Network Learning for Neural Model Interpretability

Aakanksha Naik (anaik) [1] **Alankar Jain** (alankarj) [1] **Aldrian Obaja Muis** (amuis) [1]

Abstract

Interpretability for neural network models is a trending topic in NLP, with some preliminary definitions of what it means to have an interpretable model. In this project, we focus on one form of interpretability (called *partial decomposability*) by learning an undirected dependency graph between neurons in a neural network, in the hope of uncovering groups of correlated nodes. We treat these groups as "concepts" learned by the network and manifest the interpretation of said concepts through prototypical examples (Melis & Jaakkola, 2018). Finally, we compare the interpretability of groups extracted by our approach to components extracted via Singular Value Decomposition (SVD). We apply our method on both synthetic and real NLP data (sentiment analysis; NLI) and uncover interesting insights.

1. Introduction and Motivation

Model interpretability is a key problem in machine learning, as it helps users understand the reasoning behind model predictions better. For example, in survival analysis, understanding why a model predicts a certain lifespan for a person can greatly help justify the risk assessment for insurance purposes. When using linear models with handcrafted features for a task, it is easy to identify which features influenced the final decision based on feature weights. However, as we see neural models being used more owing to their superior performance, interpretability has started to emerge as an important open challenge.

The precise definition of model interpretability that is useful practically is still not agreed upon by the community at large. Lipton (2017) discusses various definitions of model interpretability. In this work, we focus on the notion of interpretability as (partial) *decomposability*, where some components of the model in question lend themselves to an intuitive explanation. The research question that we are tackling is: given a model that is not interpretable in the decomposability sense, how can we uncover intuitive explanations for each component of the model?

As noted by Raghu et al. (2017), neurons in a neural model might be correlated. These correlations can result in redundancy while assigning intuitive explanations to network components, so they need to first be removed. Another application of decorrelating neurons is in co-training (Yu et al., 2011), which requires that the feature sets used by different views be independent. To remove this correlation, they use SVD to extract orthogonal vectors representing the same information as the original network.

In this project, we propose an alternative method, which is based on network learning with LASSO (Meinshausen et al., 2006), to discover correlations between neurons, and subsequently use the learned network structure as the basis to form units of interpretations. To complete the interpretability pipeline, we follow the idea proposed by Melis & Jaakkola (2018) of extracting prototypical examples as intuitive explanations for components in a neural model.

In summary, our contributions are as follows:

1. We propose to use network learning to find correlations between neurons in a neural network, and to use this as a basis for units of interpretation.
2. We compare our proposed method with SVD, an existing baseline for decorrelating neurons.
3. We show the feasibility of our proposed method through experiments on a synthetic dataset and on two concrete NLP tasks: sentiment analysis and natural language inference.

2. Related Work

Model Interpretability The problem of interpretability for neural network models has been approached from various angles, as summarized by Lipton (2017). For this project, we focus on the notion of "interpretability" as partial *decomposability*. We interpret decomposability as human-interpretable explanations for sub-modules

[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA.

10708 Class Project, Spring, 2019.

of a neural model. Prior work on creating such human-interpretable proxies for neural models can broadly be divided into two categories. The first class of techniques focus on training networks to generate these explanatory models while making predictions jointly. Al-Shedivat et al. (2017) propose contextual explanation networks (CEN), which use neural models on raw input data (C) to generate parameters for simpler probabilistic models that use interpretable features (X). The simpler model is then used to generate the final prediction, with its parameters serving as an explanation for the prediction. However, they assume that the interpretable features (X) are available for the task. Melis & Jaakkola (2018) do away with this assumption with SENN. They learn a set of interpretable basis concepts $h(.)$ which are analogous to high-level feature combinations from raw input data. To interpret these basis concepts, they extract a sample from their data which maximizes concept value and use it as a prototype for the concept. Lei et al. (2016) also build a model which learns to produce explanations and predictions jointly, however they focus on "extractive" explanations, i.e., using subsets of input sequence as explanations instead of weights from an interpretable classifier.

The second class of techniques in network interpretability focus on generating a posteriori explanations for previously trained models. Such techniques usually focus on generating explanations for model predictions on specific instances. Ribeiro et al. (2016) propose LIME (Local Interpretable Model-Agnostic Explanations), a framework which learns simple classifiers that are "locally faithful" to the model being explained, around the instance in consideration. These classifiers use an interpretable representation, which ensures that their parameters serve as an instance-specific explanation of the original model's prediction. Alvarez-Melis & Jaakkola (2017) generate similar post-hoc causal "explanations" for sequence-to-sequence model outputs. Following Lei et al. (2016), their explanations for any particular input-output pair consist of a mapping from each output token to a set of causally related input tokens.

Our proposed approach falls into the second class of models, where units of explanations for network sub-modules are uncovered via a post-hoc analysis. However, to interpret groups of nodes in our learned network, we will follow Melis & Jaakkola (2018) and generate prototypical subsets from our training data which have high activations for a neuron and use them to infer the feature captured by it.

Structure learning Multiple algorithms have been proposed to learn correlation structure between variables. Since covariation selection methods (Dempster, 1972) tend to break down when the covariance matrix is not invertible, we prefer L_1-regularization based methods such as the Meinshausen-Bühlmann algorithm (Meinshausen et al., 2006). (Raghu et al., 2017) propose SVCCA, a method that combines singular value decomposition and cannonical correlation analysis. SVCCA learns a *representative set of directions*, rather than combinations of individual neurons i.e variables. This technique makes it difficult to construct *prototypes* as suggested in this proposal for concept discovery. Nevertheless, it appears to be an effective technique for understanding network dynamics in a variety of applications (Saphra & Lopez, 2018).

3. Proposed Method

Our method consists of a three-stage pipeline: (1) network learning (2) defining units of interpretability (3) concept interpretation.

3.1. Network Learning

For network learning, similar to SVCCA, given the hidden node activations from a particular layer in a neural network, we want to discover subsets of correlated neurons. In contrast to SVCCA which performs SVD on hidden activations that produces a set of orthogonal dimensions, we propose to explicitly capture relationships between nodes by learning a correlation network. Further, motivated by prior observations that hidden layers in neural networks often capture redundant information (Raghu et al., 2017), we seek to enforce sparsity during network learning.

Given activations for n input samples from a network layer consisting of N nodes, we perform the following steps, largely following Meinshausen et al. (2006):

- For each node $i \in N$, we train a linear regression model M_i to predict i, given the values of all other nodes. We impose L_1 regularization during regression to induce sparsity. $M_i(j)$ now gives an estimate of the importance of node j in predicting i. For each regression model, the L_1 regularization coefficient is calculated according to the following equation:

$$\lambda(\alpha) = \frac{2\hat{\sigma}_i}{\sqrt{n}} \tilde{\phi}^{-1}\left(\frac{\alpha}{2N^2}\right) \tag{1}$$

  In the above equation, $\tilde{\phi} = 1 - \phi$, where ϕ is the cumulative density function of $\mathcal{N}(0,1)$ and $\hat{\sigma_i^2} = n^{-1} < X_i, X_i >$ is the empirical covariance, where X_i denotes the activation values for node i and $< a, b >$ stands for the dot product of two vectors. Choosing a regularization coefficient following the above procedure ensures that the probability of introducing false edges (i.e. learning non-zero weights between node-pairs that are not correlated) is bound by α. We can tune this bound based on our requirements. In our experiments we use $\alpha = 0.05$.
- Given N regression models, one for each node, we

introduce edges between all node pairs with non-zero weights. Since our network is undirected, for each pair (i, j), we choose $max(|M_i(j)|, |M_j(i)|)$ as the edge weight.

- The resulting graph serves as our correlation network, which is subsequently used to define the units of interpretation for the next step.

3.2. Defining Units of Interpretability

The main goal of network learning is to use it to interpret the underlying neural network. Our main idea is that some neurons might be correlated, and the correlation structure between neurons can shed some insights on how to interpret the neural network. [1]

We considered a few ways to define units of interpretability from the learned network structure, but eventually settled in using *communities*, as detected by community detection algorithms, as our units of interpretability.[2] In our experiments, we use the Louvain Community Detection algorithm (Blondel et al., 2008), an iterative method that optimizes modularity in a greedy way. This method is fast compared to most other community detection algorithms, while maintaining a very good modularity score. We refer the details of the algorithm to the original paper.

After running the community detection algorithm, we get a partitioning of the graph into a set of communities. These communities, which are essentially a collection of nodes, is what we consider as units of interpretability.

3.3. Concept Interpretation

Finally, given a set of units of interpretability, we want to assign human-interpretable meaning to these units. To this end, we follow the approach used by SENN (Melis & Jaakkola, 2018), where they extract examples from the data that "maximally activate" the concepts. They proposed three alternatives of selecting these prototypes. We adapted their first two alternatives to arrive at the following criteria for selecting prototypes: for each unit of interpretability (henceforth *unit*), we take the top-k examples having maximum difference between mean activation score across all neurons in that unit and mean activation score across all other neurons not in this unit. More formally, the score for each example X for the unit C is defined to be: $mean_{c \in C}[h(c)] - mean_{c \notin C}[h(c)]$, and then we take top-$k$ examples with the highest scores.

[1] We can see this negatively as the nodes capturing redundant information, or positively as the nodes jointly working together to encode richer concepts.

[2] We considered using connected components as units of interpretability. However, this definition of units of interpretability is too strict, as it often leads into a very small number of units, often 1, since the graph learned are often a connected graph.

Algorithm 1 Pseudo-code to generate the synthetic data.

$p \sim \text{Uniform}([0,1]^2, p)$ ▷ Sample p points from the unit square
for each pair of points (p_i, p_j) **do**
  prob $\leftarrow \phi\left(\frac{\text{distance}(p_i, p_j)}{\sqrt{p}}\right)$
  if $\text{Bernoulli}(\text{prob})$ **then**
    Put (p_i, p_j) to the edge set E
  end if
end for
while There are nodes with more than 4 neighbors **do**
  Randomly remove such edges
end while
for each pair of points (p_i, p_j) **do**

$$\Sigma^{-1}_{ij} \leftarrow \begin{cases} 0.245 \text{ if } (p_i, p_j) \in E \\ 1 \text{ if } i = j \\ 0 \text{ otherwise} \end{cases}$$

end for
Scale Σ such that $\Sigma_{ii} = 1$
$L \leftarrow \text{Cholesky}(\Sigma)$ ▷ L is a $p \times p$ matrix
samples $\sim \mathcal{N}(0, 1)$ ▷ Sample n points from $\mathcal{N}$
samples $\leftarrow L \times$ samples ▷ Realize the correlation

3.4. Research Questions

Overall, from this work we aim to gain deeper insight into model interpretability using network learning techniques. To do so, in our experiments we follow a two-pronged analysis of our networks along the following two dimensions: consistency and interpretability.

Consistency How consistent is the network learned from activations of the same layer across multiple random initializations and train/test data? If the resulting network stucture and interpretability are consistent, then we can establish that learned network structures indeed correspond to some high-level representations *important* to the task.

Human Interpretability How interpretable is the neural network that results from extracting prototypical examples? If there are concepts that can be associated with each unit of interpretability, then we can establish that the units of interpretability are indeed meaningful, and that they lead to better interpretability by humans.

4. Synthetic Data: Gaussian RV

To start, we first reproduce the results from (Meinshausen et al., 2006) with the same synthetic dataset that they use, and perform more in-depth analysis of the impact of number of datapoints on the quality of the recovered graph.

4.1. Synthetic Data Creation

Graph Creation The graph structure between nodes was created by first sampling p points in the unit square. Then

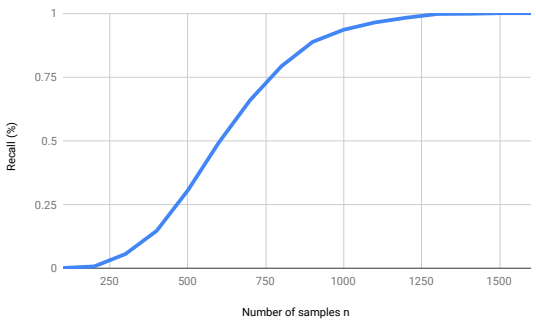


Figure 1. The number of correct edges recovered as a function of number of data points n.

for any two points, we assign an edge with a probability $\phi(\frac{d}{\sqrt{p}})$, where ϕ is the PDF of the normal distribution, and d is the Euclidean distance between the two variables. To simulate sparsity, edges were randomly removed until the maximum degree of any point is 4.

Generating Samples Then, each point is interpreted as a Gaussian random variable, with inverse cross-correlation of 0.245 between connected points (i.e., $\Sigma^{-1}_{ij} = 0.245$ for $i \neq j$), and zero otherwise. To ensure constant variance, all variables are finally rescaled so that the diagonal elements of Σ are all ones. We then transform this covariance matrix using Cholesky transformation to draw n independent samples drawn from the corresponding Gaussian distribution. The complete pseudo-code is shown in Algorithm 1.

4.2. Experiments

We run the method proposed by Meinshausen et al. (2006) on this dataset, with $p = 1000$ and $n = \{100, 200, \ldots, 1600\}$. Similar to the results of Meinshausen et al. (2006), we obtain a very high precision (100% in almost all cases). We also find that the recall increases with the number of data points n. We plot this in Figure 1.

This result is encouraging in two ways: (1) It confirms the results of Meinshausen et al. (2006), and (2) It gives us insight that providing more samples results in a graph that is closer to the true graph, under the Markov Random Field assumption. In the next section, we also experiment with various sample sizes to see how it affects the number of edges, and also how the units of interpretability behave.

5. Real Data: Stanford Sentiment Task

5.1. Task and Data Description

In order to evaluate the proposed method on a real task, we train a neural network to perform sentiment classification on the Stanford Sentiment Dataset (Socher et al., 2013). This dataset consists of 8544, 1101 and 2210 sentences in the training, validation and the test datasets with each sentence possessing one of three possible labels: positive, negative or neutral.

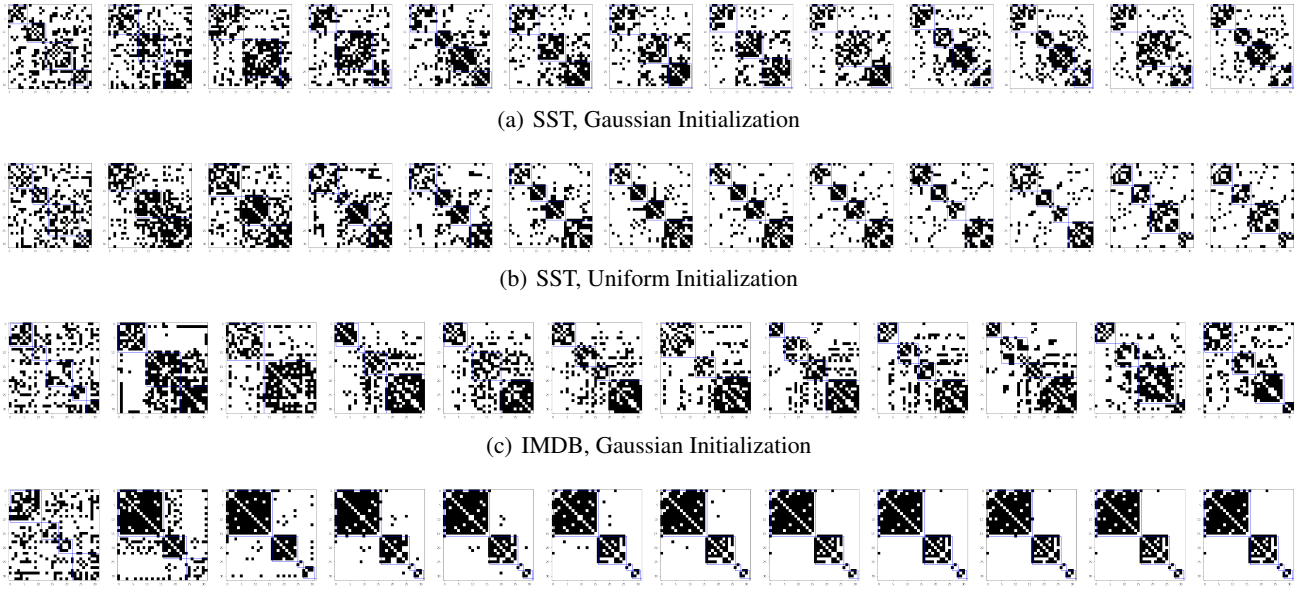
5.2. Experiments

For the Stanford Sentiment Task (SST) described above, we trained a 1-layer Multi-layer Perceptron (MLP) with 32 hidden nodes. Vocabulary of the dataset was fixed to 25000 words and the input to the MLP comprised of a Bag-of-Words (BoW) representation for each sentence with term frequencies instead of boolean entries representing presence/absence of a word. The output layer contained 3 nodes. This network was trained using Cross-Entropy Loss through minibatch training with 128 samples per minibatch. We observed the network overfitting on the training data quite quickly into the training, achieving 95% (and increasing) accuracy on the training dataset and saturating at $60 - 65\%$ accuracy on the validation dataset, so we chose to perform early stopping using only the first 5 epochs for analysis.

After training the neural network, we obtained activations of the hidden nodes on the training dataset and used them to learn an undirected network on the 32 nodes (now "observed") to analyze the dependency structure of the hidden layer. We analyze the evolution of these networks during the training phase, along with variations in the learned structure depending upon various parameters next.

5.3. Analyzing Learned Networks

We observed that the accuracy on the training dataset increased to almost 60% by the end of the first epoch, so, in order to assess the evolution of the learnt network at a greater granularity, we learnt the network structure at every one-tenth of the epoch. Figure 2 shows the evolution of the network for the first epoch. Further, we also analyzed the effect of different forms of initialization (Gaussian/Uniform) and different datasets (SST/IMDB) for the same task (sentiment classification) on the structure of the learnt network. The IMDB dataset comprised of 25,000 train and test sentences. We divided the train set into 17,500 sentences for training and the remaining 7,500 for validation. It differs from the SST dataset in that it only has two sentiment classes: positive and negative. The simple BoW MLP architecture described above achieves a very high accuracy on this dataset (88.34% on validation set). It is evident from Figure 2 that as the training progresses, the modularity of the graph increases and dense communities get formed. Further, uniform initialization (Figure 2 (b) and (d)) results in sparser networks as compared to Gaussian initialization. It should be noted that (Morcos et al., 2018) found that larger neural networks result in more consistent canonical representations, and since we only used 32 nodes in our network, it is possible that learned networks would



(a) SST, Gaussian Initialization

(b) SST, Uniform Initialization

(c) IMDB, Gaussian Initialization

(d) IMDB, Uniform Initialization

Figure 2. Network evolution during first epoch of training.

be even more consistent for greater number of nodes.

It is also interesting to see that the network learned in IMDB with uniform initialization shows very high modularity. Due to time constraint we do not investigate the results here further. But it is an interesting direction for future work.

5.4. Interpretability Results

As the main goal of our work is getting units of interpretability, we compare our method to a baseline method using SVD. With SVD, we can extract the top-k directions and treat them as units of interpretability. Then for both our method and the baseline, we extract prototypical examples as described in Section 3.

Label Distribution As described previously, for each detected community we extract top-100 prototypical example sentences from the training data. As a preliminary experiment, we compare the label distribution of these 100 examples to the label distribution in the full training data. The idea is that if the communities detected are meaningful, each of them should have a different distribution of labels, since otherwise the feature represented by the community is not distinguishing. We plot the label distributions for our method and SVD in Figure 4 and 5.

Comparing the distribution of labels across communities in our method, we see that the proportion of neutral class remains constant across all communities. However, we see one community (`Community2`) having more negative examples, and another community having more positive examples. `Community1` seems to be more sensitive to positive examples compared to `Community0`.

In contrast, in the label distribution for SVD, we see varying distribution of labels. However, in all SVD directions, the positive class, which is also the majority class in the training data, is always the majority class. So it is not very clear what each SVD direction actually represents.

The difference between our method and the baseline will be more pronounced when we compare the top words.

Word Salience When looking at the top words that maximally activate each community, as shown in Table 1 and 2, we see that our method gives more polarized word examples. `Community2` seems to have mostly negative words, `Community1` mostly positive words, and `Community0` a mix of some positive words and some neutral words. Comparing this result to the label distribution discussed previously, we see a strong alignment that the community with the highest negative class (`Community2`) has many more negative words, and the community with the most positive labels (`Community 1`) is associated with the positive words.

In contrast, the top words in SVD do not really have interpretable connection with the label distribution, apart from the set of positive words associated with `SVD2`, which shows a greater number of positive examples compared to other SVD directions. For example, `SVD0` and `SVD1` do not seem to be show a coherent set of words, unlike the

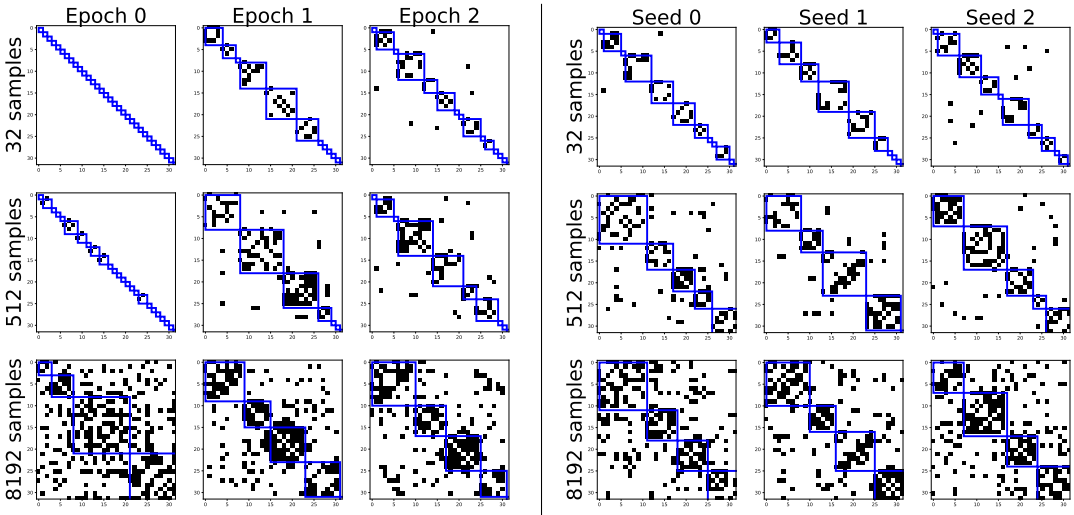


Figure 3. (left) These adjacency matrices shows the evolution of communities (shown in blue squares) across epochs for various sample sizes for the Stanford Sentiment Task. Epoch 0 is the untrained network. Notice how the community structure (shown by more edges outside the communities) is more random at the beginning, compared to after training. This shows that both the network learning and the community detection is meaningfully related to how trained the neural network is. *(right)* The resulting units of interpretation at the final epoch across different seeds and sample sizes for the Stanford Sentiment Task.

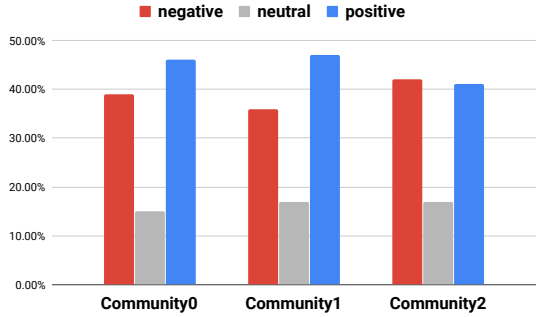


Figure 4. Label distribution in the top-100 prototypical examples for the 3 units discovered using our method.

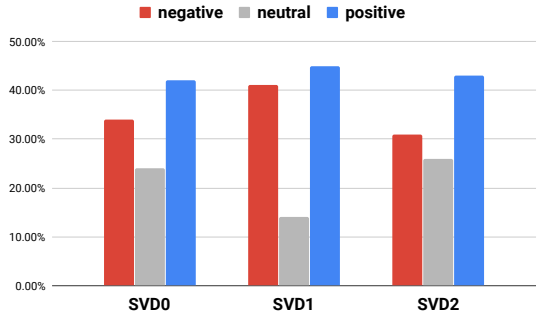


Figure 5. Label distribution in the top-100 prototypical examples for the top-3 SVD directions.

words discovered in our method.

6. Real Data: Natural Language Inference

6.1. Task and Data Description

The second task which we evaluate our proposed method on is NLI. Natural Language Inference (NLI), also known as Recognizing Textual Entailment (RTE), is a benchmark task in natural language understanding (Cooper et al., 1996; Condoravdi et al., 2003; Bos & Markert, 2005; Dagan et al., 2006). In this task, a model must determine whether a natural language hypothesis can be justifiably inferred from a given premise. Often, this is posed as a three-way decision where the hypothesis can be inferred to be true (entailment), false (contradiction) or its truth value cannot be determined (neutral).

Due to its importance, significant prior work (Dagan et al., 2006; 2009; 2013; Marelli et al., 2014) has focused on developing datasets and models for this benchmark task. Most recently, this task has been concretely implemented in two large-scale datasets: Stanford NLI (SNLI; (Bowman et al., 2015)), and Multi-genre NLI (MultiNLI; (Williams et al., 2017)). Both these datasets were collected via a crowdsourcing task wherein workers were given a premise sentence and asked to generate novel hypothesis sen-

Community0	Community1	Community2
surprise	perfectly	failure
provocative	thoughtful	Unfortunately
aftertaste	powerful	bore
Offers	brilliant	thinks
smarter	inventive	nowhere

Table 1. Top-5 maximally activating words for the three communities discovered by our method in the Stanford Sentiment Task.

SVD0	SVD1	SVD2
toughest	minimum	solid
techies	vs	enjoyable
redefinition	shocked	powerful
inform	Who	brilliant
subjected	Or	hilarious

Table 2. Top-5 maximally activating words for the three SVD directions discovered in the Stanford Sentiment Task.

tences representing the three categories of entailment relations. A key difference between these datasets is that for SNLI, premise sentences were selected from captions in the Flickr30k corpus (Young et al., 2014), while for MultiNLI, premise sentences were selected from ten distinct genres of written and spoken English. These datasets spurred the rapid development of neural models which performed extremely well on the entailment task. However, recent work has uncovered the presence of "annotation artifacts" (i.e. shallow heuristic patterns used by crowdworkers while generating hypotheses) in these datasets (Gururangan et al., 2018; Poliak et al., 2018). Several studies have shown that neural models exploit the presence of such biases instead of capturing linguistic information needed for the task (Naik et al., 2018; Glockner et al., 2018). Inspired by these observations, we "rediscover" biases in NLI datasets, by showing that these biases emerge as human-interpretable concepts on using our interpretability pipeline to analyze neural models. For all experiments in this report, we use SNLI, which has training, development and test set sizes of 549367, 9842 and 9824 respectively.

6.2. Neural Model

To "rediscover" some of the dataset biases reported by prior work, we build a *hypothesis-only* classifier. This model only uses the hypothesis sentence to predict the entailment label, completely ignoring the premise sentence. It consists of three layers: Embedding layer, Hidden layer, Output layer. For the embedding layer, we use fastText (Joulin et al., 2016) to construct 100-dimensional sentence embeddings (we use the default setting which computes bag of words+bigrams). The hidden layer consists of 32 nodes with ReLU activation while the output layer is a 3-class softmax layer (analogous to model settings used in our sentiment classification experiments). We experimented with larger hidden layers but observed no additional improvements in classification accuracy. The classification accuracy of this model on SNLI development and test sets hovers around 65.2% and 64.4% respectively, depending on the seed used. This matches results reported by Gururangan et al. (2018). Since a classifier which ignores the premise completely is able to perform much better than a majority-class baseline (33%; SNLI is balanced), this indicates that hypotheses generated by crowdworkers contain shallow patterns which make NLI artificially easier for models. We run our network learning approach to uncover some of these patterns from hypothesis sentences.

6.3. Analyzing Learned Networks

Our observations from analyzing networks learned for NLI match our insights from the sentiment classification experiments. To save space, adjacency matrix visualizations are included in the appendix.

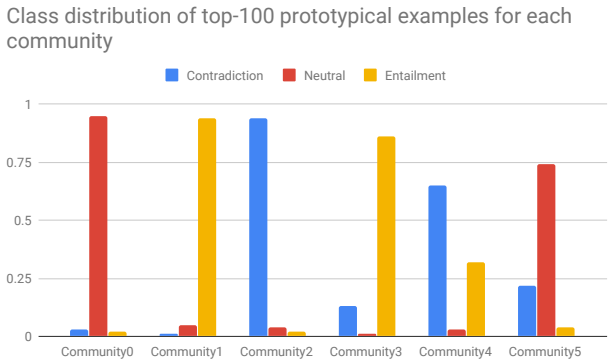


Figure 6. Class distribution of top-100 prototypical examples for each community created by our method

Label Distribution After running our network learning technique, we discover six communities among the hidden nodes of the hypothesis-only classifier. We then extract the top-100 prototypical examples for each community. The label distributions of prototypical examples for each community are shown in figure 6. From the graph, we can see that in each community, there is a clear majority label. Communities 0,1 and 2 in particular seem to be highly dominated by a single label (94-95% of examples have the same label). Henceforth, all further analyses of prototypical examples focus on these three *label-dominated* communities.

6.4. Interpretability Results

Prototypical Words Gururangan et al. (2018) noticed that hypotheses generated by crowdworkers contained lexical biases i.e. certain words were highly correlated with particular labels. For example, they observed that negation words (eg: no, nothing, never etc.) were present in a large number of contradictory hypotheses and hence were very predictive of the contradiction label. To study whether our communities can help us uncover such word-label correlations, we extract *prototypical words* for the three label-dominated communities. To extract prototypical words, we simply treat each word in the vocabulary as a hypothesis sentence and extract the top-100 words which maximally activate each community. The top 5 prototypical words for each community are shown in table 3. We can see that 4 out of 5 top words for community 2 (dominated by contradiction) are indeed negation words, corroborating observations made by previous work. This indicates that our technique can be used to identify word-label correlations to some extent. However, the top 5 words for communities 0 and 1 (neutral;entailment) seem less interpretable.

Prototypical Example Construction Strategies In addition to detecting lexical biases, Gururangan et al. (2018) also identified some common heuristics used by

Community1	Community0	Community2
least	proximity	nobody
age	picture	Nobody
picture	seems	No
motion	age	nothing
proximity	except	Mars

Table 3. Top-5 maximally activating words for the most label-dominated communities. Note that communities 1, 0 and 2 are dominated by entailment, neutral and contradiction labels respectively.

crowdworkers to generate hypotheses efficiently. We study whether such strategies can be detected by our method by looking at prototypical examples for the three label-dominated communities. Table 4 presents the distribution of construction strategies exhibited by prototypical examples. Note that an example may exhibit multiple or none of these strategies. The construction strategies detected are briefly explained below:

- **Entailment Strategies:** We observe that prototypical examples for entailment exhibit high usage of generic words (eg: animal, person) and non-gender words (eg:human), which were probably chosen to generalize over specific premise words (eg: dog, girl). They also demonstrate moderate usage of approximation words (eg:some,at least) used to replace exact numbers from the premise. 15% of the examples don not exhibit any of these strategies
- **Neutral Strategies:** Prototypical examples for the neutral category show moderate usage of modifiers (eg:tall,sad), superlatives (eg: first, favorite) and discourse markers (eg: because). Modifiers and superlatives can be used as a simple strategy to generate hypotheses that are not obviously entailed but still plausible. For example, for a premise sentence "A girl is playing", a neutral hypothesis can easily be generated as "A tall girl is playing". Discourse markers can be used to attach extra information to generate a neutral hypothesis. Finally, we also observe high usage of present participle verbs, which may be used to generate neutral hypotheses when the premise involves periodic actions. For example, for a premise "she plays everyday", a neutral hypothesis can be "she is playing". 36% examples exhibit none of these strategies
- **Contradiction Strategies:** Analogous to our observations from word-label experiments, we see that using negation words is the most dominant strategy among prototypical examples. We also observe moderate usage of lack-of-action words (eg: sleeping) to generate contradictory hypotheses when the premise involves an action. An interesting high-scoring feature in this class is the usage of "cats". This was also noticed by Gururangan et al. (2018), who speculated that it might be due to high presence of "dogs" in the premise sentences which were extracted from image captions. 33% examples exhibit none of these strategies

Unit	Top Features
Community1	Generic Words (32%)
	Non-Gender Words (60%)
	Approximation Words (19%)
Community0	Modifiers (14%)
	Superlatives (7%)
	Discourse Markers (5%)
	Present Participle (49%)
Community2	Negation Words (48%)
	Lack of Action (10%)
	Cats (13%)

Table 4. Strategies exhibited by top-100 prototypical examples for the most label-dominated communities. Note that communities 1, 0 and 2 are dominated by entailment, neutral and contradiction labels respectively.

Our observations match prior work, indicating that identifying groups of correlated neurons and interpreting them via prototypical examples can be used to uncover dataset biases, in addition to interpreting trained models. This suggests another interesting application of our interpretability pipeline: using it for adversarial filtering in a data collection process. Current work using adversarial filtering during data collection (Zellers et al., 2018; Dua et al., 2019) usually runs a state-of-the-art model and discards crowdsourced examples which can be solved by it, ensuring that only "hard" examples get included. However, having an intermediate module that identifies strategies which make examples easy for SOTA models can make filtering more efficient by prompting users to *not* use these strategies which result in their examples getting rejected.

7. Conclusion

In this project, we proposed an approach to learn sparse networks (Gaussian graphical model) over hidden nodes in a neural network, with the goal of decomposable interpretability and potential network compression. We then performed experiments using one synthetic dataset and two real NLP datasets. The results show some evidence that units of interpretability as found by our proposed method are quite meaningful, as they show different behavior with respect to the label distribution of the examples that maximally activate them. However, there is still little consistency of learned units across different initialization, which may be explained by the fact that each initialization leads to different local optima.

References

Abu-Aisheh, Z., Raveaux, R., Ramel, J.-Y., and Martineau, P. An exact graph edit distance algorithm for solving pattern recognition problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*, 2015.

Al-Shedivat, M., Dubey, A., and Xing, E. P. Contextual explanation networks. *arXiv preprint arXiv:1705.10301*, 2017.

Alvarez-Melis, D. and Jaakkola, T. A causal framework for explaining the predictions of black-box sequence-to-sequence models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 412–421, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D17-1042.

Blondel, V. D., Guillaume, J.-l., Lambiotte, R., and Lefebvre, E. Fast unfolding of communities in large networks. In *arXiv*, pp. 1–12, 2008. URL https://arxiv.org/pdf/0803.0476.pdf.

Bos, J. and Markert, K. Recognising textual entailment with logical inference. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 628–635. Association for Computational Linguistics, 2005.

Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. URL http://aclweb.org/anthology/D15-1075.

Condoravdi, C., Crouch, D., De Paiva, V., Stolle, R., and Bobrow, D. G. Entailment, intensionality and text understanding. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning-Volume 9*, pp. 38–45. Association for Computational Linguistics, 2003.

Cooper, R., Crouch, D., Van Eijck, J., Fox, C., Van Genabith, J., Jaspars, J., Kamp, H., Milward, D., Pinkal, M., Poesio, M., et al. Using the framework. Technical report, 1996.

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In *Machine learning challenges. evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, pp. 177–190. Springer, 2006.

Dagan, I., Dolan, B., Magnini, B., and Roth, D. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(4):i–xvii, 2009.

Dagan, I., Roth, D., Sammons, M., and Zanzotto, F. M. Recognizing textual entailment: Models and applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220, 2013.

Dempster, A. P. Covariance selection. *Biometrics*, pp. 157–175, 1972.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*, 2019.

Glockner, M., Shwartz, V., and Goldberg, Y. Breaking NLI systems with sentences that require simple lexical inferences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 650–655, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-2103.

Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S., and Smith, N. A. Annotation artifacts in natural language inference data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2017. URL https://www.aclweb.org/anthology/N18-2017.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

Karpathy, A., Johnson, J., and Fei-Fei, L. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Lei, T., Barzilay, R., and Jaakkola, T. Rationalizing neural predictions. *arXiv preprint arXiv:1606.04155*, 2016.

Lipton, Z. C. The Mythos of Model Interpretability. Technical report, 2017. URL https://arxiv.org/pdf/1606.03490.pdf.

Marelli, M., Bentivogli, L., Baroni, M., Bernardi, R., Menini, S., and Zamparelli, R. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. In *Proceedings of the 8th*

International Workshop on Semantic Evaluation (SemEval 2014), pp. 1–8, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University. URL http://www.aclweb.org/anthology/S14-2001.

Meinshausen, N., Bühlmann, P., et al. High-dimensional graphs and variable selection with the lasso. *The annals of statistics*, 34(3):1436–1462, 2006.

Melis, D. A. and Jaakkola, T. Towards robust interpretability with self-explaining neural networks. In *Advances in Neural Information Processing Systems*, pp. 7786–7795, 2018.

Morcos, A. S., Raghu, M., and Bengio, S. Insights on representational similarity in neural networks with canonical correlation. In *NeurIPS*, 2018. URL http://arxiv.org/abs/1806.05759.

Naik, A., Ravichander, A., Sadeh, N., Rose, C., and Neubig, G. Stress test evaluation for natural language inference. In *Proceedings of the 27th International Conference on Computational Linguistics*, pp. 2340–2353, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/C18-1198.

Poliak, A., Naradowsky, J., Haldar, A., Rudinger, R., and Van Durme, B. Hypothesis only baselines in natural language inference. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*, pp. 180–191, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/S18-2023. URL https://www.aclweb.org/anthology/S18-2023.

Raghu, M., Gilmer, J., Yosinski, J., and Sohl-Dickstein, J. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. (Nips):1–17, 2017. ISSN 10495258. doi: 1706.05806. URL http://arxiv.org/abs/1706.05806.

Ribeiro, M. T., Singh, S., and Guestrin, C. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144. ACM, 2016.

Saphra, N. and Lopez, A. Understanding Learning Dynamics Of Language Models with SVCCA. 2018. URL http://arxiv.org/abs/1811.00225.

Shi, X., Knight, K., and Yuret, D. Why neural translations are the right length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2278–2282, 2016.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Young, P., Lai, A., Hodosh, M., and Hockenmaier, J. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, December 2014. doi: {10.1162/tacl_a_00166}. URL https://www.aclweb.org/anthology/Q14-1006.

Yu, S., Krishnapuram, B., Rosales, R., Rao, R. B., and Com, B. R. Bayesian Co-Training. *Journal of Machine Learning Research*, 12:2649–2680, 2011. URL http://jmlr.csail.mit.edu/papers/volume12/yu11a/yu11a.pdf.

Zellers, R., Bisk, Y., Schwartz, R., and Choi, Y. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 93–104, Brussels, Belgium, 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1009.

A. Adjacency Matrices for NLI

Figure 7 shows the evolution of the learned network structure across epochs, as well as the network structure at the final epoch for different initializations. As training progresses, graphs become more modular (epoch 0 shows more randomness than epochs 1 and 2). The network converges fairly quickly, not showing much increase in modularity after the first epoch. Finally, we again observe that using different seeds results in different network structures.

B. Prototypical Examples for NLI

C. Toy Task: Premier League

Our idea of learning the network structure between the neurons in a neural model can be applied to virtually any neural model. For the purpose of this project, we start with a toy task: predicting the probability of a premier league team getting certain number of goals given the opposing team.

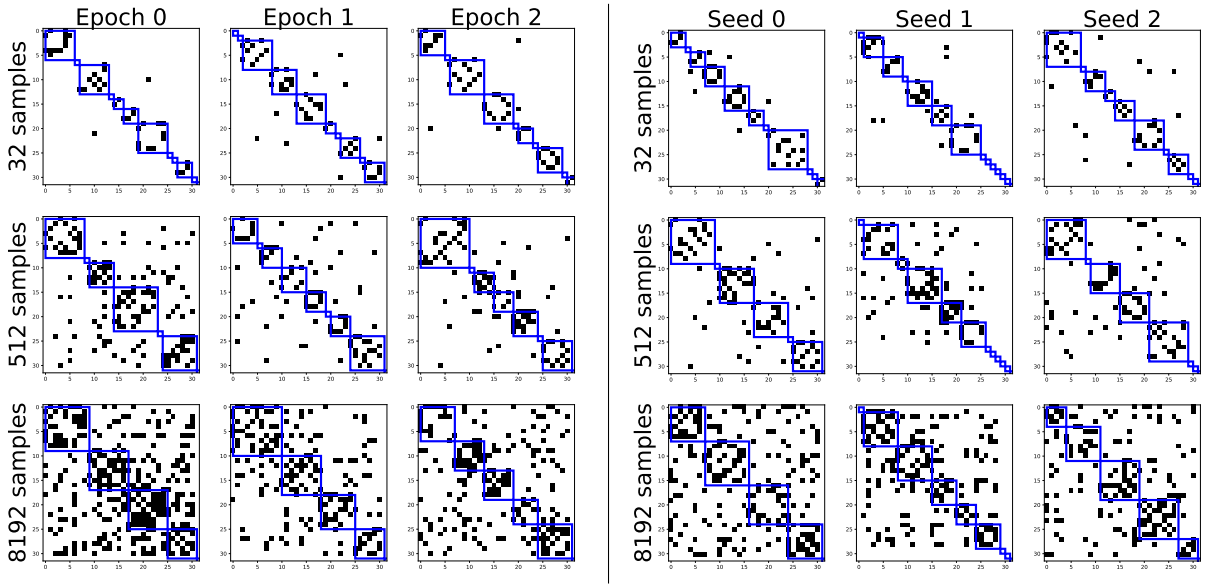


Figure 7. (left) Adjacency matrices for the Natural Language Inference task across epoch, with communities marked with blue squares. *(right)* The visualization at final epoch across different initialization.

Unit	Top Examples
Community1	*Humans are interacting* *Humans holding something* *Humans outside*
Community0	*Three men are joyously smiling* *Tall humans waiting* *Tall humans enjoying*
Community2	*Nobody is holding anything* *Two cats sleeping* *Nobody is sitting*

Table 5. Top 3 prototypical examples for the most label-dominated communities. Note that communities 1, 0 and 2 are dominated by entailment, neutral and contradiction labels respectively.

After testing our idea on this toy task to show its feasibility, we then show the application of our idea on an existing NLP task: Natural Language Inference (NLI).

For the toy task on premier league score probability prediction, we start by modeling the problem as a Bayesian graphical model. We assume that each team has a specific attack and defense strength as numeric value, and the difference between the attack of current team and the defense of the opposing team determines the distribution of the number of goals made by this team. To break the symmetry between the two teams, we also model the concept of having home advantage on one of the teams, which affect the distribution of one of the teams. A graphical summary of the model is shown in Figure 8.

In our experiments, we synthetically generate samples from

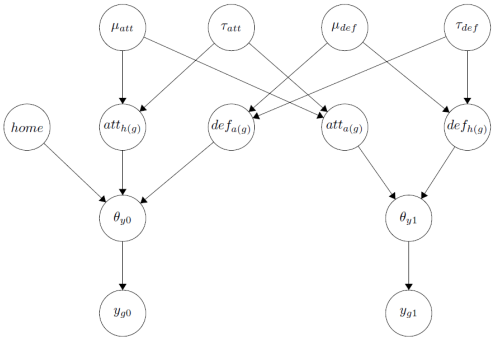


Figure 8. The dependency network between variables modeled in the toy task.

this network using the following distributions, where $h(g)$ and $a(g)$ represent the home team and the away team in the match g, respectively:

$$\mu_{att} \sim \mathcal{N}(0, \sigma_1) \qquad \mu_{def} \sim \mathcal{N}(0, \sigma_1)$$
$$\sigma_{att} = 1 \qquad \sigma_{def} = 1$$
$$att_t \sim \mathcal{N}(\mu_{att}, \sigma_{att}^2) \qquad def_t \sim \mathcal{N}(\mu_{def}, \sigma_{def}^2)$$
$$\log \theta_{g0} = home + att_{h(g)} - def_{a(g)}$$
$$\log \theta_{g1} = att_{a(g)} - def_{h(g)}$$
$$home \sim \mathcal{N}(0, \sigma_0) \qquad y_{gj} \sim \text{Poisson}(\theta_{gj}), j = 0, 1$$

Here y_{gj} represents the number of goals sampled for the team $h(g)$ when $j = 0$ and the team $a(g)$ when $j = 1$.

We consider T participating teams, and we first sample the attacks and defenses of each team. Then, using this parameters, we sample M leagues, where each league consist of a team playing with all other teams twice, once as the home team, and once as the away team (so in total we have $MT(T-1)$ matches.

For our final report, we also wish to explore NLI, whose details have been described in the appendix.

D. Preliminary Experiments

For the purpose of this report, we have focused only on the toy task. Extending our experiments to real-world NLI data will be one of our goals for the final report.

D.1. Model and Data

We train an MLP with a single hidden layer with different number of hidden nodes ($N_h \in \{4, 8, 16, 32, 64\}$) with ReLU activation. The input to our MLP is comprised of the IDs of the teams in the game (one-hot vector of size N_T, the number of teams) and the goals scored by each of the teams. The output is two-dimensional: probability assigned to the goals scored by the each of the teams. We apply a sigmoid activation to the output layer to learn these probabilities. The network is trained for 100 epochs to optimize mean squared error loss using minibatch stochastic gradient descent (batch size $= 100$) with Adam optimizer. Further, we experiment with two different data sizes for 4 teams: ($N_T = 4$): number of matches between each team pair, $N_M \in \{20, 100\}$, resulting a total dataset size of $D \in \{240, 1200\}$. Note that $D = N_M \cdot 2^{N_T} C_2$. We used 80% of the generated dataset for training and the remaining 20% for testing purposes.

D.2. Basic Analysis

We evaluated the learning of the trained model by calculating the goodness-of-fit using the coefficient of determination, R^2, averaged over 5 different seed values for the training dataset. We show training R^2 values here because they were highly correlated with the test R^2 values. It is evident from Figure 9 that the learnt model runs very well, resulting in almost perfect predictions $R^2 \approx 1$ for higher number of hidden nodes. Also, model performance predictably improves (both in terms of mean and standard deviation) as the network capacity improves. Larger training data expectedly improves performance for every number of hidden nodes. Next, we learn a sparse Gaussian Graphical Model over the hidden nodes and assess its density by compute a graph density metric, $\frac{|E|}{{}^{n}C_2}$, which expresses the number of edges in the graph as a fraction of the maximum number of edges possible (those in a complete graph). As Figure 10 shows, graph density reduces, i.e., graph becomes sparser as the number of hidden nodes increase. Further, the learnt graph seems to be denser for larger training data.

D.3. Consistency Analysis

These experiments evaluate the consistency of correlation networks learned by our method, across different random initializations and across train and test data. To evaluate graph inconsistency, we propose the following metric:

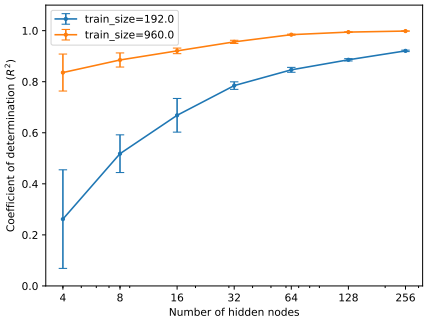


Figure 9. R^2 on training set for different number of hidden nodes.

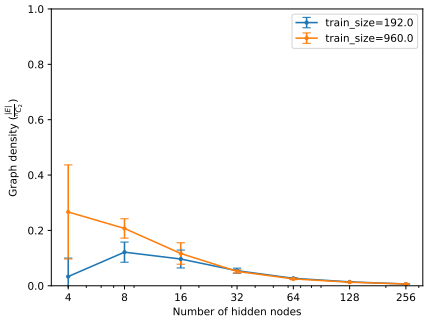


Figure 10. Graph density for different number of hidden nodes.

$$GI(G1, G2) = \frac{GED(G1, G2)}{max(|E_{G1}|, |E_{G2}|)} \quad (2)$$

In the above equation, E_{G1}, E_{G2} refer to the edge-sets for graphs $G1, G2$ respectively, while $GED(G1, G2)$ gives the graph edit distance (Abu-Aisheh et al., 2015) between the two. We normalize raw graph edit distance in order to allow comparison across different hidden node sizes. Since computing graph edit distance is expensive, to improve tractability, we remove $min(|S_{G1}|, |S_{G2}|)$ singleton nodes from both $G1$ and $G2$. Here S_G is the set of all singleton nodes present in graph G.

Before calculating inconsistency scores, we first qualitatively analyze learned graphs for 4,8 and 16 hidden nodes across multiple random initializations. For 4 hidden nodes, we see some consistency across initialization, but this drops very quickly as we increase the number of hidden nodes, dropping to nearly 0 for 16 nodes. Hence, we compute inconsistency scores for networks 4 and 8 hidden nodes and see that inconsistency across initialization is high, which is an issue we need to resolve.

While learned graphs are not consistent across initializations, we do observe that train and test graphs for each random seed show high levels of isomorphism. This behavior can be observed in figure 11. As the number of hidden nodes increases, train-test isomorphism decreases, but increasing the amount of training data leads to increase in isomorphism at higher hidden node settings.

Hidden	D1	D2
4	0.8	1.0
8	0.6	0.6
16	0.4	0.0

Table 6. Percentage of isomorphic graphs on training data across multiple random initializations for each hidden node and dataset setting. D1 and D2 refer to the 240 and 1200 example datasets respectively

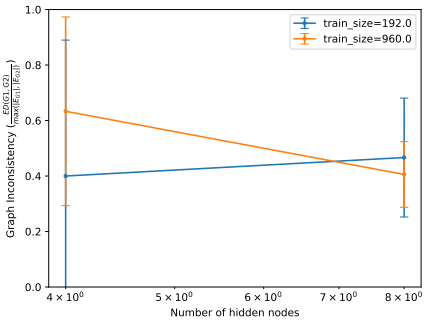


Figure 11. Percentage of isomorphic graphs across different seeds vs number of hidden nodes

D.4. Human Interpretability Analysis

From our observation of the prototypes extracted by this method on the graphs at the bottom of Figure 13, it seems that the activations of the large connected component is capturing the probability distribution of the number of goals for matches between team 2 as the attacker and team 3 as the defender.

E. Limitations of Toy Task

There are several directions that we would like to explore for our final project. One of the shortcomings of our toy problem is that because the probability density assigned to the input is Poisson, which involves exponentiation of attack and defense strengths, the model to be learned does not necessitate a decomposable structure over the hidden nodes of an MLP. To address this, we would like to simplify the task a little bit (so that the "expected" behaviour of the neural network is decomposable). Next, we would like to extend our analysis to problem settings where previous research has demonstrated decomposability of hidden layers (Shi et al., 2016) (Karpathy et al., 2015). We would like to explore if our proposed approach confirms these findings. Finally, we would like to apply the proposed approach (suitably altered to ensure it works on the two simple tasks described above) to test decomposable interpretability for NLI. The old and updated versions of the plan of activities are described in the appendix.

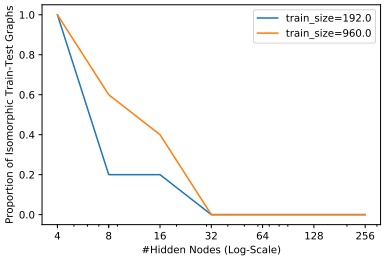


Figure 12. Percentage of isomorphic train-test graph pairs vs nu

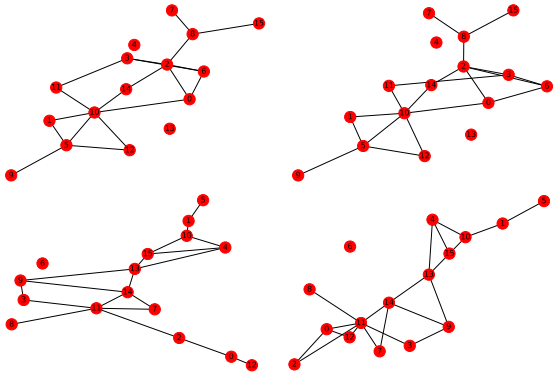


Figure 13. (top-left) The graph learned from the activations of the training data. (top-right) The graph learned from the activations of the test data. (bottom-left and bottom-right) The graphs learned on training and test data by the network using different initialization of the MLP.

F. Recovering Network Structure using MCMC Samples

One of the experiments we tried for sanity check of network learning was to train it on the MCMC samples (after sufficient burn-in) generated in HW-2 to see if it recovers the network structure of the Bayesian Network for the premier league. This, for appropriately tuned level of sparsity resulted in the learned model as shown in Figure 14.

The orange nodes refer to the teams' defenses, while the blue nodes refer to the teams' attacks. The pink node represents prior parameter related to defense, and the brown node represents the prior parameter related to attack.

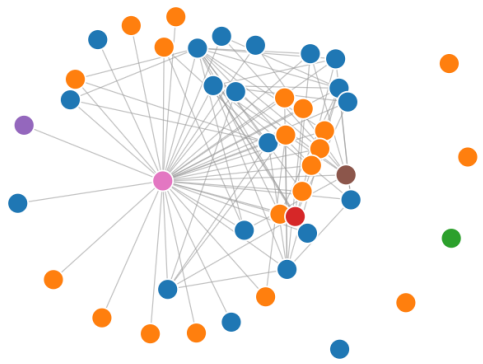


Figure 14. The network learned on MCMC samples.