2-MAP: Aligned Visualizations for Comparison of High-Dimensional Point Sets

Xiaotong Liu[1], Zeyu Zhang[1], Roxana Leontie[1], Abby Stylianou[2], Robert Pless[1]
[1]George Washington University, [2]Saint Louis University
liuxiaotong2017@gwu.edu

Abstract

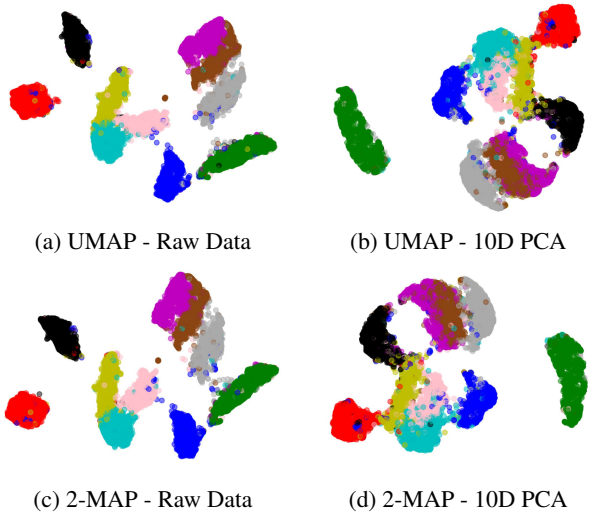
Visualization tools like t-SNE and UMAP give insight into the high-dimensional structure of datasets. When there are related datasets (such as the high-dimensional representations of image data created by two different Deep Learning architectures), roughly aligning those visualizations helps to highlight both the similarities and differences. In this paper we propose a method to align multiple low dimensional UMAP visualizations by adding an alignment term to the UMAP loss function. We provide an automated procedure to find a weight for this term that encourages the alignment but only minimally changes the fidelity of the underlying embedding.

1. Introduction

The visualization of high-dimensional data is an increasingly important tool in the machine learning development pipeline. One approach to this visualization is to use dimensionality reduction tools like t-SNE [1] or UMAP [2], which produce a low-dimensional embedding of the high d bimensional points that attempts to preserve and display the local neighborhoods and clustering relationships that exist in the original representation.

These approaches, however, are not well suited for the comparison of multiple related high-dimensional representations, even though this sort of comparison is often the reason for the visualization, for example, to compare the relationships between raw data and a learned representation of that data, or to compare two different representations of the same dataset. In these cases, the low-dimensional visualizations produced by dimensionality reduction tools like t-SNE and UMAP may not be well aligned, making it difficulty to see how the representations compare and contrast.

There is often flexibility in how a high-dimensional point set might be mapped from high-dimensions to low-dimensions — for example, points in clusters in the high-dimensional space may be mapped well to clusters in low-dimensional space, but there may be various, equally good, configurations of those clusters relative to each other. This flexibility suggests an opportunity to align multiple visualizations of the related high-dimensional data without substantially changing the fidelity of each visualization, making them significantly easier to compare. This is the goal of this paper.



(a) UMAP - Raw Data (b) UMAP - 10D PCA

(c) 2-MAP - Raw Data (d) 2-MAP - 10D PCA

Figure 1: Using MNIST digits as a simple example, we show (a) a UMAP visualization of the raw data (1a) and, (b) a 10-dimensional PCA representation of the data. (c),(d) show the aligned 2-MAP visualization. Because the points are loosely aligned with each other, it is easier to find similarities and differences between the two visualizations.

Figure 1 (top) shows an example of the UMAP visualization of raw MNIST digits, and the UMAP visualization of the 10-D PCA representation of the MNIST digits. The bottom shows our aligned visualizations. Two things are notable. First, comparing the bottom left and bottom right visualizations of two related high-dimensional point sets, the clusters for each class are in about the same place. For example, the round, red class is on the left side, the elongated, green class is on the right. This makes it easier to visually see the differences between the representations. Second, each visualization is locally similar to the unaligned version. For example, in the raw data on the left, the "teapot"

shape of the blue cluster, and the "rabbit ear" shape of the overlapping of the three clusters in the center are preserved. Similarly, in the PCA visualization on the right, the red and chartreuse clusters remain connected, as do the purple and gray clusters. This suggests that the visualizations, while aligned, remain similarly good at visualizing the original high-dimensional point set.

This alignment is achieved by running a modified version of UMAP, called 2-MAP, which simultaneously embeds both high dimensional representations, and includes an addition alignment loss to encourage the low dimensional embeddings to be well aligned. The key question in implementing such an alignment is how to balance the fidelity of each embedding to its high dimensional representation with the degree of alignment of the low dimensional embeddings.

In this work we explore this balance, and build an end-to-end, parameter free tool to create aligned visualizations. Specifically, our contributions are:

- the derivation of a simple algorithm that modifies the popular and robust UMAP dimensionality reduction tool to optimize the embedding of two related datasets with an additional weighted alignment term;
- an approach to automatically choose the weight of the alignment term to minimally affect the fidelity of each embedding;
- an extension of this to work for more than two sequentially related representations, such as the evolution of deep learning representations during training; and
- several examples of the 2-MAP visualization approach in different application domains.

Code for reproducing all experiments in this paper, and to create the aligned 2-MAP embedding of any pair of related high-dimensional point sets can be accessed at: `https://github.com/GWUvision/2-MAP`.

2. Background

t-SNE [1] and UMAP [2] are popular tools to create 2D visualizations of high-dimensional data. While these visualizations are known to have limitations, they offer visual depictions of how the points are clustered and/or distributed. Both approaches optimize a cost function the encourages the local distribution of points in the low-dimension space be similar to the local distribution of points in the high-dimensional space. Both are popular because their results often give good intuition about the structure of point sets when more classical approaches like PCA, MDS [3] or Sammon Mapping [4] fail, or when the assumptions on the manifold structures used by more complex non-linear approaches including Isomap [5], LLE [6], and Laplacian Eigenmaps [7] do not apply.

t-SNE is known to have limitations [8] arising from mapping a high-dimensional space down to two dimensions. UMAP shares some of these limitations that arise from any projection of points onto a much lower dimension, but more explictly preserves larger scale structures [2], is usually faster to compute, and is becoming the standard for many problem domains [9], so we focus on making an aligned version of UMAP.

While often the purpose of showing t-SNE or UMAP visualization is to give qualitative comparisons of two embeddings, allowing one to see, for example, generally how many clusters there are. But richer comparisons require some way to find correspondences between the representations to each other. One approach to this is interactive visualizations, for example, the SleepWalk system provides a number of interactive tools, so that hovering over a point in one embedding highlights the same point in another embedding [10], allowing a user to compare the local neighborhood of the same point.

We know of only one work [11] that explicitly seeks to align the t-SNE embeddings of related high-dimensional point sets together. Our approach is similar to theirs – they suggest simultaneously optimizing the standard t-SNE on each each dataset and including an alignment error term that penalizes the Euclidean distance between the embedded location of corresponding points. However, they give no guidance about how to set that parameter, creating the risk of incorrect visualizations of each high-dimensional point set because they are too aligned (and constraints from one high-dimensional representation have leaked into the other), or joint visualizations that are not as aligned as they could be.

3. 2-MAP

In this section, we will introduce our visualization method for data comparison, 2-MAP, which is based on Uniform Manifold Approximation and Projection (UMAP) [2].

3.1. UMAP Algorithm

The UMAP algorithm for dimensionality reduction has two phases: in the first phase, a weighted graph of the high-dimensional data is created, by turning distance between nearby high-dimensional points into weighs; the second phase involves the optimization of a cost function to find a low-dimensional embedding which has the closest structure to the original data. In the following description of the UMAP algorithm, we follow the notation from [12].

In the first phase, the weights of the graph are given by:

$$v_{ij} = e^{-[(r_{ij}-\rho_i)/\sigma_i]} \tag{1}$$

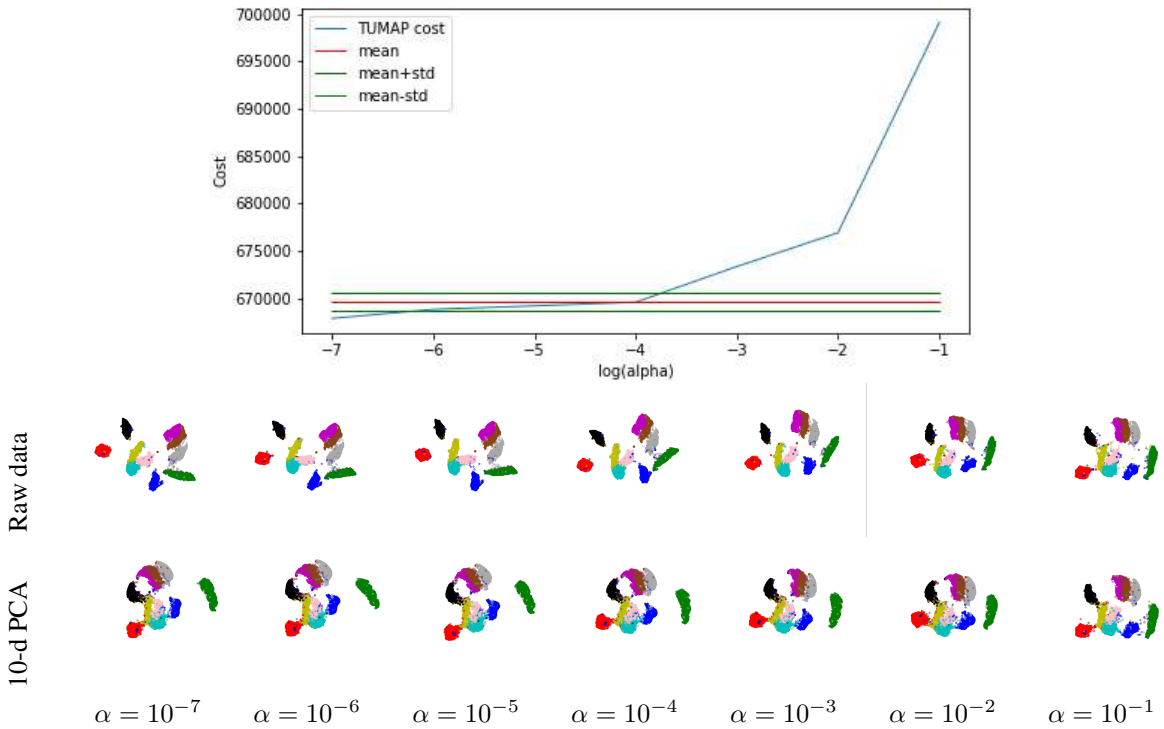


Figure 2: A visualization of the scale selection process aligning the MNIST raw data and MNIST 10-D PCA result. (top) Shows the mean and standard deviation of the U-MAP optimization score (for unaligned embeddings) from 10 random initializations, as well as and the U-MAP cost when optimized with different weights α for the alignment errors.

Where r_{ij} is distance between high dimensional points i and j, ρ_i is the distance of point i with its nearest neighbor, k is the neighborhood size, and σ_i is a normalization term which makes $\sum_j v_{ij} = \log_2 k$.

When point j isn't in the k-nearest neighborhood of point i, the weight v_{ij} is set to 0. The adjacency matrix V contains all v_{ij}, and is symmetrized by: $V_s = V + V^T - V \circ V^T$, where $\circ$ is the Hadamard product.

In the second phase, UMAP solves for the low-dimensional topological structure that optimizes the cross entropy loss with the high-dimensional topological structure:

$$C = \sum_{ij} \left[v_{ij} \log \frac{v_{ij}}{w_{ij}} + (1 - v_{ij}) \log \frac{1 - v_{ij}}{1 - w_{ij}} \right] \quad (2)$$

The distances in the low-dimensional approximation are estimated by the following family of curves:

$$w_{ij} = \frac{1}{(1 + a d_{ij}^{2b})} \quad (3)$$

where d_{ij} is the pairwise distance between points i and j, and a and b are hyperparameters that modify the shape of the curve (the default parameters in the original UMAP implementation are $a = 1.277$ and $b = 0.895$ [13]).

3.2. 2-MAP Algorithm

While UMAP is useful for visualizing high-dimensional representations of a dataset with a low-dimensional embedding that preserves the local structure of the data, it is not easy to compare different high-dimensional representations of the same data (e.g., the original MNIST digits versus a low-dimensional embedding of the data as seen in Figure 1). To address this need, we introduce the 2-MAP approach to visualizing multiple high-dimensional representations of the same data.

In order to compare two or more representations of the same data easily, we attempt to align corresponding points in the different representations to the same location in the low-dimensional embedding. We do this by simultaneously performing the UMAP optimization while adding an additional pairwise penalty to the loss function. UMAP is usually optimized by directly computing the gradient of the loss function, , or, adding a new gradient to ensure the different low dimensional embeddings converge to similar global structures while maintaining local structures that are faithful to the high dimensional representation.

Assume we want to visualize two high-dimensional representations, A and B, of the same data. The 2-MAP loss function is computed as follows:

$$C = C_A + C_B + \alpha \sum_i ||y_{Ai} - y_{Bi}|| \tag{4}$$

This equation combines two UMAP processes together by adding a penalty term to the two original UMAP losses (C_A and C_B). This penalty term is the sum of Euclidean distances between all corresponding points in both maps (y_{Ai} and y_{Bi}). This encourages the two low dimensional mappings to be similar in global structure, enabling clear comparisons between the two visualizations. A hyper-parameter α defines the relative importance of this alignment penalty in the overall cost function.

3.3. Hyper-parameter Selection

A large alignment penalty in the 2-MAP loss function results in low-dimensional mappings that do not accurately represent the local structure of the high-dimensional representations (preserving this local structure is the benefit of UMAP for visualization). In order to avoid this, we first perform a slight modification to the UMAP loss function that facilitates comparison between embeddings, and additionally propose a method to automatically select an appropriate α (the scaling parameter in Equation 4) that sufficiently preserves the local structure.

3.3.1 Cost Function for Comparison

In the original UMAP algorithm, cross entropy loss is used to measure the difference between the high-dimensional and low-dimensional topological structures. UMAP optimizes this loss based the neighborhood graph structure.

This is, however, problematic in the 2-MAP application. In the construction of the weighted graph of the high-dimensional data, UMAP sets the weights, v_{ij}, between points outside of the k-nearest neighborhood to zero.

The optimization then seeks low-dimensional point locations that make the corresponding w_{ij} weights 0, which can only be realized if the points are infinitely far apart in the low-dimensional embedding. If there are multiple connected components in the original graph, this implies that the optimization can always improve by pushing the disjoint components farther apart. The is a problem faced by any UMAP implementation, but because UMAP is usually used for visualization the exact cost is not so important and the optimization is cut off after some number of steps.

In our case, to validate that an aligned visualization is as good as an un-aligned visualization we need to compare the costs. Stopping after the same number of steps does not work because the joint optimization of 2-MAP does not converge at the same rate as separately optimizing each UMAP separately, and we end up comparing embeddings that have gone through different amounts of the "I've found the right embedding, now I'm just pushing the clusters farther apart" phase.

In order to address this, we measure a modified version of the cross entropy loss for 2-MAP defined in in Equation 6 (For keep UMAP own properties, the original cross entropy loss is still used for the actual optimization):

$$C_M = \sum_{ij} C_{ij} \tag{5}$$

$$C_{ij} = \begin{cases} v_{ij} \log \frac{v_{ij}}{w_{ij}} & \textit{if } v_{ij} = 0 \\ v_{ij} \log \frac{v_{ij}}{w_{ij}} + (1 - v_{ij}) \log \frac{1 - v_{ij}}{1 - w_{ij}} & \textit{otherwise} \end{cases} \tag{6}$$

This modification removes the part of the cost function that encourages disconnected points to be pushed farther apart, and therefore gives a measurements that can be better compared between different UMAP optimizations.

3.3.2 Scale Selection

In order to select an appropriate scaling term, α, that sufficiently preserves the local structure, we consider running UMAP on a single high-dimensional dataset many times with random initializations – each of these runs will result in a different low-dimensional embedding of the same high-dimensional data. We use the modified loss function from Equation 6, and get the cost for a number of randomly initialized low-dimensional UMAP embeddings. We can then compute the mean and standard deviation of these (unaligned) embeddings as statistics of the cost of a 'normal' UMAP embedding. This provides a bound for acceptable costs for an aligned 2-MAP embedding: if the cost of an aligned low-dimensional 2-MAP embedding less than the mean plus one standard deviation of the unaligned UMAP embedding, then we accept it as preserving sufficiently preserving the local structure of the high-dimensional data.

In order to search for the optimal α value, we first compute the mean and standard deviation for randomly initialized unaligned embeddings. We then run 2-MAP with $\alpha = 0.1$ and measure the 2-MAP cost. If it isn't in acceptable range described above, we decrease α to 0.1 times α, repeating this step until the 2-MAP cost is in the acceptable range.

In Figure 2, we use such strategy on the MNIST dataset, comparing the raw MNIST digit data with a 10-dimensional PCA embedding of the digits. We compute the loss from Equation 6 for the two dataset separately ten times using random initialization, and get the mean and standard deviation of those costs to define the acceptable cost for the aligned embeddings. Figure 2 shows the acceptable range (the mean cost of the unaligned representations are shown in red, and the $+/-$ one standard deviation bounds in green),

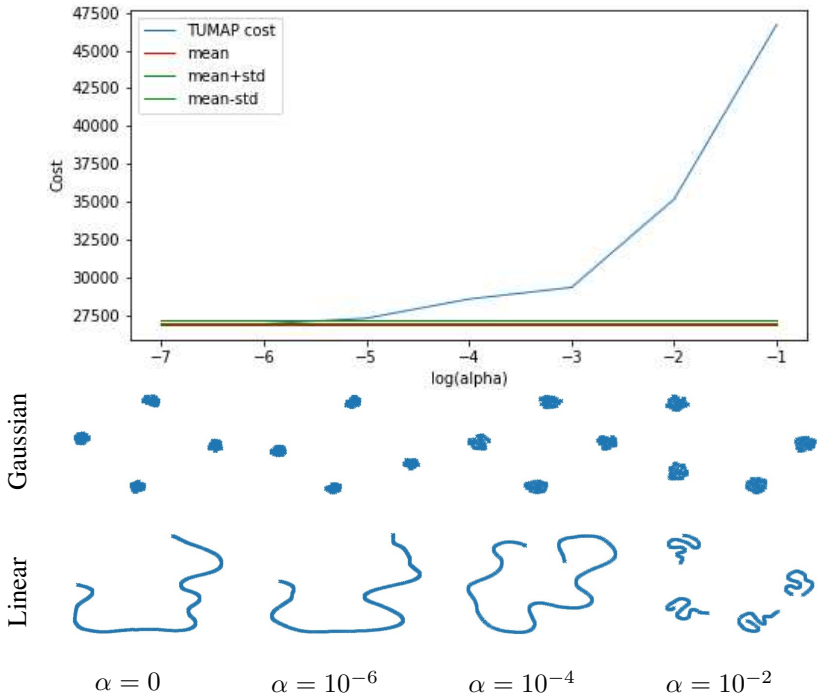


Figure 3: The effect of the alignment parameter α when aligning a data set with four 100-D Gaussian distributions with a 100-D random walk. The top plot shows that as the scale of the alignment term increases, the 2-MAP cost also increases. The four pairs of figures on the bottom show the aligned 2-MAP visualizations for increasing values of α.

with the cost of the aligned 2-MAP embeddings for different α values, as well as their corresponding visualizations.

According to our algorithm, when $\alpha < 10^{-3}$ for this data, the resulting 2-MAP visualizations can reflect the high-dimensional structure well, while still preserving each representation's local structure. When α is not large enough, the 2-MAP visualizations keep their own structure and are well aligned. When α is too large, the 2-MAP visualizations are very similar to each other, but not representative of the actual local structure of each high-dimensional representation.

If the two high-dimensional datasets are sufficiently different to each other, the only acceptable weight of α may be extremely small, resulting in 2-MAP visualizations that are not aligned – this is the desired behavior. We do not want to artificially create aligned visualizations that are not faithful to the original high-dimensional data. In order to demonstrate this, we perform an experiment with two fake datasets. One is built by combining four Gaussian distributions in 100 dimensions, while the other is a random walk path in 100 dimensions. These two datasets have two very different high-dimensional structures. In Figure 3, we show that with the highest acceptable scaling term ($\alpha < 10^{-6}$), 2-MAP does not yield a well-aligned visualization result. The 2-MAP embeddings are essentially identical to the original UMAP embeddings. When we set a large penalty $\alpha = 10^{-2}$, the 2-MAP result does not accurately reflect the high-dimensional data.

3.4. Summary of Algorithm

To summarize the 2-MAP algorithm, for a given pair of high-dimensional datasets, A and B:

1. Measure the variation of the normal, unaligned low-dimensional embeddings, a and b, by running UMAP 10 times and recording the mean ($\mu_{ab} = \mu a + \mu b$)) and standard deviation ($\sigma_{ab} = \sigma a + \sigma b$)) of the modified cost from Equation 6.

2. For decreasing values of α, solve for the aligned low-dimensional embedding, a_{aligned} and b_{aligned}, that minimize the modified cost in Equation 6, stopping when that cost is less than $\mu_{ab} + \sigma_{ab}$.

We have found the embedding to be not very sensitive to the exact value of α, so we decrease α by a factor of 10 each time.

3.4.1 thruMAP

The 2-map algorithm can be extended to a sequence of high-dimensional datasets, $A_1, A_2, \ldots, A_n$, to create a temporal sequence of aligned embeddings. We call this version of the algorithm thruMAP, and it is implemented as follows:

1. Run 2-MAP(A_1, A_2)

2. For $i = 3 \ldots n$:

   (a) Estimate the mean μ_{a_i} and variation σ_{a_i} of the normal, unaligned low-dimensional embeddings, a_i, by running UMAP 10 times and recording modified cost from Equation 6.

   (b) For decreasing values of α, optimize the 2-MAP loss function, only allowing the new embedding a_i to vary (holding a_{i-1} constant), stopping when the modified cost is less than $\mu_{a_i} + \sigma_{a_i}$

This does not give any guarantee that the overall set of embeddings are the best aligned embeddings, but it does guarantee that each embedding is similar to the un-aligned embedding of that representation.

4. Example Use Cases

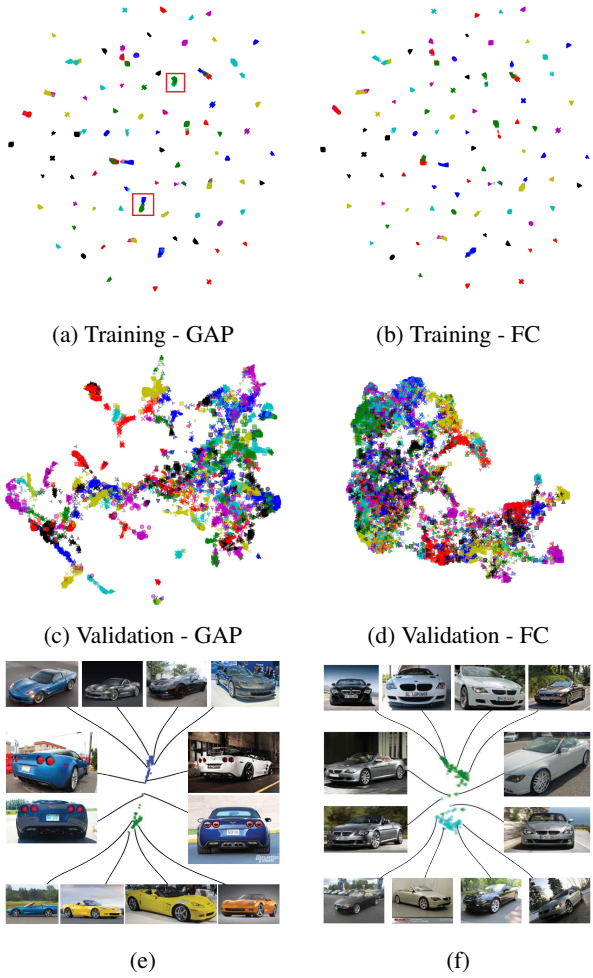
To highlight the potential applications of aligned visualizations, we explore three possible use cases with representations learned from words and images. The first case compares image embeddings defined by the features at different layers of a convolutional neural network. In the second case we show that the alignment makes it easier to compare word embeddings trained with two different datasets. The third case, shows the evolution of a network during training, with aligned visualizations of the representations learned after different epochs.

In all cases, the weight of the 2-MAP alignment term was set with the fully automatic approach defined in the previous section, making this approach to 2-MAP require no more user input than it would take to run UMAP on one dataset.

4.1. Learned Image Representations

An extensive line of research in image retrieval seeks the learn image representations that capture semantic similarities. Recent work compares the image retrieval performance of representations based on the final fully connected (FC) layer of a CNN, with representations based on the penultimate Global Average Pooling (GAP) layer [16]. That work shows that features extracted from the GAP layer generalize significantly better to new data and new classes. Here, we use the 2-MAP algorithm to visualize this phenomenon.

Training on the popular CAR-196 dataset[14], which includes 196 different classes of cars, we use Resnet-50 [17] as our backbone network, optimized by N-Pair Loss [15]. After training, we pass all training data and validation data through this trained model, recording the representation vector from the global average pooling (GAP) layer output and the fully connected (FC) output layer. Figure 4 shows



(a) Training - GAP (b) Training - FC

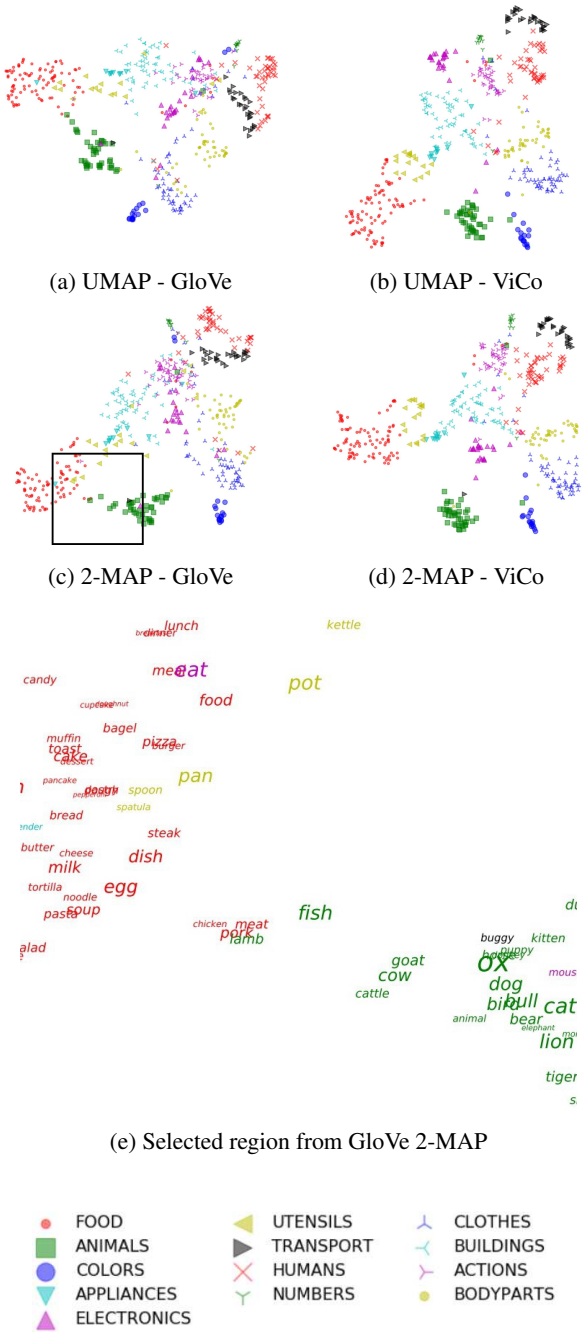
(c) Validation - GAP (d) Validation - FC

(e) (f)

Figure 4: Using the CAR-196 [14] dataset, trained with N-Pair loss [15], we show on the top a 2-MAP visualization of the training data, with the global average pooling (GAP) layer representation on the left, and the fully connected (FC) layer representation on the right. On the middle, we show the 2-MAP visualizations of the validation data. The training data 2-MAP results are well aligned, making it to notice clusters that are well separated in the FC representation but not well separated in the GAP representation. Examples of these clusters are shown in Figures 4e and 4f. In the validation data, the two datasets are sufficiently different that our strategy does not align them.

the aligned 2-MAP embeddings of the GAP and FC representations of the training data (top), and of the validation data (middle).

The structure of the FC and GAP representations in the validation data is quite different, and there is no good alignment, so our alignment procedure returns unrelated embeddings (although the additional structure in the embedding of the validation data still highlights why the GAP layer



(e) Selected region from GloVe 2-MAP

Figure 5: Comparing GloVe and ViCo word embedding features using UMAP and 2-MAP. We show (a) UMAP visualizations for the GloVe (5a) and (b) ViCo (5b) features. Corresponding aligned 2-MAP visualizations are shown in (c,d). Zooming in shows words that are similar in the GloVe embedding but not in the ViCo embedding. The aligned 2-MAP visualizations make it easy to find these sorts of differences.

may give better image retrieval results). The relationship between the GAP layer and the FC layer is more interesting to compare in the aligned embeddings of the training data representations. Because they are aligned, it is easier to find the regions that have differences between the two embeddings. We highlight two examples with small red boxes in places where the GAP layer embedding nearly merges two classes, but the FC layer separates them well. The alignment helps to find these locations. Zooming into these regions, the (bottom row) shows a few examples highlighting that the nearby clusters are very similar car models. Furthermore, the specific images that are embedded between clusters in the GAP embedding are the viewpoints from which those models look even more similar, with extremely similar tail lights.

4.2. Word Embeddings

Similarly to learned image representations, there is also significant work in learning how to represent language. In the word embedding task, the goal is to learn a representation where meaningfully similar words have similar representations. In this task, a low dimensional (2D or potentially 3D) visualization is a powerful tool to show and analyse the ability of a model. In this section we show how 2-MAP can be used to compare two different word embedding approaches on the same dataset. The GloVe word embedding features are trained using text only [18], while the ViCo word embedding features are trained using both text and associated images [19]. Figure 5 shows the UMAP and 2-MAP visualizations of both embeddings.

The 2-MAP visualization facilitates clear comparisons between the two approaches. In particular, our eyes were immediately drawn to the differences between the red and green clusters in the bottom left of the 2-MAP visualizations – in the GloVe visualization (bottom left), there are interspersed points from the two classes (outlined in black) that are obviously not confused in the ViCo visualization (bottom right). The red cluster corresponds to words in the class 'food', while the green cluster corresponds to the 'animal' class. In Figure 5e, we zoom in on this region and look at the actual words that are being confused. The words that GloVe embeds very nearby include 'chicken', 'meat', 'lamb' and 'fish' – all words whose correct class is disambiguated when images are able to help clarify whether the word is referring to the animal or to meat, as in the ViCo word embedding approach.

4.3. Visualizing Network Evolution

Aligning visualizations to each other makes it easier to observe how representations change over time. We again consider the CARS-196 dataset and use 2-MAP to observe how the representation changes during the training. Figure 6 shows the output of the thruMAP algorithm giving

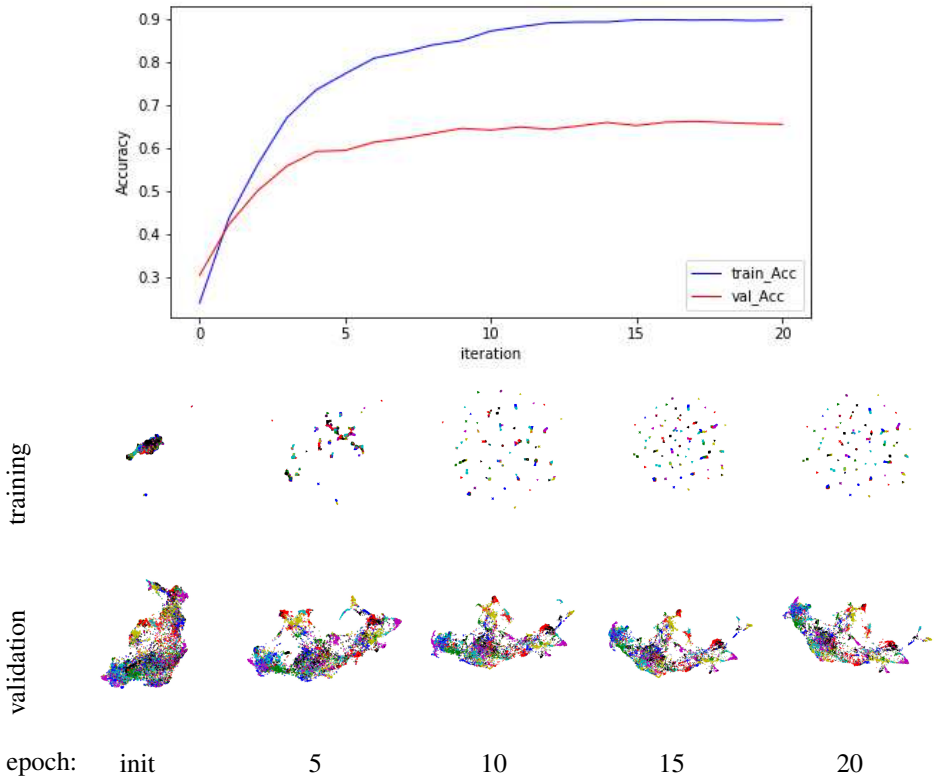


Figure 6: For the CAR dataset, we learn an image representation by training Resnet50 [17] with N-pair loss [15]. This figure shows how representations of training data and validation data change during the training process. The plot on the top shows the training and validation accuracy during training. The aligned 2-MAP visualizations on the bottom show how the representations of the training data and validation data change as training progresses.

aligned embeddings of both the training data (top) and the validation data (bottom).

Because they are aligned, it is easy to see how quickly most of the training data for this problem is separated into clusters, with a few classes remaining more overlapping and only becoming separated after many iterations. The validation data is less well clustered (sensibly, since the network has not seen these classes), and the evolution of this representation shows progress at a similar time-scale. Both the timing of when substantial changes are happening, and the final clustering is consistent with the change in accuracy as a function of training iterations.

5. Discussion

Comparing multiple high-dimensional representations of the same dataset is challenging. In the domain of image representation learning, this is mostly done by comparing image retrieval results. Standard visualizations based on UMAP or t-SNE are not optimal for such comparisons because they may embed similar representations in very different ways. Sometimes these differences reflect truly important differences in the structure of the high-dimensional representation, but sometimes the differences arise from arbitrary choices in how to map local relationships in the high-dimensional space onto the low-dimensional embedding.

Variations in those arbitrary choices are why one high-dimensional representation may have many different embeddings with approximately the same cost, but this also suggests that it is possible to re-arrange the two embeddings to be similar to each other without increasing the cost. We give a simple approach to do this, and show several examples of how this could be used by the research community to better understand or explain different behaviors in the representation learning field.

We extended this approach to a sequence of high-dimensional representations. Future work could consider a larger collection of representations that all need to be aligned. However, it can be challenging to compare many different embeddings side by side, even if they are well aligned, suggesting the need for additional work in how best to visualize and interact with this information.

References

[1] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[2] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

[3] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. Chapman and Hall/CRC, 2000.

[4] John W Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on computers*, 100(5):401–409, 1969.

[5] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[6] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[7] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[8] Martin Wattenberg, Fernanda Viégas, and Ian Johnson. How to use t-SNE effectively. *Distill*, 1(10):e2, 2016.

[9] Etienne Becht, Charles-Antoine Dutertre, Immanuel WH Kwok, Lai Guan Ng, Florent Ginhoux, and Evan W Newell. Evaluation of UMAP as an alternative to t-sne for single-cell data. *bioRxiv*, page 298430, 2018.

[10] Svetlana Ovchinnikova and Simon Anders. Exploring dimension-reduced embeddings with sleepwalk. *bioRxiv*, 2019.

[11] Paulo E. Rauber, Alexandre X. Falcão, and Alexandru C. Telea. Visualizing time-dependent data using dynamic t-sne. In *Proceedings of the Eurographics / IEEE VGTC Conference on Visualization: Short Papers*, EuroVis '16, pages 73–77, Goslar Germany, Germany, 2016. Eurographics Association.

[12] James Melville. smallvis documentation. Available at `https://jlmelville.github.io/smallvis/`.

[13] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

[14] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[15] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1857–1865. Curran Associates, Inc., 2016.

[16] Nam Vo and James Hays. Generalization in metric learning: Should the embedding layer be embedding layer? In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 589–598. IEEE, 2019.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[18] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[19] Tanmay Gupta, Alexander Schwing, and Derek Hoiem. ViCo: Word embeddings from visual co-occurrences. *arXiv preprint arXiv:1908.08527*, 2019.