USING LSTMS TO CREATE GENERATIVE MODELS FOR PREDICTING FMRI TIME SERIES

May 2, 2019

Hendrick Townley
Yale University
Department of Computer Science
Department Advisor: Professor Steven Zucker
De Facto Advisor: Professor John Murray

Special thanks to Max Shinn for his very generous time and advice throughout the process of building and researching this project.
Contents

1 Background and Overview 3

2 Description 4

3 Implementation 5
   3.1 Libraries ................................................................. 5
   3.2 Vanilla LSTM ......................................................... 5
   3.3 Discretized Probability Distribution LSTM .......................... 5
   3.4 Overfitting .............................................................. 6
   3.5 Evaluation ............................................................... 6
   3.6 Data Pipeline .......................................................... 7

4 Results and Examples 7
   4.1 Vanilla LSTM ............................................................ 7
   4.2 Discretized Probability Distribution LSTM .......................... 9
   4.3 Varying the Number of Bins .......................................... 12
   4.4 Varying Number of Units .............................................. 13
   4.5 Analysis of fMRI Data: Comparison of Regions with Similar and Different Functionality ........................... 14
   4.6 Discussion of Overfitting with fMRI ................................. 19

5 Conclusion and Future Work 21
Abstract

Despite significant existing analysis of fMRI data sets representing human brain activity, there still exist large gaps in our understanding of them. For example, we still do not have a full understanding of the temporal properties of the BOLD signal represented by fMRI time series. This research project investigates the feasibility of using LSTMs to construct a generative model trained on and predicting fMRI signals of brain regions. Altogether, we successfully constructed and trained a discretized probability distribution (DPD) LSTM architecture capable of producing stable models, which accurately reproduce the autoregressive (AR) and moving average (MA) properties of the original time series used for training. This was accomplished both for artificially generated ARMA time series and fMRI scans from various isolated human brain regions. The output of the DPD-LSTM model is bounded, ensuring stability, and probabilistic, aiding in the capture of the stochastic nature of brain activity. Time series generated by “vanilla” LSTM models, on the other hand, which predict the next timestep directly, were prone to exploding or collapsing to zero. Of the architectures tested, two-layer DPD-LSTMs with 10 to 20 bins and 200 to 300 total units were the most effective. The networks tested trained very quickly and significant precautions had to be taken to avoid overfitting. We are hopeful that the success of this DPD-LSTM model will lead to the future success of similar models incorporating data from many different regions of the brain at once and aiding in the discovery of new insights into the structural and functional connectivity of the brain.

1 Background and Overview

Advances in the application of deep neural networks in recent years have proven successful in analyzing large data sets with complex feature spaces. Even though these methods have achieved widespread success only within the last half decade, there are many fascinating questions in a variety of scientific disciplines that neural networks can help us investigate.

Studying time series in the context of computational neuroscience is one such example. In particular there exist rich data sets collected by means of Functional Magnetic Resonance Imaging (fMRI) representing the brain activity associated with changes in blood flow (Ogawa und Sung (2007)). Although there are instances in which neural networks have been applied to fMRI data for purposes of classification (Dvornek u. a. (2017)), there is little to no work which uses neural networks to create generative models of fMRI time series.

The Long Short Term Memory (LSTM) network is an obvious candidate for this analysis (Hochreiter und Schmidhuber (1997)). Similar to Recurrent Neural Networks (RNNs) from which they are adapted, LSTMs are capable of analyzing signals that change over time thanks to their recurrent structure which allows information to persist in the network. Unlike RNNs, LSTMs include additional cell structures allowing the network to effectively “remember” high value past information while “forgetting” low value information over long periods of time.
It is unclear whether existing fMRI data sets are sufficient in size for data intensive deep network methods. In order to construct the generative model, I will train the network on a data set which includes approximately 300 brains. There are 360 brain regions associated with each scan and 1350 time samples were collected for each region. The data set includes fMRI data of “healthy” participants with no known disorders.

Despite the significant previous analysis of these data sets using other methods, there still exist large gaps in our understanding. For example, we still do not fully understand the temporal properties of the BOLD signal represented by fMRI time series. We are hopeful that a successful neural network model will shed light on this lacuna and others.

2 DESCRIPTION

This research project seeks to determine whether we can train an LSTM network using existing fMRI data to develop a useful generative model. In other words – can we train an LSTM on a certain number of points in a particular time series in order to predict any of the remaining timesteps? Moreover, do the time series generated by the trained model mimic any of the underlying properties of the original data set?

The first models developed for this purpose involved a very straightforward LSTM architecture that will hereafter be referred to as the vanilla LSTM model. The vanilla LSTM attempted to directly predict the value of the next timestep of a given time series. The vanilla LSTM proved to be unworkable given the instability of the resulting generative model. Any lengthy prediction more than a few hundred timesteps in length resulted in values exploding towards infinity or collapsing to zero. Although stability improved dramatically with multiple layers, this change did not prove sufficient.

The instability of the vanilla LSTM motivated the construction of a second more sophisticated architecture – a discretized probability density LSTM (DPD-LSTM). Instead of predicting a single value the DPD-LSTM model predicted a bounded and discretized probability distribution for the next step of a given time series (for similar architectures see (Flunkert u. a. (2017)) and (Yeo u. a. (2018))). Models trained from the DPD-LSTM architecture were robust and stable, even after thousands of predictions, and proved capable of accurately reflecting autoregressive and moving average components of the original data.

The success of the DPD-LSTM architecture motivated its application to time series of fMRI data from various regions of the human brain.
3 IMPLEMENTATION

3.1 Libraries

The LSTM networks developed over the course of this research project were written using the Python programming language and Keras, a popular API for building neural networks built on top of the Tensorflow library. Keras offered a sufficient level of customization for the architectures developed – including experimentation with custom loss, training, and output postprocessing procedures.

3.2 Vanilla LSTM

The vanilla LSTM consisted of a very straightforward and familiar neural network architecture. One or two LSTM layers, each with the same number of units, preceded a final dense layer with one unit. The input size of the first LSTM layer was equivalent to the length of the time series being used as training samples.

The samples used for training could be collected from a time series of any arbitrary length. Supposed the LSTM model expected samples of size \( n \). A time series of length \( m \) could be split into \((m - n)\) training samples each of length \( n \). Each possible group of \( n \) consecutive timesteps would serve as the training data, \( X \), while the \((n + 1)^{th}\) timestep would serve as the target data, \( y \). Mean square error was used as the loss function.

A trained model could then be used to generate a time series of arbitrary length as follows. The model is provided with “seed” data of length \( n \) to initialize its predicted time series. Given the 1\(^{st}\) through \( n^{th}\) timesteps as input the model predicts the \((n + 1)^{th}\) timestep. The model then uses the 2\(^{nd}\) through \((n + 1)^{th}\) steps to predict the \((n + 2)^{th}\) step and so on and so forth.

The models constructed used an input of size \( n = 10 \).

3.3 Discretized Probability Distribution LSTM

The DPD-LSTM model trains and predicts in a process that is fundamentally similar to that described above for the vanilla model with several important modifications.

First, the output of the network must be equal to the number of bins in the discretized probability distribution function. Suppose we are trying to predict a probability distribution function discretized into \( b \) bins. In this case the last dense fully connected output layer of the DPD-LSTM must have \( b \) units as well (not just 1). This is because the DPD-LSTM does not attempt to predict the next time step itself but rather the probabilities that the time step will have particular values (the possible values have been discretized into \( b \) buckets).
Second, for training we must modify the target data to make each target timestep value directly comparable to the predicted probability distribution. To do so we will convert each target value into a discretized delta function. Instead of one target value there will be a target vector of length $b$ representing each of the bins. The bin containing the original target value will have value 1 whereas all other bins will have value 0. This new target vector can now be directly compared to the predicted probability distribution using various loss functions. This conversion is completed as a preprocessing step by the provided code before training for each model begins. While training the model seeks to maximize the likelihood that its chosen bucket contains the true target value.

Finally, for prediction using the resulting generative model we choose the next timestep probabilistically based on the probability distribution returned for each timestep. Choosing values probabilistically from a bounded probability distribution will be helpful in capturing the stochastic nature of the brain signals.

### 3.4 Overfitting

Precautions needed to be taken in every case to prevent the model from overfitting the provided data.

In the case of the ARMA models this was accomplished by generating a new training sample of size $n$ for each epoch. This means that during training the model was never exposed to the same time series twice.

In the case of the fMRI models an analogous methodology was used such that each fMRI scan was used during only one training epoch. Given a specific brain region, a list of every scan of that region across all subjects and samples was curated and then shuffled randomly. The model was fitted using a unique scan for every epoch of training until there were no more scans remaining. As a result the model was never exposed to the same the same fMRI scan twice. Fortunately, this was reasonable given the quantity of fMRI data available.

### 3.5 Evaluation

To evaluate the success of each model, AR and MA models were fitted to both the original and newly predicted time series. For any given set of parameters, 100 unique LSTM models were trained from scratch independently of one another. Then, each of the 100 models was used to generate 10 independent time series with 5000 timesteps each. This resulted in 1000 time series total for any given set of LSTM parameters. 1000 time series of the original training data were also prepared to serve as comparison. Then AR and MA models were fitted to each time
series in both sets. The coefficients found for each group of time series were compared to one another either by (1) taking the difference between the means for each set of coefficients or (2) finding the \( p \)-value for both sets of coefficients. Lower values for the first metric and higher values for the second metric indicated a good model.

To aid in the development and testing of potential LSTM architectures that could serve as successful generative models, artificially generated ARMA data first served as the “original data” as described above. This data had the distinct advantage of having a definite ground truth – each coefficient could be explicitly specified in the training data used.

3.6 Data Pipeline

Tools to streamline the model training and data analysis processes were written from scratch as needed.

Code was written to automatically save the model, create multidimensional visual plots of training progress and model performance, and to fit ARMA models to predicted time series at the conclusion of each successfully trained LSTM model. This functionality was abstracted into a python function library and could be applied as needed to any training script.

Scripts were written to allow for submitting jobs with various parameters in bulk to the Grace high performance computer available to Yale researchers. Other scripts parsed output summarizing the fitted coefficients of the ARMA model and consolidated the relevant data for convenient entry into spreadsheet software for analysis.

4 Results and Examples

4.1 Vanilla LSTM

The first architecture tested as a candidate for a generative LSTM model was the vanilla LSTM. In its simplest form the vanilla LSTM was very unstable and unable to accurately reproduce time series of any length without collapsing to zero or exploding as shown in the figures below.
Figure 1: Time series generated by a one-layer vanilla LSTM network with the following parameters. AR(1) = [0.25], input_size = 20, epochs = 500, n_training_samples = 1000. The values predicted by the trained model explode to $10^{23}$ after only 500 epochs. Blue indicates original data generated by the AR model. Orange represents predicted data generated by the trained LSTM model.

Adding multiple layers to the architecture was enough to improve the stability of the resulting architectures dramatically. However, these models were still not robust enough over the long term. As demonstrated in the figure below, the time series generated by multi-layer vanilla LSTM models collapsed to zero given enough time.

Figure 2: The time series generated by a trained multi-layer vanilla LSTM model collapses after a little more than 10,000 timesteps. (Most other models collapsed much sooner than this.) Parameters: AR(1) = [0.25], input_size = 10, epochs = 1000, n_training_samples = 10,000, units_per_layer = 50. Blue indicates original data generated by the AR model. Orange represents predicted data generated by the trained LSTM model.

Moreover, although the two-layer vanilla LSTM network was successful in producing time series, which reproduced the autoregressive properties of an original AR(1) time series, it was unable able to do so for higher order models. These failings made the vanilla LSTM architecture unworkable for our purposes of training a stable, accurate generative model.
4.2 Discretized Probability Distribution LSTM

The DPD-LSTM architecture proved to be dramatically more effective at producing models, which accurately reflected the underlying properties of the original training data. The time series generated by trained models were necessarily stable due to the bounded probability distribution returned by the model at each time step ($[-5, 5]$ in our case as can be seen with the generated data depicted in orange in Figure 3).

![Figure 3: In blue, the original AR(10) training data with coefficients provided in Table 1. In orange, the time series generated by the trained DPD-LSTM model with input_size = 10, n_training_samples = 1000, units_per_layer = 125, and n_bins = 11.](image)

The models trained using the DPD-LSTM were capable of generating time series of indeterminate length, which proved to be convincing in comparison to the original training data.

To evaluate the relative effectiveness of models trained with various starting parameters, we trained the DPD-LSTM model on time series generated by a moderately complex AR(10) model. After training was complete for a given DPD-LSTM model, AR(10) models were fitted both to time series generated by the DPD-LSTM model and to the original time series used for training. Comparing the coefficients found in each case provided us with a sense for how well the DPD-LSTM was learning properties inherent in the training data. The particular AR(10) model used for training data and fitting in each case is detailed by Table 1 below.

<table>
<thead>
<tr>
<th>Coefficient #:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value:</td>
<td>0.50</td>
<td>0.05</td>
<td>0.20</td>
<td>0.0</td>
<td>0.0</td>
<td>0.10</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.10</td>
</tr>
</tbody>
</table>

*Table 1: The values for each of ten coefficients of the AR(10) model used to generate training data for the DPD-LSTM.*
Below is a summary of the data collected for just one set of parameters: 11 bins and 100 units per layer (or 200 units total). For the two metrics used to judge the effectiveness of the trained models (see Evaluation section), (1) smaller differences between the means of the fitted coefficient distributions and (2) higher \( p \)-values, were considered to indicate better models. The ideal trained model would generate time series with AR coefficients identical to that of the original data used for training after fitting both with an AR(10) model.

**Figure 4:** Distributions of coefficient values fitted to both training data and generated data (11 bins, 100 units per layer (200 total)). Each plot represents the coefficient values fitted for one of the ten coefficients. Each point represents the coefficient value for a single, unique time series generated by one of the models. Blue indicates the original data. Orange represents the generated prediction. Notice that both distributions overlap relatively closely for each of the ten coefficients.
**Figure 5:** A different representation of the same data depicted in Figure 4. The only difference is that the data points are spread horizontally over a dummy dimension represented by the x axis for better visualization. Each group of ten consecutive points corresponds to a single model (since ten time series were generated from each unique model (see the Evaluation subsection earlier)). Again note that the distributions overlap closely for each of the ten coefficients.

Figure 5 makes it easy to visualize that the time series generated by the DPD-LSTM models have AR properties that are good approximations to that of the training data if not perfect. In Table 2 below we see that the means of both distributions never differ by more than 0.03 (and often by much less). The $p$-values are very low, but this is not surprising considering that the original training data represents a perfectly generated AR time series.
### Table 2

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
<th>five</th>
<th>six</th>
<th>seven</th>
<th>eight</th>
<th>nine</th>
<th>ten</th>
</tr>
</thead>
<tbody>
<tr>
<td>original data:</td>
<td>0.49991</td>
<td>0.04952</td>
<td>0.19965</td>
<td>-0.00036</td>
<td>0.00035</td>
<td>0.09979</td>
<td>0.00002</td>
<td>0.00025</td>
<td>0.00002</td>
<td>0.09910</td>
</tr>
<tr>
<td>predicted:</td>
<td>0.47683</td>
<td>0.05226</td>
<td>0.19080</td>
<td>0.00916</td>
<td>0.01415</td>
<td>0.07113</td>
<td>0.00966</td>
<td>0.00042</td>
<td>0.01863</td>
<td>0.08921</td>
</tr>
<tr>
<td>Diff in mean:</td>
<td>0.02309</td>
<td>0.00274</td>
<td>0.00886</td>
<td>0.00952</td>
<td>0.01380</td>
<td>0.02866</td>
<td>0.00964</td>
<td>0.00016</td>
<td>0.01861</td>
<td>0.00988</td>
</tr>
<tr>
<td>P value</td>
<td>1.07E-06</td>
<td>6.41E-03</td>
<td>1.56E-24</td>
<td>9.49E-27</td>
<td>1.24E-65</td>
<td>4.96E-192</td>
<td>3.42E-37</td>
<td>0.82E-1</td>
<td>5.65E-114</td>
<td>8.76E-42</td>
</tr>
</tbody>
</table>

The first two rows represent the mean value for each coefficient found from the original time series and the generated prediction respectively. The third row represents the difference between those two means. The fourth row represents the \( p \)-value of both distributions of coefficient values.

Altogether the models trained from the DPD-LSTM architecture are both stable and accurately reflect the AR properties of the original training data.

### 4.3 Varying the Number of Bins

To determine whether or not the number of bins of the estimated probability distribution had a significant influence on model quality, models with identical parameters, except with varying number of bins, were trained. Among these models, the number of bins was varied granularly between 3 and 31 bins and more coarsely between 31 and 101 bins.

![Figure 6: The difference in means for each of the ten coefficients (y axis) plotted as a function of the number of bins (x axis).](image)

Figure 6: The difference in means for each of the ten coefficients (y axis) plotted as a function of the number of bins (x axis).

![Figure 7: The \( p \)-values for each of the ten coefficients (y axis) plotted as a function of the number of bins (x axis).](image)

Figure 7: The \( p \)-values for each of the ten coefficients (y axis) plotted as a function of the number of bins (x axis).
These plots together indicate that model performance tends to suffer for small numbers of bins (fewer than 11) and increasingly large numbers of bins (greater than 21). It is to be expected that for very small bin numbers the model output is too coarse to approximate the original data. The same is true for very large bin numbers as the output grows in complexity and begins to approximate the original vanilla model in certain ways.

Interestingly the optimal number of bins is actually not consistent among different coefficients. In particular the behavior of coefficient nine deviated distinctly from the rest in ways that are not immediately obvious.

4.4 Varying Number of Units

To determine whether or not the number of units in the network had a significant influence on model quality, models with identical parameters, except with varying number of total units, were trained. Among these models, the number of units was varied uniformly from 20 to 500.

![Image of plots showing the difference in means for each of the ten coefficients (y axis) plotted as a function of the total number of units in the network (x axis). One-layer model shown in blue. Two-layer model shown in orange. Notice that the two-layer model improves with smaller numbers of units much more quickly than the one-layer model for coefficients three through six.](image)

Figure 8: The difference in means for each of the ten coefficients (y axis) plotted as a function of the total number of units in the network (x axis). One-layer model shown in blue. Two-layer model shown in orange. Notice that the two-layer model improves with smaller numbers of units much more quickly than the one-layer model for coefficients three through six.
Both sets of plots indicate that two-layer DPD-LSTM networks with 200 to 400 units total seem to result in the best models. With too few units the network has trouble learning the underlying properties, but at a certain point more units no longer help.

Again, there are peculiar inconsistencies among the various coefficients. In particular the one-layer network seems to perform best in modeling the 1st, 9th and 10th coefficients while the two-layer network is superior for the rest. There is no obvious reason why this should be the case, however it may have something to do with discrepancies of the ARMA fitting function in estimating coefficients which bookend the chosen model.

4.5 Analysis of fMRI Data: Comparison of Regions with Similar and Different Functionality

The success of the DPD-LSTM architecture in modeling artificially generated ARMA data motivated its application to raw fMRI data of brain regions from various human subjects. This application proved to be similarly successful. The tables and graphs provided immediately below demonstrate that the distributions of AR and MA coefficients of the data generated by the LSTM model were similar to those distributions found by fitting AR and MA models directly to the original fMRI time series.
The accuracy of the models in reflecting the properties of the fMRI training data ranged from excellent (especially in the case of the AR models. See Table 3, Figure 10, and Figure 11) to decent (especially in the case of a handful of the MA models. See Table 4, Figure 12, and Figure 13).

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
<th>five</th>
<th>six</th>
<th>seven</th>
<th>eight</th>
<th>nine</th>
<th>ten</th>
</tr>
</thead>
<tbody>
<tr>
<td>original data:</td>
<td>0.3321</td>
<td>0.2012</td>
<td>0.1056</td>
<td>0.0433</td>
<td>0.0060</td>
<td>-0.0087</td>
<td>-0.0060</td>
<td>-0.0138</td>
<td>-0.0014</td>
<td>0.0003</td>
</tr>
<tr>
<td>predicted:</td>
<td>0.3226</td>
<td>0.2068</td>
<td>0.1020</td>
<td>0.0413</td>
<td>0.0076</td>
<td>-0.0052</td>
<td>-0.0095</td>
<td>-0.0063</td>
<td>-0.0043</td>
<td>-0.0006</td>
</tr>
<tr>
<td>diff in means:</td>
<td>0.0095</td>
<td>0.0056</td>
<td>0.0037</td>
<td>0.0019</td>
<td>0.0016</td>
<td>0.0035</td>
<td>0.0035</td>
<td>0.0075</td>
<td>0.0029</td>
<td>0.0009</td>
</tr>
<tr>
<td>p-value:</td>
<td>0.1516</td>
<td>0.0001</td>
<td>0.9912</td>
<td>0.5237</td>
<td>0.5300</td>
<td>0.4214</td>
<td>0.1513</td>
<td>0.0147</td>
<td>0.3597</td>
<td>0.7506</td>
</tr>
</tbody>
</table>

**Table 3:** This table is analogous to Table 2. Coefficients values were collected by fitting an AR(10) model to time series generated from the DPD-LSTM trained on region 186 (VIS - visual functionality) and to the fMRI data itself. Each column represents a different coefficient.

**Figure 10:** A plot analogous to that depicted in Figure 4 previously. Coefficients distributions were collected by fitting an AR(10) model to time series generated from the DPD-LSTM trained on region 186 (VIS - visual functionality) and to the fMRI data itself. Each column represents a different coefficient. Notice that the predicted distributions fall within the distributions of coefficients fitted to the original data, which has greater variance in this case unlike the original figure.
Figure 11: A plot analogous to that depicted in Figure 5 previously (i.e. we see the same distribution as in Figure 10 but extended over a dummy dimension on the x axis). Notice the significant overlap of both distributions.

Table 4: This table is also analogous to Table 2. Coefficients values were collected by fitting an MA(10) model to time series generated from the DPD-LSTM trained on region 64 (DMN - default mode network functionality) and to the fMRI data itself. Each column represents a different coefficient. Notice that the difference in means are higher and the p-values are lower than in the previous table.
**Figure 12:** Another plot analogous to that depicted in Figure 4. Coefficients distributions were collected by fitting an MA(10) model to time series generated from the DPD-LSTM trained on region 64 (DMN - default mode network functionality) and to the fMRI data itself. Each column represents a different coefficient. Notice that the distributions in this case have much greater variance than those in the previous two figures.

**Figure 13:** Another plot analogous to that depicted in Figure 5. (i.e. we see the same distribution as in Figure 12 but extended over a dummy dimension on the \(x\) axis). Again notice the much greater variance.

In the plots below we compare the AR and MA components of brain regions with similar and varying functionality. Each line corresponds to the predicted coefficients of a DPD-LSTM
time series trained using fMRI scans from a particular brain region. Note that the VIS-4 region (depicted in purple) is present in all four of the plots.

**Figure 14:** Plotting mean fitted AR coefficient value (y axis) as a function of coefficient number (x axis) for regions with visual functionality.

**Figure 15:** Plotting mean fitted AR coefficient value (y axis) as a function of coefficient number (x axis) for regions with varying functionality.
It is interesting that the AR and MA coefficients are not more similar among fMRI scans taken from brain regions with similar functionality. Clearly regions with similar functionality still have distinct temporal properties.

### 4.6 Discussion of Overfitting with fMRI

The LSTM networks were very prone to overfitting without the precautions taken as described previously in the Implementation section. To see that this is case, consider the loss history for a vanilla LSTM, which is exposed to the same 10,000 training samples over the course of 1000 epochs (see Figure 18), and the loss history for the DPD-LSTM, which for which a fresh set of ARMA time series training data is generated anew for every epoch (see Figure 19).
Figure 18: Loss history collected while training one-layer "vanilla" model with parameters: \( \text{AR(1)} = [0.25] \), input\_size = 5, epochs = 1000, n\_training\_samples = 10000, n\_per\_layer = 50. The extremely low error is indicative of overfitting.

Figure 19: Loss history collected while training a two-layer DPD-LSTM model depicted in Figure 18. (Parameters: input\_size = 10, n\_training\_samples = 1000, units\_per\_layer = 125, and n\_bins = 11. Notice how quickly all possible training is completed.

Whereas the loss in Figure 19 falls to almost zero in less than a thousand epochs, the loss in Figure 18 drops just a hair after less than 250 epochs.

Despite the precautions taken, there is still a chance that the success of the DPD-LSTM could be due to its memorizing common sequences within the fMRI data. To determine whether this was the case, 300 scans were left out of training during one training session. Then the distribution of log likelihoods for each time series were compared for the trained and untrained sets of scans.
Figure 20: After log-likelihood values were calculated for each of the 1350 scans associated with a single brain region, the values were discretized into buckets of size ten. The number of values to appear in each bucket are plotting above. Notice that the two distributions are very similar.

The similarity of distributions indicate that it is unlikely any overfitting is occurring. The $p$-value for both distributions is very high: 0.411, and the mean values are very close: the difference between the two was 2.52.

5 Conclusion and Future Work

Altogether we successfully constructed and trained a discretized probability distribution (DPD) LSTM architecture capable of producing stable models, which accurately reproduce the AR and MA properties of the original time series used for training. This was accomplished both for artificially generated ARMA time series and individual fMRI scans from various isolated human brain regions.

These models indicate that relatively little training time is needed to successfully train models, but precautions must actively be taken to avoid overfitting. Moreover, existing fMRI data sets are sufficient in size for certain meaningful analysis using neural networks such as LSTMs. Of the architectures tested, two-layer DPD-LSTMs with 10 to 20 bins and 200 to 300 total units were the most effective.

There are many future avenues of inquiry, which this research opens up. First, we can further improve the performance of the current model by continuing to experiment with new and different parameters and network features. Second, there is analysis to be done to understand the physical interpretations of the current fMRI findings. Why do some regions have more similar temporal properties than others? Third, we can expand upon our model to incorporate data from many different brain regions at once. Significant aspects in the feature space identified by a generative model trained on and predicting fMRI signals of multiple
brain regions would aid in the discovery of new insights into the structural and functional connectivity of the brain.

REFERENCES


