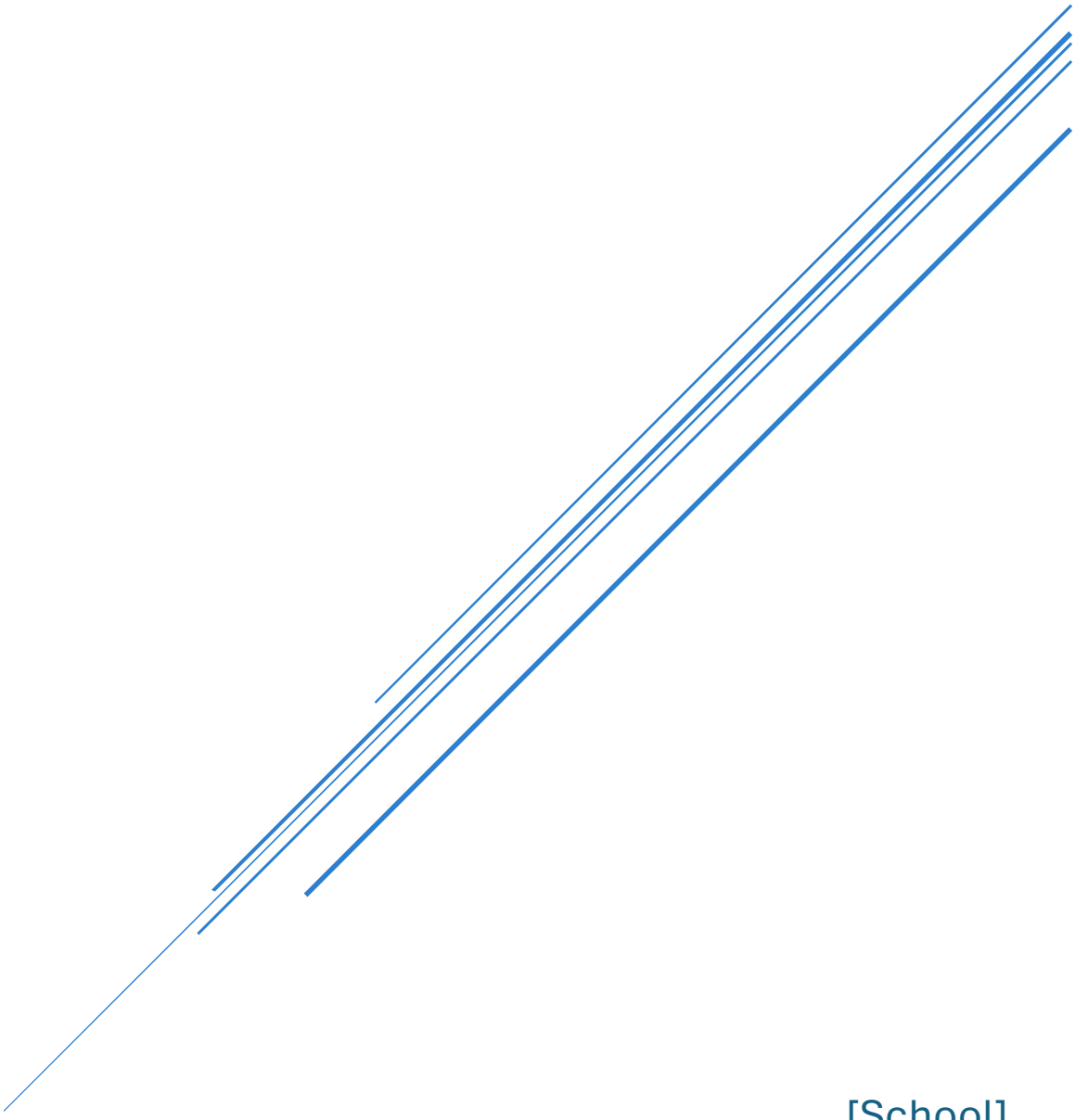
[DOCUMENT TITLE]

[Document subtitle]



[School]
[Course title]

Problem Statement

The Fuzzy C-Means (FCM) algorithm is an unsupervised clustering algorithm and is considered one of the most effective and widely used algorithms for medical image segmentation. It is like the traditional K-means algorithm but with "soft" memberships. Instead of binary memberships (0 or 1) that indicate whether a data point belongs to a cluster, each point in FCM has a weighted membership value, a number between 0 and 1. This value represents the degree of membership or probability of the point belonging to each cluster (Pseudocode 1).

Step 1: Determine the number of clusters c and ε

Step 2: Initialize the center of the cluster $v_i^{(0)}$ and $u_{ij}^{(0)}$

Step 3: k=1

Step 4: While $\left\| v_i^{(k)} - v_i^{(k-1)} \right\| > \epsilon$

Calculation of $u_{ij}^{(k)}$ and $v_i^{(k)}$ using equations (3-4)

$$u_{ij}^{(k)} = \frac{\left(1/\left\|x_j - v_i^{(k-1)}\right\|^2\right)^{\frac{1}{m-1}}}{\Sigma_{j=1}^{c}\left(1/\left\|x_j - v_i^{(k-1)}\right\|^2\right)^{1/m}}, \forall i = 1, 2, \ldots., c, j = 1, 2, \ldots., n \quad (3)$$

$$v_i^{(k)} = \frac{\Sigma_{j=1}^{n}\left(u_{ij}^{(k)}\right)^m x_j}{\Sigma_{j=1}^{N}\left(u_{ij}^{(k)}\right)^m}, \forall i = 1, 2, \ldots., c$$

k=k+1 (4)

Step 5: return cluster centers v_i and membership function u_{ij}

Pseudocode 1: FCM Algorithm

Although the FCM algorithm is simple to implement, it has a few shortcomings such as sensitivity to the cluster center initializations, getting stuck in the local minima and low convergence rate.
Boulanouar et al. propose enhancing the quality of segmentation and the speed of convergence by using the Bat Algorithm for determining the initial cluster centers and defining a cluster-based fitness function. This fitness function combines intra-cluster

distance with fuzzy cluster validity indices. They refer to the combined algorithm as **MFBAFCM**.

BAT Algorithm

The Bat Algorithm (BA) is a metaheuristic optimization technique inspired by natural processes. Specifically, it is inspired from the echolocation behavior of bats, which they use to sense distances. Bats hunting at night emit brief, intense sound pulses and analyze the returning echoes to detect obstacles or prey. Their unique auditory system enables them to determine both the size and location of objects with precision.
In the BA, the location of a bat x_i, represents a potential solution to an optimization problem, evaluated by a fitness function that measures how close the bat is to the optimal solution (or "prey"). The goal is to optimize this fitness value, guiding the bat toward the optimal solution.
Bats fly randomly with velocity v_i at position x_i and emit sounds with loudness A or at varying frequency (fmin, fmax) to search for a prey.

Parameter	Description
nBats	Number of bats
IterMax	Maximum number of iterations
f_{min}, f_{max}	Minimum and maximum frequency
Loudness Coefficient	Constant parameter in range [0, 1] used to update the loudness of each bat
Gamma	Constant parameter in range [0, 1] used to decay pulse rate

The steps of the algorithm could be summarized as follows:
Step 1 - Initialize the BAT algorithm parameters:
Initialize randomly bat positions, set initial velocities to 0. Compute fitness values for each bat with initial position and data. Set up initial best position using initial bat positions.
Step 2 – Update using best position X*, pulse frequency, the velocity, and position of the i^{th} bat as follows as:

$$f_i = f_{\min} + (f_{\max} - f_{\min})\,\beta, \quad \beta \in [0,1],$$
$$V_i^{t+1} = V_i^t + (X_i^t + X^*)\,f_i,$$
$$X_i^{t+1} = X_i^t + V_i^t,$$

Where V_i^t, and X_i^t are the velocity and position at time t (iteration t), V_i^{t+1} and X_i^{t+1} are the velocity and position at time t+1 (iteration t+1).

Step 3 – If the random number is greater than r_i, the pulse rate of the i^{th} bat, a new solution for the bat is generated by the following equation:

$$X_{\text{new}} = X_{\text{old}} + \epsilon A^t,$$

where ϵ is a random number, $\epsilon \in [-1, 1]$, and A^t represents the average loudness of all bats at time t.

Step 4 - If the random number is lower than A_i and $f(X_i) < f(X^*)$, the new solution is accepted. Next, update A_i and r_i, respectively, as follows:

$$A_i^{t+1} = \alpha A_i^t,$$
$$r_i^t = r_i^0 \left[1 - e^{-\gamma t}\right],$$

where A_i^{t+1} and A_i^t denote the loudness at times t and $t+1$, respectively; r_i^0 and r_i^t are the initial pulse rate and pulse rate at time t, respectively, α is a constant parameter in range $[0, 1]$, γ is a constant parameter, and $\gamma > 0$. As $t \to \infty$, $A_i^t \to 0$ and $r_i^t \to r_i^0$.

Step 5. Sort the bats based on their fitness and find the current optimal solution X^*.
Step 6. Return to Step 2 until the maximum number of iterations is reached; output the globally optimal solution.

MFBAFCM

When the BAT algorithm and FCM algorithm are combined, the process involves two main steps. In the first step, the BAT algorithm selects the optimal initial clusters for the FCM algorithm. In the second step, the FCM algorithm refines these clusters to find the optimal cluster assignments and their centers. Both algorithms operate using the same cost function (Pseudocode 2).
Boulanouar et al. define a fitness function, which is minimized when the value of PC is high and the value of (Intra_cluster + SC) is low.

Fitness = (Intra_Cluster + SC) / PC

- **Intra Cluster Distance**: this metrics measures the compactness of the clusters. The goal is to minimize the distance between data points and their assigned cluster centers.
- **Partition Coefficient (PC)**: This metric quantifies the overlap between clusters, with higher values indicating better-defined clusters.

- **Classification Entropy (CE)**: Like PC, CE measures the fuzziness in cluster assignments. Lower values are preferred as they suggest more distinct cluster boundaries.
- **Partition Index (SC)**: This index measures cluster validity based on individual cluster characteristics, normalized by the fuzzy cardinality of each cluster. A higher SC value indicates better separation between clusters.
  To have more control and flexibility, we weight each term of the fitness function with scaling coefficient alpha, beta and zeta and defined an improved fitness function:

$$\text{fitness} = \alpha \cdot \text{intraCluster} + \beta \cdot SC + \zeta \cdot \left(\frac{1}{PC} + CE \right)$$

Dry Run (FCMWithWindow.m)

We load an MRI image and focus on a specific slice of the brain to compare the clustering performance between segmenting the slice using the FCM algorithm alone and the combined BAT+FCM algorithm (MFBAFCM).
We apply median filtering to the image to smooth it and reduce the noise, followed by scaling the intensity values between 0 and 1 (Figure 1). Using the processed image, we create various feature vectors with a moving window approach, incorporating padding and filtering. To capture the intensity gradient landscape and preserve edges for tumor segmentation, we experimented with different filters, including Prewitt and Laplacian operators. Ultimately, we found that a simple 3x3 sliding kernel filter provided the best clustering performance. However, more in-depth exploration of window designs will be necessary in future work.
We tried different parameters for FCM and MFBAFCM and use in final:

Parameter	Value
nClusters	15
FCM fuzzy exponent	2
FCM max. iterations	10
FCM distance metric	Euclidean
FCM min. improvement	10^(-5)
Fitness Alpha	1
Fitness Beta	0.5
Fitness Zeta	1.5
Bat number	50
Bat max. iterations	10
Bat f_{min}, f_{max}	0 and 2
Bat loudness coefficient	0.9
Bat Gamma	0.95

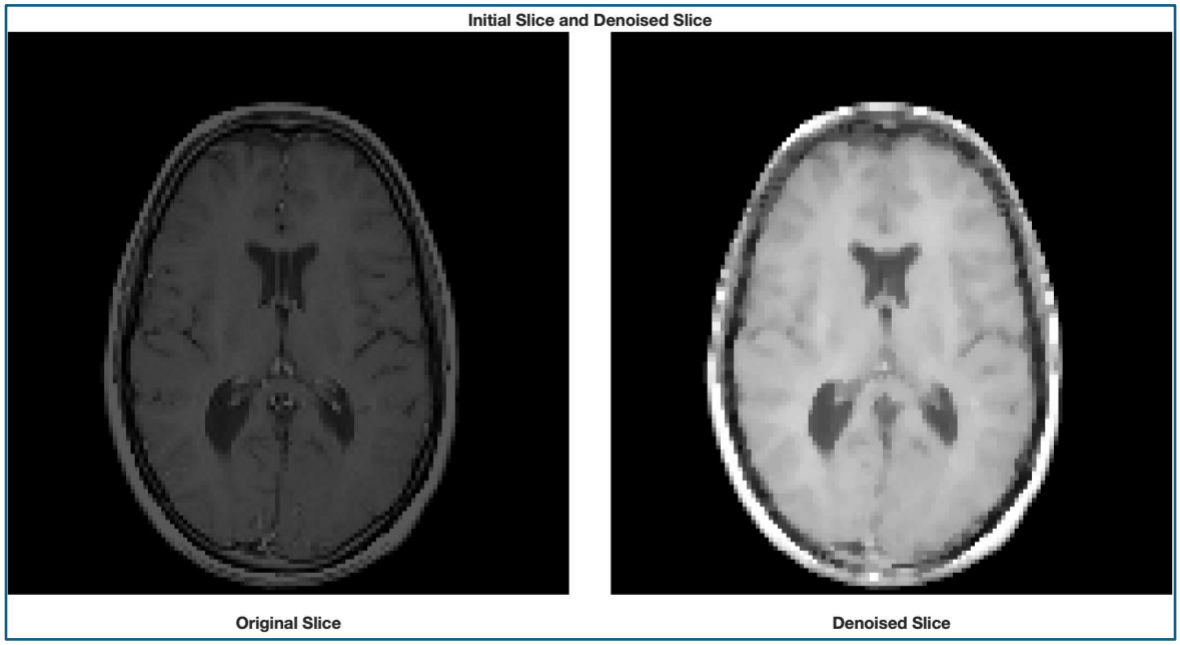


Figure 1: original and denoised slice.

- MFBAFCM clustering metrics are better than FCM: PC is larger, and CE, SC and S values of MFBAFCM are smallest than FCM. The proposed method MFBAFCM provides a better separation of the clusters.

Algorithm	PC	CE	SC	S
FCM	0.654	0.892	0.001	1771.868
BAT + FCM	0.667	0.865	0.001	1.759

Fuzzy Separation Index (S): This metric assesses the separation quality of clusters. Thus, minimizing S ensures that the clusters are both compact and well-separated, which is the desired goal in clustering.

S = (Intra-cluster-compactness) / [N . minimum inter-cluster-distance]

Notice how the metric S is smaller with the BAT+FCM algorithm compared to FCM alone, indicating that MFBAFCM achieves better clustering performance.

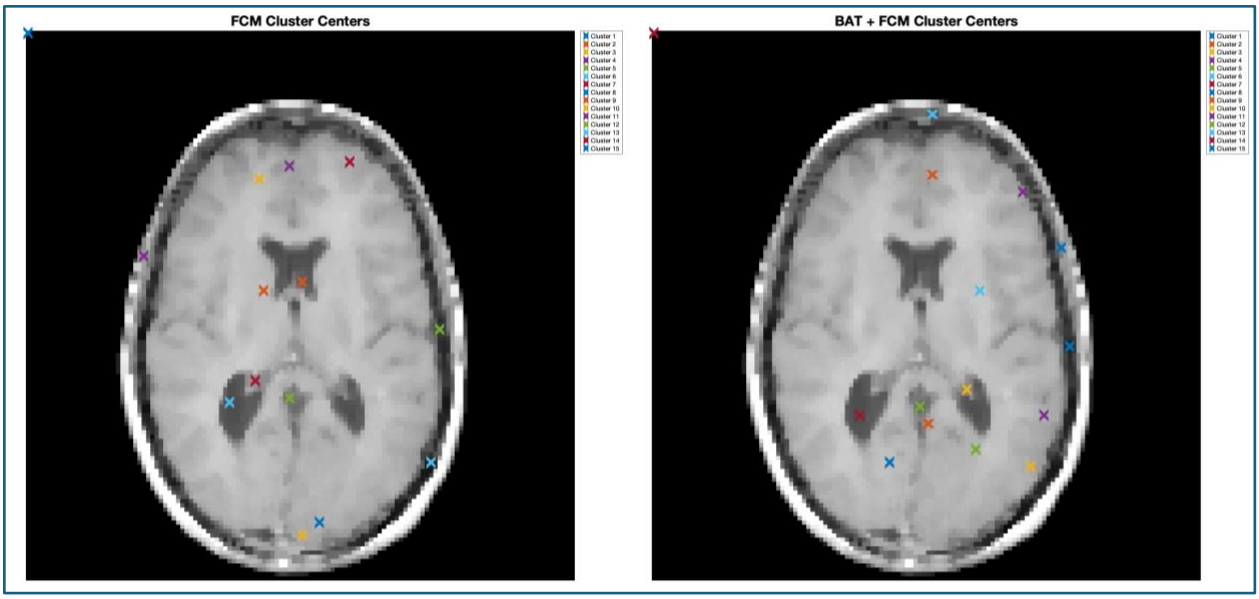


Figure 2: cluster centers on FCM segmented image (left) and BAT+FCM (right).

When looking at figure 2, we observe that:

- **FCM (left image):** The cluster centers appear to be more dispersed, with some potentially located in less relevant regions of the brain slice. The cluster centers appear less compact and more spread out, possibly leading to higher intra-cluster distances and reduced clustering quality. This dispersion may indicate that FCM alone is more sensitive to initial conditions and lacks the optimization power to fine-tune cluster center placement effectively.
- **BAT+FCM (right image)**: The cluster centers are more concentrated in relevant regions, with fewer centers placed in areas without significant image features. The cluster centers are more compact and strategically positioned, which likely reduces intra-cluster distances and improves separation between clusters. This suggests that BAT+FCM provides a more optimized initialization, allowing the algorithm to focus on meaningful regions of the image.

Looking at side by side the segmented slice using FCM alone or BAT+FCM; confirms further that with BAT+FCM, the segmentation is more refined, with sharper and more distinct boundaries between clusters, indicating better clustering performance (Figure 3). The results suggest that the BAT+FCM algorithm (MFBAFCM) leads to better-defined clusters and is more suitable for tasks requiring high-quality segmentation, such as medical image analysis.

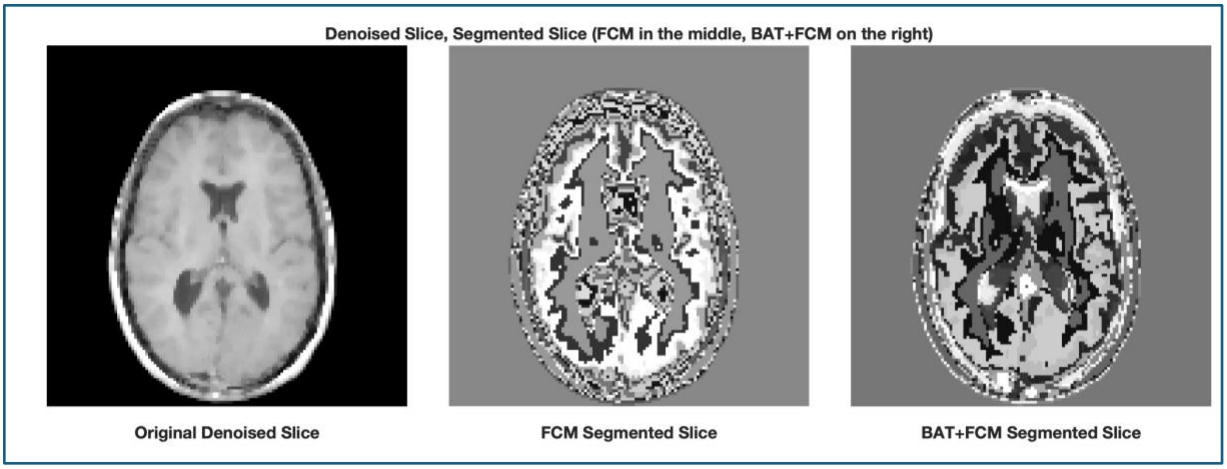


Figure 3: denoised slice (left), denoised segmented slice with FCM (middle), and MFBAFCM (right)

Automatic Segmentation of Vestibular Schwannoma from Magnetic Resonance Imaging (fcm_predict_tumor.m)

This dataset includes MRI images of 242 patients diagnosed with **vestibular schwannoma (VS)**. The patients underwent **Gamma Knife stereotactic radiosurgery (GK SRS)**. The focus is on **T1-weighted images**. The dataset is part of **The Cancer Imaging Archive (TCIA)**, an open dataset repository.

Initial Preprocessing

1. This dataset first was processed using the python script TCIA_data_convert_into_convenient_folder_structure.py provided in the GitHub repo: https://github.com/KCL-BMEIS/VS_Seg/tree/master. The script converts the dataset into a standardized folder structure with clearer organization and file naming.
2. We developed a custom Python script *process_DICOM_create_mask.py* which processes the DICOM files, to reconstitute the brain images while creating a binary mask using the RTSS DICOM file indicating whether a pixel belongs to a tumor. The script also, aligns the generated mask with the tumor in the brain image using padding or cropping techniques. However, we obtained better tumor annotation with the padding method. Finally, both masks and brain images were saved into NIFTI format.
3. The mask and brain images are divided into training and test datasets, with 80% of the images and masks allocated to the training set (52 images) and 20% to the test set (13 images).
4. To determine the range of pixel intensities of the tumors in the training dataset (**tumorIntensityRange**) , the mean and the standard deviations of the radiuses of the bounding boxes around these tumors (**roiCenter**) and the min. and max. areas of the training set tumor areas (**minArea**, **maxArea**), we compute the following for each image in the training set:
    a. Identify the slice with the largest tumor region

b. Calculate the centroid, bounding box, and area of the tumor region in this slice
c. Store the computed data for further analysis

Tumor Segmentation

For each image in the test dataset, we identify the slice with the largest tumor area in the binary tumor mask. The detection function, **fcm_predict_tumor,** is then called with the test image and the statistical parameters: **tumorIntensityRange, minArea, maxArea, roiCenter, and roiRadius**; learned during the training step.

fcm_predict_tumor

This function applies either **FCM alone** or **BAT + FCM**, depending on the useBAT option provided as an input parameter. The steps are as follows:

1. **Cluster Selection**: Select the cluster that falls within the specified tumor intensity range (tumorIntensityRange).
2. **Region Restriction**: Restrict the predicted tumor mask to the region of interest (ROI) defined during training.
3. **Area Filtering**: Filter the detected tumor mask using minimum and maximum area constraints (**bwareafilt** function in MATLAB).
4. **Mask Dilation**: Dilate the detected tumor mask based on the training-derived statistical parameters: mean and standard deviation of the radius of the tumor bounding box (**imdilate** function in MATLAB).

Results

The results are not perfect as we can see in the plots below (Fig 4,5 and 6):

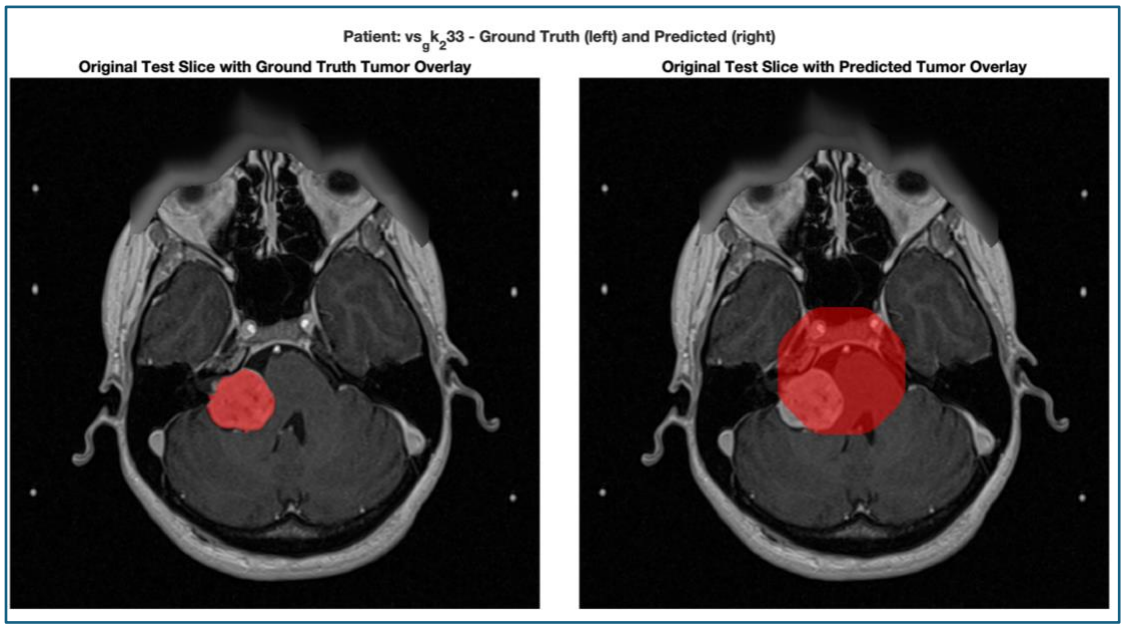


Figure 4: Patient 233 - The tumor appears reasonably localized, but the predicted mask significantly overestimates the tumor size, especially along the boundaries.

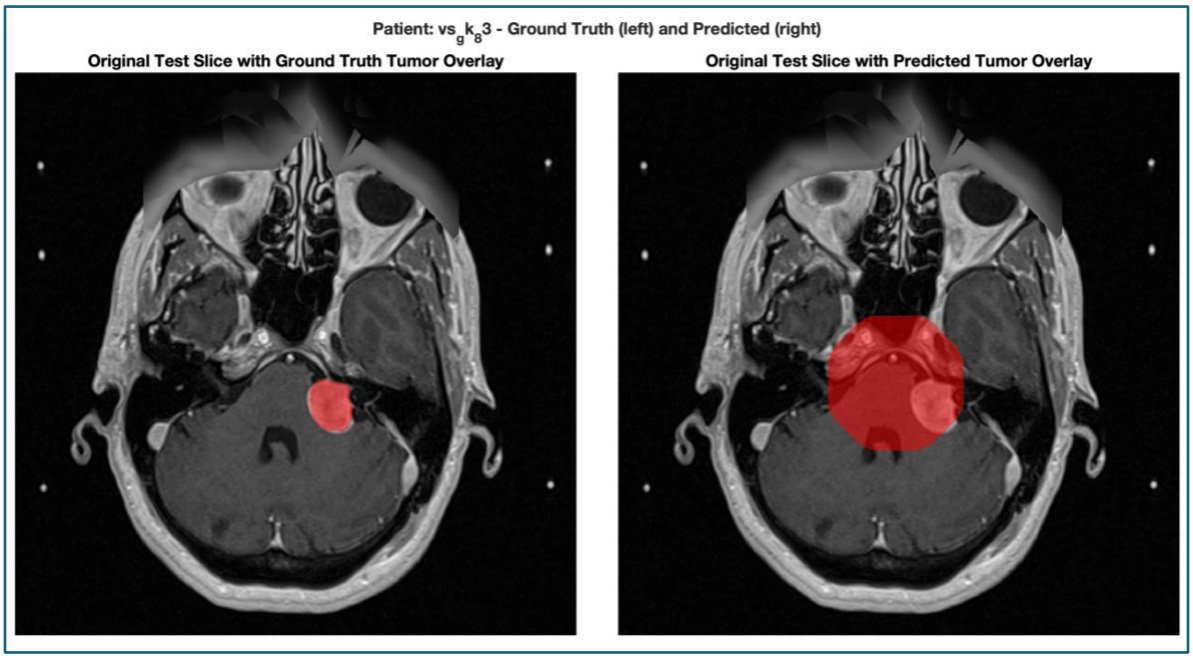


Figure 5: Patient 83 - the prediction seems overly generous, extending to areas outside the tumor. Slight under-segmentation in certain regions can also be observed.

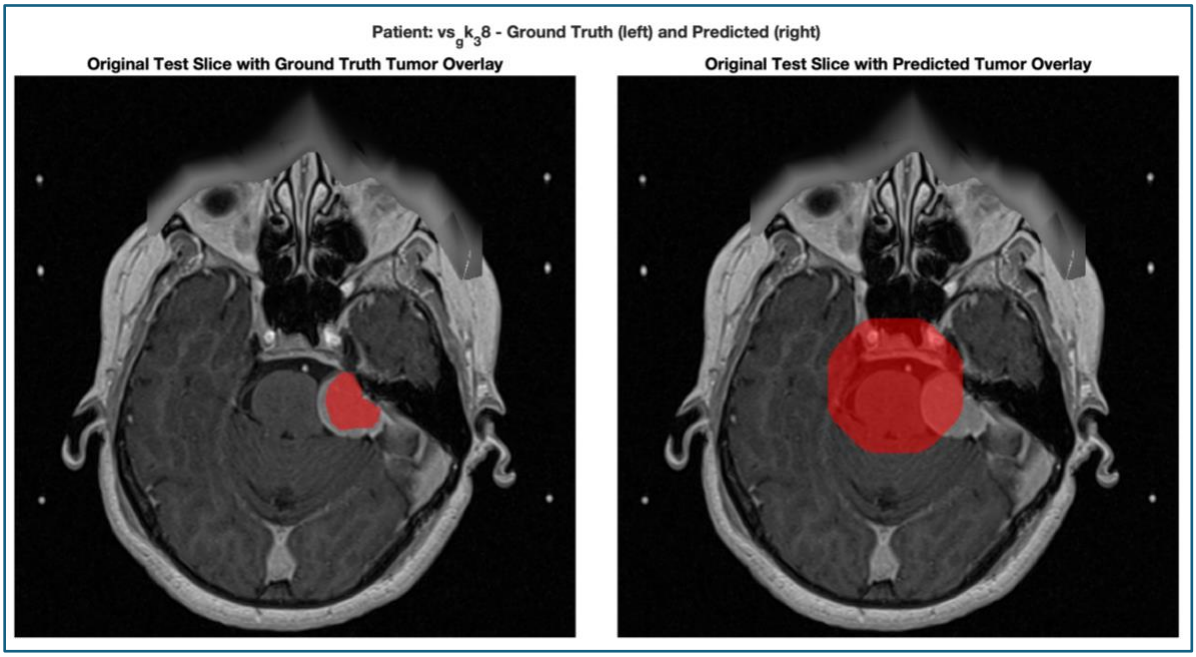


Figure 6: Patient 38 - Similar over-segmentation is observed, with predictions extending beyond the actual tumor region.

The model shows promise in correctly localizing tumors, but over-segmentation and boundary issues reduce its accuracy. Fine-tuning the intensity thresholds, clustering steps, and post-processing (e.g., dilation) could help improve the results. Additionally, stricter constraints on the ROI and better statistical parameter optimization from the training phase might address the over-segmentation problem.

Metrics – Key Take Away

- Both methods have moderate sensitivity (0.66), meaning they correctly identify 66% of actual tumor pixels. While this is reasonable, there is room for improvement to reduce false negatives.
- Both methods show very low PPV (0.08), meaning that only 8% of predicted tumor pixels are tumors. This suggests significant over-segmentation or inclusion of false positives in the tumor predictions.
- Both methods have very high NPV (0.99), meaning that almost all pixels predicted as non-tumor are indeed non-tumor. This aligns with the high specificity.
- The Area Under the Curve (AUC) score of 0.80 indicates decent overall classification performance when considering both tumor and non-tumor regions. However, it is not excellent.
- Jaccard and DICE Similarity Coefficients:

Both overlap metrics are very low, showing poor agreement between the predicted and ground truth tumor masks. This indicates that while the models correctly identify the presence of tumors (e.g., in terms of accuracy and sensitivity), the spatial localization and boundaries of the predicted tumors are not accurate. Metrics are computed in the function **evaluateTestSetSlice**.

Metric	FCM	BAT + FCM
Accuracy	0.95	0.95
Sensitivity	0.66	0.66
Specificity	0.95	0.95
PPV	0.08	0.08
NPV	0.99	.0.99
AUC	0.80	0.80
Jaccard Coefficient	0.07	0.07
DICE Coefficient	0.14	0.14

To improve classification and tumor detection, we applied histogram matching using patient 5 as the reference image. Patient 5's predicted tumor showed almost perfect overlap with the ground truth, making it a suitable choice for intensity normalization. However, the overall results were not better after histogram matching (fig. 7 and 8), as reflected in the evaluation metrics. Further investigation is required to understand the reasons for this performance.

Metric	FCM	BAT + FCM
Accuracy	0.95	0.95
Sensitivity	0.66	0.66
Specificity	0.95	0.95
PPV	0.08	0.08
NPV	0.99	.0.99
AUC	0.80	0.80
Jaccard Coefficient	0.07	0.07
DICE Coefficient	0.14	0.14

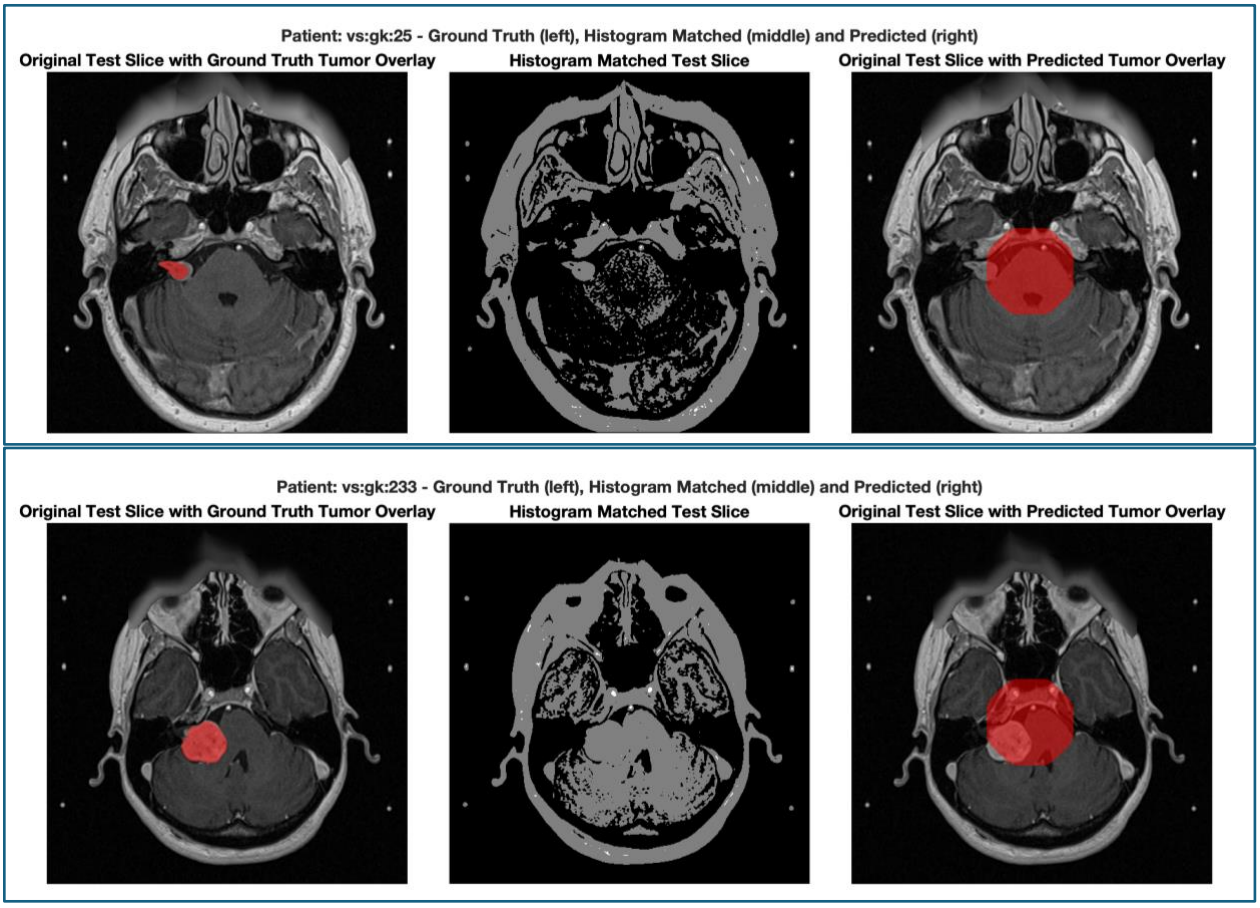
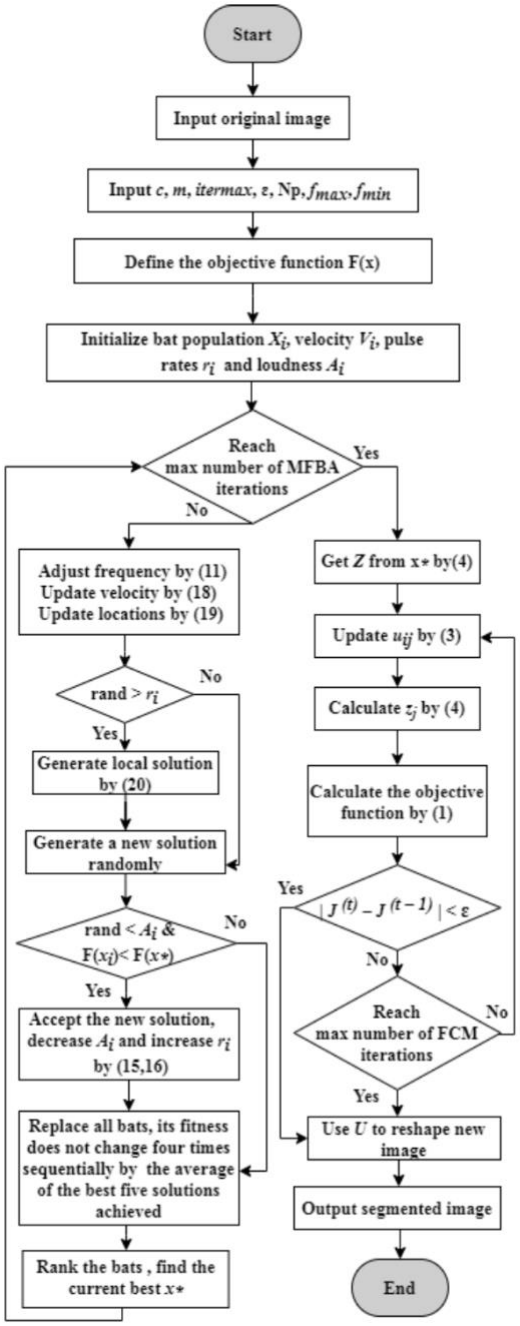


Figure 7 & 8: tumor detection using histogram matching and BAT+FCM.



Pseudocode 2: MFBAFCM flow chart

[1] Gandomi A. H. and Yang X.-S., Chaotic bat algorithm, *Journal of Computational Science*. (2014) **5**, no. 2, 224–232, https://doi.org/10.1016/j.jocs.2013.10.002, 2-s2.0-84897588368.