CPSC 553 Final Project

Introduction

In the development of drugs, molecules need to bind together to create new molecules that have different properties. In the field of computational chemistry, we have the docking score to measure the binding affinity between two molecules. Docking is a technique that samples conformations of small molecules in protein binding sites and gives a score to assess which of these conformations best complements the protein binding site. In short, if the dock score is higher, the bond between the ligand and the protein will be stronger. In this paper we look to predict the docking score while finding representations of the molecule structure that we can draw inferences from. The main goal is to explore the molecule space using a combination of supervised and unsupervised methods and find potential relationships related to the docking score. Since molecular structures are historically complex, we use the SMILES of each molecule. The SMILES is the simplified molecular-input line-entry system. which is a way of representing the chemical structure of a molecule using only ASCII characters. Using these SMILES, we can predict the corresponding docking scores to rank protein-protein docked poses.

The SMILES are in the form of ASCII strings, which tend to not work too well with numeric operations, which leads us to use the Python package RDkit. RDkit is an extensive package for computational chemistry/cheminformatics, however we will just use the function that converts SMILES to fingerprints. Fingerprints give us the same amount of information as SMILES, but it is in a less human-readable format in the form of 1024-digit long binary strings. We will use five different neural network-based approaches including two separate networks acting as regressors, two autoencoders, and a conditional generative adversarial network or cGAN. We also use PCA and t-SNE on the embeddings of the regressors and autoencoders.

Related Works

There have been various deep learning approaches related to SMILES and attempting to predict binding affinity. The paper by Zhu et al [1] attempts to input the protein, the ligand, and the distance between them and try to predict a binding affinity. A paper that we base part of our work on is by Ozturk et al [2] known as DeepDTA. DeepDTA inputs the protein sequence and the SMILES to try to predict the binding affinity. Instead of using the fingerprint like in our work, DeepDTA uses embedded SMILES to feed into the network. It also uses two convolutional networks where one takes the protein sequence as the input and the other takes the embedded SMILES as input. Each network produces an embedding, the embeddings are then concatenated and fed to a fully connected network and the docking score is predicted.

As far as autoencoders, Castro et al [3] used an autoencoder approach to embed the structure of molecules to find important features and variations. This approach known as a geometric scattering autoencoder (GSAE) employs an auxiliary network that predicts meta properties of

the structure. Since the meta properties are known in the dataset, these properties can be predicted using supervised learning.

The paper by Amodio et al [4] serves as inspiration to the GAN aspect of this work. This paper employs a cGAN to generate hard to obtain information based on its easy to obtain counterpart. This framework is known as Feature Mapping GAN or FMGAN. The FMGAN takes the easy to obtain data concatenated with noise and feeds it to the generator. The generator will then output generated hard to obtain data. The discriminator will determine if the generated hard to obtain data is comparable to the true hard to obtain data. The significance of this is after training, only easy to obtain data would need to be collected while hard to obtain data could be generated based on the easy to obtain data.

Methods

Regressors

As previously stated, we use five different methods in order to explore different scenarios. The first two of which are regressors. We use regressors simply because we want to try to predict docking scores and get the embedding of the fingerprint. First, we use a standard four layer network with four fully connected layers. We use this network as a baseline for the following networks. The baseline network takes in the 1024-digit fingerprint as input and tries to predict the dockscore, which is in the range [-0.08, -73.47]. We get the embedding by passing the data through the network and taking the output at the third layer, which has a size of 128 neurons. We will later use this embedding in PCA and t-SNE.

Second, similar to the DeepDTA work, we use a convolutional network to predict binding affinity. The differences are that we only use one network because we are only passing in the SMILES data equivalent, the fingerprint, and as opposed to using the fingerprint the authors of DeepDTA used an embedding. Since our input data is one-dimensional, we use two one-dimensional convolutional layers with max pooling and two fully connected layers. Similar to the baseline network, we get the embedding by passing the data through the network and getting the output after the first linear layer, the third layer overall.

Autoencoders

Another method that we use is two autoencoders. We use autoencoders because they tend to provide great embeddings, as the decoder needs to reconstruct the embedding after it is encoded. The first of which is a vanilla autoencoder with two fully connected layers for the encoder and two fully connected layers for the decoder with a bottleneck of size 1. The bottleneck of size 1 is intended to act as a regressor, trying to predict the value of the dockscore. The decoder then tries to take this singular value and tries to reconstruct the input. Since PCA and t-SNE cannot create embeddings and components based off of one value, the autoencoder uses the embedding after the first fully connected layer.

The second autoencoder is also a vanilla autoencoder with two fully connected layers for the encoder and two fully connected layers for the decoder. The difference being the bottleneck size is 128 and the output of the encoder is fed to an auxiliary network which works to predict the docking score. The auxiliary network is similar to the GSAE by Castro et al. [3] which uses the auxiliary network to predict the meta properties of the input. The losses of the auxiliary network as well as the decoder trying to reconstruct the input is added to train the network. Because the size of the bottleneck is 128, we are able to use this to get the embedding for PCA and t-SNE.

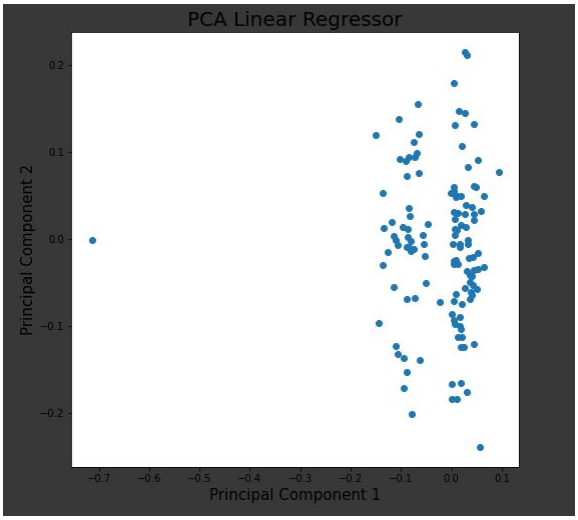
GAN

Finally, we use a cGAN, similar to the FMGAN by Amodio et al [4]. We want to use a generative network so we would be able to ideally be able to input a docking score and output a generated molecule fingerprint. The cGAN we use has a 4 layer neural network for the generator, and a 4 layer neural network as the discriminator. All 8 layers are fully connected layers. In the case of a regular GAN, noise is fed in and the output is generated data, usually an image. In the case of a cGAN, noise is also fed in, but there is the addition of the label. In a multi-label setting, this would allow the GAN to know to produce a dog or a cat. In our case, we use the docking score as the label. The discriminator is also fed the label, again, so it knows which case it is classifying for. For this project, it gives some point of reference to the discriminator. So even though the generated data looks like a perfectly acceptable fingerprint, it may be unrealistic to have it associated with a docking score of -50, for example.

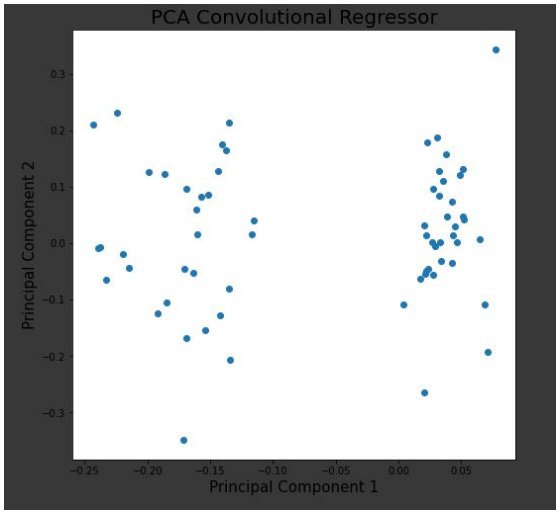
Results

PCA

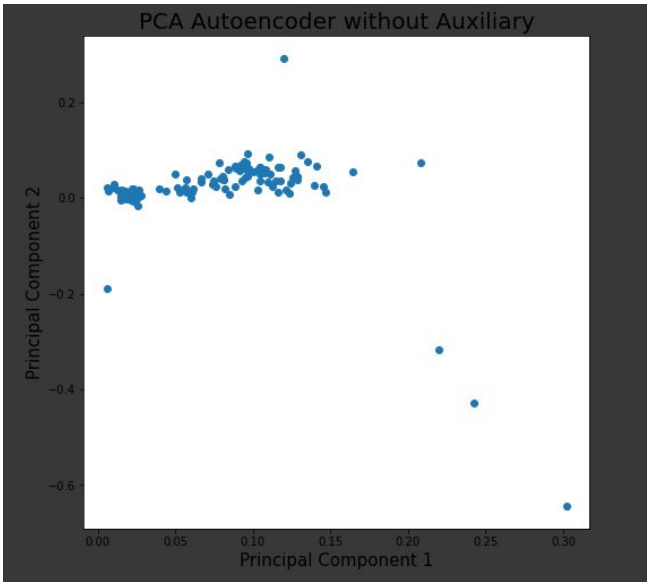
We base our results mainly on the components of PCA and t-SNE and visually inspect to see if there is any significant clustering. First, we find the principal components using PCA with 2 components.



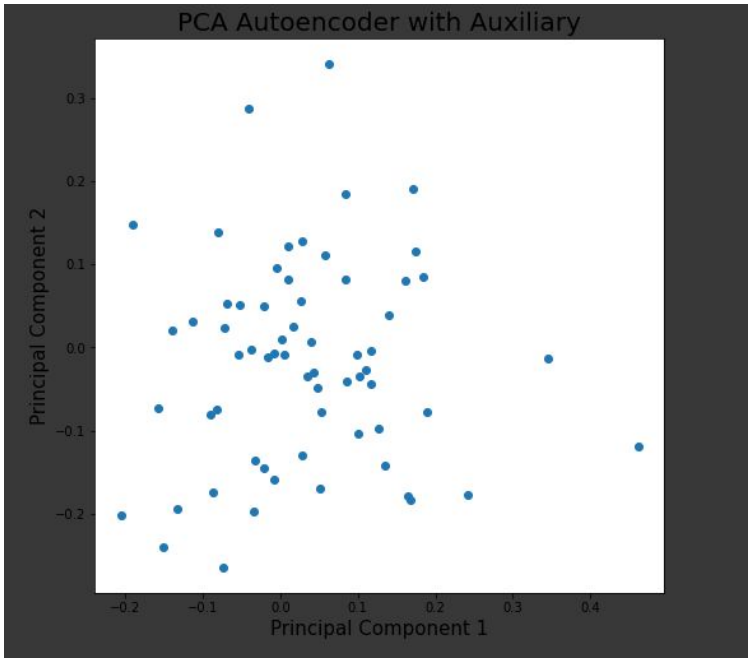
We see here the result of PCA on the baseline network. We see there are two columns of clustering on the right side of the graph, and an outlier off to the left. They are mostly clustered by principal component 1 and do not seem to have any clustering along the second principal component.



Above is the result of PCA on the convolutional network. Again, we see clustering along the first principal component with an outlier in the top right corner of the graph.



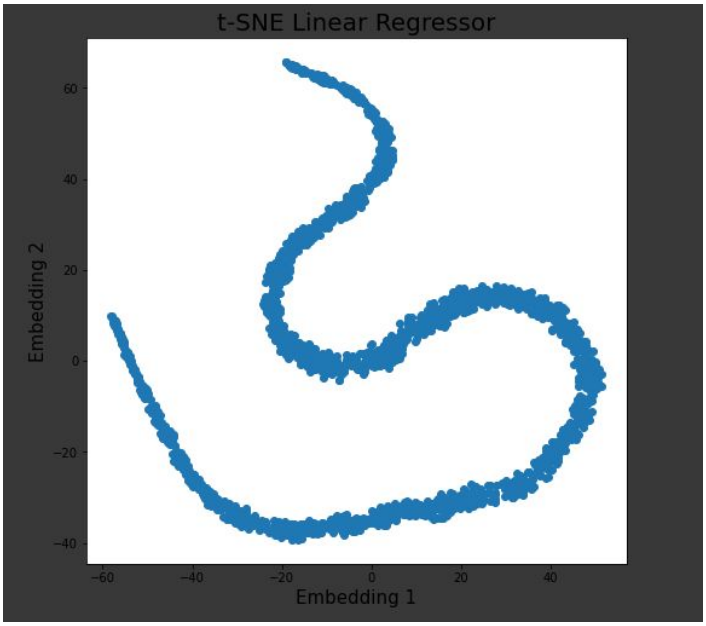
Here is the PCA autoencoder without the auxiliary network. There seems to be a much tighter cluster than either of the regression models. The cluster is along the second component with a relatively small range across the first. This cluster may also be because this was taken after only a single linear layer as opposed to two.



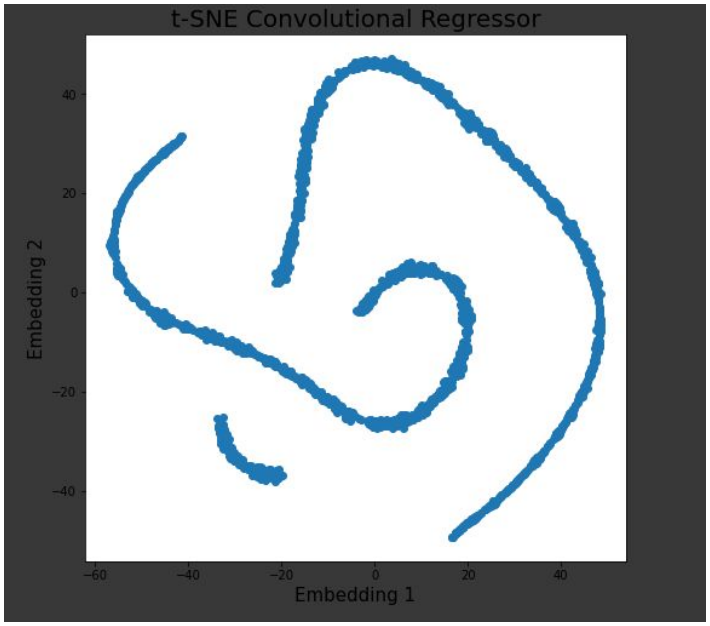
PCA on the autoencoder with the auxiliary network had almost no clustering. There are a few outliers, but we mostly see a very loose cluster around the center of the graph, but not much else.

t-SNE

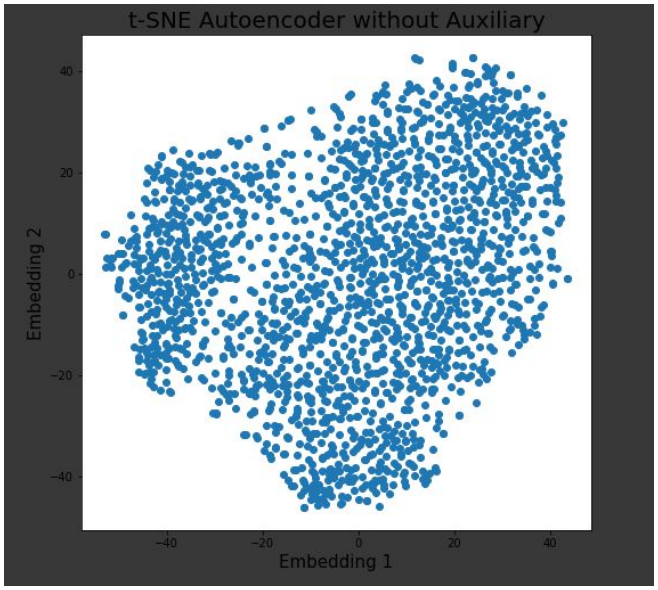
We also run t-SNE on the embeddings, again with 2 components



In the case of the baseline model, there is a definite “snake” shape. This is something a neural network would be able to capture, but it is undetermined if the docking scores have any relation to the snake.

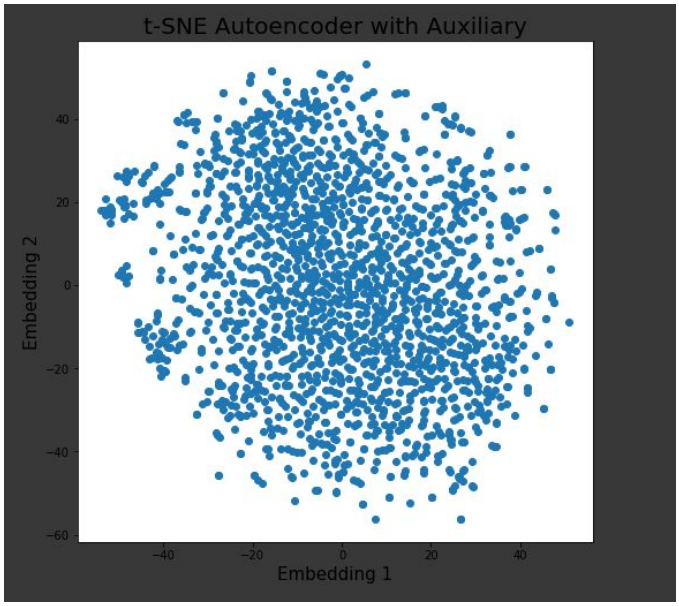


For the convolutional regressor, we have three connected clusters. The most interesting is the smallest cluster, this model may have learned some very particular way of embedding the results of the regression. Again, having the labels of the embeddings would be most helpful in determining if this is the model’s doing or the doing of t-SNE.



In the case of the autoencoder without the auxiliary network, there doesn’t seem to be any particular shape. There is simply a large cluster in the middle. There is, however, more sparse

data points to the left of the cluster, but that may also have to do with the data distribution. Again, it is worth noting that this was the embedding after only one layer.



Similar to the autoencoder without the auxiliary network, the autoencoder with the auxiliary network seems to be pretty shapeless. There are a few small clusters around the main cluster in the middle, but for the most part all data points are a part of the same main cluster in the middle.

Conclusion

In this project, we trained models and used their embeddings to create PCA and t-SNE visualizations on the fingerprints of molecules. The goal was mostly exploratory, to see if we can find interesting information by clustering the embeddings. We found that there is clustering, particularly with the regressor approaches; the baseline 4 layer neural network and the neural network that uses convolutional layers. Using the autoencoders, there was some clustering in the first layer of a vanilla autoencoder using PCA, but t-SNE didn't have the same effect. For an autoencoder with an auxiliary network, there was very little clustering using both approaches, PCA and t-SNE. Future work should include labeling the clusters to see if the positions in the clusters correlate to the dockscore.

Code

https://github.com/idjoannachen/Unsupervised-Learning/blob/master/Final%20Project/Chen%20Joanna%2C%20Final%20Project.ipynb

Reference

[1] Binding Affinity Prediction by Pairwise Function Based on Neural Network. J Chem Inf Model. 2020 Jun 22;60(6):2766-2772. doi: 10.1021/acs.jcim.0c00026.

[2] Öztürk, Hakime, Arzucan Özgür, and Elif Ozkirimli. "DeepDTA: deep drug–target binding affinity prediction." Bioinformatics34.17 (2018): i821-i829.

[3] Castro, Egbert, et al. "Uncovering the Folding Landscape of RNA Secondary Structure with Deep Graph Embeddings." arXiv preprint arXiv:2006.06885 (2020).

[4] Amodio, Matthew, et al. "Generating hard-to-obtain information from easy-to-obtain information: applications in drug discovery and clinical inference." bioRxiv (2020).