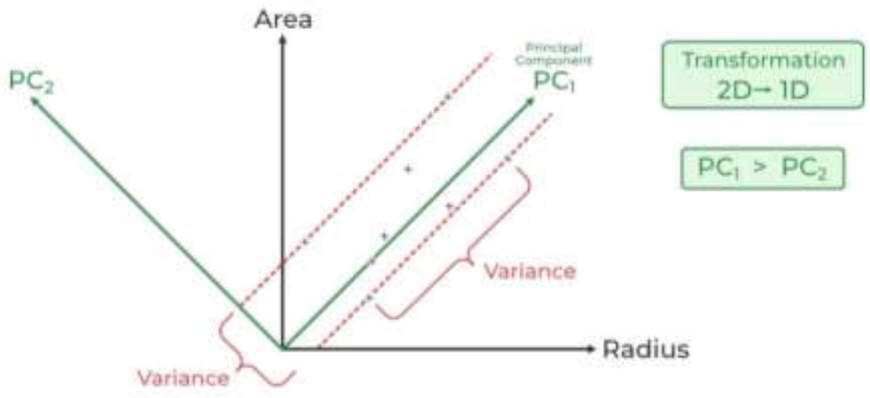
MNIST Data Reduction and Performance Analysis

By Inam ul haq azad

Introduction

- Most datasets contain correlated variables
- Principal Component Analysis (PCA) – dimensionality reduction technique
- Identify a set of orthogonal axis that capture maximum variance in data
- Methods for PCA :-
    1. Singular Value Decomposition (SVD) of data matrix $X = USV^T$
    2. Eigenvalue decomposition of covariance matrix X^TX
- PCA given by top k SVDs
- Choice of k – Horn's parallel analysis



$$\sum_{i=1}^{k} \hat{\lambda}_i / \mathrm{tr}(\hat{\Sigma}_n) > q, \quad \text{e.g.} \quad q = 0.95$$

Horn's Parallel analysis

- Randomly permute sample features for decorrelation
- Compute singular values of random matrices
- Repeat for R times, we get R set singular values
- Define p-value for i-th eigenvalue

$$\mathrm{pval}_i = \frac{1}{R}\#\{\widehat{\lambda}_i^r > \widehat{\lambda}_i\},$$

- Keep λ_i if $\mathrm{pval}_i < 0.05$

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,n} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ X_{p,1} & X_{p,2} & \cdots & X_{p,n} \end{bmatrix}$$

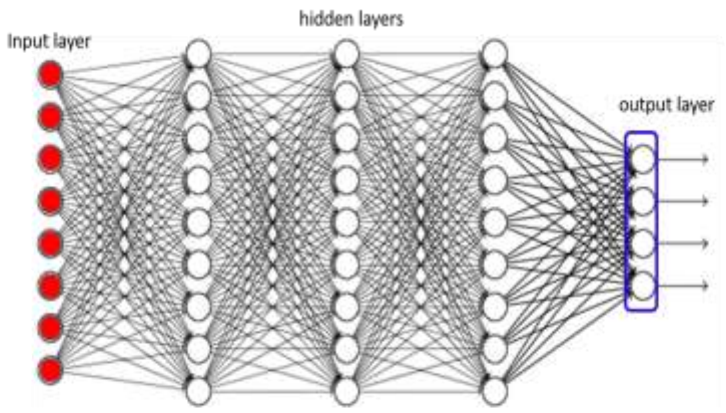
$$X^1 = \begin{bmatrix} X_{1,\pi_1(1)} & X_{1,\pi_1(2)} & \cdots & X_{1,\pi_1(n)} \\ X_{2,\pi_2(1)} & X_{2,\pi_2(2)} & \cdots & X_{2,\pi_2(n)} \\ \vdots & \vdots & \ddots & \vdots \\ X_{p,\pi_p(1)} & X_{p,\pi_p(2)} & \cdots & X_{p,\pi_p(n)} \end{bmatrix}$$

MNIST Dataset

- 28*28 grayscale images of handwritten digits (0-9)
- 70,000 images – 60k for training, 10k for testing
- Data Preprocessing
  1. Flatten to get a single array of 784 dimension
  2. Normalize pixel values – divide by 255
  3. Data centering – subtract the mean from data

Methodology

- Deep Learning architecture – MLP with 3 hidden layers
- Train on the original dataset – 784 features
- Data reduction with PCA – 47 Principal components
- Performance reported in both cases
- Adam optimization algorithm
- Trained for 5 epochs

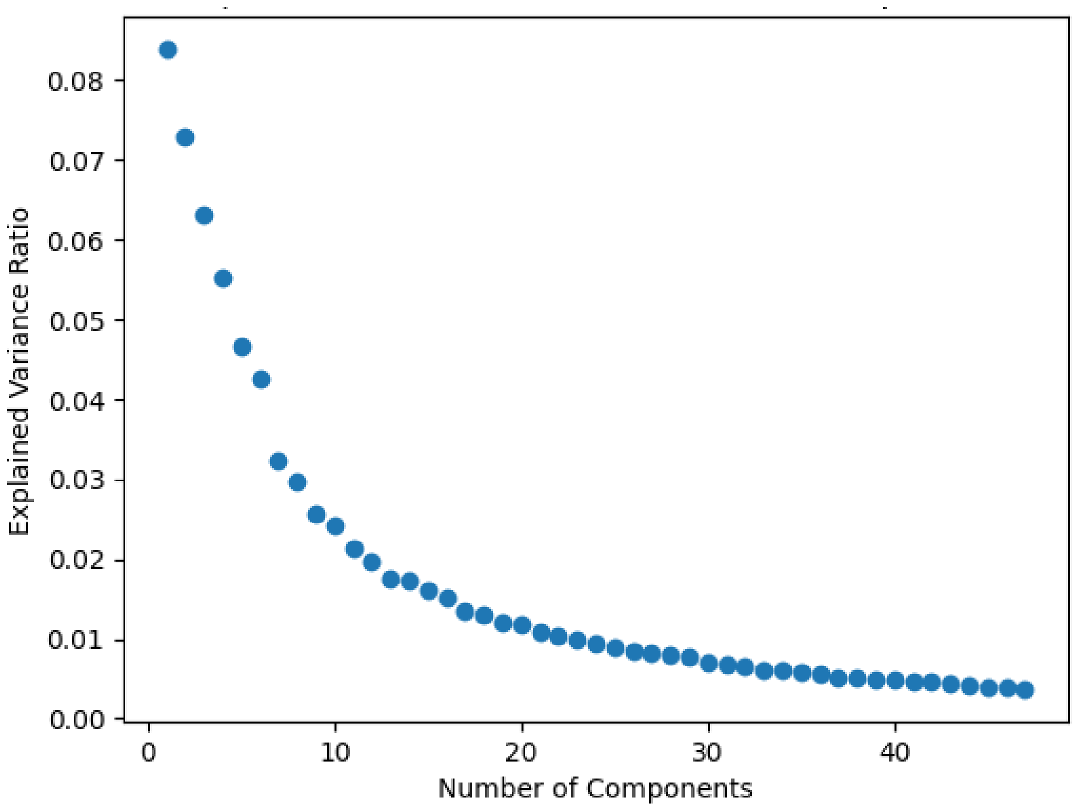



```
model = models.Sequential([
    layers.Dense(512, activation='relu',
    layers.Dropout(0.2),
    layers.Dense(128, activation = 'relu
    layers.Dropout(0.2),
    layers.Dense(64, activation = 'relu'
    layers.Dropout(0.2),
    layers.Dense(10, activation='softmax
])
```

Results

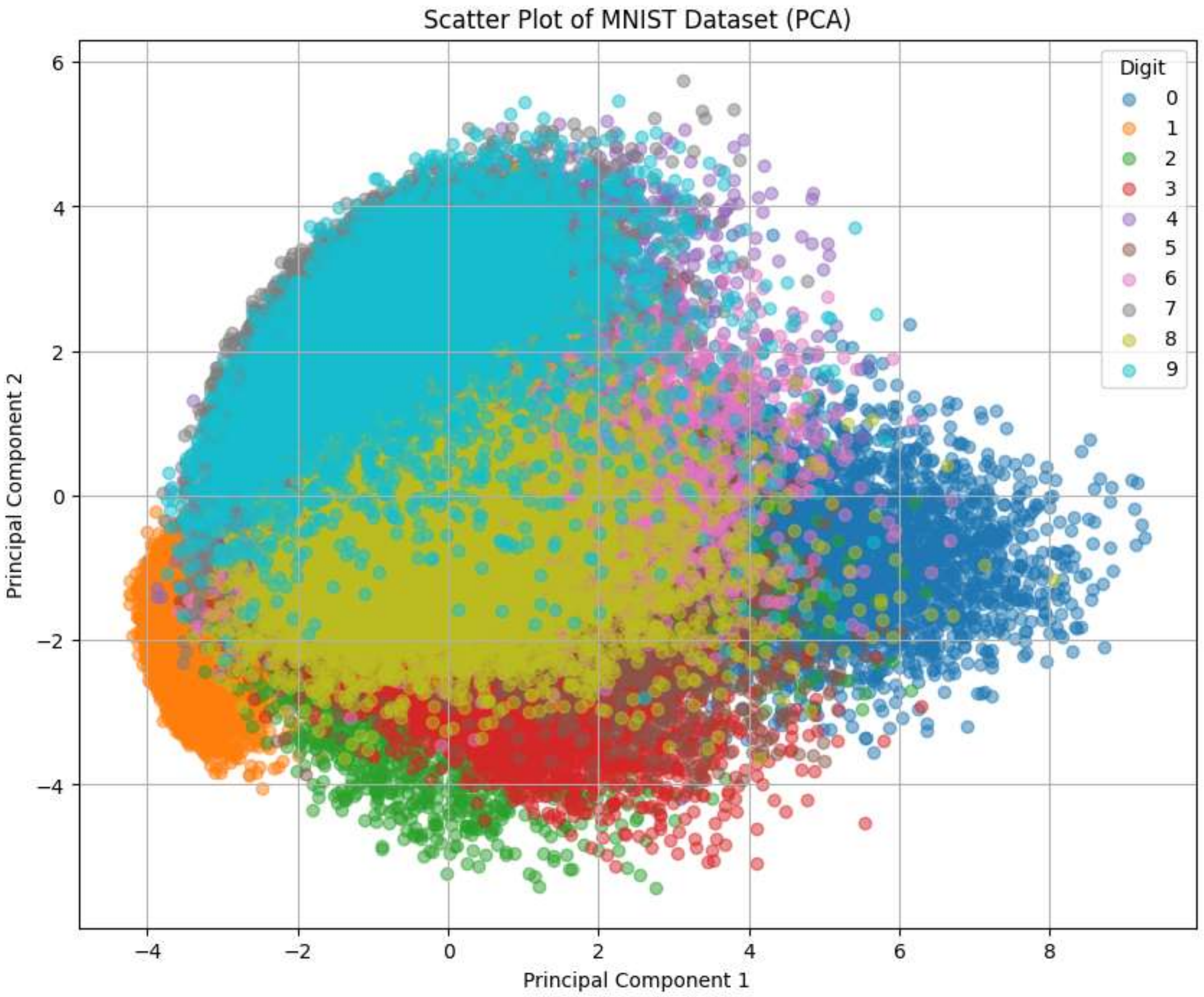
Input Data	Input features	Model parameters	Training time	Accuracy
MNIST Dataset	784	476k	46 sec	97.79 %
Reduced MNIST data	47	99k	11 sec	98.08 %

Top 47 PCs for explained variance ratio

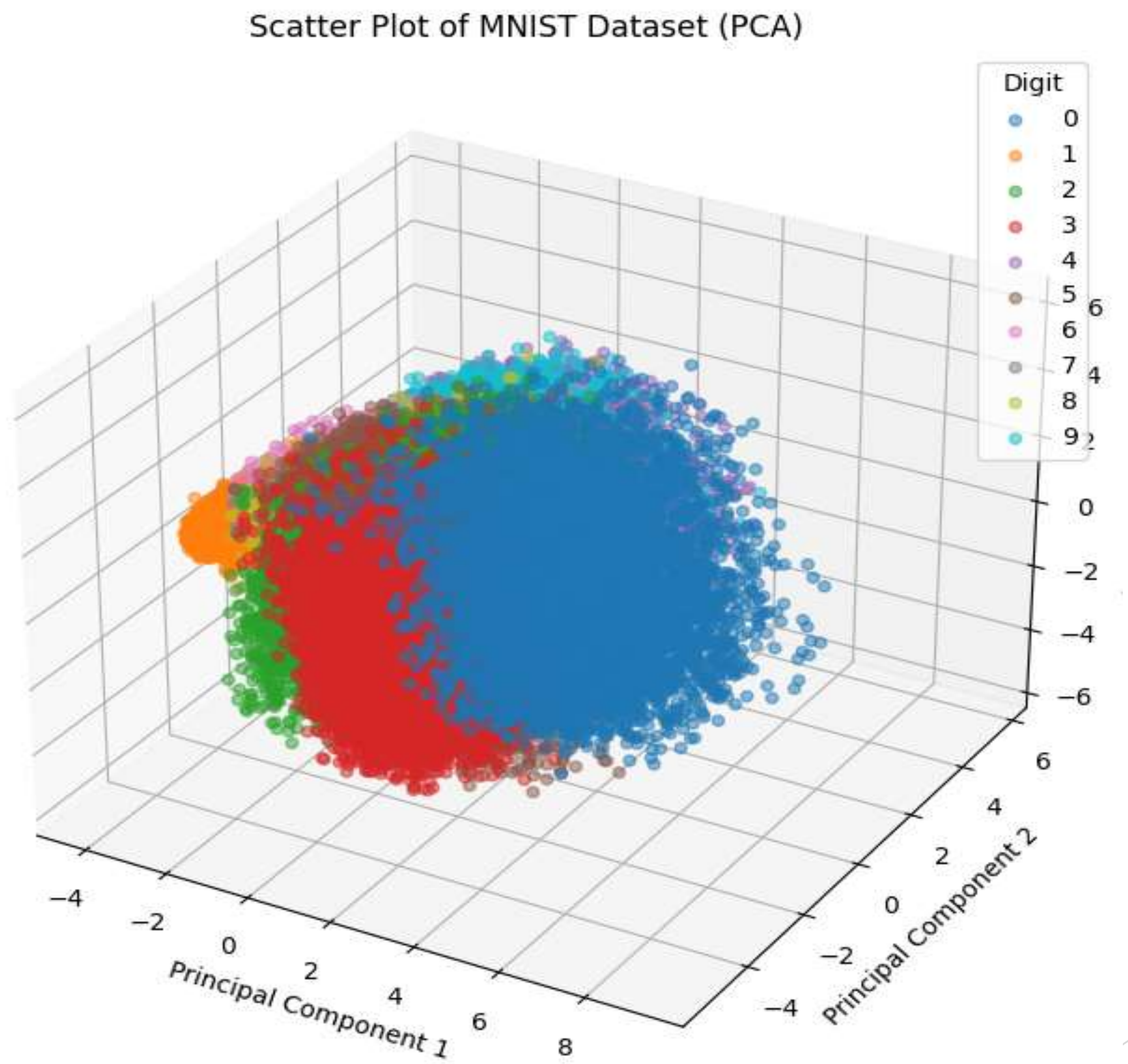


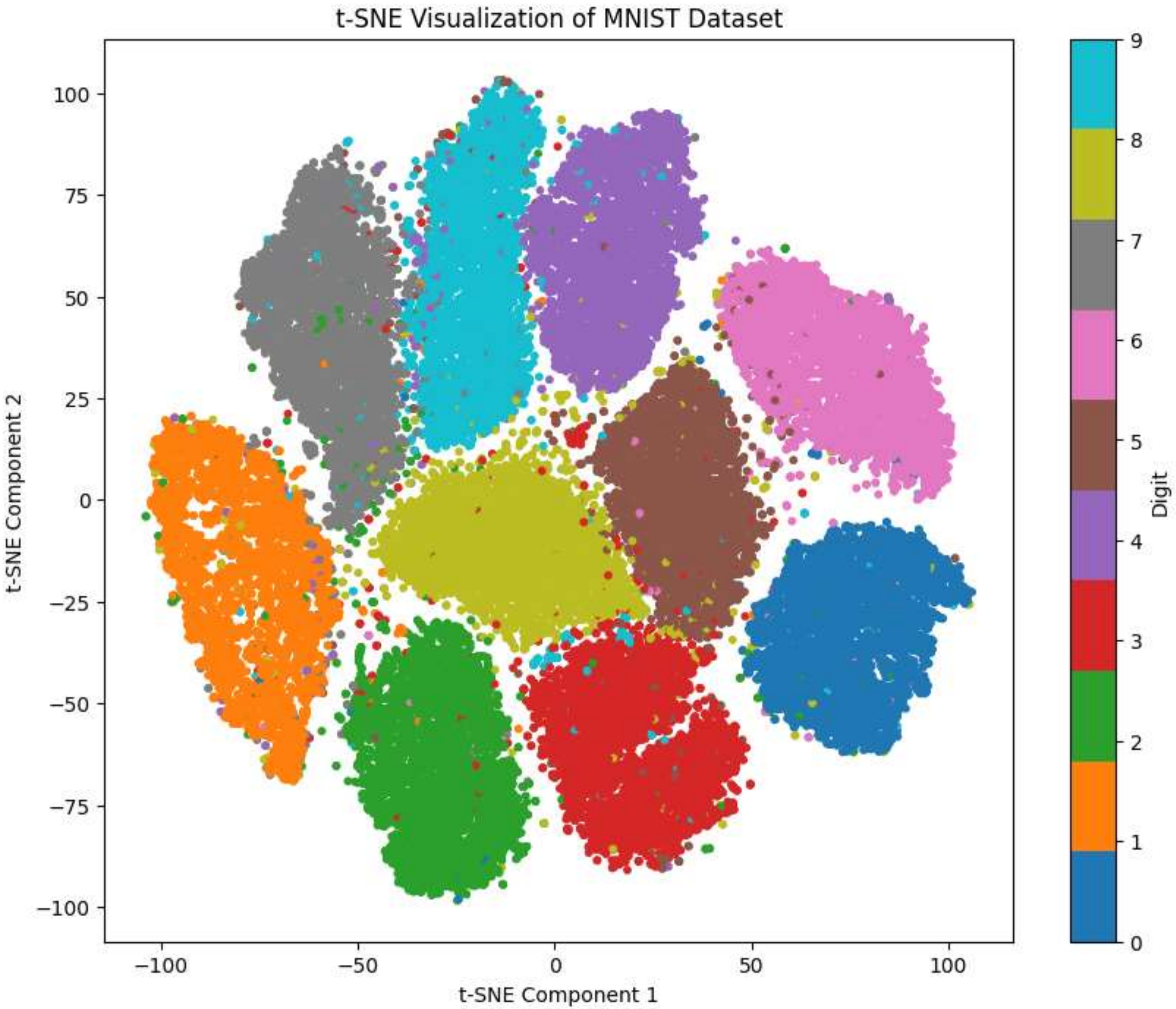
- Top 47 PCs capture more than 80% of the total variance in dataset
- First PC captures most variance and decreases subsequently

Data Visualization (2D)



3D visualization (3 PCs)




t-SNE Visualization of MNIST Dataset
t-SNE Component 1
t-SNE Component 2
Digit
100
75
50
25
0
−25
−50
−75
−100
−100
−50
0
50
100
0
1
2
3
4
5
6
7
8
9

PCA Vs t-SNE

- **Non-Linearity**: PCA is linear dimensionality reduction technique, while t-SNE is non-linear
- **Preservation of Local Structure**: t-SNE groups similar instances together in low-dimensional space (e.g. multiple instances of digit '5')
- PCA focuses on global variance, not effective for preserving local structure
- **Sensitivity to perplexity:** number of nearest neighbours considered when constructing high-dimensional similarities
- **Speed and Scalability**: PCA is computationally faster and more scalable compared to t-SNE

Conclusion

- PCA reduces model complexity, decreases training time
- Retains the crucial information for classification boundaries
- Original data might have noise and all features not important for drawing perfect boundaries
- 2D visualization PCs do not yield significant boundaries
- We need the order of 10 PCs to capture useful data
- t-SNE embeddings good visualization in 2D space
- t-SNE tends to provide more visually appealing and informative visualizations for moderate-sized datasets like MNIST

Acknowledgement

1. Yao Yuan, Topological and Geometric Data Reduction and Visualization Spring 2024, Lecture Notes
2. Pearson, K. (1901). LIII. On lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11), 559–572.
3. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. 2011; 12:2825–2830
4. Abadi M, Barham P, Chen J, et al. TensorFlow: A system for large-scale machine learning
5. Hinton, Geoffrey; Roweis, Sam (January 2002). Stochastic neighbor embedding (PDF). Neural Information Processing Systems.

Thank You

Questions