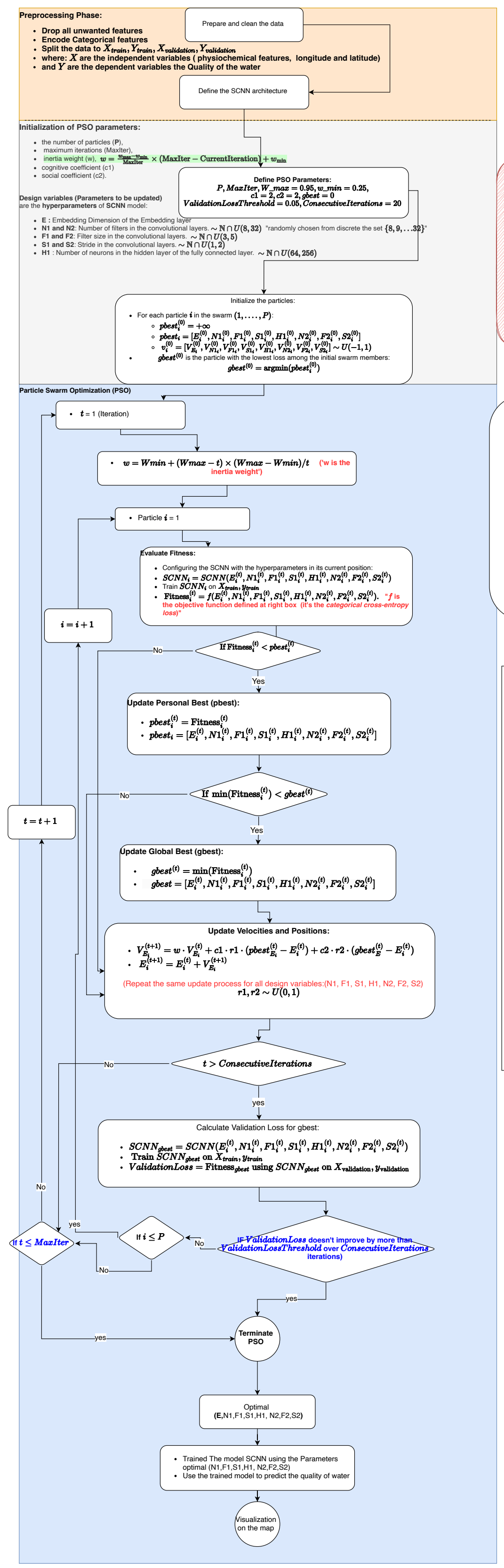

Preprocessing Phase:
Drop all unwanted features
Encode Categorical features
Split the data to $X_{train}, Y_{train}, X_{validation}, Y_{validation}$
where: X are the independent variables (physiochemical features, longitude and latitude)
and Y are the dependent variables the Quality of the water
Prepare and clean the data
Define the SCNN architecture
Initialization of PSO parameters:
the number of particles (P),
maximum iterations (MaxIter),
inertia weight (w), $w = \frac{w_{max}-w_{min}}{MaxIter} \times (MaxIter - CurrentIteration) + w_{min}$
cognitive coefficient (c1)
social coefficient (c2).
Define PSO Parameters:
$P, MaxIter, W\_max = 0.95, w\_min = 0.25, c1 = 2, c2 = 2, gbest = 0$
$ValidationLossThreshold = 0.05, ConsecutiveIterations = 20$
Design variables (Parameters to be updated) are the hyperparameters of SCNN model:
E : Embedding Dimension of the Embedding layer
N1 and N2: Number of filters in the convolutional layers. $\sim \mathbb{N} \cap U(8,32)$ "randomly chosen from discrete the set $\{8, 9, \ldots 32\}$"
F1 and F2: Filter size in the convolutional layers. $\sim \mathbb{N} \cap U(3,5)$
S1 and S2: Stride in the convolutional layers. $\sim \mathbb{N} \cap U(1,2)$
H1 : Number of neurons in the hidden layer of the fully connected layer. $\sim \mathbb{N} \cap U(64,256)$
Initialize the particles:
For each particle i in the swarm $(1, \ldots, P)$:
$pbest_i^{(0)} = +\infty$
$pbest_i = [E_i^{(0)}, N1_i^{(0)}, F1_i^{(0)}, S1_i^{(0)}, H1_i^{(0)}, N2_i^{(0)}, F2_i^{(0)}, S2_i^{(0)}]$
$v_i^{(0)} = [V_{E_i}^{(0)}, V_{N1_i}^{(0)}, V_{F1_i}^{(0)}, V_{S1_i}^{(0)}, V_{H1_i}^{(0)}, V_{N2_i}^{(0)}, V_{F2_i}^{(0)}, V_{S2_i}^{(0)}] \sim U(-1,1)$
$gbest^{(0)}$ is the particle with the lowest loss among the initial swarm members:
$gbest^{(0)} = \text{argmin}(pbest_i^{(0)})$
Particle Swarm Optimization (PSO)
t = 1 (Iteration)
$w = Wmin + (Wmax - t) \times (Wmax - Wmin)/t$ ('w is the inertia weight')
Particle i = 1
Evaluate Fitness:
Configuring the SCNN with the hyperparameters in its current position:
$SCNN_i = SCNN(E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)})$
Train $SCNN_i$ on X_{train}, y_{train}
$\text{Fitness}_i^{(t)} = f(E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)})$. "$f$ is the objective function defined at right box (it's the categorical cross-entropy loss)"
$i = i + 1$
If $\text{Fitness}_i^{(t)} < pbest_i^{(t)}$
No
Yes
Update Personal Best (pbest):
$pbest_i^{(t)} = \text{Fitness}_i^{(t)}$
$pbest_i = [E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)}]$
$t = t + 1$
If $\min(\text{Fitness}_i^{(t)}) < gbest^{(t)}$
No
Yes
Update Global Best (gbest):
$gbest^{(t)} = \min(\text{Fitness}_i^{(t)})$
$gbest = [E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)}]$
Update Velocities and Positions:
$V_{E_i}^{(t+1)} = w \cdot V_{E_i}^{(t)} + c1 \cdot r1 \cdot (pbest_{E_i}^{(t)} - E_i^{(t)}) + c2 \cdot r2 \cdot (gbest_E^{(t)} - E_i^{(t)})$
$E_i^{(t+1)} = E_i^{(t)} + V_{E_i}^{(t+1)}$
(Repeat the same update process for all design variables:(N1, F1, S1, H1, N2, F2, S2)
$r1, r2 \sim U(0,1)$
$t > ConsecutiveIterations$
No
yes
Calculate Validation Loss for gbest:
$SCNN_{gbest} = SCNN(E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)})$
Train $SCNN_{gbest}$ on X_{train}, y_{train}
$ValidationLoss = \text{Fitness}_{gbest}$ using $SCNN_{gbest}$ on $X_{validation}, y_{validation}$
No
yes
If $t \le MaxIter$
If $i \le P$
No
No
IF $ValidationLoss$ doesn't improve by more than $ValidationLossThreshold$ over $ConsecutiveIterations$ iterations)
yes
yes
Terminate PSO
Optimal (E,N1,F1,S1,H1, N2,F2,S2)
Trained The model SCNN using the Parameters optimal (N1,F1,S1,H1, N2,F2,S2)
Use the trained model to predict the quality of water
Visualization on the map

- **Design variables (Parameters to be updated)** are the **hyperparameters (E, N1, N2, F1,F2,S1,S2, and H1)** of **SCNN** model.
- **Encode representation of a particle:** we represent a particle by **an array or vector** in which each element corresponds to one of the **design variables (E, N1, N2, F1,F2,S1,S2, and H1)** for SCNN.
- **The objective function**, denoted as "**f(E, N1,F1,S1,H1, N2,F2,S2)**," represents the mathematical expression that measures the classification accuracy of the groundwater quality classification model based on the given hyperparameters for the Spatial Convolutional Neural Network (SCNN)
- we have a multi-class classification problem, one commonly used **objective function** is the **categorical cross-entropy loss**.

$$f(E, N1, F1, S1, H1, N2, F2, S2) = \frac{-1}{m} \sum_{i=1}^{m} \sum_{c=1}^{4} y_{i,c} \times \log(p_{i,c}); \text{ where } p_{i,c} = SCNN(E, N1, F1, S1, H1, N2, F2, S2)(X_i)$$

"**we are using categorical cross-entropy loss (multiclass) not binary cross-entropy loss (2 classes)**"

- where: m is the number of data samples. $y_{i,c}$ is an indicator that is 1 if the true label of the ith sample is in class c (e.g., 1 for "Excellent" if the true label is "Excellent"), and 0 otherwise. $SCNN(E, N1, F1, S1, H1, N2, F2, S2,)_{i,c}$: use the $SCNN_{i,c}$ model trained using **(E, N1,F1,S1,H1, N2,F2,S2)** hyperparameters to predict the correspondent labels of the sample ith in dataset.
- **The fitness function** Is the categorical cross-entropy loss of SCNN model. $\text{Fitness}_i^{(t)} = f(E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)})$

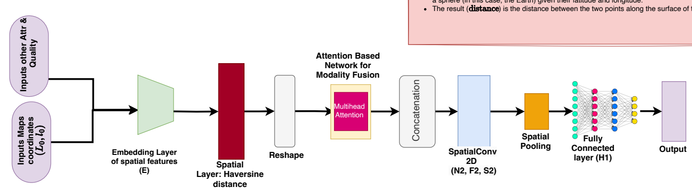
- **Evolution of Pbest and gbest:**
  - **Update Personal Best (pbest):** For each particle i, compare its fitness to its personal best fitness ($pbest_i$).
    - If the fitness of the current position is better than the previous personal best, update $pbest_i$ and $pbest_i^{(t)}$ with the current position and fitness.
    - If $\text{Fitness}_i^{(t)} < pbest_i^{(t)}$, then $pbest_i^{(t)} = \text{Fitness}_i^{(t)}$ and $pbest_i = [E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)}]$
  - **Update Global Best (gbest):** Determine the particle with the best fitness among all particles in the current **iteration t**.
    - If this particle has a better fitness than the current global best fitness ($gbest^{(t)}$), update $gbest^{(t)}$ with the fitness and position of the best particle.
    - If $\min(\text{Fitness}_i^{(t)}) < gbest^{(t)}$, then $gbest^{(t)} = \min(\text{Fitness}_i^{(t)})$, and $gbest = [E_i^{(t)}, N1_i^{(t)}, F1_i^{(t)}, S1_i^{(t)}, H1_i^{(t)}, N2_i^{(t)}, F2_i^{(t)}, S2_i^{(t)}]$
- **Stopping conditions:**
  - The algorithm is terminated when the convergence criteria are met or the maximum iteration ($maxIter$) is reached.
    - Convergence criteria :
      - $If Iteration > ConsecutiveIterations$ (example 20) :
        - **Calculate ValidationLoss for gbest**:
        - Train $SCNN$ on X_{train}, y_{train} using $(gbest = [E, N1, F1, S1, H1, N2, F2, S2])$
        - *ValidationLoss* = Fitness_{gbest} using $SCNN_{gbest}$ on $X_{validation}, y_{validation}$
      - *If (**ValidationLoss** doesn't improve by more than **ValidationLossThreshold** over **ConsecutiveIterations** iterations) or (or the maximum iteration ($maxIter$) is reached): Terminate PSO*

- The ***Haversine distance*** between two points on the Earth specified by their latitude (L_0, L_1) and longitude (l_0, l_1) can be calculated using the Haversine formula.
- The formula is given by:

$$a = \sin^2(\tfrac{\Delta L}{2}) + \cos(L_0) \cdot \cos(L_1) \cdot \sin^2(\tfrac{\Delta l}{2})$$
$$c = 2 \cdot \text{atan2}\left(\sqrt{a}, \sqrt{1-a}\right)$$
$$\text{distance} = R \cdot c$$

where:
- $\Delta L = L_1 - L_0$
- $\Delta l = l_1 - l_0$
- R is the Earth's radius (mean radius =6371000 m)
- atan2 is the arctangent function with two arguments.

- The ***Haversine distance*** calculates the great-circle distance between two points on the surface of a sphere (in this case, the Earth) given their latitude and longitude.
- The result (distance) is the distance between the two points along the surface of the sphere.


Inputs other Attr & Quality
Inputs Maps coordinates (L_0, l_0)
Embedding Layer of spatial features (E)
Spatial Layer: Haversine distance
Reshape
Attention Based Network for Modality Fusion
Multihead Attention
Concatenation
SpatialConv 2D (N2, F2, S2)
Spatial Pooling
Fully Connected layer (H1)
Output