Semi-Supervised Deep Learning for Molecular Structures

Alvin Wan
alvinwan@berkeley.edu
Computer Science
Third Year Undergraduate

Allen Guo
allenguo@berkeley.edu
Computer Science
Third Year Undergraduate

Problem Definition and Motivation

Recent advances in microscopy have made it possible to image biomolecules in 3D at the scale of nanometers. However, interpreting these images is onerous, and typically requires manual labeling of molecular structures using 2D projections of the data. We sought to streamline this process using deep learning, drawing inspiration from existing research in point cloud classification.

Our work involved classifying 3D images of clathrin, a protein involved in cell transport. During clathrin-mediated endocytosis (CME), clathrin surrounds molecules awaiting transport, forming a spherical coat. Our goal was to pick out images depicting clathrin undergoing CME.

The clathrin data was provided by the Ke Xu lab in the College of Chemistry, whose research work we are supporting. Their ultimate goal is to better understand how clathrin interacts with other proteins (actin, in particular) during cell transport. While this is a fundamental problem in biology in and of itself, broader applications that might result from a stronger understanding of CME include more efficient, more specific drug delivery.

Related Work and Comparison

In general, existing work in classifying point clouds relies on previously segmented objects. However, large bodies of labeled data do not exist alongside the inception of super-resolution molecular imaging.[1] Our work employs unsupervised techniques to sidestep this issue. More importantly, most existing work is tailored to urban settings and indoor environments, whereas this project is focused on objects with potentially many orientations and related structures.

One proposed speedup found in the literature considers reducing the 3D classification problem to multiple 2D projections, to save time. However, microscopies have highest variance in only one hyperplane, minimizing the efficacy of this approach.[2] In short, previous solutions are insufficient for challenges specific to microscopies and molecular data.

Approach

Datasets

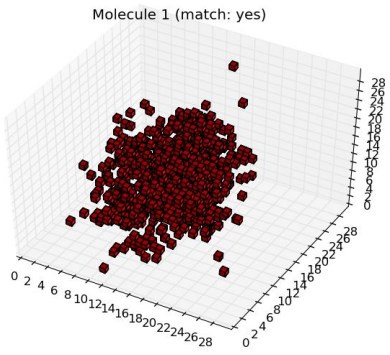
Our primary dataset consists of 899 images of clathrin, created using a super-resolution microscopy technique called Stochastic Optical Reconstruction Microscopy (STORM). Each image is presented a point cloud, i.e., a set of

[1] "Convolutional-Recursive Deep Learning for 3D Object Classification" [Ng et al., NIPS 2012] https://papers.nips.cc/paper/4773-convolutional-recursive-deep-learning-for-3d-object-classification.pdf

[2] "Fast and Robust Multi-View 3D Object Recognition in Point Cloud" [Pang, Neuman '16] http://graphics.usc.edu/cgit/publications/papers/3dv2015.pdf

points in 3D space. Out of the 899 images, only 280 have labels (assigned manually by the Ke Xu lab).

Below is an example of a clathrin point cloud, voxelized into a 30x30x30 binary image:

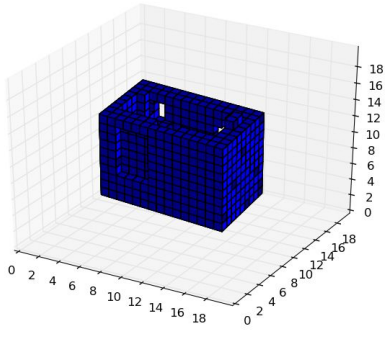


The label ("match: yes") indicates that it is in fact an example of a clathrin undergoing CME.

In addition to the STORM dataset, we used two other datasets to validate our approach:

- Geometries, a homegrown dataset containing 600 geometric figures (cubes, spheres, and diamonds) with added noise.
- Princeton ModelNet10, which contains ~57,000 real-world objects drawn from 10 categories (bed, chair, etc.).[3]

Below is an example from ModelNet10 (a bathtub):



[3] "3D ShapeNets: A Deep Representation for Volumetric Shapes" [Wu, et al.CVPR 2015] http://modelnet.cs.princeton.edu/.

Models

Because we had far more unlabeled examples than labeled examples, we chose a semi-supervised approach: we trained unsupervised models to generate featurizations (encodings) for each data point. We then trained a support vector machine (SVM) on the learned features of the labeled subset of the data. In this report, we run various neural network autoencoders against classical encoding methods.

Our primary unsupervised model was a 3D convolutional autoencoder (3DCAE), implemented in TensorFlow. We experimented with a variety architectures. We also implemented a fully-connected autoencoder (FAE) that does not use convolutions.

We also experimented with:

- Soft k-means clustering, which is relatively simple but has, in past comparisons of unsupervised learning approaches, proven to be quite powerful.[4] In soft k-means, we use the standard k-means algorithm to find clusters, then featurize a single data point x_i by considering its distance to each cluster center.
- Principal components analysis (PCA), which learns a linear projection of the data points into a lower-dimensional subspace.

We also implemented a regular 3D convolutional neural network in TFLearn, which we used to directly classify the data. We tried this supervised method to provide a contrast to the unsupervised approaches.

[4] "An Analysis of Single-Layer Networks in Unsupervised Feature Learning" [Coates et al. AISTATS 2011] https://cs.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf.

Evaluation Method

To assess our performance on the STORM dataset, we computed classification accuracies on the labeled subset of the data. The Ke Xu lab previously ran logistic regression on this subset and obtained 63% classification accuracy, which we were able to reproduce. The features used by the lab were size, angular density, and a score indicating roundness.

Results

We spent a significant amount of time building groundwork: getting up to speed with TensorFlow, figuring out how to both generate and visualize 3D data, writing software for creating voxels from point clouds, and so on.

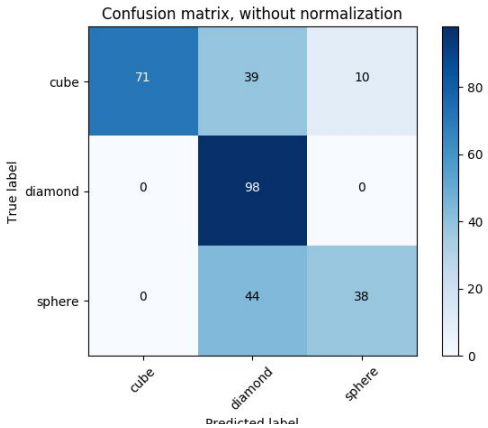
We implemented and evaluated our 3D convolutional autoencoder (3DCAE) and our k-means learner on our geometries dataset. The results are summarized below.

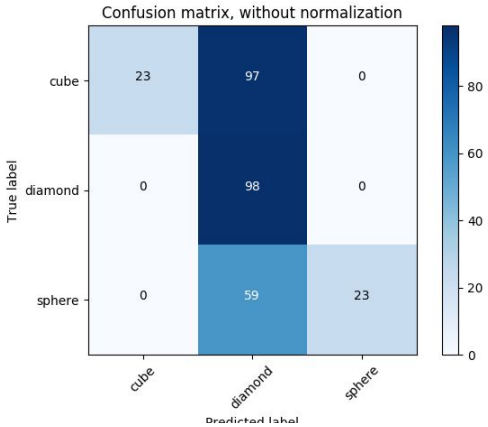
Model	Cube acc.	Diamond acc.	Sphere acc.	Overall acc.
3DCAE (sparse)	50.8%	100%	70.7%	72.3%
3DCAE	59.1%	100%	46.3%	69%
k-means	19.2%	100%	28.0%	48%

The 3DCAE result labeled "sparse" corresponds to the regular 3DCAE with an added sparsity penalty. (See Andrew Ng's notes on sparse autoencoders for details.[5]) Both the sparse and non-sparse autoencoders performed significantly better than random guessing (33.3%) across all accuracy categories. When training our SVM on the featurized data, we ran an exhaustive grid search over the possible kernels (RBF, polynomial, linear) and values of C, the regularization hyperparameter.

[5] "Sparse Autoencoder" [Ng, Andrew] https://web.stanford.edu/class/cs294a/sparseAutoencoder.pdf

Looking at the confusion matrices for the three models, it seems that both cubes and spheres are commonly misclassified as diamonds. For instance, here are the confusion matrices for the 3DCAE and k-means learner respectively:

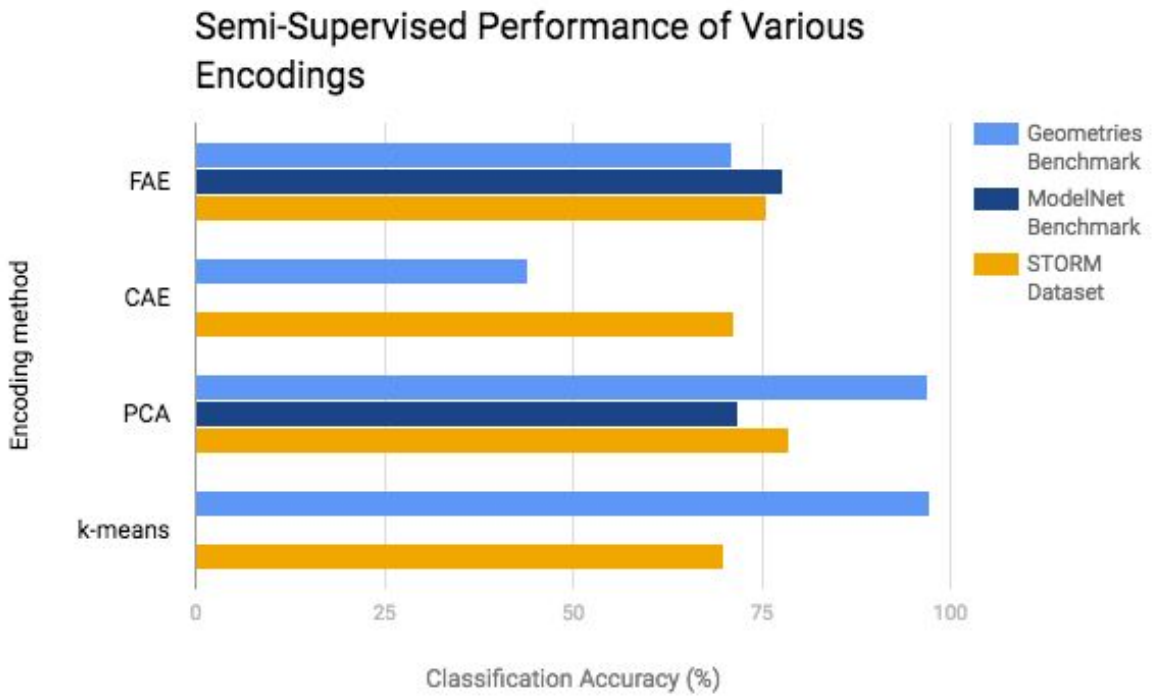




We worked extensively to verify that the models were in fact learning properly. For example, we looked at the 3DCAE outputs at different point during the training process:

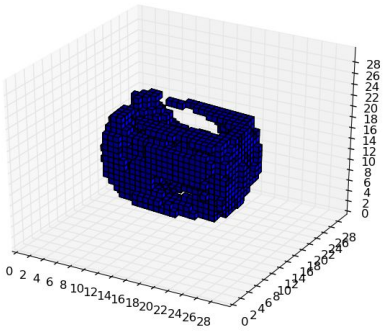
(Click here for larger view.)

This is a timelapse of the 3DCAE learning to reconstruct a 12x12x12 shape (shown in *red*) consisting purely of random noise. The *blue* shapes are the outputs from the decoder at various stages of training, from the start of training (leftmost) to convergence (rightmost).



All encodings used 64 latent dimensions. The highest accuracy (not listed) was PCA with SVM for 100 latent dimensions at 78.57%. K-means did not finish running for ModelNet, and the 3DCAE ran into a memory error. Both neural networks were ran for 10 epochs. We used a grid search for the optimal SVM hyperparameters. We noted reconstruction accuracy was not correlated with classification accuracy.

After we were satisfied with our results on the geometries dataset, we attempted to apply our neural network to the aforementioned ModelNet dataset, which is far more challenging. 3DCAE ran into memory issues, so we simplified the neural network by removing convolutional layers; this resulted in nearly 80% accuracy. State-of-the-art neural networks, tailored to supervised tasks, achieved 91%. We noted that the autoencoder's reconstructions performed fairly well. The following is a reconstruction of the bathtub pictured in the Dataset section above:



Our models performed reasonably on both benchmarks, so we moved onto the STORM dataset and optimized over various hyperparameters - learning rates, learning schedules, filter sizes, hidden layer sizes, and

input sizes - to obtain our final results. To bolster the dataset for deep learning to have a chance at success, we employed label spreading to 619 additional samples. With a combined dataset, we used translations and rotations to generate over 60,000 training samples. Despite this, deep learning approaches did not perform better than classical methods. Namely, PCA with an SVM resulted in the best performing classifier, at 78.57%. In light of superior deep learning performance with the ModelNet task, where we had over 50,000 un-augmented, unique sample points, we suspect that insufficient variance in our STORM dataset led to sub-par deep learning results.

Efficacy of Semi-Supervised Approach

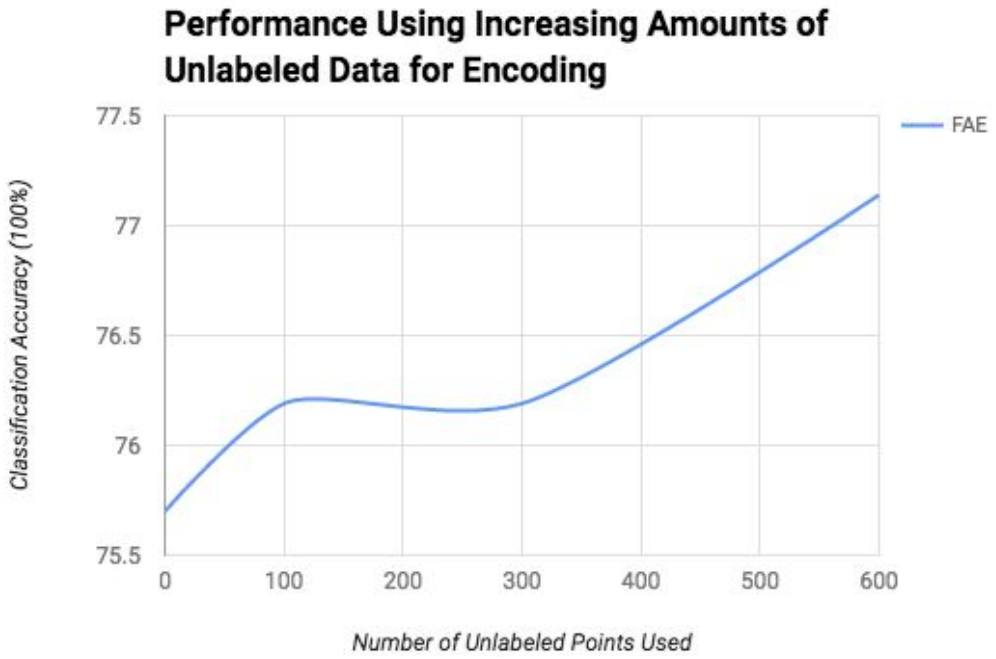
Purely supervised methods performed subpar, and semi-supervised methods benefited from an increasing number of unlabeled samples.

Supervised Approach

We tested two end-to-end 3D convolutional neural network architectures, both of which attained 61.43% at most, failing to surpass our baseline performance of 63% with logistic regression. The first involved just one convolutional layer and two fully-connected ones, and the second involved two convolutional layers and three fully-connected ones, with various filter sizes and number of filters, ranging from 100 to 2000 parameters.

Semi-Supervised Approach

We used label spreading to include increasing numbers of unlabeled samples, which were then used to train the various encoding techniques. On the fully-connected autoencoder, we found that the number of unlabeled samples used, on top of already-labeled samples, was positively correlated with classification accuracy. Improvements were not drastic but were reliable and measurable, validating our semi-supervised approach.



Conclusion

Our best unsupervised learning method, PCA, performed respectably on the clathrin dataset, achieving nearly 80% accuracy. Conversely, deep neural networks excelled on the ModelNet benchmark dataset, but did not perform well on the STORM dataset. We believe this is because there is simply not enough data to train on. Ultimately, we expect that our work will enable the Xu lab to classify clathrin and other molecular structures more efficiently.