Information Bottleneck for Multi-Task LSTMs

Anonymous Author(s)
Affiliation
Address
email

Abstract

Neural networks, which have had a profound effect on how researchers , do so through a complex, nonlinear mathematical structure which can be difficult to interpret or understand. This is especially true for recurrent models, as their dynamic structure can be difficult to measure and analyze. However, interpretability is a key factor in understanding certain problems such as text and language analysis. In this paper, we present a novel introspection method for LSTMs trained to solve complex language problems, such as sentiment analysis. Inspired by Information Bottleneck theory, our method uses a state-of-the-art information theoretic framework to visualize shared information around labels, features, and between layers. We apply our approach on simulated data, and real sentiment analysis datasets, providing novel, information-theoretic insights into internal model dynamics.

1 Introduction

Recurrent neural networks (RNN) used to solve problems with sequential text data already encounter unique performance issues such as vanishing or exploding gradients which can make established model introspection methods such as saliency and sensitivity analysis less effective. The sequential structure inherent in these models inclines researchers to create "inherently" interpretable architectures using mechanisms like attention, for example. Relatively few methods exist, however, which do not require architectural changes to the model, while still exploring the unique sequential interactions.

There is a need for model introspection schemas which are agnostic to the particular issues encountered by RNNs. One such potential framework is provided by the Information Bottleneck method from Tishby et al. [9, 8]. The Information Bottleneck method describes deep learning in terms of the trade-off in mutual information between a model's hidden layers and its input and output layers. Regardless of whether or not the various theoretical underpinnings of the Information Bottleneck theory of deep learning are correct, the idea of measuring shared information within a model is promising. At the very least, measuring how layers share information can provide some unique insight to RNNs, while remaining agnostic to the gradient.

In this work, we apply the Information Bottleneck (IB) method to a multi-task text-classification RNN. Our architecture uses a long-short term memory (LSTM) model to two sentiment analysis tasks simultaneously using a shared LSTM for both tasks. We will analyze the trained model with the Information Bottleneck framework developed by Tishby et al. in order to better interpret what the model learns from both tasks. These analyses will be compared to the single-task problems to show what the model has to do differently in order to learn two tasks at once. Finally, we will embolden our findings by comparing them to other RNN interpretation methods such as saliency and activation maps. Our hypothesis is that using MI in a fashion inspired by the IB principal will provide unique insight into sequential and multi-task dynamics which is not afforded by other techniques.

Our method provides an altogether new perspective on interpreting LSTMs and RNNs in general. Mutual information as a measure between sets of activation can be applied to interpret models without

altering the model structure, and can be organized to visualize the sequential changes in information unique to recurrent models.

1.1 The Information Bottleneck

The Information Bottleneck (IB) principal was introduced by Naftali Tishby as a general information theoretic principle for describing the relevant information between a pair of random variables $\mathbf{X}$ and $\mathbf{Y}$, and how that information can be captured by some encoding $\mathbf{H}$ [9]. Namely, if we assume that all *relevant* information in $\mathbf{X}$ about $\mathbf{Y}$ is given as the mutual information $I(\mathbf{X};\mathbf{Y})$, the goal is to find a set of *minimal sufficient statistics*, $\mathbf{H}$ which best captures $I(\mathbf{X};\mathbf{Y})$. Finding this $\mathbf{H}$ is then formulated as minimizing the Lagrangian

$$\mathcal{L}[p(h|x)] = I(\mathbf{X};\mathbf{H}) - \beta I(\mathbf{H};\mathbf{Y}). \tag{1}$$

In 2015, Tishby extended the general IB principal to analyze how DNNs learn to predict information around a label space $\mathbf{Y}$, given some input $\mathbf{X}$ [8]. Specifically, he argues that the set of hidden layers $\mathbf{H}$ in a well-trained, feed-forward DNN attempt to minimize the Lagrangian described in equation 1. Additionally, this work first proposes the idea of visualizing an **Information Plane**, where information between hidden layers and the input are plotted on the X axis, and information between hidden layers and output are plotted on the Y axis. This method can be used, Tishby proposes, to study DNN learning dynamics.

2 Methods

Our goal with this work is to evaluate and interpret LSTM-based models through the perspective of IB. Although IB has been used as an objective for RNN learning elsewhere [11, 7], it has never been used to interpret LSTMs using standard LSTM training techniques. Instead, much of the IB research for LSTMs focuses on using mutual information to optimize model training. Our work expands the current body of IB and LSTM interpretability in two primary ways: analyzing a shared-weights multi-task model and by looking at the information transfer between subsequent layers, which has not been done before.

More formally, we write the Mutual Information between the input at a given timepoint $\mathbf{X}_t$ and the hidden layer at timepoint t_1, $\mathbf{H}_t$ as

$$I(\mathbf{X}_t;\mathbf{H}_t) = E(\mathbf{X}_t) - E(\mathbf{X}_t|\mathbf{H}_t) = \sum_{x\in\mathbf{X}_t, h\in\mathbf{H}_t} p(x,h)\log\frac{p(x,h)}{p(x)\,p(h)} \tag{2}$$

where $E(x)$ is the entropy of x. Typically this measure can only be *estimated*, since true distributions are not available. Shwartz et al. provide a histogram-based binning procedure for estimating mutual information [6]. In future work, it may be prudent to test multiple estimators, as Saxe et al. found that the choice of estimator may influence trajectory in the information plane [5].

Provided that we have a reasonable estimator, we can then do the same for our single output features $I\left(\hat{\mathbf{Y}};\mathbf{H}_t\right)$. This provides us with a pair of coordinates per layer which can be visualized in the X and Y axis of an Information Plane plot, similar to the strategy proposed in Shwartz et al. [6].

Multi Task Learning. We use the above formulations and the methods from Shwartz et al. [6] to analyze a multi-task sentiment analysis model, structured similarly to Liu et al. [2]. Multi-task learning leverages shared information between related tasks to improve final classification by learning tasks simultaneously. For multi-task analysis, researchers have suggested different model architectures based on different focuses [1, 10], but always the ultimate goal of these architectures is to provide a mean to share some lower layers among the tasks to determine common features. All of these features make multi-task learning a particularly appealing target for our mutual-information method.

The two tasks we perform will be the binary classification problem from the Internet Movie Database [3] and the five-class sentiment analysis from the rotten tomatoes dataset [4]. Both tasks predict the emotional sentiment of text-based movie reviews. The first task classifies the movie reviews as either positive or negative. The second task classifies the reviews as very negative, mostly negative, neutral,

mostly positive, or very positive. We chose these two particular tasks due to their similarities both in problem and input data. The similarities between the two tasks make it easier to train the model, and make the mutual information results more interpretable, as we expect the information results to be similar for both tasks. To make the number of layers uniform, the IMDB dataset is batch-padded to 300 layers, so that the earliest words are trimmed from any sample with more than 300 words, and samples with fewer than 300 words are padded at the beginning of the sample. This 300 number was calculated so that fewer than 40% of samples were padded. The rotten tomatoes (RT) dataset was padded to 10 timesteps, in a similar fashion as the IMDB dataset.

3 Experimental Setup

First, we utilize a simulated data set provided as part of Aymeric Damien's Tensorflow examples on github[1]. The data set is generated as two classes of numeric sequences with 10 timepoints each. One class is generated as gaussian noise, and the other is generated as a linear trend starting from an arbitrary integer. We use the default generation parameters provided in the source code for the data set, and train a one-layer LSTM to classify between kinds of sequences for 1000 epochs.

For real data, we use sentiment analysis datasets from both IMDB [3] and Rotten Tomatoes (RT) [4], which are commonly used in the literature. We use a skip-gram model to create three sets of embeddings: IMDB only, RT only, and both datasets combined, or shared embeddings. The shared embeddings are concatenated with both single-task embeddings, as described in [1]. Then, an LSTM is trained on both datasets, so that each sample (or batch) can be from either dataset. Each dataset has a separate feed-forward model that trains a final, task-specific output layer, such that for each sample from a given task, only the feed-forward model for that task is updated. The feed-forward model for both tasks consists of one additional hidden layer, and the output layer. For comparative purposes, we also train single-task models for both datasets, which are evaluated in the appendix.

Next, we apply the evaluation framework proposed in section 2 to all three models, which produces a pair of coordinates in the Information Plane for each layer. We then provide the corresponding Information Plane visualizations, following the practice in [6] and [5]. For the multi-task model, two separate plots will be generated, corresponding to the measures for each task.

4 Results and Discussion

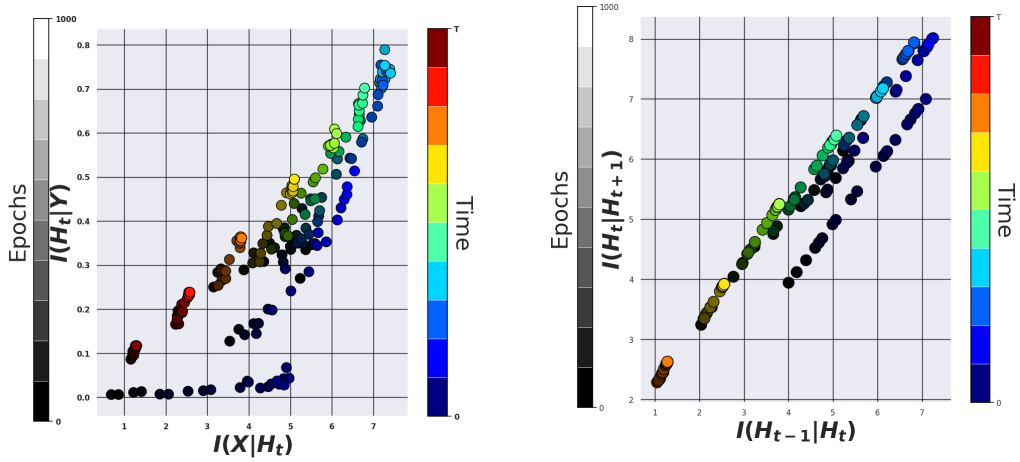
First, we apply the IB analysis to the simple simulated experiment described above, and provide the corresponding information planes in figure 1. In panel 1a we provide the information plane plot following the standard Tishby approach, plotting each LSTM module over an 1000 epoch training period with a learning rate of 0.001. Panel 1b plots the information at each timepoint as measured between that timepoint and the previous timepoint (x-axis) and the following one (y-axis).

Figure 1a shows increasing feature representation in the initial timepoint, with simultaneously increasing label and feature representation around timepoints 2-4. Both feature and label representations then drop from about timepoints 5 onward. Visualization over the training period shows that different modules are responsible with feature and label representation, with modules toward the middle of the sequence carrying the most information around both labels and input features. Additionally, we see that the information in earlier timepoints changes much more drastically than later timepoints.

In figure 1b we see that with the exception of the first timepoint, the amount of shared information between adjacent timepoint decreases linearly across the sequence, with the later timepoints illustrating the most independence with their neighbor representations. Additionally, we observe a linear increase in information shared with adjacent timepoints which again increases more rapidly at earlier modules in the sequence, and less rapidly at later modules in the sequence.

Multi-Task Analysis. Next, we measure how mutual information changes when applied to real text data in a multi-task LSTM. Figure 2 (in appendix A) provides the information plane plots for both the IMDB and Rotten Tomatoes (RT) tasks described above, trained for 100 epochs with a learning rate of 0.0001. Panels 2a and 2b provide the information plane plot for the LSTM modules in the IMDB and Rotten Tomatoes task respectively. Panels 2c and 2d provide the information plane plot for the first hidden linear layers for the IMDB and Rotten Tomatoes task respectively. Panels 2e

[1] `https://github.com/aymericdamien/TensorFlow-Examples`



(a) Information Plane measured between the hidden LSTM layers and the labels (y-axis) and the hidden layers and the input features (x-axis). Different colors indicate different timepoints, according to the provided colorbar. Darker points come from earlier epochs, and lighter points come from later epochs.

(b) Information Plane measured between the hidden LSTM layers and the LSTM module at the previous timepoint (x-axis) and the hidden layers the module at the next timepoint (y-axis). Different colors indicate different timepoints, according to the provided colorbar. Darker points come from earlier epochs, and lighter points come from later epochs.

Figure 1: Information Plane for the Simulated Experiment. A single-layer LSTM was trained to classify gaussian noise versus a linear trend trained for 1000 epochs with a learning rate of 0.001.

and 2f provide the information plane plot for the output layer for the IMDB and Rotten Tomatoes task respectively. For the IMDB task, the last 100 timepoints have been provided in figure 4 in the appendix to allow for closer analysis.

First, looking at the LSTM layers for either task, we see a clear organization of information representation, with later modules in the sequence holding more information both around the input features and labels. For the IMDB task in panel 2a we do observe some compression of input features in the last few timepoints, though unlike the simulated case, information around the labels is not lost. This compression is more clearly visible when looking only at the last few timepoints, which we provide in panel 4a in the appendix.

While there are no clear changes in the information during training for the Rotten Tomatoes task, we can observe some changes during training in the LSTM layer for the IMDB task, though the changes are subtle. In figure 4a in the appendix, we observe that the last few timepoints do not begin with perfect label representation, though this is very quickly learned. Additionally, we can see that for lower timepoints, the feature representation increases through training.

In the output layers for both tasks, we again can observe the compression of features across the sequence of LSTM modules, and in additional visualization over epochs lets us see an increase in the overall compression of features at each timepoint; however, the increase in compression comes with a loss in label information about halfway through the training period. We note that the information around the input features initially increases while label information also increases; however, once feature compression begins about halfway through the training period, label information is also lost.

5 Conclusion

In recurrent models, where it's difficult to measure and analyze the dynamic structure, interpretability is extremely important. Inspired by Naftali Tishby's work with the Information Bottleneck principal, we apply mutual information to the analysis of LSTMs. Using a simulated and real data demonstrations, we show how mutual information provides unique insight into how feature and label information propagates throughout time in LSTM modules.

References

[1] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2873–2879. AAAI Press, 2016. ISBN 978-1-57735-770-4. URL `http://dl.acm.org/citation.cfm?id=3060832.3061023`.

[2] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning, 2016.

[3] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P11-1015`.

[4] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL `https://doi.org/10.3115/1219840.1219855`.

[5] Andrew Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Tracey, and David Cox. On the information bottleneck theory of deep learning. In *Sixth International Conference on Learning Representations*, 2018.

[6] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[7] Lei Sun, Jun Du, Tian Gao, Yu-Ding Lu, Yu Tsao, Chin-Hui Lee, and Neville Ryant. A novel lstm-based speech preprocessor for speaker diarization in realistic mismatch conditions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5234–5238. IEEE, 2018.

[8] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE, 2015.

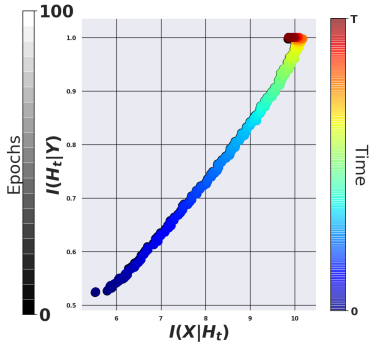
[9] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[10] Liqiang Xiao, Honglun Zhang, and Wenqing Chen. Gated multi-task network for text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 726–731, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2114. URL `https://www.aclweb.org/anthology/N18-2114`.

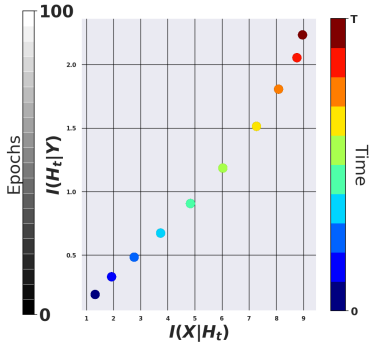
[11] Duo Xu and Faramarz Fekri. Time series prediction via recurrent neural networks with the information bottleneck principle. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2018.

A Real Data Results

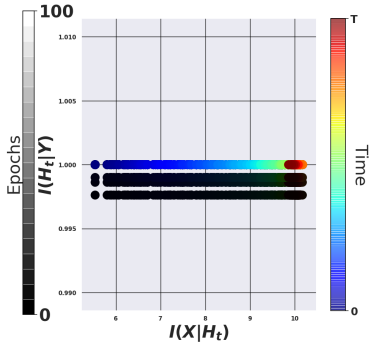
In this section, we include the real-data results for the multi-task experiments described in the main body of the text.



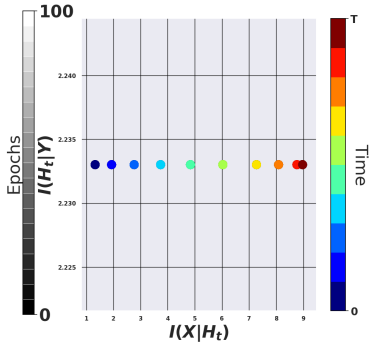
(a) The LSTM layer's Information Plane for the IMDB task trained in a multi-task framework.



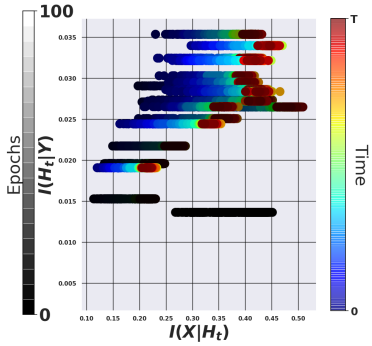
(b) The LSTM layer's Information Plane for the RT task trained in a multi-task framework.



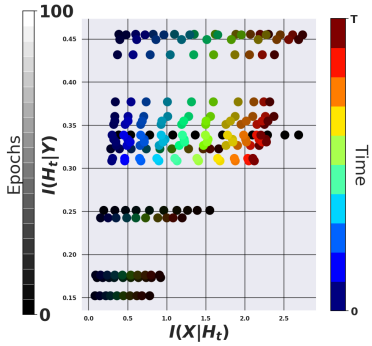
(c) The first Hidden layer's Information Plane for the IMDB task trained in a multi-task framework.



(d) The first Hidden layer's Information Plane for the RT task trained in a multi-task framework.



(e) The Output layer's Information Plane for the IMDB task trained in a multi-task framework.

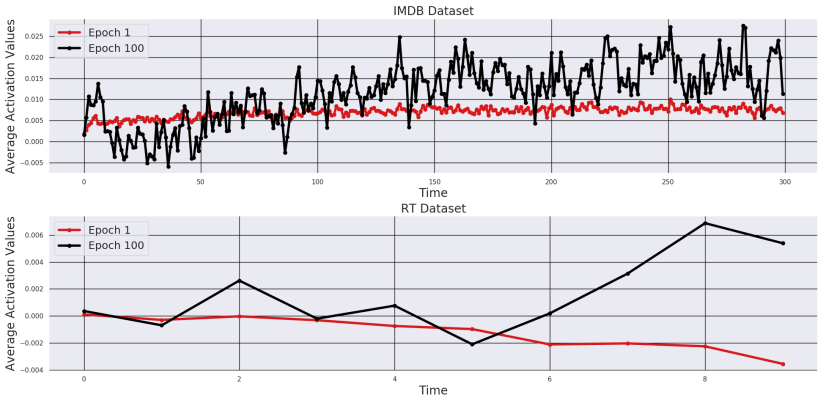


(f) The Output layer's Information Plane for the RT task trained in a multi-task framework.

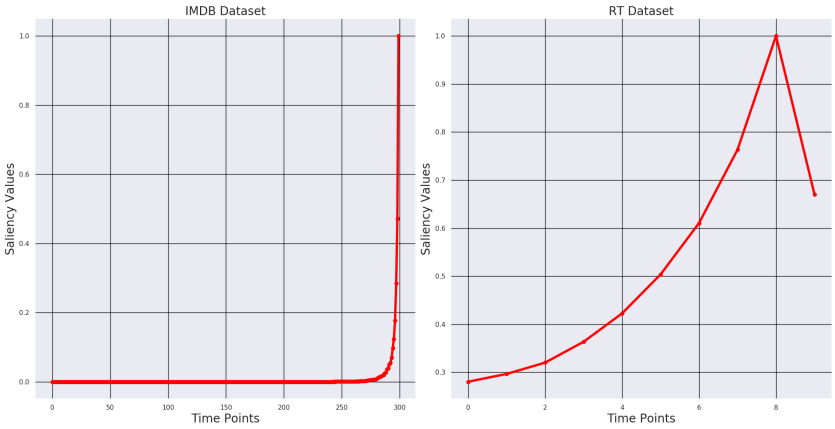
Figure 2: Information Plane for the Multi-Task Experiment over the first 100 epochs. A single-layer LSTM was trained to classify random noise versus a linear trend over a 100 epoch training period.

B Comparitive Methods

In figure 3 we provide two other comparitive methods for model introspection. Panel 3a provides the averaged activations at the first and last training epochs, and panel 3b provides saliency. At the end of training, both measures provide a general increase over the sequence of LSTM modules.



(a) Averaged activations for the mult-task paradigm for the IMDB and RT sentiment analysis tasks. The y axis plots the averaged activations, and the x axis plots the corresponding time point in the LSTM. The red curve plots the activations at epoch 0, and the black curve plots teh activations at the last epoch.



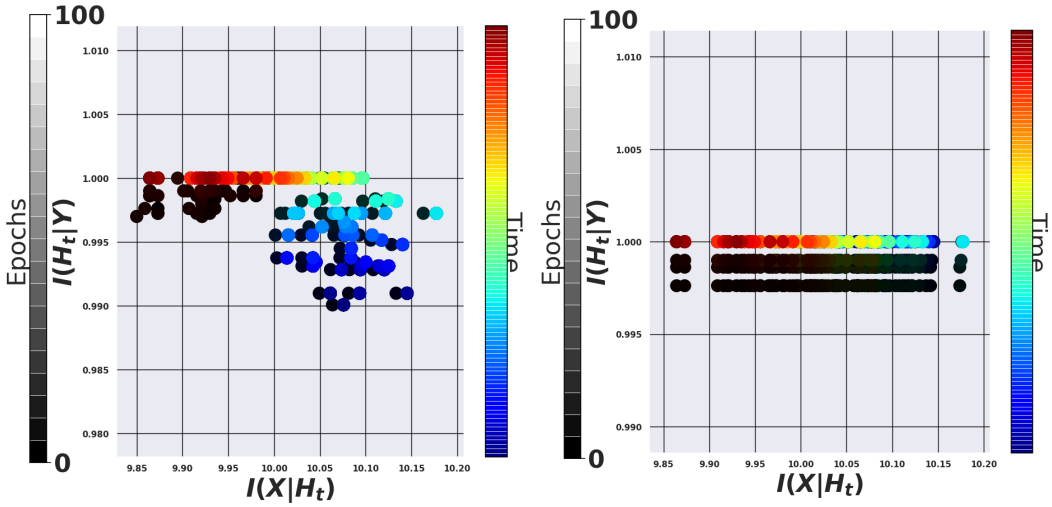
(b) Saliency for the mult-task paradigm for the IMDB and RT sentiment analysis tasks. The y axis plots the saliency, and the x axis plots the corresponding time point in the LSTM.

Figure 3: Other introspection methods which we implemented for comparison on our architecture.

C Additional Results

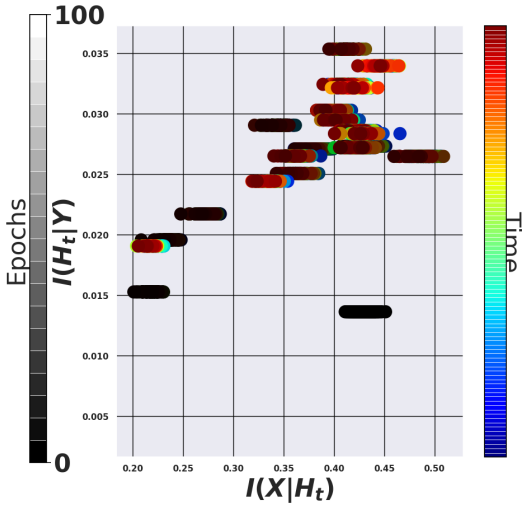
C.1 Closer Analysis

In figure 4, we plot the information plane for the last 100 timepoints in the IMDB task. Panel 4a provides the information plane for the LSTM layer, panel 4b provides the information plane for the hidden layer, and panel 4c provides the information plane for the output layer.



(a) The LSTM layer's Information Plane for the IMDB task trained in a multi-task framework.

(b) The first Hidden layer's Information Plane for the IMDB task trained in a multi-task framework.



(c) The Output layer's Information Plane for the IMDB task trained in a multi-task framework.

Figure 4: Information Plane for the Multi-Task Experiment over the first 100 epochs. A single-layer LSTM was trained to classify random noise versus a linear trend over a 100 epoch training period with a learning rate of 0.0001. In all panels, we only plot the information for the last 100 timepoints.

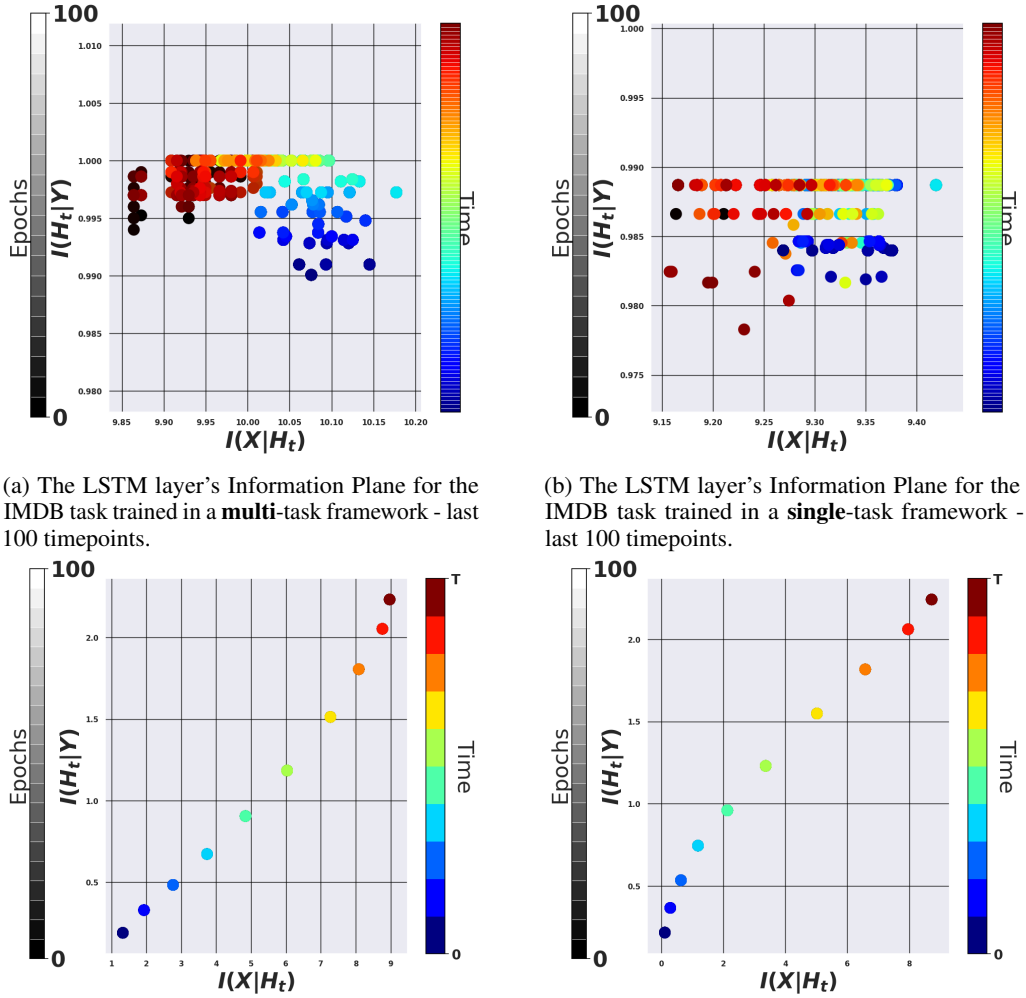
C.2 Single-Tasks

In figure 5, we compare the multi-task analysis framework with the single-task analysis using Information-Bottleneck analysis. For the IMDB task, comparing the multi-task information (panel 6b) and the single-task information (panel 6a) shows increased label representation in the last 100 timepoints of the multi-task LSTM, while the single-task LSTM loses some information in the later timepoints, along with an overall lower feature and label representation.

Conversely, for the RT task the multi-task LSTM (panel 5c) shows lower label representation in the majority of modules, but higher feature representation across modules than the single-task case (panel 5d).

These results speak to intuitions about multi-task learning. Namely, we see an increase in label information for the IMDB task, and feature information for both tasks, perhaps indicating that the additional features learned for the RT task provide insight for predicting the IMDB task. At the same

time, the multi-task framework does lose some information around the label space for the RT task, perhaps indicating an asymmetric relationship in the multi-task framework.



(a) The LSTM layer's Information Plane for the IMDB task trained in a **multi**-task framework - last 100 timepoints.

(b) The LSTM layer's Information Plane for the IMDB task trained in a **single**-task framework - last 100 timepoints.

(c) The LSTM layer's Information Plane for the RT task trained in a **multi**-task framework.

(d) The LSTM layer's Information Plane for the RT task trained in a **single**-task framework.

Figure 5: Information Planes for the multi-task and single-task experiments trained over 100 epochs with learning rate of 0.001.

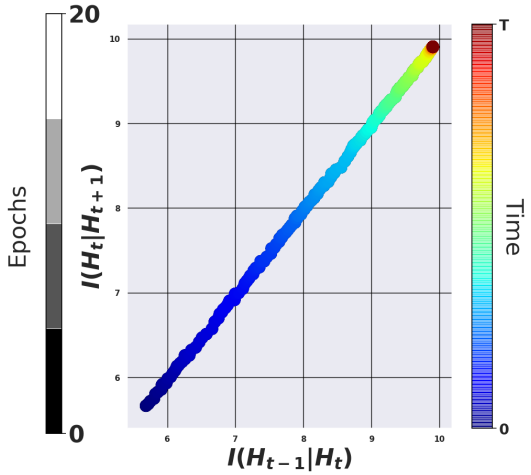
C.3 Inter-Layer Information For Real Data

In figure 6, we provide the inter-layer information for the single-task IMDB and RT data sets. Due to time constraints, we will leave the multi-task inter-layer information for future work. We see for both tasks that the story is relatively straightforward: information with adjacent timepoints increases linearly in sequence with the LSTM modules. This result departs from our findings with the simulated data, which may be attributed to the complexity of the real data.

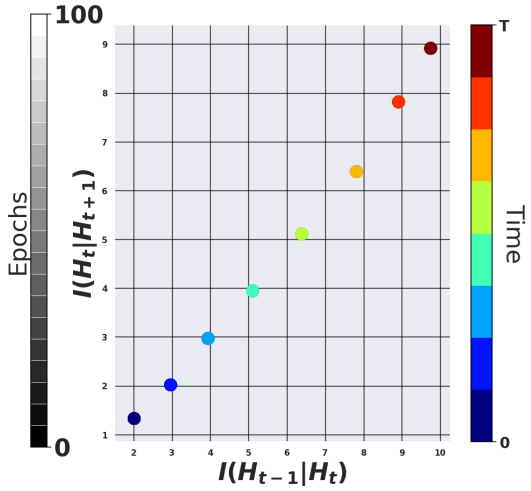
C.4 Increased Learning Rate

In figure 7, we compare how the information in each of the layers of our multi-task model changes with adjusting the learning rate. In panels 7a, 7c, and 7e, we use a learning rate of 0.001, and in panels 7b, 7d, and 7f, we use a learning rate of 0.0001.

We observe a loss in label information in later epochs, especially in the output layer for the larger learning rate, perhaps indicating overfitting. Interestingly, the LSTM and hidden layers show similar



(a) Inter-Layer information plane for the IMDB task.



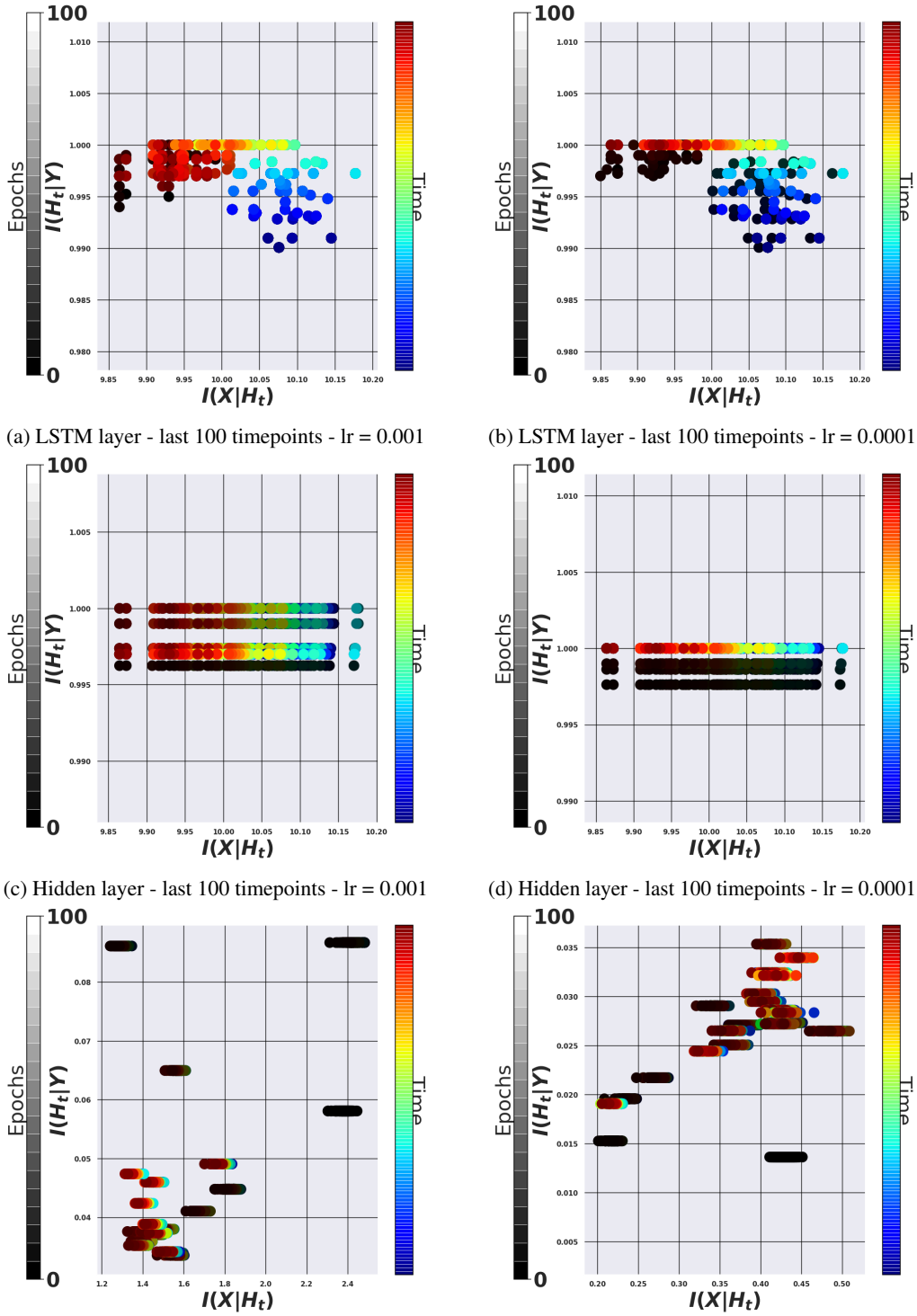
(b) Inter-Layer information plane for the RT task.

Figure 6: Inter-Layer information for the real-data task. The models were trained on single tasks for as 20 and 100 epochs respectively with a learning rate of 0.001. Information shared with the previous timepoint is plotted in the x-axis, and information shared with the next timepoint is plotted in the y-axis.

distributions of feature representation regardless of the learning rate, and the label representation in earlier timepoints of the LSTM layer is nearly identical for both learning rates.

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material, and will not necessarily be considered during the review process. Therefore, authors should make sure that their main (4 page) paper submission is self-contained.



(a) LSTM layer - last 100 timepoints - lr = 0.001

(b) LSTM layer - last 100 timepoints - lr = 0.0001

(c) Hidden layer - last 100 timepoints - lr = 0.001

(d) Hidden layer - last 100 timepoints - lr = 0.0001

(e) Output layer - last 100 timepoints - lr = 0.0001

(f) Output layer - last 100 timepoints - lr = 0.001

Figure 7: Caption