Supplementary Material
Visualizing Deep Neural Nets with UMAP Tour

Anonymous Authors[1]

1. Computing Infrastructure

All computations (the training and forward pass of neural networks, computation of UMAP embeddings) are conducted on a 12-core Intel(R) Xeon(R) CPU (E5-2603 v3 @ 1.60GHz) with 264 GB of memory. We also used one Tesla K80 GPU (with 11 GB memory) in the training and forward passing steps.

2. Detail of Experiments

We visualized two models of GoogLeNet: one pre-trained model provided by PyTorch [1] and one trained on the above infrastructure [2]. The training takes 30 epochs ($\sim$ 60 hours) over the training set of ImageNet with simple image preprocessing (resize, central crop and normalize). On both models, we observed cluster of humans, orientation of animals and animal faces. On the model we trained, we further observed the migration of dog-with-human images from the cluster of humans to the cluster of dogs from the 12-Inception layer to 19-Linear Layer [3].

3. Runtime Complexity

When processing data for UMAP Tour, we pass image examples through the network, downsample the neuron activations through average pool and fit UMAP embeddings. The dimensionality (before and after average pooling) and the runtime on each layer of GoogLeNet are listed in Figure 1a. In Figure 1b, we plot run time of UMAP embedding as a function of its input dimensionality. The runtime is based on the passing 50000 ImageNet validation images to GoogLeNet and fitting the UMAP embeddings.

4. Impact of UMAP Embedding Dimensions

We analyzed the impact of embedding dimension on the quality of UMAP. For each layer we embed its neuron activations in different dimensions through UMAP and evaluate its embedding loss. UMAP is a randomized algorithm but we only considered one embedding for each case, which already gives us enough information on the general trend. In practice, one can run UMAP multiple times on the same data and pickup the one that gives the best quality. Figure 2 shows the embedding loss on every layer of GoogLeNet. In general, the embedding loss depends on both the complexity of data (i.e. layers) and the dimensionality of the embedding. Later layers typically to have a lower UMAP loss.

5. More Results

Figure 3, 4a and 4b show other patterns we found within GoogLeNet, when probing it with the ImageNet or FairFace datasets. On the 4-BasicConv2d layer, we see images mostly arranged by colors (Figure 3). On the 13-Inception layer, we find orientation of faces (Figure 4a). On the 15-Inception layer, we observed clusters of cars, birds and food (Figure 4b).

6. Comparing UMAP with Principle Component Analysis

We compare our method to principle component analysis (PCA) in Figure 5. On each layer, we found 15 principle directions by PCA and compare it against the views given by UMAPs. We see 15-dimensional PCA fails to capture interesting patterns within each layer, especially in layer layers. Due to the batch normalizations in most layers, the distribution of neuron activation vectors in these layers is almost spherical (i.e. having unit standard deviation in every direction along the standard basis). In this case, the variance in data can not reliably reflect the importance of each direction, therefore PCA fails.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

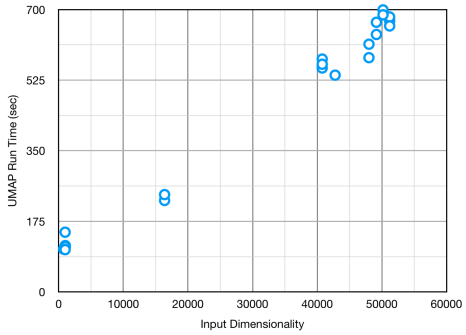
[1]https://umap-tour.github.io/googlenet.html

[2]https://umap-tour.github.io/googlenet-train.html

[3]https://www.dropbox.com/s/zbxadke4gr6r479/2-layer-dynamics-dogs.mov?dl=0

	Data Dimensionality		Run Time (sec)		
Layer	Before Pooling	After Pooling	Forward	UMAP	Total
0-Input	3x224x224	3x16x16	205.5	106.1	311.7
1-BasicConv2d	64x112x112	64x16x16	262.8	226.8	489.6
2-MaxPool2d	64x56x56	64x16x16	208.2	241.3	449.5
3-BasicConv2d	64x56x56	64x16x16	208.9	239.9	448.8
4-BasicConv2d	192x56x56	192x16x16	242.3	638.1	880.5
5-MaxPool2d	192x28x28	192x16x16	217.1	668.4	885.4
6-Inception	256x28x28	256x14x14	217.4	699.1	916.5
7-Inception	480x28x28	480x10x10	219.3	580.7	800.0
8-MaxPool2d	480x14x14	480x10x10	217.1	614.0	831.1
9-Inception	512x14x14	512x10x10	217.9	671.7	889.7
10-Inception	512x14x14	512x10x10	213.6	681.6	895.2
11-Inception	512x14x14	512x10x10	216.8	659.3	876.0
12-Inception	528x14x14	528x9x9	214.6	537.4	751.9
13-Inception	832x14x14	832x7x7	216.9	555.1	772.0
14-MaxPool2d	832x7x7	832x7x7	210.6	577.2	787.8
15-Inception	832x7x7	832x7x7	213.1	564.4	777.5
16-Inception	1024x7x7	1024x7x7	216.5	686.9	903.4
17-AdaptiveAvgPool2d	1024x1x1	1024x1x1	213.5	114.9	328.4
18-Flatten	1024	1024	205.2	111.7	316.9
19-Linear	1000	1000	207.2	104.8	312.1
20-Softmax	1000	1000	206.4	148.1	354.5

(a) Runtime of UMAP Tour when passing 50000 ImageNet examples through GoogLeNet.



(b) UMAP's input dimensionality vs run time. The run time of UMAP embedding scales almost linearly with its input dimensions.

7. Alignment Induced by Centered Kernel Alignment

Orthogonal Procrustes works by finding the best rotation (Q) that aligns two p-dimensional configurations X and Y.

$$argmin_Q \; ||XQ - Y||_F^2, \quad \text{subject to } Q^TQ = I$$

Similarly, centered kernel alignment (CKA) also comes with an alignment. As mentioned in (Kornblith et al., 2019), the alignment (W) induced by CKA can also be formulated into a similar optimization criterion:

$$argmin_W \; ||XW - Y||_F^2, \quad \text{subject to } tr(W^TW) = 1 \tag{1}$$

This only differs from orthogonal Procrustes in the constraint. The constraint is more flexible than orthogonal Procrustes, since $tr(Q^TQ) = p$ is also a constant. To better

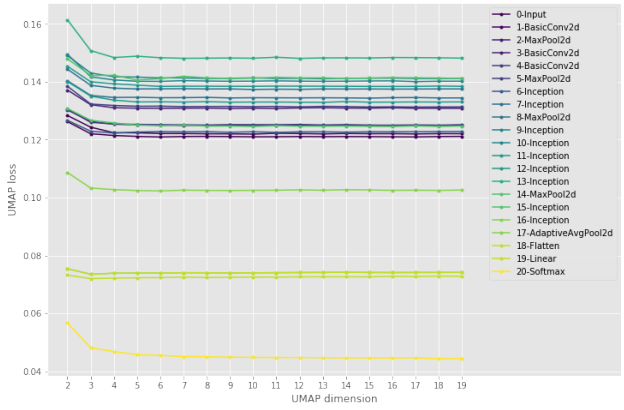


Figure 2. Analysis of UMAP embedding dimensions. UMAP's embedding loss depends on both the complexity of data and the dimensionality of the embedding. In general, later layers tend to have a lower UMAP loss. The UMAP loss decreases with the increase of embedding dimensions. In other words, for most layers the embedding quality benefits from having more than 3 dimensions.

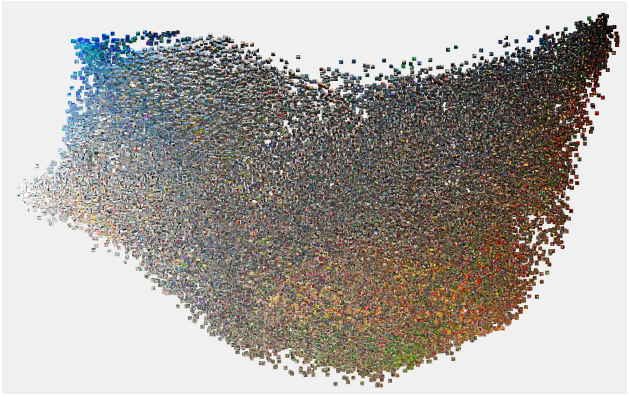


Figure 3. On the 4-BasicConv2d layer, images are mostly arranged by colors.

compare these two constraints, let us scale W by $\sqrt{p}$:

$$\hat{W} = \sqrt{p}W$$

so that

$$tr(\hat{W}^T\hat{W}) = p = tr(Q^TQ)$$

Geometrically, the new constraint requires the (scaled) alignment $\hat{W}$ to be a 'diagonal preserving' transform. To see why, consider a unit box that is aligned with the left singular vectors of $\hat{W}$. Since $tr(\hat{W}^T\hat{W})$ is the sum of squared singular values of $\hat{W}$,

$$tr(\hat{W}^T\hat{W}) = \sum_{i=1}^{p} \hat{\sigma}_i^2$$

$\hat{W}$ stretches the unit box in every (left singular) direction by σ_i while preserving the diagonal of it. The optimal alignment under the criterion in Eq.1 can be found via Lagrange multiplier.

$$W^* = cX^TY$$

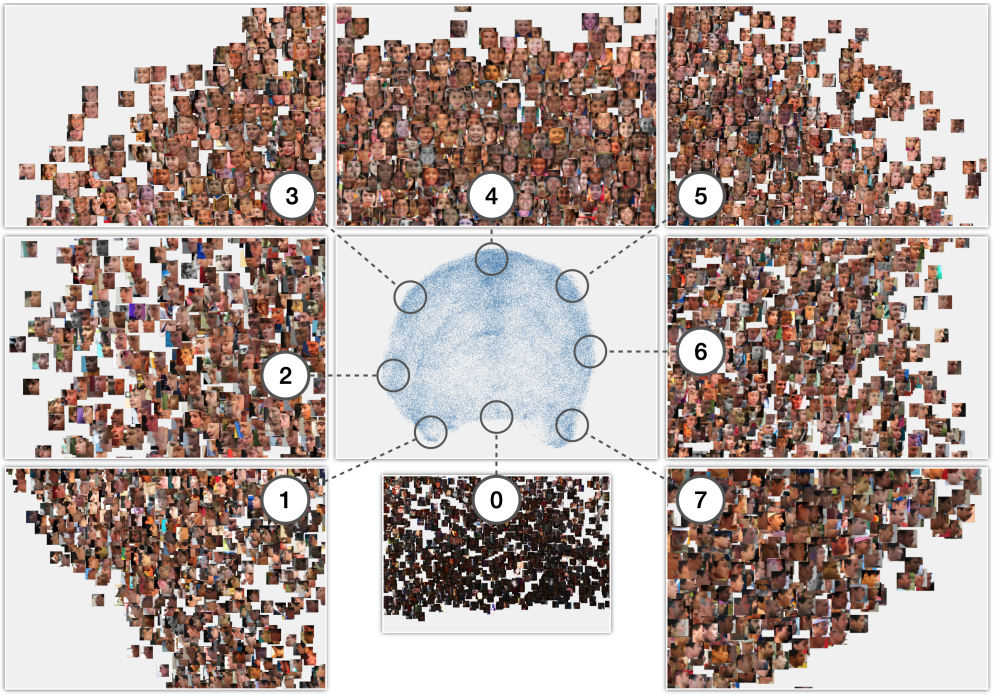
where $c \in \mathbb{R}$ is a normalizing factor which makes W satisfy the constraint. It is not clear to us how this alignment may change the appearance of data in UMAP Tour. We would like to look into this direction in the future.

8. Links

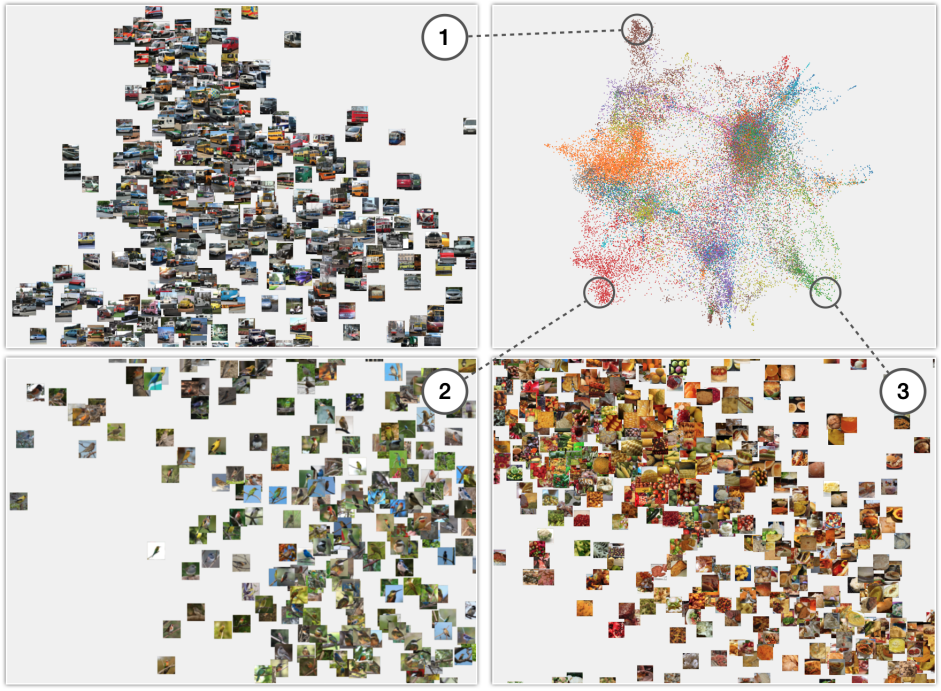
Source code:
https://github.com/umap-tour/source
Demo:
https://umap-tour.github.io
Video:
https://www.dropbox.com/sh/2za8rt3tpz35w7u/AAAKDpvgKDKvjRcthSTtexbja?dl=0
We share source code to this work through a GitHub account (umap-tour) that was created for this submission only. Our script for the demo website are hosted through GitHub Pages of the same account. In the demo, we neither reveal the identity of ours nor track the visitors. To our knowledge, GitHub Pages do not provide access to logs of its visitors either.

References

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019.



(a) Orientation of faces (① - ⑦: left to right) found in the 13-Inception layer.



(b) Clusters found in the 15-Inception layer: ① cars, ② birds and ③ food.

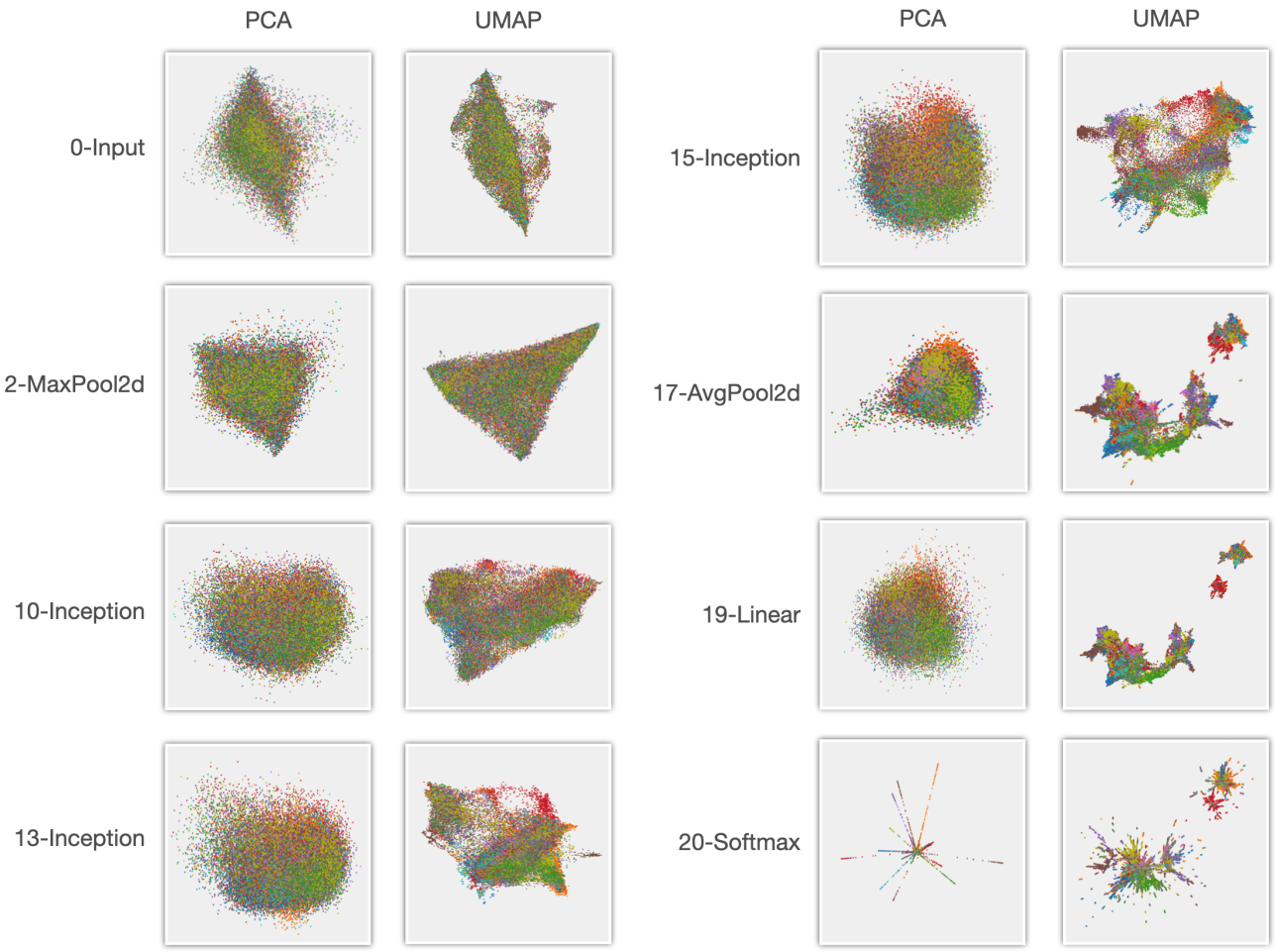


Figure 5. UMAP vs PCA. PCA fails to capture interesting patterns within each layer.