	mean $\|\hat{\Sigma} - \Sigma\|_2^2$	mean $\|\hat{\Sigma} - \Sigma\|_\infty$
Empirical	0.0267	0.543
Graph Lasso	0.0223	0.680
DeepGraph	0.0232	0.673

Table 5: Covariance prediction of ABIDE data. Averaged over 50 trials of 35 samples from the ABIDE Control data

A Supplementary Experiments and Analysis

A.1 Predicting Covariance Matrices

Using our framework it is possible to attempt to directly predict an accurate covariance matrix given a noisy one constructed from few observations. This is a more challenging task than predicting the edges. In this section we show preliminay experiments which given an empirical covariance matrix from few observations attempts to predict a more accurate covariance matrix that takes into account underlying sparse data dependency structure.

One challenge is that outputs of our covariance predictor must be on the positive semidefinite cone, thus we choose to instead predict on the cholesky decompositions, which allows us to always produce positive definite covariances. We train a similar structure to DeepGraph-39 structure modifying the last layer to be fully connected linear layer that predicts on the cholesky decomposition of the true covariance matrices generated by our model with a squared loss.

We evaluate this network using the ABIDE dataset described in Section 3. The ABIDE data has a large number of samples allowing us to obtain a large sample estimate of the covariance and compare it to our estimator as well as graphical lasso and empirical covariance estimators. Using the large sample ABIDE empirical covariance matrix. We find that we can obtain competitive ℓ_2 and ℓ_∞ norm using few samples. We use 403 subjects from the ABIDE Control group each with a recording of $150 - 200$ samples to construct covariance matrix, totaling 77 330 samples (some correlated). This acts as our very approximate estimate of the population Σ. We then evaluate covariance estimation on 35 samples using the empirical covariance estimator, graphical lasso, and DeepGraph trained to output covariance matrices. We repeat the experiment for 50 different subsamples of the data. We see in **??** that the prediction approach can obtain competitive results. In terms of ℓ_2 graphical lasso performs better, however our estimate is better than empirical covariance estimation and much faster then graphical lasso. In some applications such as robust estimation a fast estimate of the covariance matrix (automatically embedding sparsity assumptions) can be of great use. For ℓ_∞ error we see the empirical covariance estimation outperforms graphical lasso and DeepGraph for this dataset, while DeepGraph performs better in terms of this metric.

We note these results are preliminary, as the covariance predicting networks were not heavily optimized, moreover the ABIDE dataset is very noisy even when pre-processed and thus even the large sample covariance estimate may not be accurate. We believe this is an interesting alternate application of our paper.

A.2 Additional Synthetic Results on Sparsity

We investigate the affect of sparsity on DeepGraph-39 which has been trained with input that has sparsity $96\% - 92\%$ sparse. We find that DeepGraph performs well at the 2% sparsity level despite not seeing this at training time. At the same time performance begins to degrade for 15% but is still competitive in several categories. The results are shown in Table **??**. Future investigation can consider how alternate variation of sparsity at training time will affect these results.

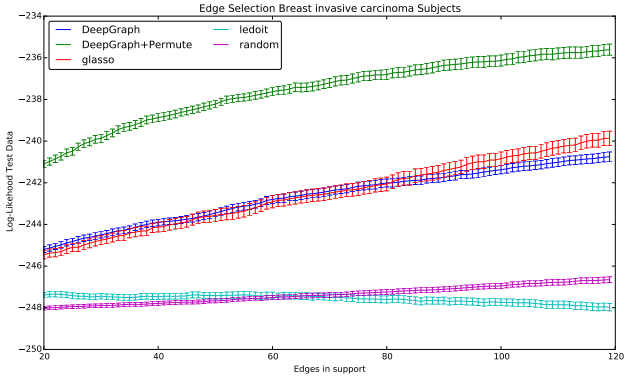

Figure 6: Average test likelihood over 50 trials of applying a network trained for 500 nodes, used on a 175 node problem

A.3 Application of Larger Network on Smaller Input

We perform preliminary investigation of application of a network trained for a larger number of nodes to a smaller set of nodes. Specifically, we consider the breast invasive carcinoma groups gene data. We now take all 175 valid genes from Appendix C.2 of (Honorio et al., 2012). We take the network trained on 500 nodes in the synthetic experiments section. We use the same experimental setup as in the gene experiments. The 175×175 covariance matrix from 40 samples and padded to the appropriate size. We observe that DeepGraph has similar performance to graph lasso while permuting the input and ensembling the result gives substantial improvement.

A.4 Permutation as Ensemble Method

As discussed in Section 2.3, permuting the input and averaging several permutations can produce an improved result empirically. We interpret this as a typical ensembling method. This can be an advantage of the proposed architecture as we are able to easily use standard ensemble techniques. We perform an experiment to further verify that indeed the permutation of the input (and subsequent inverse permutation) allows us to produce separate classifiers that have uncorrelated errors.

We use the setup from the synthetic experiments with DeepGraph-39 in Section 3 with $n = 35$ and $p = 39$. We construct 20 permutation matrices as in the experimental section. Treating each as a separate classifier we compute the correlation coefficient of the errors on 50 synthetic input examples. We find that the average correlation coefficient of the errors of two classifiers is 0.028 ± 0.002, suggesting they are uncorrelated. Finally we note the individual errors are relatively small, as can already be inferred from our extensive experimental results in Section 3. We however compute the average absolute error of all the outputs across each permutation for this set of inputs as 0.03, notably the range of outputs is 0 to 1. Thus since prediction error differ at each permutation but are accurate we can average and yield a lower total prediction error.

Finally we note that our method is extremely efficient computationally thus averaging the results of several permutations is practical even as the graph becomes large.

Experimental Setup	Method	Prec@5%	AUC	CE
Gaussian Random Graphs (n=35,p=39,sparsity=2%)	Glasso	0.464 ± 0.038	0.726 ± 0.021	0.02
	Glasso (optimal)	0.519 ± 0.035	0.754 ± 0.019	0.02
	BDGraph	0.587 ± 0.033	0.811 ± 0.017	0.15
	DeepGraph-39	0.590 ± 0.026	0.810 ± 0.019	0.03
	DeepGraph-39+Perm	0.598 ± 0.026	0.831 ± 0.017	0.03
Gaussian Random Graphs (n=35,p=39,sparsity=15%)	Glasso	0.732 ± 0.046	0.562 ± 0.013	0.32
	Glasso (optimal)	0.847 ± 0.029	0.595 ± 0.011	0.33
	BDGraph	0.861 ± 0.015	0.654 ± 0.013	0.33
	DeepGraph-39	0.678 ± 0.032	0.643 ± 0.012	0.33
	DeepGraph-39+Perm	0.792 ± 0.023	0.660 ± 0.011	0.33

Table 6: For each scenario we generate 100 graphs with 39 nodes, and corresponding data matrix sampled from distributions with those underlying graphs. The number of samples is indicated by n.