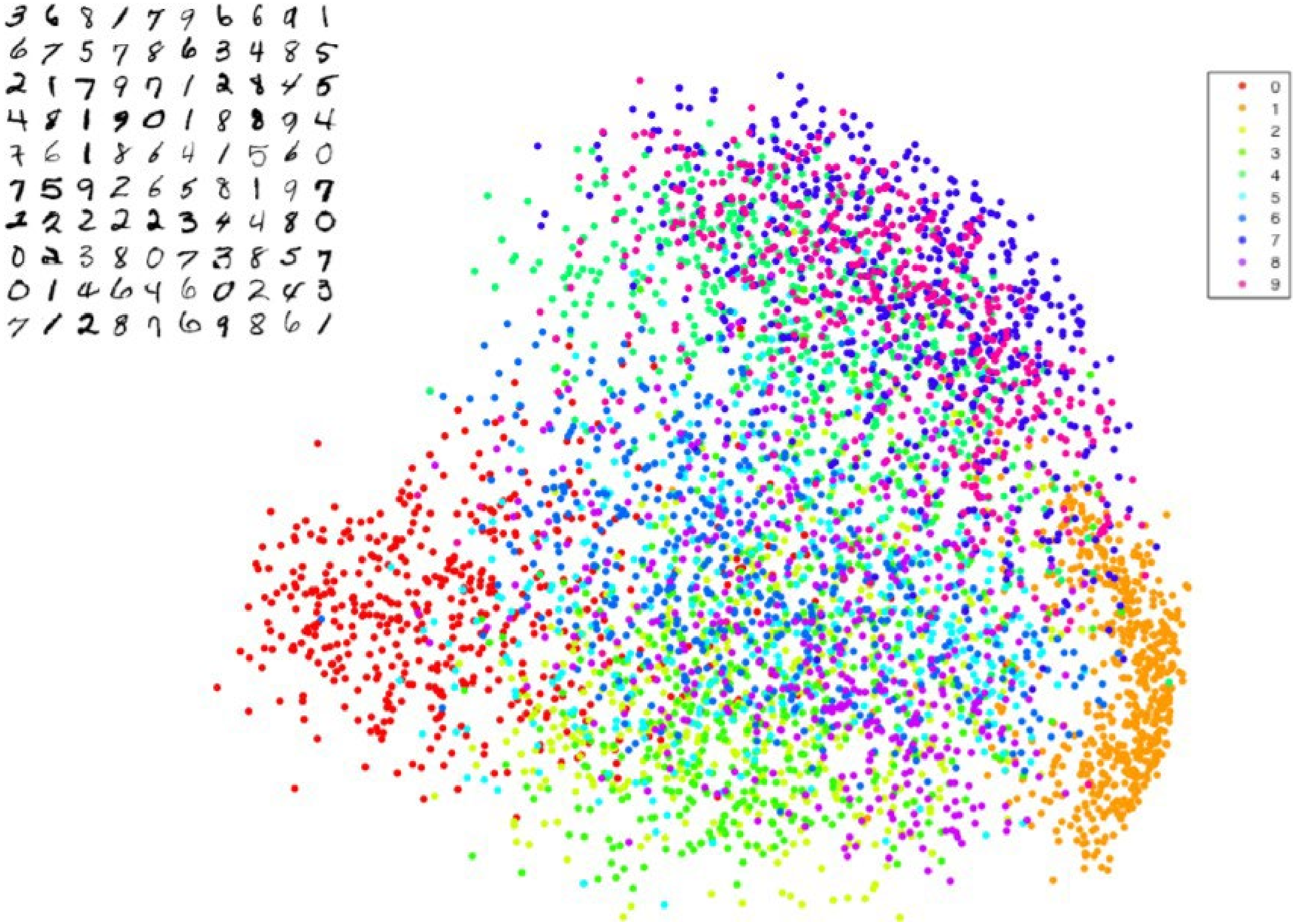
t-Distributed Stochastic Neighbour Embedding (t-SNE)

Adopted from slides by Ethan Fetaya, James Lucas and Emad Andrews at University of Toronto

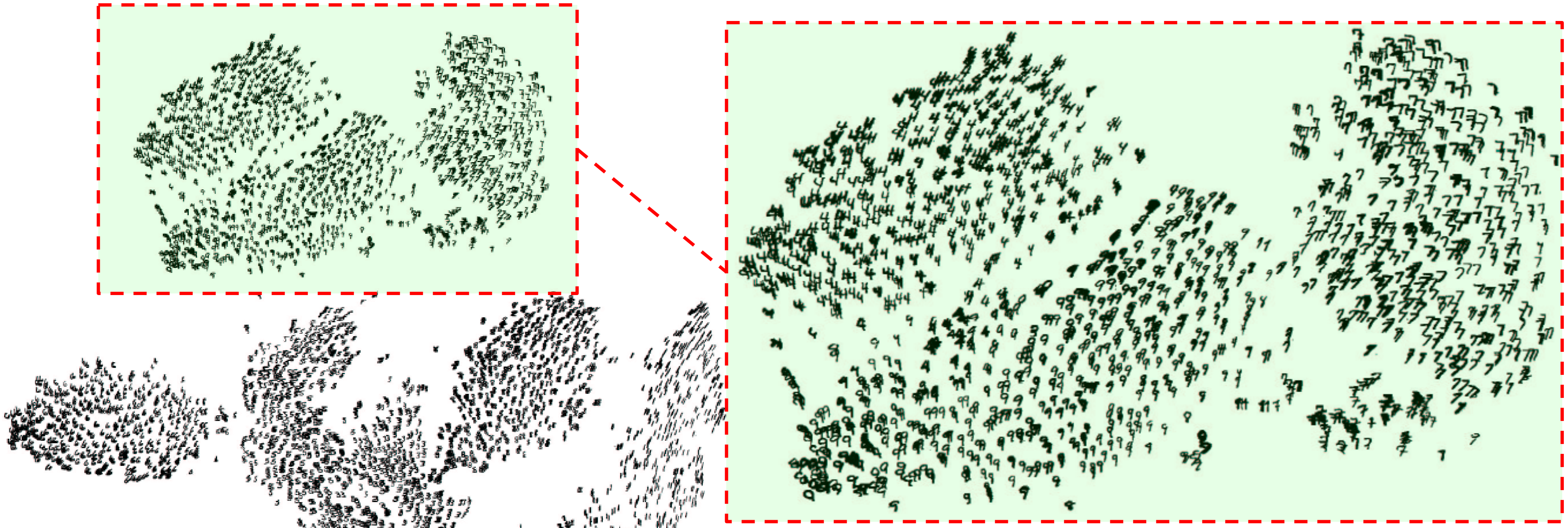
Local embedding

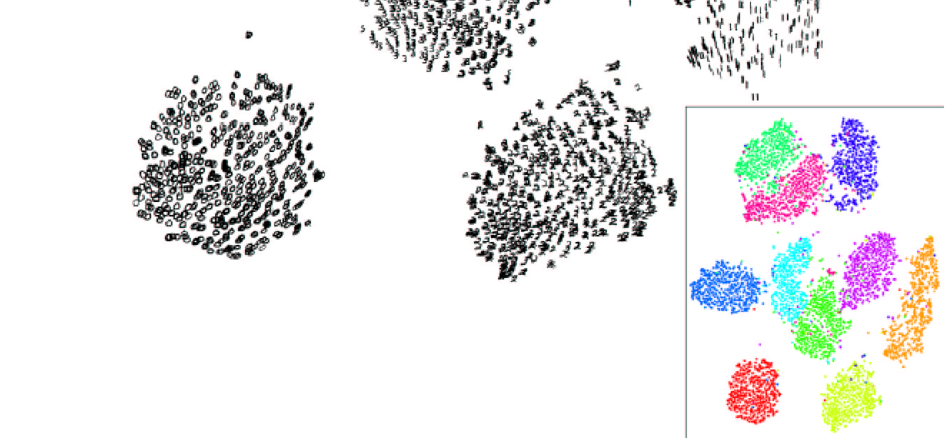
- t-SNE is an alternative dimensionality reduction algorithm.
- PCA tries to find a global structure
  - Low dimensional subspace
  - Can lead to local inconsistencies
  - Far away point can become nearest neighbors
- t-SNE tries to preserve local structure
  - Low dimensional neighborhood should be the same as original neighborhood.
  - Unlike PCA, t-SNE almost only used for visualization
  - No easy way to embed new points

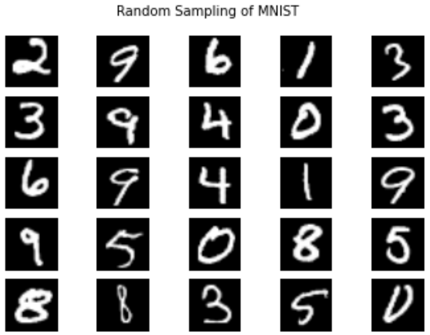
PCA 2 dimensions embedding for MNIST



t-SNE 2 dimensions embedding for MNIST







Stochastic Neighbor Embedding (SNE)

- SNE basic idea:
  - “Encode” high dimensional neighborhood information as a distribution Intuition: Random walk between data points.
    - High probability to jump to a close point
  - Find low dimensional points such that their neighborhood distribution is similar.
  - How do you measure distance between distributions?
    - Most common measure: KL divergence

Neighborhood Distributions

- Consider the neighborhood around an input data point $\mathbf{x}_i \in \mathrm{R}^d$
- Imagine that we have a Gaussian distribution centered around $\mathbf{x}_i$
- Then the probability that $\mathbf{x}_i$ chooses some other datapoint $\mathbf{x}_j$ as its neighbor is in proportion with the density under this Gaussian
- A point closer to $\mathbf{x}_i$ will be more likely than one further away

Probabilities P_{ij}

- The $i \rightarrow j$ probability is the probability that point $\mathbf{x}_i$ chooses $\mathbf{x}_j$ as its neighbor

$$P_{j|i} = \frac{\exp\left(-||\mathbf{x}^{(i)} - \mathbf{x}^{(j)}||^2/2\sigma_i^2\right)}{\sum_{k\neq i} \exp\left(-||\mathbf{x}^{(i)} - \mathbf{x}^{(k)}||^2/2\sigma_i^2\right)}$$

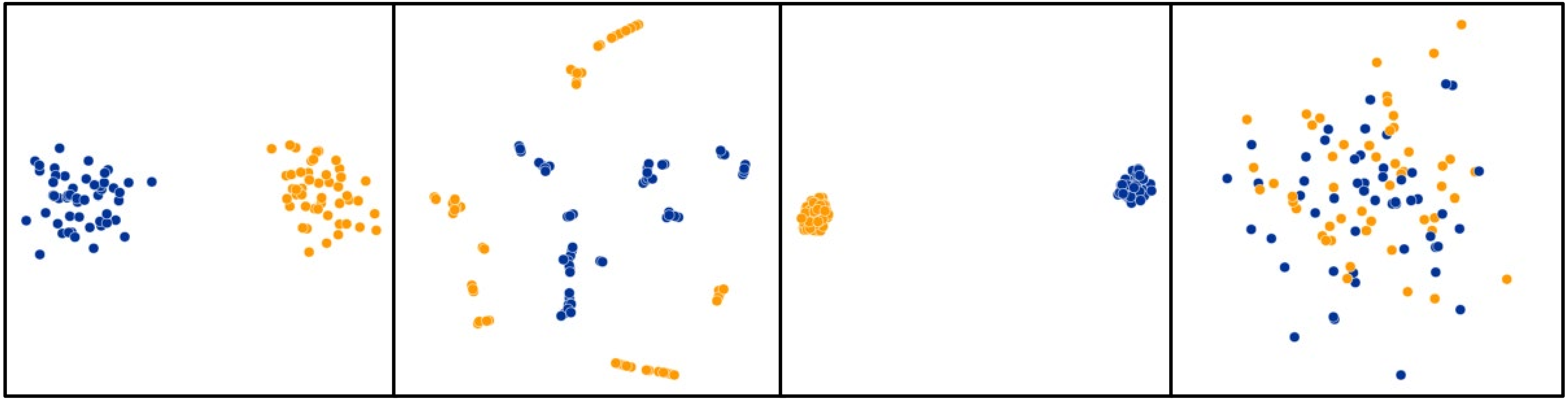
- With $P_{i|i} = 0$
  - The parameter σ_i sets the size of the neighborhood
    - Very low σ_i - all the probability is in the nearest neighbor.
    - Very high σ_i - Uniform weights.
  - Here we set σ_i differently for each data point
  - Results depend heavily on σ_i - it defines the neighborhoods we are trying to preserve.
  - Final distribution over pairs is symmetrized: $P_{ij} = 1/2N(P_{i|j} + P_{j|i})$

Perplexity

- For each distribution P_i (depends on σ_i) we define the perplexity
  - $perp(P_i) = 2^{H(P_i)}$ where $H(P) = -\Sigma_j P_{j|i} \log(P_{j|i})$ is the entropy.
- If P is uniform over k elements - perplexity is k.
  - Smooth version of k in kNN
  - Low perplexity = small σ
  - High perplexity = large σ
- Define the desired perplexity and set σ_i to get that (binary search)
- Values between 5-50 usually work well
- Important parameter - different perplexity can capture different scales in the data

t-SNE Practical Examples

Perplexity Settings Matter

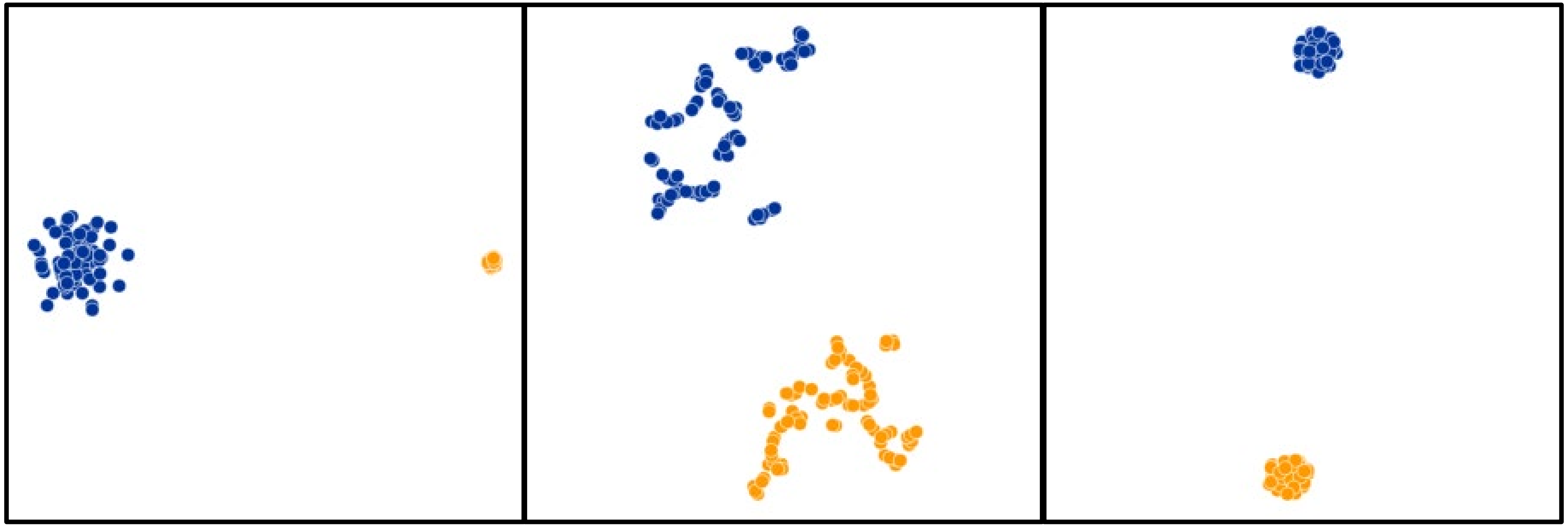


Original Perplexity = 2 Perplexity = 30 Perplexity = 100

https://distill.pub/2016/misread-tsne/

t-SNE Practical Examples

Cluster Sizes are Meaningless



Original Perplexity = 5 Perplexity = 50

https://distill.pub/2016/misread-tsne/

SNE objective

- Given $\mathbf{x}^{(1)}, .., \mathbf{x}^{(N)} \in \mathrm{R}^D$ we define the distribution P_{ij}
- Goal: Find good embedding $\mathbf{y}^{(1)}, .., \mathbf{y}^{(N)} \in \mathrm{R}^d$ for some $d < D$ (normally 2 or 3)
- How do we measure an embedding quality?
- For points $\mathbf{y}^{(1)}, .., \mathbf{y}^{(N)} \in \mathrm{R}^d$ we can define distribution Q similarly the same

$$Q_{ij} = \frac{\exp\left(-||\mathbf{y}^{(i)} - \mathbf{y}^{(j)}||^2\right)}{\sum_k \sum_{l \neq k} \exp\left(-||\mathbf{y}^{(l)} - \mathbf{y}^{(k)}||^2\right)}$$

- Optimize Q to be close to P
  - Minimize KL-divergence
- The embeddings $\mathbf{y}^{(1)}, .., \mathbf{y}^{(N)} \in \mathrm{R}^d$ are the parameters we are optimizing

SNE algorithm

- We have P, and are looking for $\boldsymbol{y}^{(1)},\ldots,\boldsymbol{y}^{(N)} \ \in \ R^{d}$ such that the distribution Q we infer will minimize $L(Q) \ = \ KL(P||Q)$.
- Note that $KL(P||Q) = \sum_{ij} P_{ij} \log\left(\frac{P_{ij}}{Q_{ij}}\right) \propto -\sum_{ij} P_{ij} \log\left(Q_{ij}\right)$
- Can show that $\frac{\partial L}{\partial \boldsymbol{y}^{(i)}} = \sum_{j}\left(P_{ij} - Q_{ij}\right)\left(\boldsymbol{y}^{(i)} - \boldsymbol{y}^{(j)}\right)$
- Main issue - crowding problem.

Crowding Problem

- In high dimension we have more room, points can have a lot of different neighbors
- In 2D a point can have a few neighbors at distance one all far from each other - what happens when we embed in 1D?
- This is the ”crowding problem” - we don’t have enough room to accommodate all neighbors.
- This is one of the biggest problems with SNE.
- t-SNE solution: Change the Gaussian in Q to a heavy tailed distribution.
  - if Q changes slower, we have more ”wiggle room” to place points at.

t-SNE

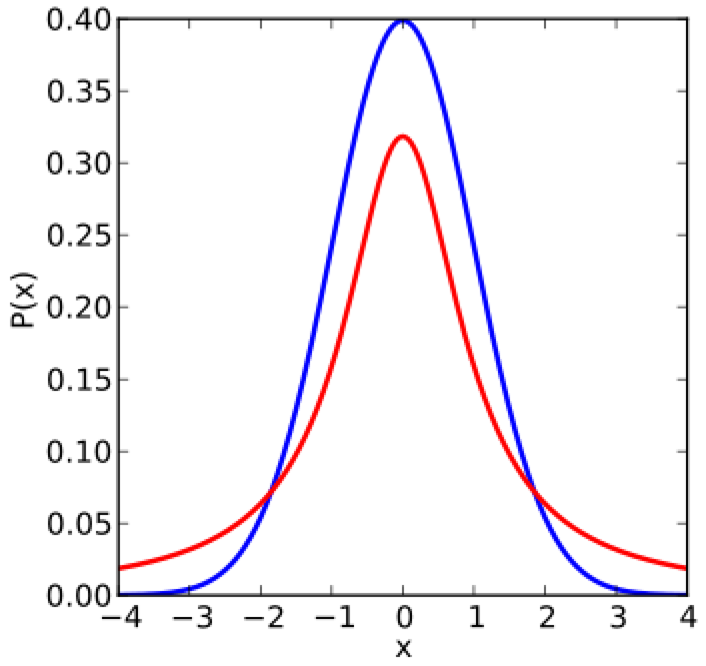
- t-Distributed Stochastic Neighbor Embedding
  - Student-t Probability density $p(x) \propto (1 + \frac{x^2}{v})^{-(v+1)/2}$
- Probability goes to zero much slower then a Gaussian.
- Can show it is equivalent to averaging Gaussians with some prior over σ
- We can now redefine Q_{ij} as

$$Q_{ij} = \frac{(1 + ||\mathbf{y}_i - \mathbf{y}_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}}$$

- We leave P_{ij} as is

t-SNE

Blue = Gaussian
Red = Student's t



t-SNE gradients

- Can show that the gradients of t-SNE objective are

$$\frac{\partial L}{\partial \mathbf{y}^{(i)}} = \sum_j (P_{ij} - Q_{ij})(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})(1 + ||y_i - y_j||^2)^{-1}$$

- Compare to the SNE gradients

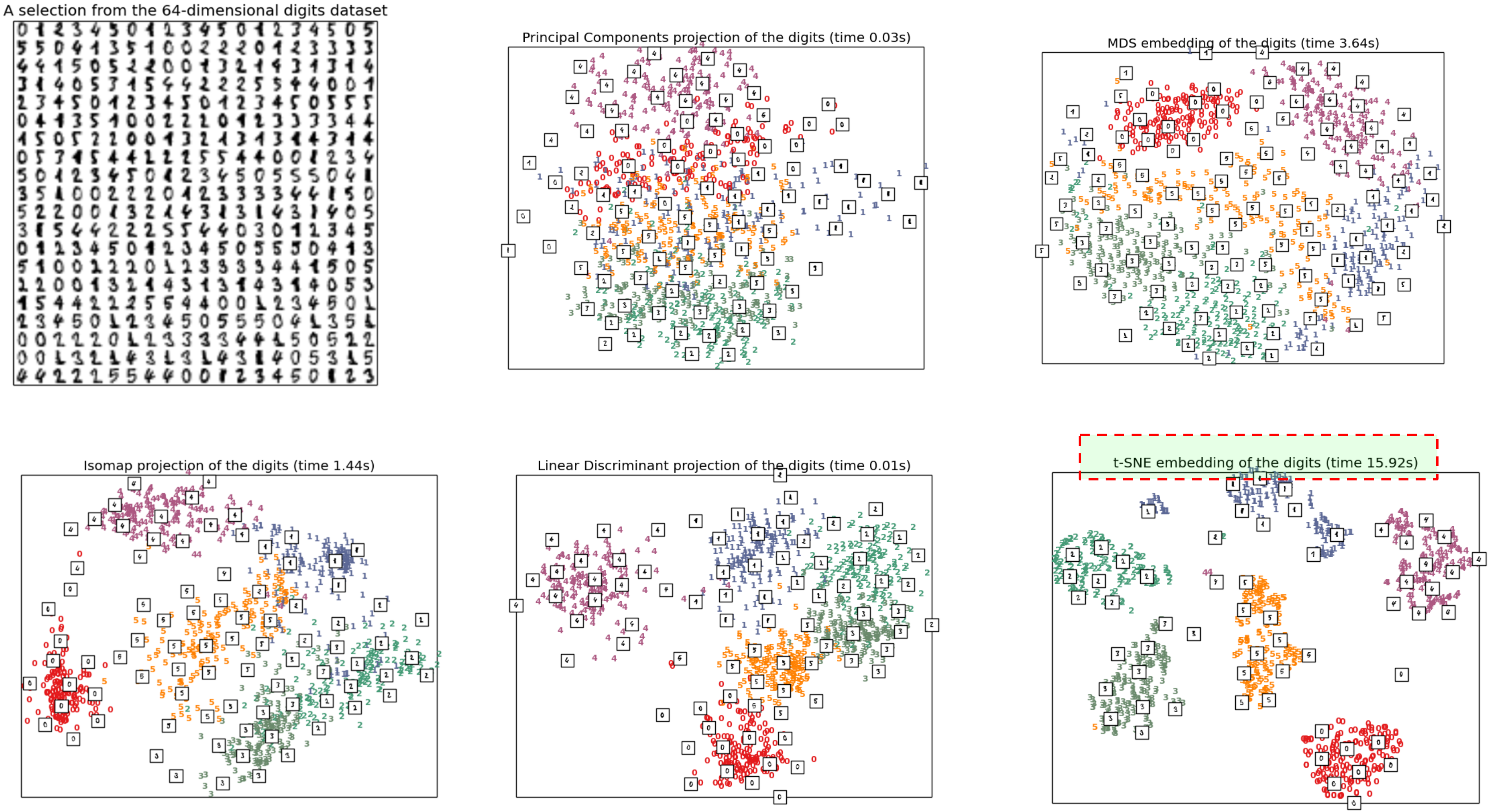
$$\frac{\partial L}{\partial \mathbf{y}^{(i)}} = \sum_j (P_{ij} - Q_{ij})(\mathbf{y}^{(i)} - \mathbf{y}^{(j)})$$

Algorithm

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

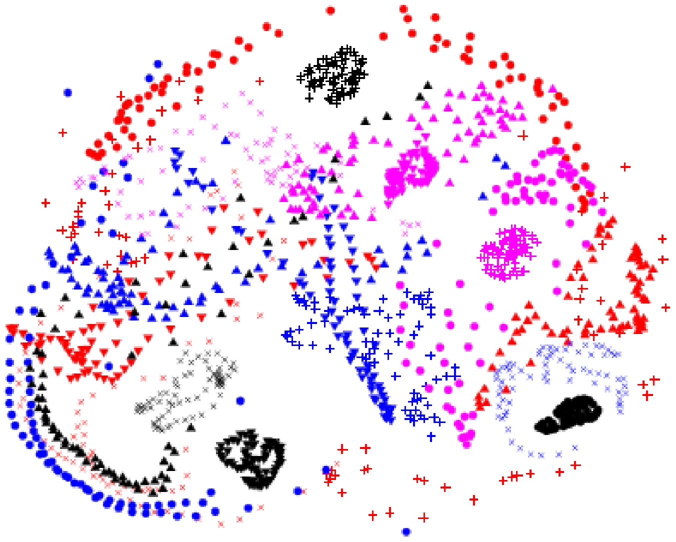
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for $t=1$ **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
  end
end

[Slide credit: "Visualizing Data using t-SNE"]

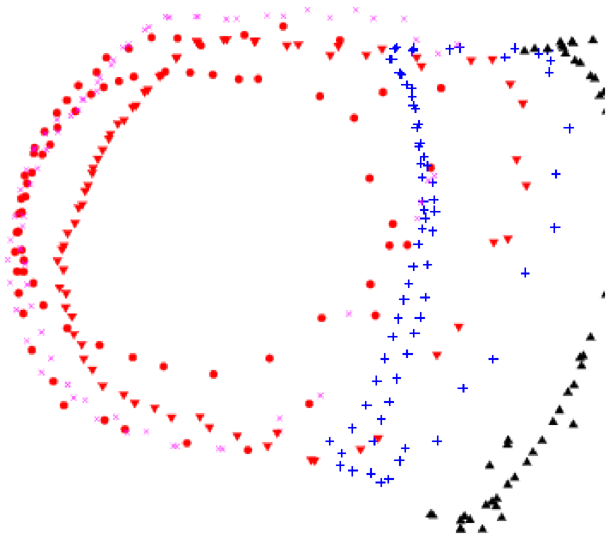


http://scikit-learn.org/stable/auto_examples/manifold/plot_lle_digits.html

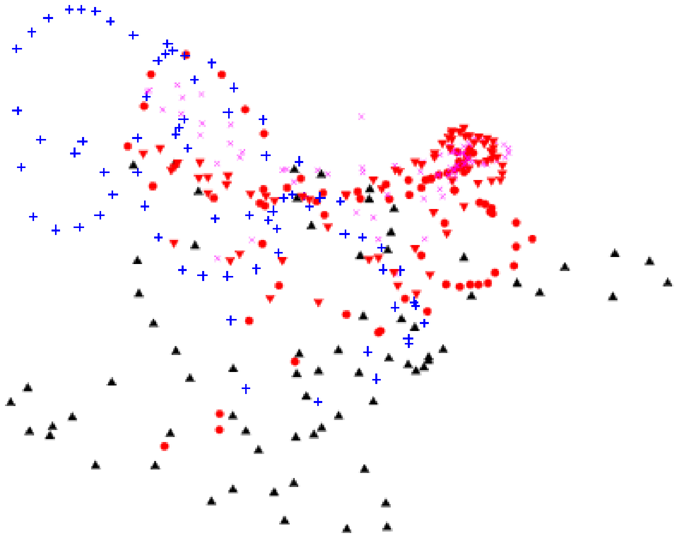
(a) Visualization by t-SNE.



(b) Visualization by Sammon mapping.

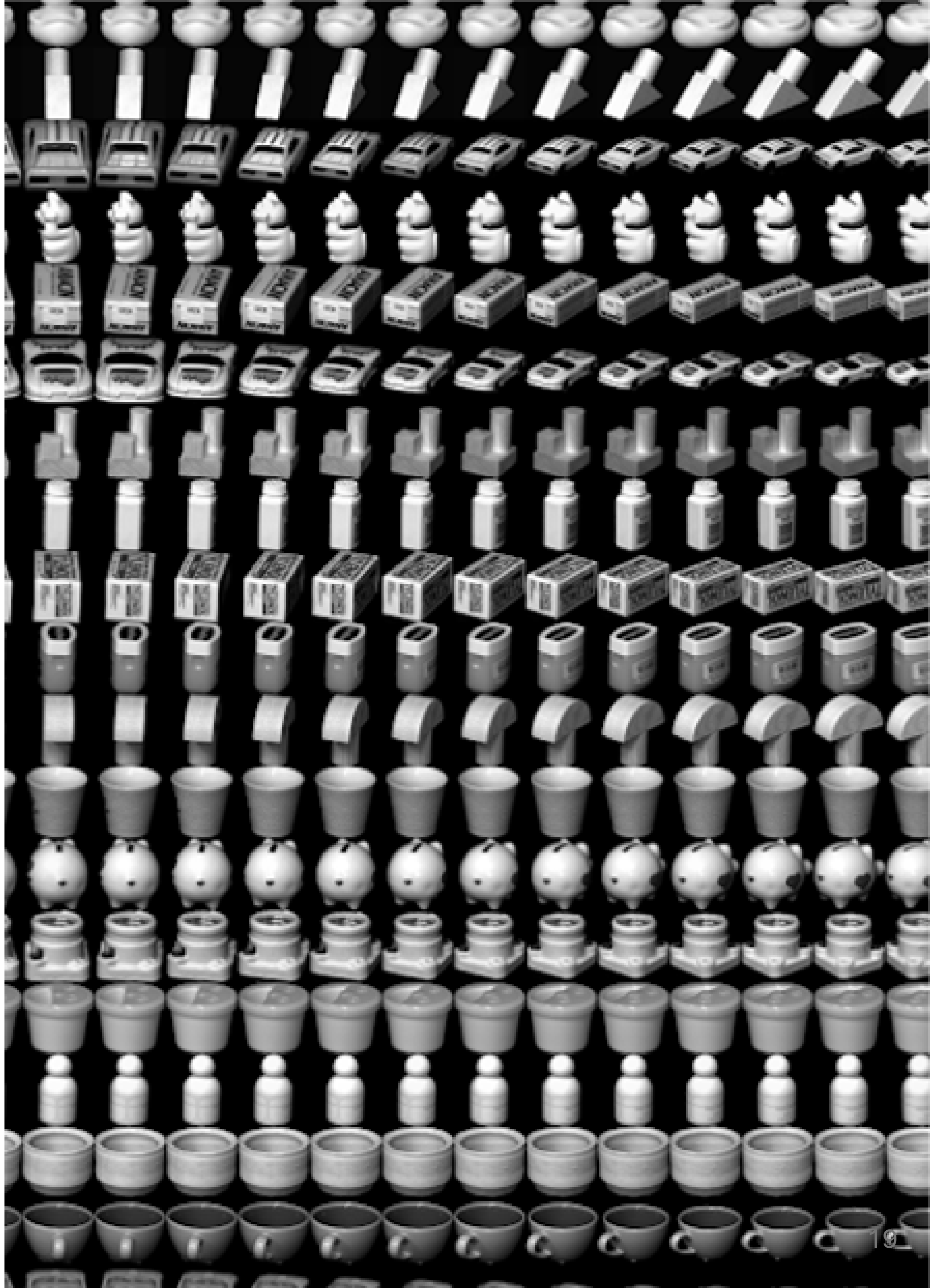


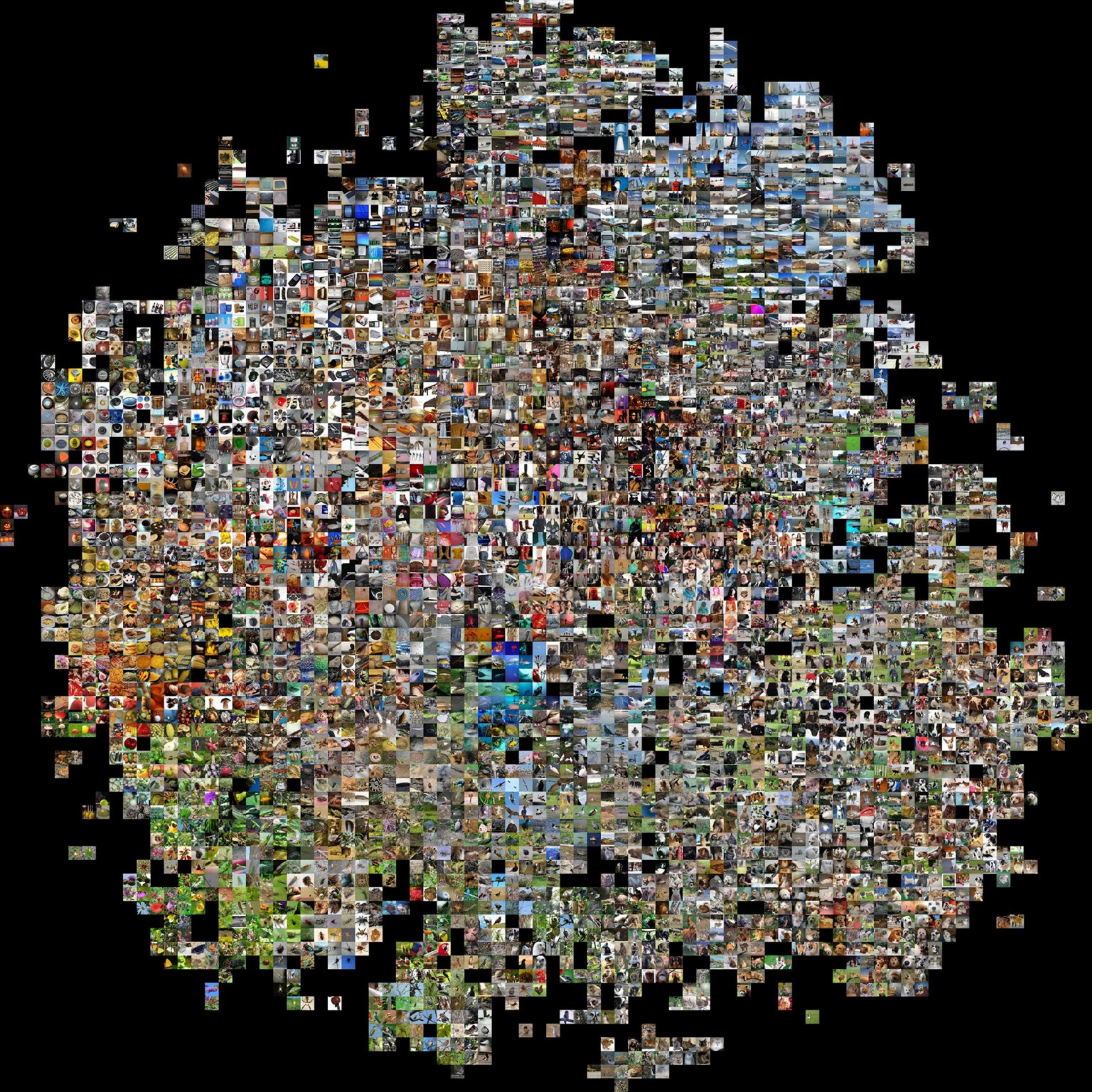
(c) Visualization by Isomap.



(d) Visualization by LLE.

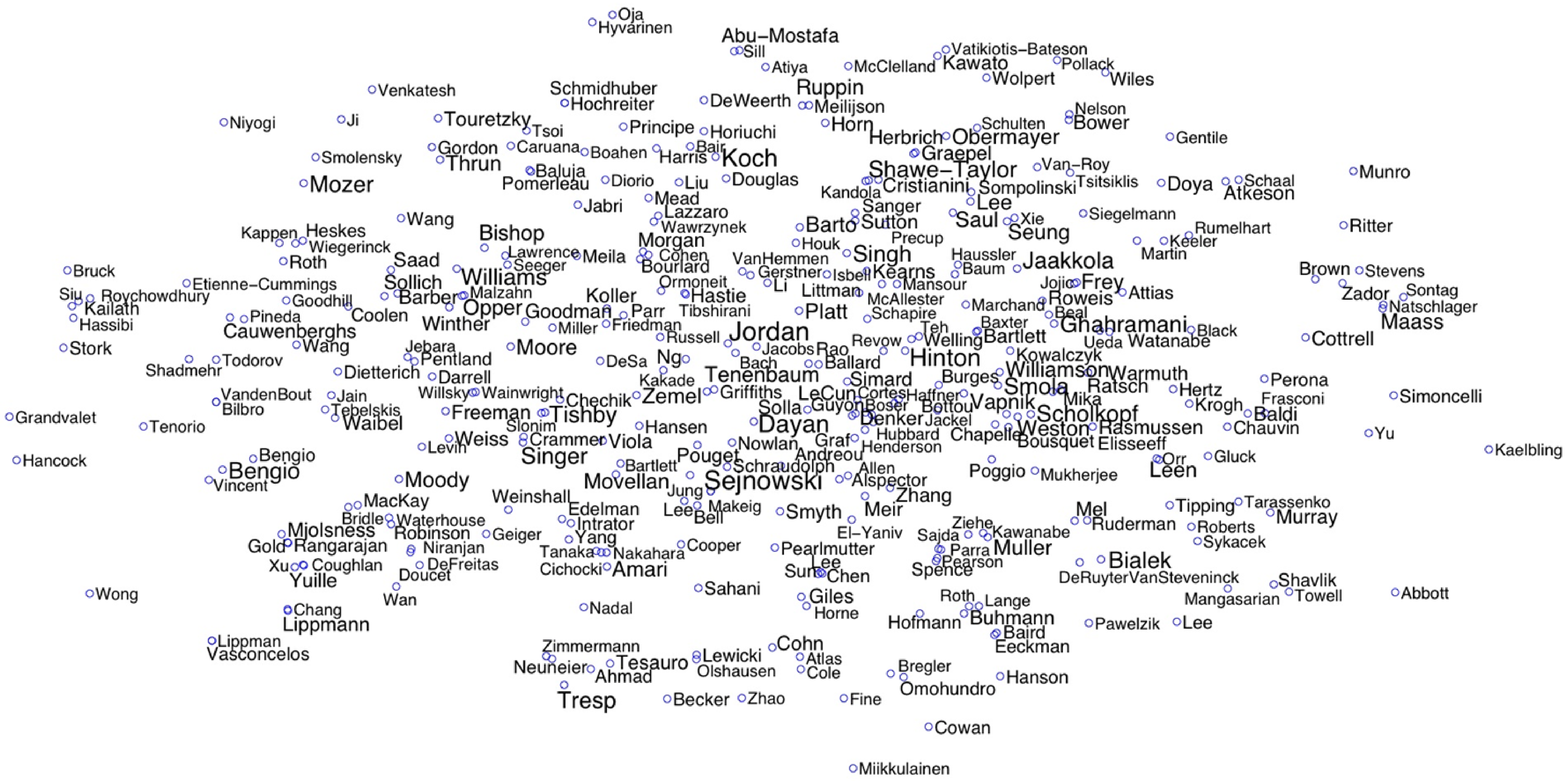
Figure 5: Visualizations of the COIL-20 data set.





ImageNet

http://cs.stanford.edu/people/karpathy/cnnembed/cnn_embed_4k.jpg



NIPS coauthorship

http://lvdmaaten.github.io/tsne/examples/nips_tsne.jpg