Google AI Blog

The latest news from Google AI

Realtime tSNE Visualizations with TensorFlow.js

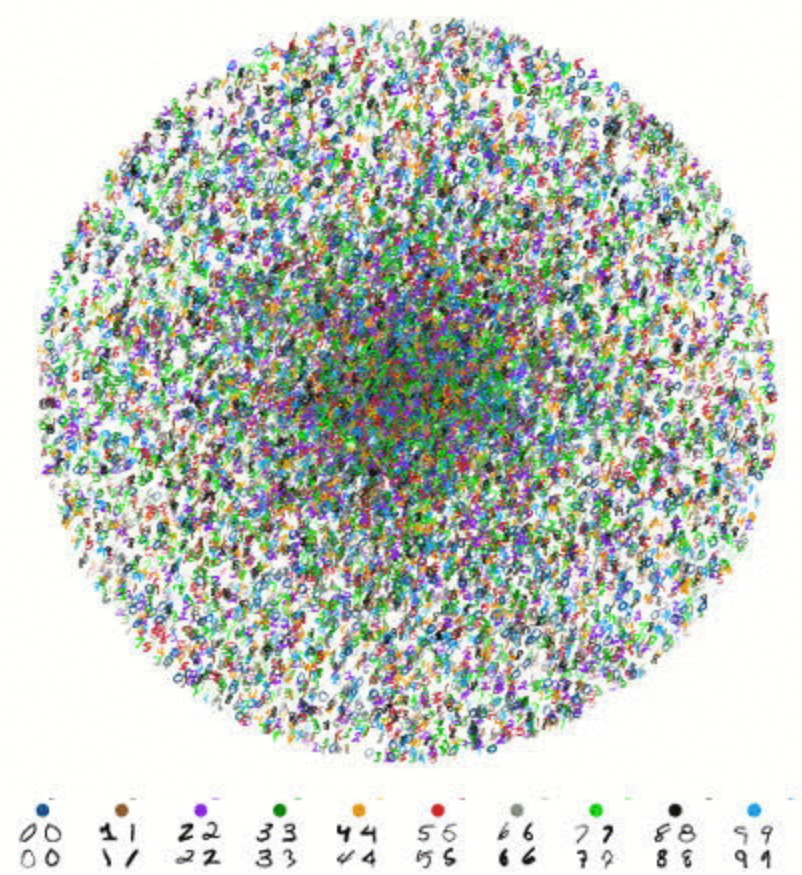
Thursday, June 7, 2018

Posted by Nicola Pezzotti, Software Engineering Intern, Google Zürich

In recent years, the t-distributed Stochastic Neighbor Embedding (tSNE) algorithm has become one of the most used and insightful techniques for exploratory data analysis of high-dimensional data. Used to interpret deep neural network outputs in tools such as the TensorFlow Embedding Projector and TensorBoard, a powerful feature of tSNE is that it reveals clusters of high-dimensional data points at different scales while requiring only minimal tuning of its parameters. Despite these advantages, the computational complexity of the tSNE algorithm limits its application to relatively small datasets. While several evolutions of tSNE have been developed to address this issue (mainly focusing on the scalability of the similarity computations between data points), they have so far not been enough to provide a truly interactive experience when visualizing the evolution of the tSNE embedding for large datasets.

In “Linear tSNE Optimization for the Web”, we present a novel approach to tSNE that heavily relies on modern graphics hardware. Given the linear complexity of the new approach, our method generates embeddings faster than comparable techniques and can even be executed on the client side in a web browser by leveraging GPU capabilities through WebGL. The combination of these two factors allows for real-time interactive visualization of large, high-dimensional datasets. Furthermore, we are releasing this work

as an open source library in the TensorFlow.js family in the hopes that the broader research community finds it useful.



Real-time evolution of the tSNE embedding for the complete MNIST dataset with our technique. The dataset contains images of 60,000 handwritten digits. You can find a live demo here.

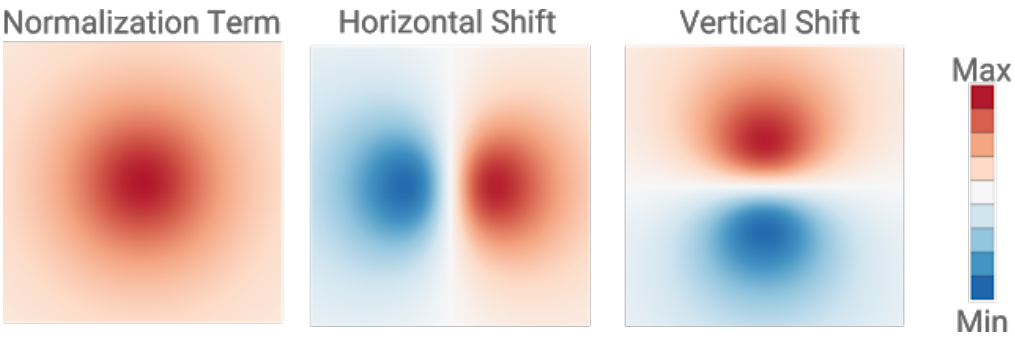
The aim of tSNE is to cluster small "neighborhoods" of similar data points while also reducing the overall dimensionality of the data so it is more easily visualized. In other words, the tSNE objective function measures how well these neighborhoods of similar data are preserved in the 2 or 3-dimensional space, and arranges them into clusters accordingly.

In previous work, the minimization of the tSNE objective was performed as a N-body simulation problem, in which points are randomly placed in the embedding space and two different types of forces are applied on each point. Attractive forces bring the points closer to the points that are most similar in the high-dimensional space, while repulsive forces push them away from all the neighbors in the embedding.

While the attractive forces are acting on a small subset of points (i.e., similar

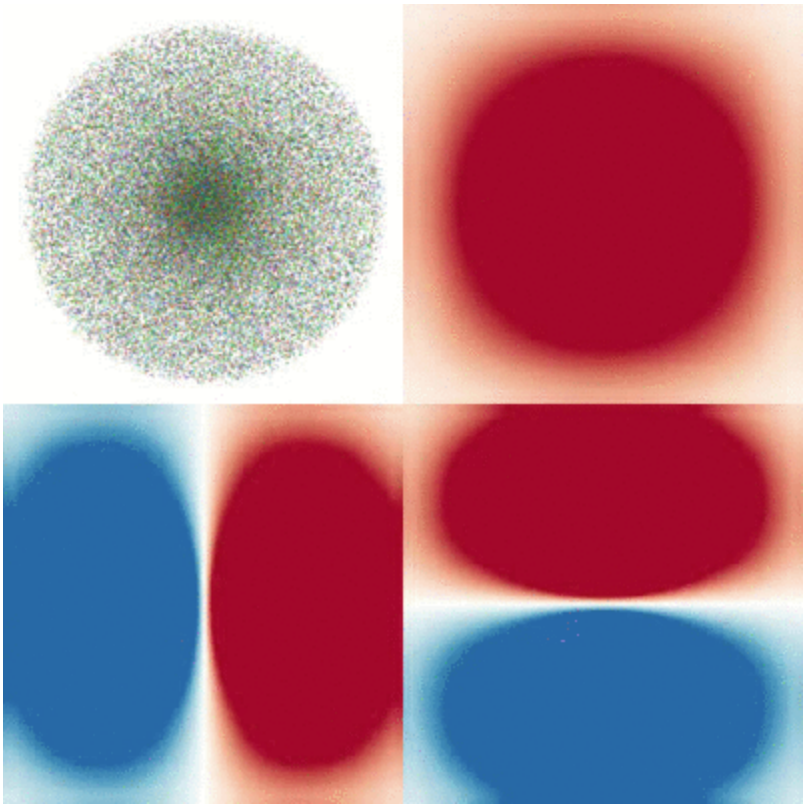
neighbors), repulsive forces are in effect from all pairs of points. Due to this, tSNE requires significant computation and many iterations of the objective function, which limits the possible dataset size to just a few hundred data points. To improve over a brute force solution, the Barnes-Hut algorithm was used to approximate the repulsive forces and the gradient of the objective function. This allows scaling of the computation to tens of thousand data points, but it requires more than 15 minutes to compute the MNIST embedding in a C++ implementation.

In our paper, we propose a solution to this scaling problem by approximating the gradient of the objective function using textures that are generated in WebGL. Our technique draws a “repulsive field” at every minimization iteration using a three channel texture, with the 3 components treated as colors and drawn in the RGB channels. The repulsive field is obtained for every point to represent both the horizontal and vertical repulsive force created by the point, and a third component used for normalization. Intuitively, the normalization term ensures that the magnitude of the shifts matches the similarity measure in the high-dimensional space. In addition, the resolution of the texture is adaptively changed to keep the number of pixels drawn constant.



Rendering of the three functions used to approximate the repulsive effect created by a single point. In the above figure the repulsive forces show a point in a blue area is pushed to the left/bottom, while a point in the red area is pushed to the right/top while a point in the white region will not move.

The contribution of every point is then added on the GPU, resulting in a texture similar to those presented in the GIF below, that approximate the repulsive fields. This innovative repulsive field approach turns out to be much more GPU friendly than more commonly used calculation of point-to-point interactions. This is because repulsion for multiple points can be computed at once and in a very fast way in the GPU. In addition, we implemented the computation of the attraction between points in the GPU.



This animation shows the evolution of the tSNE embedding (upper left) and of the scalar fields used to approximate its gradient with normalization term (upper right), horizontal shift (bottom left) and vertical shift (bottom right).

We additionally revised the update of the embedding from an ad-hoc implementation to a series of standard tensor operations that are computed in TensorFlow.js, a JavaScript library to perform tensor computations in the web browser. Our approach, which is released as an open source library in the TensorFlow.js family, allows us to compute the evolution of the tSNE embedding entirely on the GPU while having better computational complexity.

With this implementation, what used to take 15 minutes to calculate (on the MNIST dataset) can now be visualized in real-time and in the web browser. Furthermore this allows real-time visualizations of much larger datasets, a feature that is particularly useful when deep neural output is analyzed. One main limitation of our work is that this technique currently only works for 2D embeddings. However, 2D visualizations are often preferred over 3D ones as they require more interaction to effectively understand cluster results.

Future Work

We believe that having a fast and interactive tSNE implementation that runs

in the browser will empower developers of data analytics systems. We are particularly interested in exploring how our implementation can be used for the interpretation of deep neural networks. Additionally, our implementation shows how lateral thinking in using GPU computations (approximating the gradient using RGB texture) can be used to significantly speed up algorithmic computations. In the future we will be exploring how this kind of gradient approximation can be applied not only to speed-up other dimensionality reduction algorithms, but also to implement other N-body simulations in the web browser using TensorFlow.js.

Acknowledgements

We would like to thank Alexander Mordvintsev, Yannick Assogba, Matt Sharifi, Anna Vilanova, Elmar Eisemann, Nikhil Thorat, Daniel Smilkov, Martin Wattenberg, Fernanda Viegas, Alessio Bazzica, Boudewijn Lelieveldt, Thomas Höllt, Baldur van Lew, Julian Thijssen and Marvin Ritter.

14 comments

Add a comment as Nicola Pezzotti

Top comments

Google AI via Google+ 4 days ago - Shared publicly

Some new research from an intern in our Zürich office shows an approach to tS
allows real-time interactive visualization of large, high-dimensional datasets by
GPU capabilities through WebGL. Oh, and it's open source too! Check it out ↓

+62 1 · Reply

View all 5 replies

Carl Geiser 3 days ago
Mmm

gia huy 3 days ago
+Lan Nguyen Unikey đi Lan :)
· Translate

Elango Shunmugaraj shared this 19 hours ago - Google Apps Script (News)

+1 1

I.J. Atencio shared this 4 days ago - Machine Learning (Resources)

+8 1 · Reply

Boris Debic via Google+ 3 days ago - Shared publicly

https://ai.googleblog.com/2018/06/realtime-tsne-visualizations-with.html

Labels: Algorithms , optimization , TensorFlow , Visualization