1 Time complexity of the SR-loss

1. For the semantic similarity:

   - Cosine Similarity matrix \((n \times n)\) : \(O(n^2 L)\), \(L\) length of semantic feature, \(n\) total classes
   - To get the \(K\) similar classes : \(O(n^2 \log K)\), \(O(n \log K)\) for each class, heap sort.
   - Overall cost : \(O(n^2 L) + O(n^2 \log K)\)

2. For the visual similarity:

   - Cosine Similarity matrix \((B \times K)\) : \(O(BKV)\), \(V\) length of visual feature, \(B\) batch size classes, \(K\) neighbour classes.
   - Overall cost : \(O(BKV E)\), \(E\) total number of epochs.

   - Time Complexity SR-loss (Clean Attributes): \(O(BKV E) + O(n^2 L) + O(n^2 \log K)\)
   - Time Complexity SR-loss (Noisy Text): \(O(BKV E) + O(n^2 LE) + O(n^2 \log KE)\)

Clearly, the overall time complexity is linear in terms of \(E, K, B, V,\) and \(L\) and degree 2 polynomial with \(\log\) in terms of the total classes. It is worth noticing that the complexity is not exponential and the running time cost is manageable.
2 Training Algorithm

Below, we illustrate our training procedure for the LsrGAN model. We train the Generator ($G_\theta$) and Discriminator ($D_\theta$) alternately with the Adam optimizer. Notice that the training of LsrGAN contains two phases, one for the seen classes and another for the unseen classes.

**Algorithm 1** Training procedure for the LsrGAN

1: **Input:** number of epochs $N_E$, the batch size $m$, discriminator iterations $N_d = 5$ for seen classes, loss hyper parameters $\lambda_c, \lambda_{vp}$ and $\lambda_{sr}$, $N_c = 1$ or 2 discriminator iterations for unseen classes, and Adam parameters $\beta_1 = 0.9$ and $\beta_2 = 0.9$.

2: **for** iter = 1, ..., $N_E$ **do**

3: // Seen Class Training

4: **for** $i = 1, ..., N_d$ **do**

5: Minibatch sampling from $T_s$ with matching images from $X_s$ and noise $Z$

6: $\hat{x} \leftarrow G_{\theta_g}(t_s, Z)$

7: Discriminator and classifier loss computation $L_d$ and $L_c$ using Eq. 2 and 3

8: $\theta_d \leftarrow \text{Adam}(\nabla_{\theta_d} L_d, \nabla_{\theta_c} L_c, \theta_d, \lambda_c, \beta_1, \beta_2)$

9: **end for**

10: Minibatch sampling from $T_s$ and noise $Z$

11: Generator loss computation $L_G$ using Eq. 8

12: $\hat{x} \leftarrow G_{\theta_g}(t_s, Z)$

13: $\theta_g \leftarrow \text{Adam}(\nabla_{\theta_g} L_c, \nabla_{\theta_c} L_{vp}, \nabla_{\theta_g} L_s^c, \theta_g, \lambda_c, \lambda_{vp}, \lambda_{sr}, \beta_1, \beta_2)$

14: // Unseen Class Training

15: **for** $i = 1, ..., N_c$ **do**

16: Minibatch sampling from $T_u$ and noise $Z$

17: $\hat{x} \leftarrow G_{\theta_g}(t_u, Z)$

18: Classifier loss computation $L_c$ using Eq. 3

19: $\theta_d' \leftarrow \text{Adam}(\nabla_{\theta_d'} L_c, \theta_d', \lambda_c, \beta_1, \beta_2)$

20: **end for**

21: Minibatch sampling from $T_u$ and noise $Z$

22: Generator loss computation $L_G$ using Eq. 8

23: $\hat{x} \leftarrow G_{\theta_g}(t_u, Z)$

24: $\theta_g' \leftarrow \text{Adam}(\nabla_{\theta_g'} L_c, \nabla_{\theta_c} L_{sr}, \theta_g', \lambda_c, \lambda_{sr}, \beta_1, \beta_2)$

25: **end for**