM. Pattern recognition and machine learning. New York, NY, USA: Springer; 2006. 738 p.
20. Asadi B, Jiang H. On approximation capabilities of ReLU activation and softmax output layer in neural networks. arXiv:2002.04060. 2002.

21. Rumelhart DE, Hinton GE, Williams RJ. Learning representations by back-propagating errors. Nature. 1986;323(6088):533–6. doi:10.1038/323533a0.
22. Yacouby R, Axman D. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In: Proceedings of the first workshop on evaluation and comparison of NLP systems. Stroudsburg, PA, USA: ACL; 2020. p. 79–91.
23. Cheng B, Girshick R, Dollár P, Berg AC, Kirillov A. Boundary IoU: improving object-centric image segmentation evaluation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. Piscataway, NJ, USA: IEEE; 2021. p. 15334–42.
24. Al-Dhabyani W, Gomaa M, Khaled H, Fahmy A. Dataset of breast ultrasound images. Data Brief. 2020;28(5):104863. doi:10.1016/j.dib.2019.104863.
25. Wattenberg M, Viégas F, Johnson I. How to use t-SNE effectively. Distill. 2016;1(10):e2. doi:10.23915/distill.00002.