Linear Discriminant Analysis Metric Learning using Siamese Neural Networks

Network architecture: The network architecture used for training the Siamese Neural Network (SNN) is described in Table 1. The Convolutional Neural Network (CNN) architecture is shown below since the SNN consists of two parallel CNNs with shared weights.

Projection of feature vectors along eigenvector directions at the end of training: The projection of feature vectors at the end of training is shown in Figure 1. One can observe that the within-class scatter has reduced and between-class scatter has increased resulting in a class projection such that the classes are linearly separable along the projected discriminant directions.

For CIFAR-10 we used the dropout values of 0.05, 0.10, 0.15, and 0.20 with learning rates of 0.1, 0.01, and 0.001. The batch sizes used were, 100, 500, and 1000. The simulation results are summarized in Table 2. For STL-10, we measured the classification accuracy for batch sizes of 125, 200, and 250 and learning rates of 0.0001, 0.0002, and 0.0003. We used dropout values of 0.1, 0.3, 0.5, and 0.9. The highest accuracy obtained on this dataset is 71.62% which is the best value reported for LDA learning in STL-10 to the best of our knowledge. The results are summarized in Table 3.

For MNIST, we used batchsizes of 100, 500, and 750 and learning rate of 0.01, 0.02 and 0.03. The dropout was varied from 0.1, 0.3, 0.5, 0.9. The simulation results are summarized in Table 4. The main point to be noted here is that since the number of classes is 10, in all these datasets, our feature vector dimension is of length 9 which corresponds to the LDA discriminant directions.

Gradient computation for backpropagation: Here we summarize the gradient calculation of our proposed loss function for backpropogation. The KL divergence between two dissimilar class distributions p_a and p_b is given by:

$$D^{\mathbf{y}}_{KL}(p_a||p_b) = \frac{1}{2}\mathrm{tr}\left[\left(\mathbf{\Psi}^T\mathbf{\Sigma^x}\mathbf{\Psi}\right)^{-1}\mathbf{\Psi}^T\mathbf{C_{ba}}\mathbf{\Psi}\right]. \tag{1}$$

After LDA projection we have $\mathbf{\Psi}^T\mathbf{\Sigma^x}\mathbf{\Psi} = \mathbf{\Sigma^y}$ and $\mathbf{\Psi}^T\mathbf{C_{ba}}\mathbf{\Psi} = \mathbf{C^y_{ba}}$. The derivative of the KL divergence w.r.t the feature vector $\mathbf{y}$ is given by:

$$\frac{\partial}{\partial \mathbf{y}} D^{\mathbf{y}}_{KL}(p_a||p_b) = \left[\frac{\partial D^{\mathbf{y}}_{KL}}{\partial y_1} \quad \frac{\partial D^{\mathbf{y}}_{KL}}{\partial y_2} \cdots \frac{\partial D^{\mathbf{y}}_{KL}}{\partial y_{L-1}}\right]. \tag{2}$$

Let the feature vector $\mathbf{y}$ belong to class b and since for LDA we assume the covariance matrices to be equal, we have $\mathbf{\Sigma^y} = \mathbf{\Sigma^y_b}$. We computed the gradient w.r.t feature vector $\mathbf{y}$ belonging to class b as follows. The partial derivative of (1) w.r.t the i^{th} component of $\mathbf{y}$ is:

$$\frac{\partial D^{\mathbf{y}}_{KL}(p_a||p_b)}{\partial y_i} = \frac{\partial}{\partial y_i}\frac{1}{2}\mathrm{tr}\left[\mathbf{\Sigma^{y\,-1}_b} (\mathbf{u^y_b} - \mathbf{u^y_a})^T(\mathbf{u^y_b} - \mathbf{u^y_a})\right]. \tag{3}$$

The above derivative can be decomposed into two terms as:

$$\frac{\partial D^{\mathbf{y}}_{KL}(p_a||p_b)}{\partial y_i} = \frac{1}{2}\mathrm{tr}\left[\frac{\partial \mathbf{C^y_{ba}}}{\partial y_i}\cdot\mathbf{\Sigma^{y\,-1}_b} + \frac{\partial \mathbf{\Sigma^{y\,-1}_b}}{\partial y_i}\cdot\mathbf{C^y_{ba}}\right], \tag{4}$$

where $\mathbf{C^y_{ba}} = (\mathbf{u^y_b} - \mathbf{u^y_a})^T(\mathbf{u^y_b} - \mathbf{u^y_a})$. On expanding the above equation we have:

$$\frac{\partial D^{\mathbf{y}}_{KL}(p_a||p_b)}{\partial y_j} = \frac{1}{2}\mathbf{tr}\left[\frac{\partial \mathbf{C^y_{ba}}}{\partial y_j}\cdot\mathbf{\Sigma^{y\,-1}_b} - \mathbf{\Sigma^{y\,-1}_b}\cdot\frac{\partial \mathbf{\Sigma^y_b}}{\partial y_j}\cdot\mathbf{\Sigma^{y\,-1}_b}\cdot\mathbf{C^y_{ba}}\right]. \tag{5}$$

Since we consider that $\mathbf{y}$ belongs to class b, the derivative becomes

$$\frac{\partial D^{\mathbf{y}}_{KL}(p_a||p_b)}{\partial y_i} = \frac{1}{2}\mathrm{tr}\left[\frac{1}{Q_b}\left[\mathbf{u^{'}_b} - \mathbf{u^{'}_a}\right]\mathbf{\Sigma^{y\,-1}_b} - \mathbf{C^y_{ba}}\cdot\mathbf{\Sigma^{y\,-1}_b}\left[\frac{1}{Q}\left[\mathbf{y^{'}_i} - \mathbf{u^{'}_b}\right]\right]\mathbf{\Sigma^{y\,-1}_b}\right], \tag{6}$$

where Q_b, Q corresponds to number of images in class b and total number of images respectively.

MNIST	CIFAR-10	STL-10
Input $1 \times 28 \times 28$	Input $3 \times 32 \times 32$	Input $3 \times 96 \times 96$
3×3 conv, 32, BN-ReLU, (0.3)	3×3 conv, 64, BN-ReLU, (0.3)	3×3 conv, 64, BN-ReLU, (0.5)
3×3 conv, 32, BN-ReLU	3×3 conv, 64, BN-ReLU	3×3 conv, 64, BN-ReLU
2×2 max-pool	2×2 max-pool	2×2 max-pool
3×3 conv, 64, BN-ReLU, (0.4)	3×3 conv, 128, BN-ReLU, (0.4)	3×3 conv, 64, BN-ReLU, (0.5)
3×3 conv, 64, BN-ReLU	3×3 conv, 128, BN-ReLU	3×3 conv, 128, BN-ReLU
2×2 max-pool	2×2 max-pool	2×2 max-pool
3×3 conv, 128, BN-ReLU, (0.4)	3×3 conv, 256, BN-ReLU, (0.4)	3×3 conv, 128, BN-ReLU, (0.5)
3×3 conv, 128, BN-ReLU, (0.4)	3×3 conv, 256, BN-ReLU, (0.4)	3×3 conv, 256, BN-ReLU, (0.5)
2×2 max-pool	3×3 conv, 256, BN-ReLU	3×3 conv, 256, BN-ReLU
Linear $(128 - 128)$	2×2 max-pool	2×2 max-pool
Linear $(128 - 9)$	3×3 conv, 512, BN-ReLU, (0.4)	3×3 conv, 256, BN-ReLU, (0.5)
KL-divergence	3×3 conv, 512, BN-ReLU, (0.4)	3×3 conv, 512, BN-ReLU, (0.5)
LDA-Loss	3×3 conv, 512, BN-ReLU	3×3 kernel, 512, BN-ReLU
	2×2 max-pool	2×2 max-pool
	3×3 conv, 512, BN-ReLU, (0.4)	3×3 conv, 512, BN-ReLU, (0.5)
	3×3 conv, 512, BN-ReLU, (0.5)	3×3 conv, 512, BN-ReLU, (0.5)
	3×3 conv, 512, BN-ReLU	3×3 conv, 512, BN-ReLU
	2×2 max-pool	2×2 max-pool
	Linear $(512 - 512)$	Linear $(512 * 3 * 3 - 512)$
	Linear $(512 - 9)$	Linear $(512 - 9)$
	KL-divergence	KL-divergence
	LDA-Loss	LDA-Loss

Table 1: Network architecture used in our experiments for the three datasets. Conv - convolutional kernel, BN - Batch Normalization, ReLU - Rectified Linear Unit, % of introduced dropout is shown in brackets.

Lr	dropout	(bs 100)	(bs 500)	(bs 1000)
0.1	0.05	81.22	75.73	65.72
0.1	0.10	81.35	76.73	67.20
0.1	0.15	82.23	78.23	68.21
0.1	0.20	82.34	78.43	68.33
0.01	0.05	84.23	**88.17**	72.17
0.01	0.10	83.57	86.54	73.15
0.01	0.15	82.21	84.17	73.63
0.01	0.20	81.67	82.22	73.89
0.001	0.05	68.31	72.44	65.43
0.001	0.10	70.22	71.98	66.97
0.001	0.15	72.20	65.21	68.23
0.001	0.20	74.18	62.13	69.34

Table 2: Simulation results on CIFAR-10 for different batch sizes and dropout. Lr denotes the learning rate and bs denotes the batch size. Accuracy is given in %.

Lr	dropout	(bs 125)	(bs 200)	(bs 250)
0.003	0.1	64.25	**71.62**	62.77
0.003	0.3	58.17	67.41	57.69
0.003	0.5	52.59	64.81	46.22
0.003	0.9	41.19	63.71	42.77
0.001	0.1	63.21	62.34	61.32
0.001	0.3	58.23	59.33	58.90
0.001	0.5	55.22	57.68	55.98
0.001	0.9	55.10	59.98	57.98
0.002	0.1	57.21	59.93	56.23
0.002	0.3	55.32	58.58	54.43
0.002	0.5	53.29	56.64	54.23
0.002	0.9	52.59	57.71	55.55

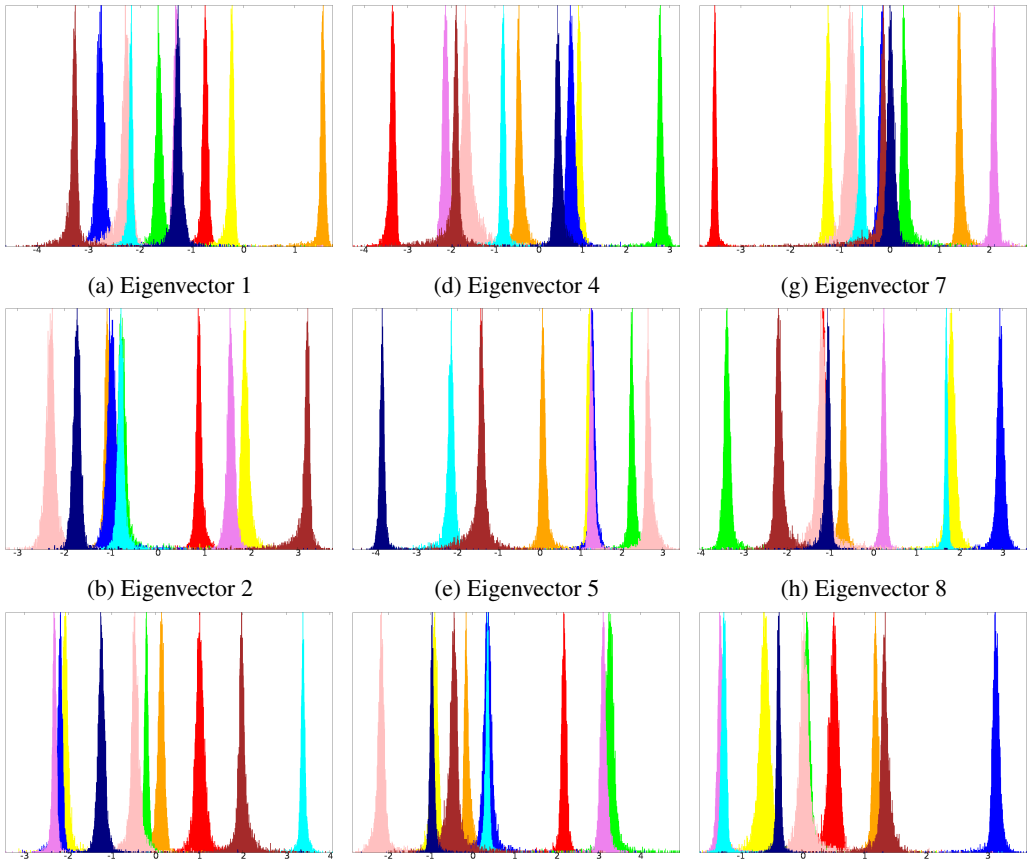
Table 3: Simulation results on STL-10 for different batch sizes and dropout. Lr denotes the learning rate and bs denotes the batch size. Accuracy is given in %.

$\mathbf{u_a^{'}}$, $\mathbf{u_b^{'}}$ and $\mathbf{y_i^{'}}$, are matrices given as:

$$\mathbf{u_l^{'}} = \begin{bmatrix} 0 & \cdots & u_{l1} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ u_{l1} & \cdots & 2u_{li} & \dots & u_{l_{L-1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & u_{l_{L-1}} & \dots & 0 \end{bmatrix}, \tag{7}$$

$$\mathbf{y_i^{'}} = \begin{bmatrix} 0 & \cdots & y_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ y_1 & \cdots & 2y_i & \dots & y_{L-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & y_{L-1} & \dots & 0 \end{bmatrix}, \tag{8}$$

where $\mathbf{l}$ indicates the class index. (6) corresponds to the derivative of the KL divergence w.r.t one component of the feature vector $\mathbf{y}$. The derivative w.r.t all the $L-1$ components can be computed by



(a) Eigenvector 1 (d) Eigenvector 4 (g) Eigenvector 7

(b) Eigenvector 2 (e) Eigenvector 5 (h) Eigenvector 8

(c) Eigenvector 3 (f) Eigenvector 6 (i) Eigenvector 9

Figure 1: Projection of image features along different eigenvector directions.

Lr	dropout	(bs 100)	(bs 500)	(bs 750)
0.01	0.1	98.71	99.57	99.43
0.01	0.3	99.12	**99.73**	99.33
0.01	0.5	99.61	99.53	99.55
0.01	0.9	99.11	99.54	99.64
0.02	0.1	98.73	98.45	98.23
0.02	0.3	97.34	97.98	97.55
0.02	0.5	98.54	98.59	98.19
0.02	0.9	99.06	98.47	98.89
0.03	0.1	98.43	98.33	98.56
0.03	0.3	98.59	98.58	98.87
0.03	0.5	98.45	98.49	98.51
0.03	0.9	99.03	99.07	98.55

Table 4: Simulation results on MNIST for different batch sizes and dropout. Lr denotes the learning rate and bs denotes the batch size. Accuracy is given in %.

concatenating the partial derivatives.