Chemical Property Prediction with Stochastic Neural Networks

Soren Dunn
sorend2@illinois.edu

Abstract

Kernel-expanded stochastic neural networks (K-StoNets) have several useful theoretical properties including (under certain assumptions) not having any local minima and having universal approximation ability even for small networks. However, little work has yet been done on evaluating the empirical importance of each of the model components of K-StoNets in the context of a real-world prediction problem. In this project, I investigate the importance of the support vector regression (SVR) layer, hidden layer(s), added noise, and SVR hyper-parameter choice in StoNets on model convergence and performance in the context of predicting activity of chemicals in inhibiting HIV replication using extended connectivity fingerprints.

1 Introduction

Deep neural networks in recent years have been increasingly used in tasks ranging from image classification Krizhevsky et al. [2012] to bioinformatics Sapoval et al. [2022] to robotics Dong et al. [2021] due to their flexibility and potential for high performance given enough compute. However, state-of-the-art neural networks often require a highly complex model structure with millions or even billions of parameters. Additionally, successfully training deep neural networks can be finicky as they are prone to becoming stuck in local minima.

To work towards solving these problems Sun and Liang [2022] proposed a novel model architecture termed kernel-expanded stochastic neural network (K-StoNet). This architecture was inspired by previous work by Gori et al. [1992] and Nguyen and Hein [2017] who showed that given particular regularity conditions, a pyramidal network structure, and linearly independent training data every critical point on a feed-forward neural network will be a global minima. Specifically they showed that for a feed forward neural network with h hidden layers and the following network construction:

1. **Input vector:** $Z_0 = X \in \mathbb{R}^{m_0}$ represents the input to the neural network, where m_0 is the input dimension.
2. **Hidden and output layers:** For i from 1 to $h+1$, the output of layer i, $Z_i \in \mathbb{R}^{m_i}$, is computed as
$$Z_i = \Psi(w_i Z_{i-1} + b_i),$$
where $w_i \in \mathbb{R}^{m_i \times m_{i-1}}$ are the weights, $b_i \in \mathbb{R}^{m_i}$ the biases, and
$$\Psi(s) = (\psi(s_1), \ldots, \psi(s_{m_i}))^T$$
applies the activation function component-wise.
3. **Activation function:** $\psi(\cdot)$ is real analytic, strictly monotonically increasing, and satisfies either a bounded condition or a growth condition given by certain positive constants ρ_1, ρ_2, ρ_3, and ρ_4.

Preprint. Under review.

4. **Loss function:** $U(\theta)$ represents the loss function of the neural network. It is given by the negative log-likelihood or log-probability of the target output Y averaged over the n training samples (index i references the training sample):

$$U(\theta) = -\frac{1}{n}\sum_{i=1}^{n} \log \pi(Y^{(i)}|\theta, X^{(i)}) \equiv \frac{1}{n}\sum_{i=1}^{n} l(Z_{h+1}^{(i)}),$$

where $\pi(\cdot)$ denotes the density/mass function, n is the sample size, and $l(\cdot)$ is a continuously differentiable function.

5. **Regularity Conditions:** Training samples are linearly independent, the activation function ψ is real analytic, strictly increasing, and satisfies the bounded or growth conditions, and the loss function l is continuously differentiable and if its derivative evaluated at 'a' is zero, then 'a' is a global optimum.

Under the above conditions, every critical point of the loss function $U(\theta)$ is a global minimum.

Such a result is possible because distinct, linearly independent training samples ensure that the input information is sufficiently diverse, full row rank of weight matrices implies that the layer transformations are capable of capturing the necessary complexity, and the activation function being real analytic and strictly increasing ensures well-behaved gradients work together to ensure that there are no stationary points or local minima. The problem with this result is that it requires the data to be linearly independent which inherently requires the sample size not be larger than the number of features. For most deep learning problems this is widely unrealistic as a datasets of thousands or millions of training samples is required to achieve state-of-the-art performance. The K-StoNet archictecture was proposed to remove this constraint.

The way K-StoNets attempted to circumvent the required linear independence of the data was by adding a radial basis function (RBF) support vector regression (SVR) on the first layer of the network to map the input data into an infinite dimensional space. The intuition behind why this can help satisfy the linear independence constraint is if the data has an arbitrary number of dimensions then the number of samples will never be larger than the feature dimension, ensuring the data's linear independence. The technical explanation is that in the infinite dimensional feature space the Gram matrix is of full rank which means the samples are linearly independent Sun and Liang [2022].

To see how the RBF kernel can be considered to map the input data into an infinite number of dimensions consider the RBF equation:

$$K(x, y) = \exp(-\gamma\|x - y\|^2). \tag{1}$$

Here, the parameter $\gamma > 0$ controls the width of the Gaussian function, dictating how quickly the similarity measure decreases with distance. The norm $\|x - y\|^2$ represents the squared Euclidean distance between the two feature vectors x and y. This can be expanded using the dot product:

$$\|x - y\|^2 = (x - y) \cdot (x - y) = \|x\|^2 - 2x \cdot y + \|y\|^2. \tag{2}$$

Thus, substituting this back into the definition of the RBF kernel, we get:

$$K(x, y) = \exp(-\gamma\|x\|^2 + 2\gamma x \cdot y - \gamma\|y\|^2). \tag{3}$$

This reformulates the RBF kernel in terms of the squared norms of x and y and their dot product.

The exponential function can be expanded using the Taylor series:

$$\exp(z) = \sum_{n=0}^{\infty} \frac{z^n}{n!}, \tag{4}$$

which applies to any real or complex number z. Taking $z = 2\gamma x \cdot y$, the Taylor series for the central term of our RBF kernel becomes:

$$\exp(2\gamma x \cdot y) = \sum_{n=0}^{\infty} \frac{(2\gamma x \cdot y)^n}{n!}. \tag{5}$$

We can now substitute this expansion into our RBF kernel formulation, separating the part that depends on x and y alone:

$$K(x, y) = \exp(-\gamma \|x\|^2) \cdot \left(\sum_{n=0}^{\infty} \frac{(2\gamma x \cdot y)^n}{n!} \right) \cdot \exp(-\gamma \|y\|^2). \tag{6}$$

This final expression shows that the RBF kernel can be understood as an infinite series expansion of the weighted dot products of x and y. Each term of this series represents an interaction of a different degree between x and y. This derivation is one way to see that the RBF kernel maps the data into an infinite dimensional embedding and thus ensures unique samples are linearly independent.

However, applying this transformation means that the model can no longer be trained using a gradient-based algorithm since the transformed input features aren't explicitly calculated. One approach to circumventing this difficulty would be to use the kernel representer theorem to try to train the model based on the kernel of corresponding to the transformed data:

$$\tilde{Y}_1 = b_1 + \sum_{i=1}^{n} w_1^{(i)} K(\mathbf{X}^{(i)}, \mathbf{X}) \tag{7}$$

where $w_1^{(i)} \in \mathbb{R}^{m_1}$ and $K(\mathbf{X}^{(i)}, \mathbf{X}) = \phi(\mathbf{X}^{(i)})^T \phi(\mathbf{X})$ is explicitly available. The issue with this model specification though is that for a nonlinear kernel regression

$$Y = g(\widetilde{Y}_1) + e = g(b_1 + \beta\phi(X)) + e,$$

where $g(\cdot)$ represents a nonlinear mapping from $\widetilde{Y}_1$ to the output layer, the kernel representer theorem may not hold. This is due to the fact that depending on the chosen activation function (which always involves a nonlinear mapping), the solution cannot be easily expressed as a linear combination of kernel functions based on training data. Instead, the mapping $g(\cdot)$ could potentially create a much more complex decision boundary that does not conform to the linear combination of kernel evaluations at each data point.

1.1 Model Specification

To avoid this issue Sun and Liang [2022] propose the K-StoNet model:

$$Y_1 = b_1 + \beta\phi(X) + e_1 \tag{8}$$

$$Y_i = b_i + w_i \Psi(\tilde{Y}_{i-1}) + e_i \; for \; i = 2, 3, ..., h \tag{9}$$

$$Y = \Psi'(b_{h+1} + w_{h+1}\Psi(Y_h) + e_{h+1}) \tag{10}$$

The variables in the K-StoNet model given above can be split into 3 types:

1. **Data and Predictions**: X is the input data, Y is the model predictions.
2. **Hyperparameters**: h: Number of hidden layers, ψ: Radial basis function (RBF) kernel, $e_1, \ldots, e_{h+1}$: Random noise, Ψ: Activation function, Ψ': Logit function, ε_i are random noise added with distribution $N(0, \sigma_i^2 I_{m_i})$ where I_{m_i} is an identity matrix with dimensions equal to the number of units on layer i and σ_i is the chosen noise level for layer i, g is a link function to convert the output to a format suitable for classification.
3. **Learned Parameters**: $b_1, \ldots, b_{h+1}$ are learned parameters (the biases), β is the learned weight for the first layer, $w_1, \ldots, w_{h+1}$ are learned parameters (the weights), $Y_1, \ldots, Y_h$ are latent (unobserved) variables defined as shown in the equations above.

The model is thus composed of an RBF kernel SVR, a lasso for each of the hidden layers (a penalty is added to ensure convexity) and a generalized linear model for the output layer (in order to allow for classification).

1.2 K-StoNet as an Approximation of KNN

To establish that K-StoNet is a valid approximator to KNN, the authors first redefine certain parameters to reflect their dependency on the training sample size n. Specifically:

- C is redefined as C_n
- ε is redefined as ε_n
- σ_i is redefined as $\sigma_{n,i}$ for $i = 2, 3, \ldots, h+1$

Next, a series of assumptions are made to establish the validity of the K-StoNet including:

(i) Θ is compact. This means the set of all parameter vectors θ is bounded and closed.

(ii) The expectation of the logarithm squared of the conditional probability is finite, ensuring reasonable behavior of the loss.

(iii) The activation function $\psi(\cdot)$ is Lipschitz continuous, indicating a bounded rate of change.

(iv) The network can grow with the size of training data.

(v) A constraint on the noise σ_n that ensures the noise does not overwhelm the signal in the network's predictions.

They use these assumptions to show that K-StoNet and KNN will have matching loss functions in the limit as the sample size grows without bound.

They next make the following assumptions:

- $Q^*(\theta)$ is continuous in θ and has a unique maximum.
- There's a positive gap between the maximum value of $Q^*(\theta)$ and any values outside a certain bound.

to ensure that the objective function has a well-defined maximum and is not too 'flat' or discrete.

They last prove that the estimated parameter vector $\hat{\theta}_n$ will converge in probability to the true parameter θ^*, meaning that the K-StoNet consistently approximates the true model as the number of samples increases.

The authors claim that since K-StoNet is a consistent and valid approximation to KNN for large training sample sizes and both will have matching loss functions asymptotically, it is adequate to train K-StoNet over KNN.

1.3 Training Algorithm

We propose an approach for training the K-StoNet utilizing the Imputation-Regularized Optimization (IRO) algorithm introduced by Liang et al. in 2018, addressing the challenge of missing data in the context of statistical learning. Let us define Z_{obs} as the observed data, Z_{mis} as the missing data, and θ as the model parameter. The primary objective of the IRO algorithm is to obtain a consistent estimate of θ by maximizing the expected log-likelihood $E \log \pi(Z_{obs}, Z_{mis}|\theta)$, with the expectation taken over the joint distribution of (Z_{obs}, Z_{mis}).

The IRO algorithm shares similarities with the Expectation-Maximization (EM) algorithm and the Stochastic EM algorithm, which focus on maximizing the marginal likelihood $\pi(Z_{obs}|\theta)$. However, IRO differentiates itself by incorporating a regularization term in the optimization step to ensure convergence, especially in high-dimensional settings.

For the K-StoNet, the goal is to estimate θ by maximizing $E \log \pi(Y, Y_{mis}|X, \theta)$, suggesting an alignment with the K-Nearest Neighbors (KNN) training paradigm if one employs the Stochastic Gradient Descent (SGD) algorithm. The IRO algorithm entails the following iterative steps starting from an initial guess $\hat{\theta}_n^{(0)}$:

I-step For each sample $(X^{(i)}, Y^{(i)})$, impute $Y_{mis}^{(i)}$ from the predictive distribution $g(Y_{mis}|Y^{(i)}, X^{(i)}, \hat{\theta}^{(t)})$.

RO-step Update the estimate $\hat{\theta}_n^{(t+1)}$ by minimizing a penalized loss function

$$\hat{\theta}_n^{(t+1)} = \arg\min_{\theta} \left(-\frac{1}{n} \sum_{i=1}^{n} \log \pi(Y^{(i)}, Y_{mis}^{(i)}|X^{(i)}, \theta) + P_{\lambda_n}(\theta) \right),$$

where $P_{\lambda_n}(\theta)$ is a chosen penalty function to form a consistent estimate of θ.

Pseudo-code for K-StoNet Training

```
Algorithm 1: Training K-StoNet
1: Initialize parameters ...
2: Repeat until convergence {
3:     Perform I-step: Impute missing data
4:     Perform RO-step: Update parameters by minimizing penalized loss function
5:     Check convergence criteria
}
6: Output final model parameters
```

Despite the potential higher cost for iteratively solving convex optimization problems, the fast convergence of the IRO algorithm typically within a few tens of iterations may offset this. If certain conditions are met regarding the sample size, the penalty term in the objective function may be omitted for computational efficiency while maintaining the asymptotic normality of the estimates.

2 Dataset

We chose to test use of these K-StoNets on a real-world dataset. We utilized a dataset originally compiled by the Drug Therapeutics Program's (DTP) AIDS Antiviral Screen, which assessed the capacity of over 40,000 chemical compounds to inhibit the replication of the Human Immunodeficiency Virus (HIV). We accessed this dataset through the MoleculeNet benchmark which provides access to a variety of datasets for molecular analysis. The compounds were tested and their effectiveness classified into one of three distinct categories, namely confirmed inactive (CI), confirmed active (CA), and confirmed moderately active (CM). For the purpose of our analysis, we merged the active (CA) and moderately active (CM) categories to simplify the task into a binary classification challenge, differentiating between inactive (CI) and active (CA/CM).

The raw dataset is provided in a CSV file format, with the following columnar information:

- "SMILES": This column contains the Simplified Molecular Input Line Entry System (SMILES) representation of each compound, describing its molecular structure in a linear text format.

- "HIV Active": A binary classification is also provided, with a value of 1 indicating active compounds (both CA and CM) and a value of 0 denoting inactive compounds (CI).

The SMILES strings are converted to Extended-Connectivity Fingerprints (ECFP) using RDKit (an open-source platform designed to support the chemical information processing) which is also provided through MoleculeNet. ECFP serve as a fundamental tool in chemical informatics for encoding molecular information into a format suited for computational processes. Creating ECFP fingerprints works roughly as follows

1. **Molecule Decomposition:** A molecule is dissected into a series of submolecular fragments according to the connectivity of its atoms.
2. **Unique Atom Identification:** Every atom in the molecule is labeled with a distinctive identifier that reflects its type, chemical properties, and bonding context.
3. **Hashing:** The identifiers are hashed into a fixed-length binary array, or *fingerprint*, to facilitate efficient data handling and comparison.
4. **Topology Capturing:** The algorithm's design accommodates the topological characteristics of the molecule, effectively distinguishing various structural features.

Scaffold splitting is used to ensure that molecular scaffolds are not shared between the training and test sets, thus providing a more stringent evaluation of the predictive model's generalizability. The dataset was split into 32,901 training samples and 4,113 testing samples. The dataset is very imbalanced with only around 1,300 total molecules having demonstrated ability to inhibit HIV replication while the remaining molecules do not exhibit this effect. We addressed the large imbalance in the dataset in the training by empoying various under-sampling and over-sampling procedures to attempt to balance the dataset. In terms of model evaluation, the imbalanced dataset was dealt with by considering the area under the receiving operator characteristic (AUC-ROC) as the evaluation metric.

2.1 Data Pre-Processing

2.1.1 Sampling

Both undersampling and oversampling procedures were applied to the dataset. For undersampling, a random selection of samples which were inactive were discarded until there was an even number of active and inactive compounds.

For oversampling the SMOTE (Synthetic Minority Over-sampling Technique) method was applied. SMOTE is an approach used to generate synthetic samples in a dataset for the minority class.

The algorithm has the following specification:

$$\mathbf{s} = \mathbf{x} + (\mathbf{n} - \mathbf{x}) \times \delta$$

where δ is a random number between 0 and 1. This operation will yield a point along the line segment between $\mathbf{x}$ and $\mathbf{n}$ but not necessarily at one of the endpoints, thus creating diversity in the sample.

By repeating this process, the SMOTE algorithm generates new, synthetic examples of the minority class, allowing models to have a more balanced view of the problem and potentially improving classification performance on imbalanced datasets.

After oversampling, the dataset comprised of over 60,000 compounds. Training a 1 hidden layer K-StoNet with 5 hidden units took over 10 minutes to train for a single epoch and a 20-unit K-StoNet took over an hour to train for a single epoch. Additionally the performance of these K-StoNets were lower than the same models trained on the undersampled dataset over only a tenth of the time they took to train a single epoch. Thus due to a combination of computational constraints and low performance no further samples were evaluated on the oversampled dataset.

2.1.2 Principle Component Analysis

Due to the high number of features in the ECFP, we also applied principle component analysis (PCA) to the features and then selected a subset of the principle components to use for prediction to lower the number of features in the dataset. This allowed for further testing of specific model configurations and training of larger models over more epochs due to the reduced size of the dataset.

The percent of variance explained by the principle components increased logarithmically with the number of components, resulting in there being no obvious cutoff for selection of principle components to use. Thus the top 100 principle components were chosen as over the top 100 components the increase in percent variance as the number of component increase is still slightly greater than linear and selecting the first 100 components decreases the size of the dataset by a factor of 10, which allows for a dramatic increase in computational speed of models trained on this dataset. The composition of the principle components was determined based on the testing set and the same transformation was applied to both the training and testing set. Two versions of the PCA subset dataset were created: one with the undersampled dataset and the other with the full dataset.

To test whether subsetting the dataset to the principle components effected trained model performance, we trained a simple 100 estimator gradient boosting classifier on the full dataset and the feature-selected version of the dataset only containing the first 100 principle performance. The model trained on the full dataset achieved 0.71 test AUC-ROC and the model trained on the data subset achieved 0.70 test AUC-ROC. Additionally the model trained on the PCA subset actually achieved higher test set accuracy (0.942) than the model trained on the full dataset (0.94). These tests indictate that the subsetted version of the dataset is able to train models to similar levels of performance while

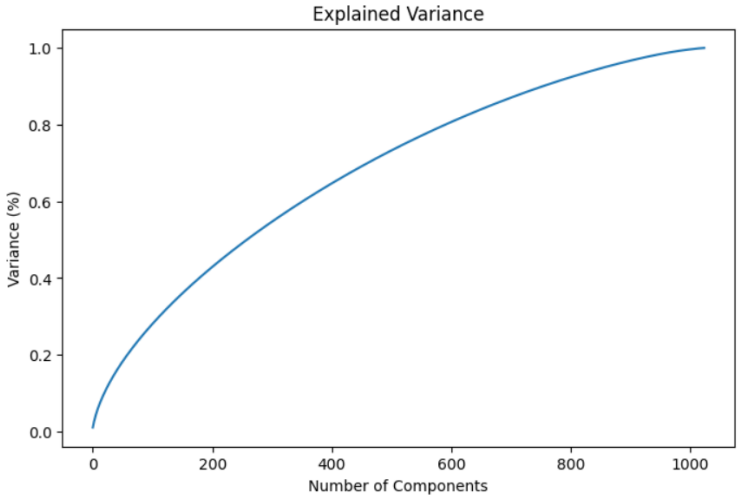

Figure 1: PCA Cumulative variance across all principle components

requiring much less computational time to run. For this reason, the model inference training runs are all performed on a version of the dataset which is undersampled and is selected for only the first 100 principle components.

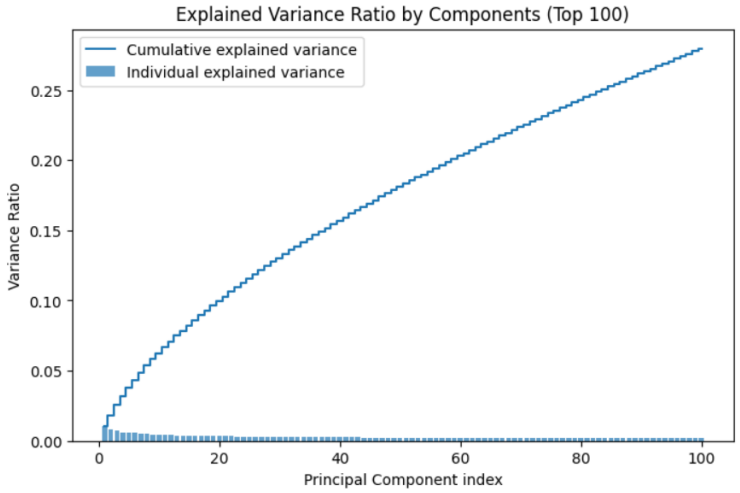

Figure 2: Plot of first 100 principle components which were selected for the feature subset dataset. All together the first 100 principle components make up around 25% of the total variance

3 Results

Four sets of analysis were conducted. The most basic of these was a basic performance comparison between the best-performing K-StoNet model specification and hyper-parameter tuned shallow machine learning baselines. The second was comparing the performance of simply the first SVR layer of the K-StoNet against the SVR combined with the hidden layer. The purpose of this comparison was to test whether the hidden layer was in fact contributing to the performance of the model over the base SVR. The third analysis was comparing the K-StoNet performance to a modified version of the model without the SVR layer. This allowed for much faster training and helped test whether the theoretical properties given by the SVR layer (namely the lack of local minima) showed up in the training of the networks without the SVR layer. Finally, the importance of the noise in the model

specification was tested by training successive versions of the K-StoNets with ever decreasing levels of noise.

3.1 Performance Compared to Baselines

The AUC-ROC between the best performing of our K-StoNets, the best performing of our feedforward neural networks, and previous baselines Wu et al. [2018] are plotted in Figure 3. In terms of raw performance, even the shallow K-StoNet trained here (with only a single hidden layer) is competitive with these baselines.

In these baselines, the feed-forward neural network, K-StoNet, and KernelSVM all display large amounts of overfitting (with training values between 0.2 and 0.25 higher than testing values). Disturbingly the K-StoNet seems just as prone to overfitting as the feedforward neural network (with training AUC-ROC 0.23 and 0.25 points larger than testing values respectively). This suggests that the K-StoNet may be converging to a minimum different from the underlying data-generating process which means it might be converging to a local minima instead of the global minima. If this is the case it is possible it is because the sample size is not large enough (since the proof that the K-StoNet is equivalent to the KNN only holds in the limit as n goes to infinity) or one of the other more technical conditions in the theoretical derivation may not be valid.

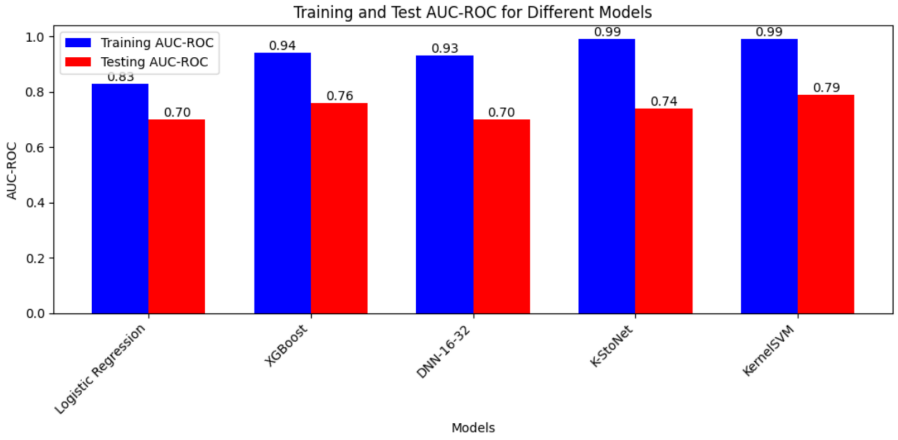

Figure 3: AUC-ROC comparison between K-StoNets and various machine learning baselines. DNN 16-32 refers to a feed forward neural network with two hidden layers of sizes 16 and 32. The K-StoNet is ours and the rest are previous baseline results from MoleculeNet

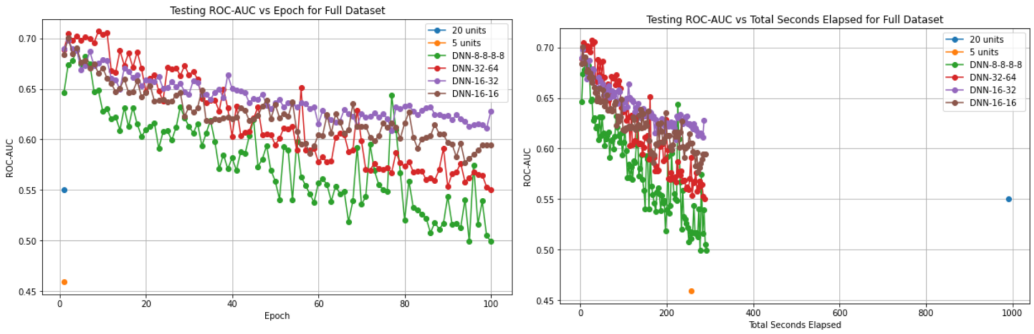

Figure 4: Testing AUC-ROC over the course of training for the full dataset. DNN refers to fully connected feed-forward networks with the specified number of hidden units. Lines which are not labeled as DNN refer to K-StoNets. Left shows results versus epochs while the right shows results over time

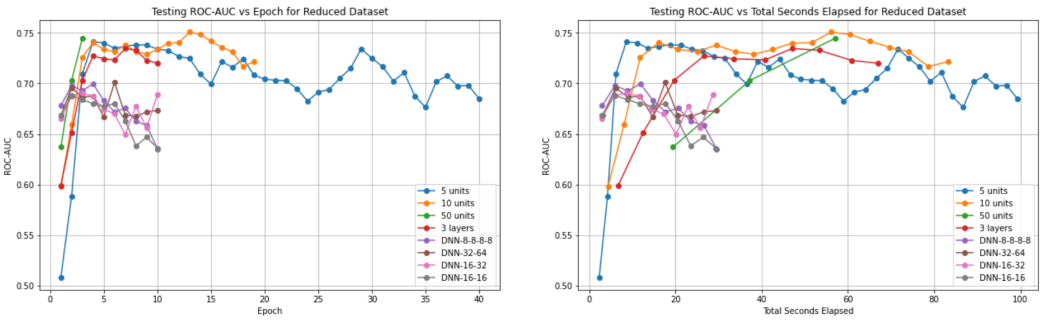

Figure 5: Testing AUC-ROC over the course of training for the undersampled dataset. DNN refers to fully connected feed-forward networks with the specified number of hidden units. Lines which are not labeled as DNN refer to K-StoNets. Left shows results versus epochs while the right shows results over time

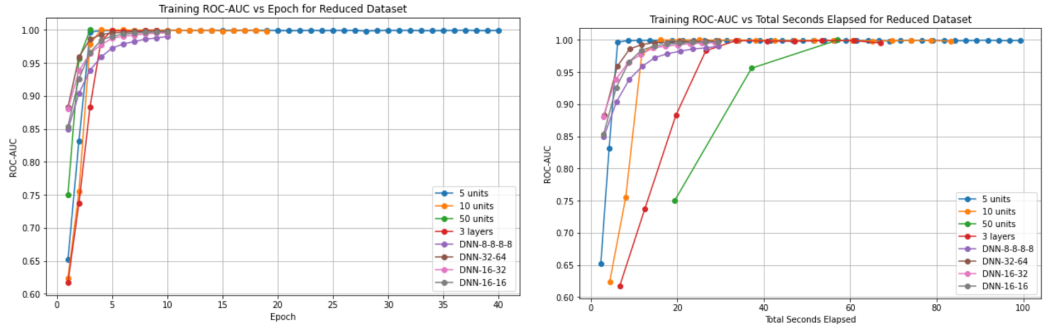

Figure 6: Training AUC-ROC over the course of training for the undersampled dataset. DNN refers to fully connected feed-forward networks with the specified number of hidden units. Lines which are not labeled as DNN refer to K-StoNets. Left shows results versus epochs while the right shows results over time

The lack of convergence to global minima is also supported by looking at the graphs of training and testing AUC-ROC over the course of training. The variation in the performance of the K-StoNets over the course of training appears very similar to the conventional feed forward networks. Additionally, the training performance of these models quickly increases to nearly one in less than ten epochs just as the regular feed forward networks do. However, as can be seen in the right side of Figure 6, this actually represents a longer total time to overfit the training data than the conventional feed forward networks. However overall based on this analysis the K-StoNets seem to have similar performance and convergence properties and feed forward neural networks trained over similar amounts of time. However these comparisons do not directly compare K-StoNet architecture to equivalent conventional neural network architecture, merely compare training K-StoNets and conventional neural networks over similar amounts of time/epochs. We will do a more direct comparison of K-StoNets and neural networks with similar architectures in Section 3.4.

3.2 SVR Parameter Search

The next analysis conducted was to identify how important the specification of the initial SVR was on the performance of the K-StoNet. Since the SVR is the most complicated component of the K-StoNet (and creates a whole new embedding of the data to feed into the next layer of the network) it is intuitive that the exact specification of the SVR layer might greatly effect the model's results. However, as shown in Figure 7 the range of the maximum testing accuracy of the trained SVR models over 3-4 orders of magnitude for each of their parameters only had around a 2% change in test set performance. This suggests that the exact specification of the SVR model likely does not have too dramatic of an effect on overall model performance. This analysis was performed by reducing the model to simply the SVR layer and conducting parameter search for the C and epsilon values of

the SVR. The best performing hyperparameter combinations were for low values of C regardless of epsilon.

C controls the strength of the L2 regularization where higher values of C correspond to lower regularization. Epsilon specifies the region where no penalty is incurred in the training loss function by points predicted within epsilon of their actual value. Thus the SVR performing best with small values of C indicates that regularization of the SVR is increasing its performance but the exact specification of the no-penalty region is not particularly significant for model performance.

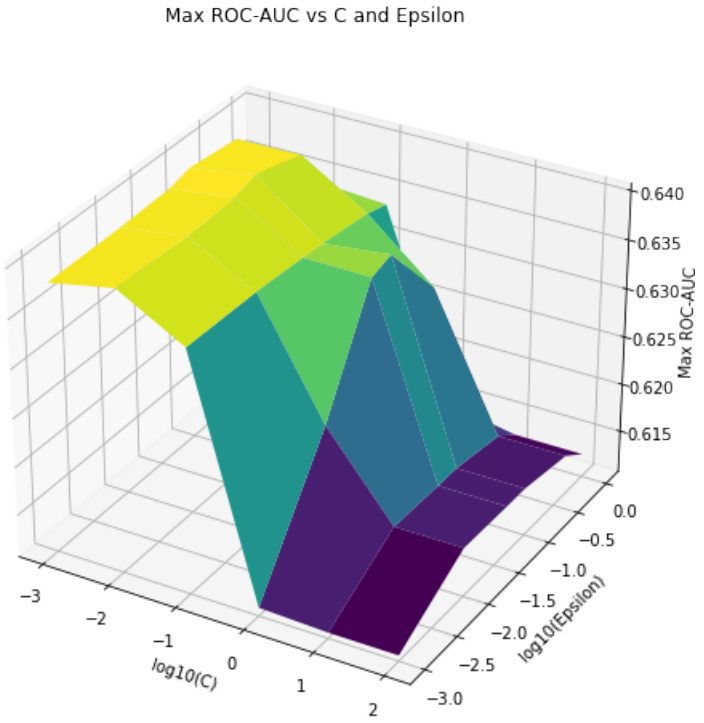

Figure 7: Maximum AUC-ROC from SVR models trained over 100 epochs plotted against their penalization coefficients trained on the PCA subsetted and undersampled dataset

3.3 K-StoNet Compared to SVR

The third analysis conducted was a test of whether the additional model layers beyond the SVM significantly increased performance. This was especially of interest after seeing that a pure SVM baseline approach outperformed the best-performing K-StoNet on the testing data.

3.4 K-StoNet Compared to Equivalent Neural Network Configuration

3.5 K-StoNet Noise Specification

One-layer five hidden unit K-StoNets were trained for ten epochs on the PCA subsetted and undersampled version of the dataset.

The parameter that is being varied here is the σ_i which controls the random noise added to each layer (which has distribution $N(0, \sigma_i^2 I_{m_i})$). Thus larger sigma values imply greater noise added at each layer. Predictably, the training AUC-ROC is low when sigma is large (greater than 1). This is to be expected because in these cases the added noise is preventing the model from learning the underlying distribution. Sigma values around 0.001 to 0.0001 achieve training performance comparable with a feed-forward neural network with the same number of hidden layers as the K-StoNet.

The trends in the testing performance mirror that of the training performance. Very large values of sigma result in low testing performance while lower levels of noise are able to yield performance

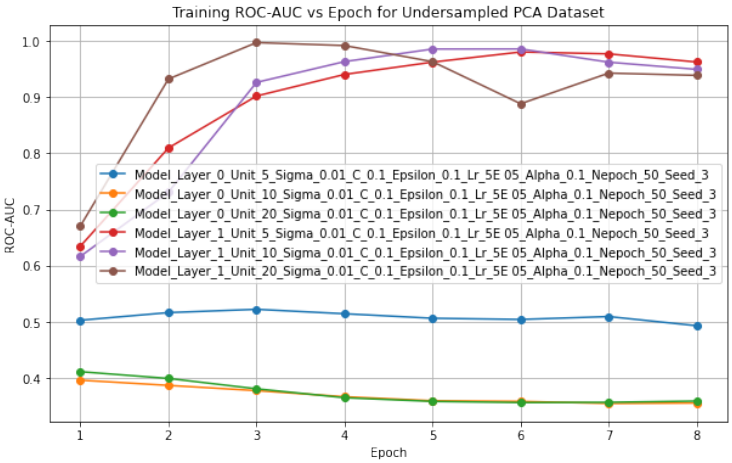

Figure 8: t

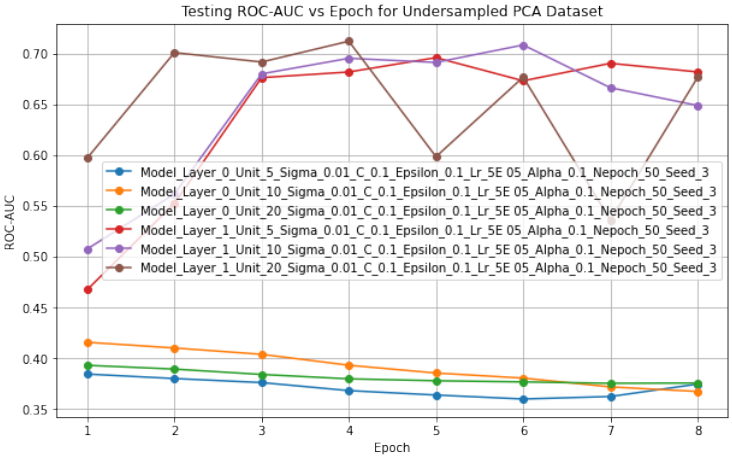

Figure 9: t

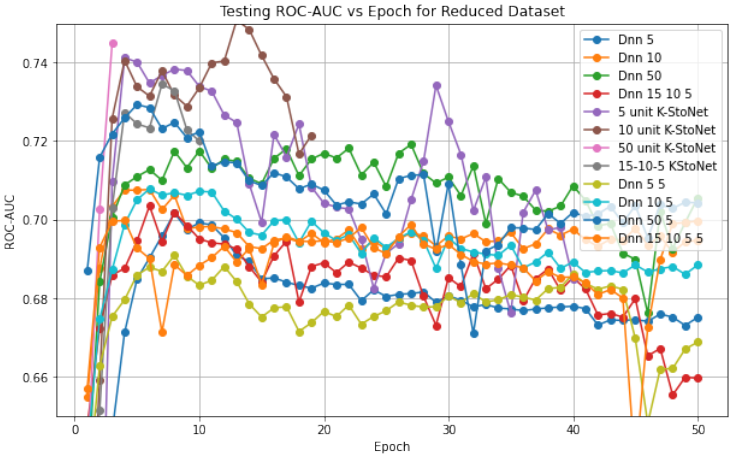

Figure 10: t

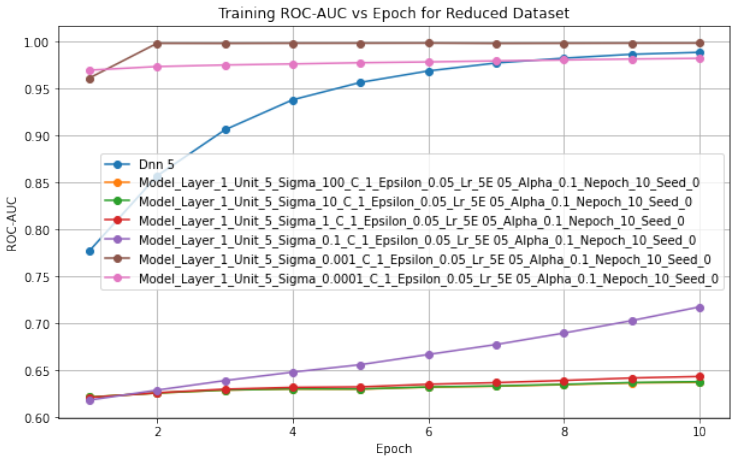

Figure 11: Training AUC-ROC for one-layer five hidden unit K-StoNets with varying sigma (is the primary parameter controlling the noise at each layer)

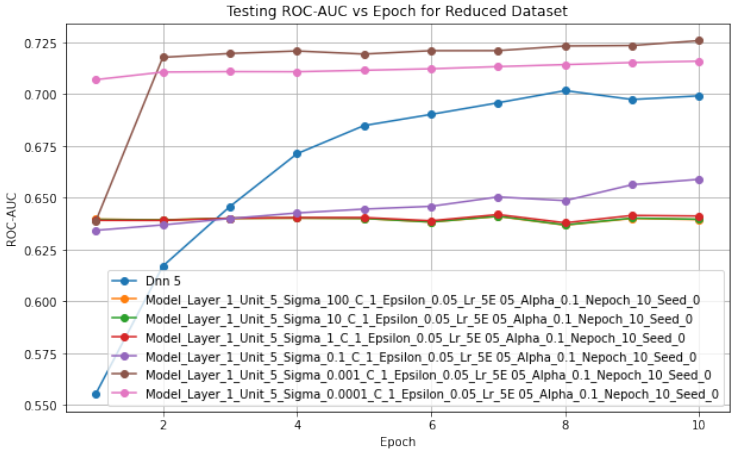

Figure 12: Testing AUC-ROC for one-layer five hidden unit K-StoNets with varying sigma (is the primary parameter controlling the noise at each layer)

greater than the regular feedforward network with the same number of hidden units. Importantly, the lowest sigma model does not display the greatest testing set performance, suggesting that for extremely low sigma values the model performance tends to decrease again (this is likely due to problems with convergence since the noise was originally added in order to allow for a valid training algorithm).

These trends demonstrate that the added noise is a necessary component of the K-StoNet but should be kept well below one to insure the model is still able to learn the underlying structure of the data.

4 Conclusion

References

Shi Dong, Ping Wang, and Khushnood Abbas. A survey on deep learning and its applications. *Computer Science Review*, 40:100379, 2021.

Marco Gori, Alberto Tesi, et al. On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(1):76–86, 1992.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

Quynh Nguyen and Matthias Hein. The loss surface of deep and wide neural networks. In *International conference on machine learning*, pages 2603–2612. PMLR, 2017.

Nicolae Sapoval, Amirali Aghazadeh, Michael G Nute, Dinler A Antunes, Advait Balaji, Richard Baraniuk, CJ Barberan, Ruth Dannenfelser, Chen Dun, Mohammadamin Edrisi, et al. Current progress and open challenges for applying deep learning across the biosciences. *Nature Communications*, 13(1):1728, 2022.

Yan Sun and Faming Liang. A kernel-expanded stochastic neural network, 2022.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay Pande. Moleculenet: A benchmark for molecular machine learning, 2018.