Algorithm: Spatial-channel collaborative multi-scale graph interaction deep transfer learning

Input:

- Source domains: S = { S_1, S_2,..., S_M }
- Target domain: T
- Batch size: N_m
- Number of categories: K
- Number of scales: H
- Spatial-channel refinement parameters

Output:

- Classification predictions for target domain samples

// ===== **3.1: Spatial-Channel Collaborative Prototype Extraction Module** =====

for each batch (X_src, y_src) from S and (X_tgt, y_tgt) from T do

// Spatial refinement

$$X_{\text{out}} = \beta + \frac{\psi(X-\mu)}{\sqrt{\sigma^2+\varepsilon}} \quad \text{// Eq. (1)}$$

γ = learnable_parameter in GN

$$W_\gamma = \{w_i\} = \gamma_i / \left(\sum_{j=1}^{D}\gamma_j\right), \quad i,j = 1,2,\ldots,D \quad \text{// Eq. (2)}$$

W = Gate(Sigmoid(W_γ))

W_1, W_2 = threshold_gate(W)

$$X^{w1} = X_{11}^{w} \oplus X_{22}^{w} \quad \text{// Eq. (3)}$$

$$X^{w2} = X_{21}^{w} \oplus X_{12}^{w} \quad \text{// Eq. (4)}$$

X^w = Concat(X^{w1} , X^{w2}) // Refined spatial feature

// Channel refinement

(X_{up} , X_{low}) = Split(X^w , ratio= α) // α is splitting ratio

$$Y_1 = X_{up}\left(\lambda^{GWC} + \lambda^{PWC_1}\right) \quad \text{// Eq. (5)}$$

$$Y_2 = \lambda^{PWC_1} X_{low} \cup X_{low} \quad \text{// Eq. (6)}$$

O_1, O_2 = GlobalAveragePool(Y_1), GlobalAveragePool(Y_2)

I_1 = exp(O_1) / (exp(O_1) + exp(O_2)) // Eq. (7)

I_2 = exp(O_2) / (exp(O_1) + exp(O_2))

$Y = I_1 * Y_1 + I_2 * Y_2$ // Eq. (8) channel refined feature

// Extract prototypes for source domains

for each source domain m = 1 to M do

23: for each category k = 1 to K do

24: $\hat{c}_k^{S_m}$ = mean(f_S (x_i)) for all x_i in category k // Eq. (9)

25: end for

26: end for

27: // ===== Temporarily skip the target domain prototype update and proceed with graph construction and propagation =====

28: // Use the current target domain prototype (obtained from the previous batch update or initial value)

29: // If it is the first iteration, initialize to the source domain mean

30: if first_batch then

31: for k = 1 to K do

32: $\hat{c}_k^T$ = mean($\hat{c}_k^{S_m}$ for m=1,2,...,M)

33: end for

34: end if

35: **// ===== 3.2: Multi-Scale Graph Interaction Transfer Network =====**

36: // Step 1: Build multi-domain knowledge graph

37: $\mathbf{F} = \left[\underbrace{c_1^{S_1}\, c_2^{S_1} \cdots c_K^{S_1}}_{\text{prototypes of } S_1} \cdots \underbrace{c_1^{S_M}\, c_2^{S_M} \cdots c_K^{S_M}}_{\text{prototypes of } S_M}\ \underbrace{c_1^{T}\, c_2^{T} \cdots c_K^{T}}_{\text{prototypes of } T} \right]^{\mathrm{T}}$

38: for i, j in 1 to (M+1)*K do

39: $\mathbf{A}_{i,j} = G(\mathbf{F}_i^{\mathrm{T}}, \mathbf{F}_j^{\mathrm{T}})$ // Eq. (10)

40: end for

41: // Step 2: Build extended graph with query samples

42: for each query sample q in batch N_{m} do

43: $\overline{\mathbf{F}} = \left[\mathbf{F}^{\mathrm{T}} f(q_1)\, f(q_2) \cdots f(q_{|N_m|}) \right]^{\mathrm{T}}$ // Eq. (11)

44: $\mathbf{S}_{i,j} = G(\mathbf{F}_i^{\mathrm{T}}, f(q_j))$ // Eq. (12)

45: $\overline{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{S} \\ \mathbf{S}^{\mathrm{T}} & \mathbf{I} \end{bmatrix}$ // Eq. (13)

46: end for

47: // Step 3: Multi-scale graph interaction

48: X_list = [$\mathbf{X}^{(0)}$, $\mathbf{X}^{(1)}$, $\mathbf{X}^{(h)}$,…, $\mathbf{X}^{(H)}$]

49: for h = 1 to H do

50: $\mathbf{A}\mathbf{X}^{(h-1)} \rightarrow \mathbf{X}^{(h)}$, $\forall h$=1, 2, ..., H // Eq. (14)

51: X_list.append($\mathbf{X}^{(h)}$)

52: end for

53: // Dynamic interaction flow construction

54: for each node i = 1 to $|\overline{V}|$ do

55: $\mathbf{Z}_i = \xi\left(\mathbf{X}_i^{(1)} \| \mathbf{X}_i^{(2)} \| \ldots \| \mathbf{X}_i^{(H)}\right)$ // Eq. (15)

56: for each propagation step l do

57: $\mathbf{X}_i^{(l)} = \mathbf{X}_i^{(l)} \| \mathbf{Z}_i$ // Eq. (16)

58: $\tilde{w}_i(l) = Sig(\mathbf{X}_i^{(l)} \cdot s)$ // Eq. (17)

59: end for

60: // Normalize over scales

61: for l = 1 to H do

62: $w_i(l) = \dfrac{e^{\tilde{w}_i(l)}}{\sum_{h=0}^{H} e^{\tilde{w}_i(h)}}$ // Eq. (18)

63: end for

64: end for

65: // Build weight matrix and fuse features

66: for h = 1 to H do

67: $\eta_h[i] = w_i(h) \quad 1 \le i \le n$ for all nodes i // Eq. (19)

68: $\mathbf{W}_h = Diag(\eta_h)$ // Eq. (20)

69: end for

70: $\mathbf{U}_{\mathbf{X}} = \sum_{h=0}^{H} \mathbf{W}_h \mathbf{X}^{(h)}$ // Eq. (21)

71: // Step 4: Cross-domain diagnosis

72: $\mathbf{P} = f_H(\overline{\mathbf{F}}, \overline{\mathbf{A}})$ // Eq. (22) - classification probabilities

73: // Extract prototypes for target domain using confidence-thresholded pseudo-labeling

74: Now proceed with updating the target domain prototype (based on graph network output)

75: Extracting the classification probability of target samples from **P**

76: HC = empty_set // High confidence sample set

77: for j = 1 to N_target_samples do

78: // Node index of the target sample in the extended graph

79: node_idx = $|F|$ + j

80: p_t = **P**[node_idx] // The classification probability of the target sample

81: // Calculate entropy value

82: entropy_t = -Σ_{k=1}^K $p_t[k]$ * log($p_t[k]$)

83:

84: if entropy_t < threshold_entropy then

85: pseudo_label = argmax(p_t)

86: // Store sample features and pseudo labels

87: HC.add

88: end if

89: end for

90: // Update target domain prototype

91: for each category k = 1 to K do

92: // Collect all high confidence sample features for this category

93: features_k = {z | (z, label) in HC and label = k}

94:

95: if features_k not empty then

96: // Update the target domain prototype using high confidence samples

97: $\hat{c}_k^T$ = mean(features_k)

98: else

99: // If there are no high confidence samples, maintain the original value or use the source domain mean

100: // $\hat{c}_k^T$ remain unchanged

101: end if

102: end for

103: // ===== **3.3: Category Constraint Loss** =====

104: // Global constraint loss

105: $L_R^g = \frac{1}{(M+1)^4} \sum_{m,n,i,j=1}^{M+1} \| \mathbf{A}_{i,j} - \mathbf{A}_{m,n} \|_F$ // Eq. (23)

106: // Local constraint loss

107: $L_R^l = \frac{1}{|N_m|} \sum_{k=1}^{K} \left(\sum_{(x_i^T, \hat{y}_i^T) \in T_k} \| f(x_i^T) - c_k^T \|_2^2 + \sum_{m=1}^{M} \sum_{(x_i^{S_m}, y_i^{S_m}) \in \mathcal{S}_m^k} \| f(x_i^{S_m}) - c_k^{S_m} \|_2^2 \right)$ // Eq. (24)

108: // Classification losses

109: $L_C^s = \frac{\sum_{m=1}^{M} \left(\mathbb{E}_{(x_i^{S_m}, y_i^{S_m}) \in \mathcal{S}_m} L_{ce}(p(x_i^{S_m}), y_i^{S_m}) \right)}{M}$ // Eq. (25)

110: $L_C^p = \dfrac{\sum_{k=1}^{K} L_{ce}(p(c_k^T), k) + \sum_{m=1}^{M}\sum_{k=1}^{K} L_{ce}(p(c_k^{S_m}), k)}{K(M+1)}$ // Eq. (26)

111: $L_C^t = -\mathbb{E}_{(x_i^T, \hat{y}_i^T)\in T} \sum_{k=1}^{K} p(\hat{y}_i^T = k \mid x_i^T) \log p(\hat{y}_i^T = k \mid x_i^T)$ // Eq. (27)

112: // Total loss

113: $L_{CCL} = L_C + L_R,$ // Eq. (28)

114: // Optimization

115: Update f_S, f_H, λ, s, γ, etc. via backpropagation

116: // Inference on target domain

117: for each target sample x_t in T do

118: Compute extended graph with x_t

119: $p_t = f_H(\overline{\mathbf{F}}, \overline{\mathbf{A}})$

120: y_pred = argmax(p_t)

121: end for

122: return y_pred for all target samples