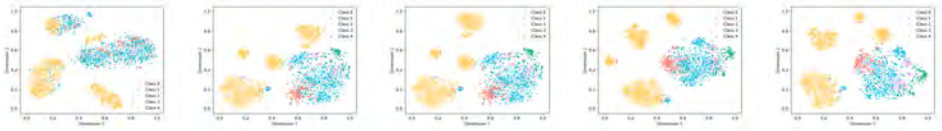
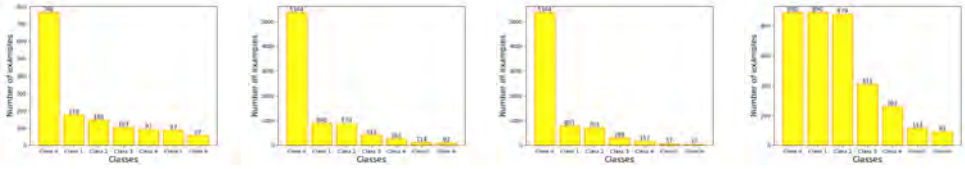
Supplementary



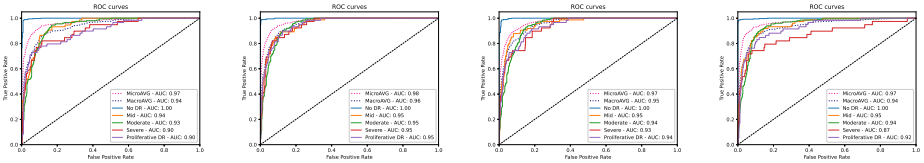
(a) Untrained (b) Scratch (c) KD (d) RKD (e) SCL-IKD

Fig. 1: The figures depict additional t-SNE plots for the APTOS dataset, illustrating different training techniques for the analysis of inter and intra-relations, as detailed in Section 3.2.



(a) Less Imbalance (b) Original Data (c) Severe imbalance (d) Balanced data

Fig. 2: The figures illustrate the composition of the datasets synthesized from HAM10000 for the analysis of class imbalance in Table 1.



(a) Scratch (M2) (b) CRCKD (c) SSD-KD (d) UDCD

Fig. 3: The figures display the ROC curves for the APTOS dataset, using various training techniques (only the most relevant and significant baselines) for KD analysis, as described in Section 3.3.

Algorithm 1: UDCD Algorithm

Input: Deep Neural Network S, Similar Teacher architecture T, Teacher and Student Memories M_t and M_s;

Parameters: Data in the form of batches $x \in X$, training epochs n, hyperparameters α_1, α_2, γ_1 and γ_2;

// From X, we synthesize x_s and x_t using different perturbations or noise

Output: $\hat{Y}_s$; // Output from an efficient version of Student Model S

Load the (X, Y) to M_s and M_t and (X_{test}, Y_{test}); // (X, Y): Train data, (X_{test}, Y_{test}): Testing data

Initialize weights of $S \leftarrow \omega$, weights of $T \leftarrow \omega'$; // ω and ω' are chosen randomly at initial stage

Apply perturbations ($Aug_t(.)$ and $Aug_s(.)$) on mini-batch x and provide inputs x_s and x_t to S and T respectively;

// Application of data augmentation on input

$i == 0$; // Initialize number of epochs

Function TRAIN_DISTIL($\mathcal{X}_t, \mathcal{X}_s, \mathcal{Y}, n, S, \delta$)**:**

    // use any optimization algorithm δ for optimizing the loss function , $\cdot$ represents the loss to be optimized for training the network

    do

        if $n <$ *warm_up* **then**

            $\mathcal{Y} = L_{wce} \cdot S(x_s)$;

            // Train S using weighted cross-entropy till it's warm-up epoch

        else

            Update $\omega' \leftarrow Exponential\_Moving\_Average(\omega)$; // Update teacher's weight with EMA of student's weight after warm up

            Extract $\boldsymbol{z_s}, \boldsymbol{z_t}$ and p_s, p_t from S and T; // By application of MLP layer and fully connected layers on the trained networks

            Calculate anchors $A^s(i)$ and $A^t(i)$ with the help of representation in M_s and M_t;

            Extract Relations $R(x_s, A_s(i))$ and $R(x_s, A_s(i))$ using $\boldsymbol{z_s}(x)$ and $\boldsymbol{z_t}(x)$ and Anchors

            Extract Unit Vectors to calculate L_{CRA}

            Calculate Supervised Contrastive Losses using $L^{(s)}_{SCL}$ and $L^{(t)}_{SCL}$;

            Calculate Categorical Relational Alignment using L_{CRA};

            Calculate Traditional Knowledge Distillation Loss L_{KL};

            Calculate the teacher's and Self Uncertainty using ψ^t_1 and psi^s_1;

            Add L_{SCL}, L_{CRA} and L_{KL} with L_{WCE} with the application of the uncertainty-based weightage to get overall loss L;

            $\mathcal{Y} = L \cdot S(x_s)$; // Train S using overall loss function

            Update $\omega' \leftarrow Exponential\_Moving\_Average(\omega)$;

            // Update teacher's weight with EMA of student's weight

            Optimize L using δ;

        $i++$;

    while *($i == n$)*

    return L

S = TRAIN_DISTIL $(x_t, x_s, Y, n, S, \delta)$; // Run the distillation training function to train the model

$\hat{Y}_s = S(X_{test})$; // Predict using trained model

Calculate Performance metrics using $\hat{Y}_s$ and Y_{test}; // End of algorithm