Deep Neural Maps

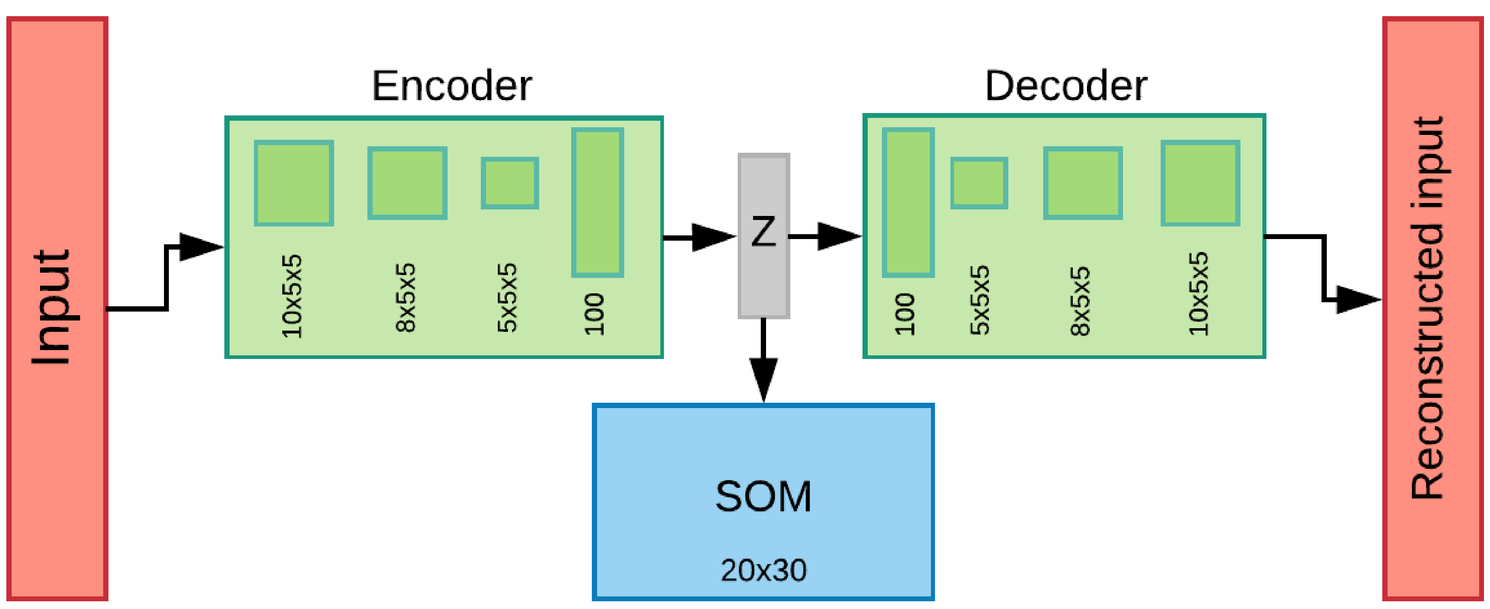
Mehran Pesteie, Purang Abolmaesumi, Robert Rohling
{mehranp, purang, rohling}@ece.ubc.ca

Introduction

Representation learning algorithms, whose goal is to capture the principal factors of the observed data [1], are a common technique for dimensionality reduction. However, the majority of these methods aim to yield an embedding of the data that are only efficient for clustering or classification [2, 3]. Hence, in order to interpret the embedding space, there is a need for an additional algorithm, such as t-SNE.
In this study, we introduce a method to learn and optimize an embedding space for visualization, called Deep Neural Maps (DNM).

DNM model

DNM model architecture:



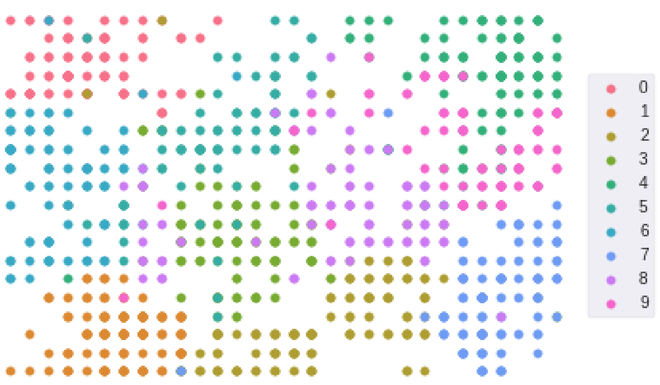
- The DNM model consists of a convolutional auto-encoder ($\mathcal{E}_\theta : X \rightarrow Z$, $\mathcal{D}_\phi : Z \rightarrow \widehat{X}$) and an SOM [4] ($\mathbf{W} = [w_j]$ for $j = 1, 2, ..., l$).
- The goal of DNM is to optimize θ and w_j such that $||Z_i - w_j||$ is minimized for all Z_i.

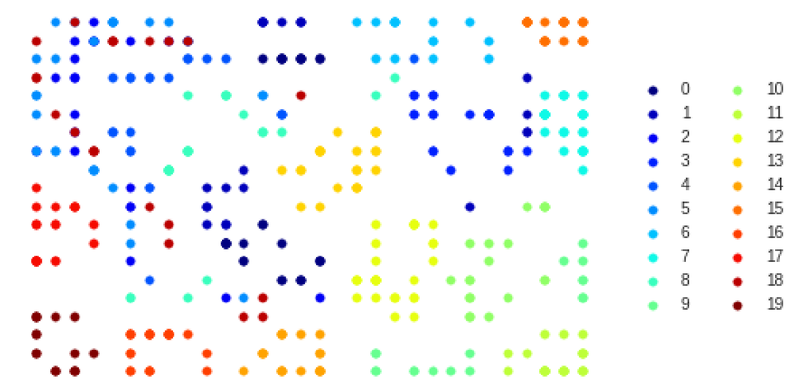
Experiments and results

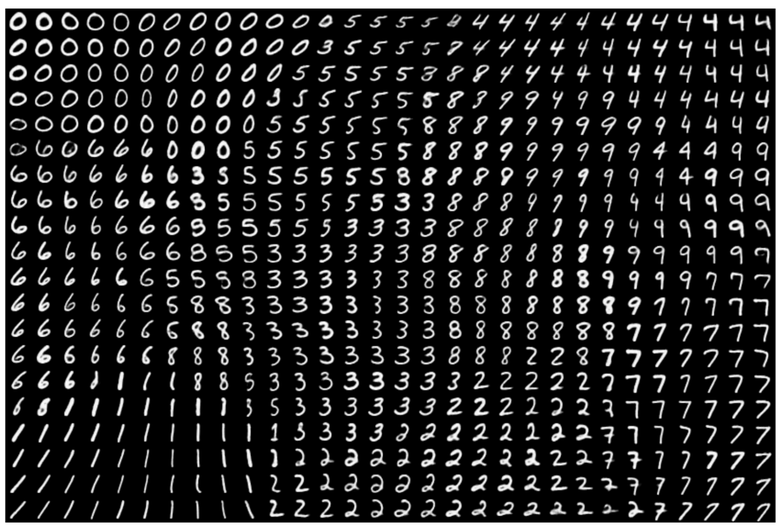
The auto-encoder filter dimensions are set to $(10 \times 5 \times 5)$-$(8 \times 5 \times 5)$-$(5 \times 5 \times 5)$-$dense100$ symmetrically for encoder and decoder. We set the lattice size of the SOM to 20×30, and pre-trained the auto-encoder and the SOM for 500 and 10000 iterations, respectively. We trained the DNM model on the following datasets:

- MNIST: Normalized, 60000 train and 10000 test images of handwritten digits of size 28×28 (a).
- COIL-20: Normalized, 1000 train, 440 test images of 20 objects re-sized to 64×64 (b).

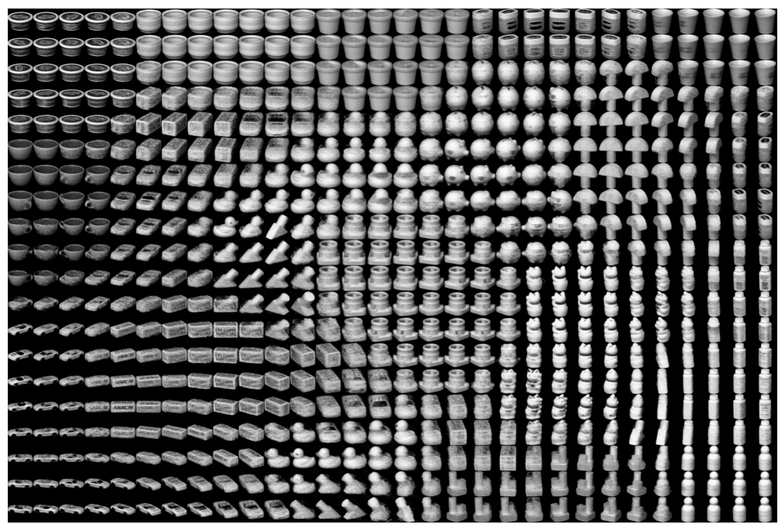
The DNM has separated each class of the data and mapped it to a particular location on the lattice, as seen as clear color separation. Moreover, it captured the variations within each class of data and mapped them to their corresponding location.







a



b

Pre-training

- Train the auto-encoder, $\mathcal{E}_\theta, \mathcal{D}_\phi$, on training dataset
- Compute the embedding Z using $\mathcal{E}_\theta$
- Train SOM weights $\mathbf{W}$ on Z
- Initialize the final model with θ, ϕ and $\mathbf{W}$

Training algorithm

Input: $\mathcal{D} \leftarrow$ training dataset, **epochs**
Output: $\mathcal{E}_\theta, \mathcal{D}_\phi, \mathbf{W} : [w_j]\ for\ j = 1, 2, ..., l$
Initialize $\theta, \phi, \mathbf{W}$ from pre-trained models
$\alpha \leftarrow 2000$, $\sigma_0 \leftarrow 10$, $\eta_0 \leftarrow 0.3$
for $n = 0$ to **epochs do**
  for each minibatch $\mathcal{B}$ in $\mathcal{D}$ **do**
    for each Z_i in $\mathcal{E}_\theta(\mathcal{B})$ **do**
      $u \leftarrow argmin||Z_i - \mathbf{W}||$
      $\sigma(n) = \sigma_0 exp\left(-\frac{n}{\alpha}\right)$
      $H_u(n) = exp\left(-\frac{||p_{w_j} - p_u||^2}{2\sigma^2(n)}\right)$
      $\eta_n = \eta_0 exp\left(-n/\alpha\right)$
      $q_{ij} = \frac{\left(1+||Z_i - w_j||^2\right)^{-1}}{\sum_{j'}\left(1+||Z_i - w_{j'}||^2\right)^{-1}}$,
      $p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_{j'} q_{ij'}^2 / \sum_i q_{ij'}}$
      $\mathbf{W}^{n+1} = \mathbf{W}^n + \eta_n H_u(n)(Z_i - \mathbf{W}^n)$
    end for
    $g \leftarrow grad(KLD(P||Q) + \gamma||X - \widehat{X}||_2^2 + \beta||\theta||_2)$
    $\theta, \phi \leftarrow Adam(g, \theta, \phi)$
  end for
end for

Acknowledgements

This study was jointly funded by grants from the Natural Sciences and Engineering Research Council of Canada (NSERC) and Canadian Institutes of Health Research (CIHR).

References

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] X. Guo, L. Gao, X. Liu, and J. Yin, "Improved deep embedded clustering with local structure preservation," in *International Joint Conference on Artificial Intelligence (IJCAI-17)*, pp. 1753–1759, 2017.

[3] C. Aytekin, X. Ni, F. Cricri, and E. Aksu, "Clustering and unsupervised anomaly detection with l2 normalized deep auto-encoder representations," *arXiv preprint arXiv:1802.00187*, 2018.

[4] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.