Hierarchical, multi-session neural activity to behavioral latent time series decoding via Convolutional and LSTM models

Nihaar Shah (ns3413@columbia.edu), Matt Whiteway, Liam Paninski

December 23, 2020

1 Introduction

This project concerns decoding the time series of neural activity captured as electrical signals from multiple trials into a latent representation of a video of mouse behavior recorded during the trial. The primary question under investigation was whether a hierarchical deep learning model consisting of session-specific layers and layers common to all sessions would improve the decoding accuracy compared to a model that simply decodes single individual sessions.

In addition, we explored two types of deep learning models to decode this time series data - 1) 1-dimensional convolution neural network for each session followed by a Multi-Layer Perceptron and 2) A Long Short Term Memory Network (LSTM), which is based on a recurrent neural network model that is capable of preserving information from a longer time horizon than a typical convolution window.

The rationale for favoring a hierarchical approach is the observation that neural activity patterns for a particular behavior could have commonalities even when captured during different trials. Hence, we could design a model that could learn the composition of representations by allowing session-specific weights to be trained in addition to weights that are common to all sessions.

The results indicated that our hierarchical model did not conclusively improve upon the single decoder model for the neural data from the two sessions that we used. The LSTM showed some improvement over the convolution-based model.

2 Results

2.1 Architecture

Figure 1 shows our hierarchical model architecture in which there are 2 input conv 1D 'heads' - one for each session for which neural activity was collected.

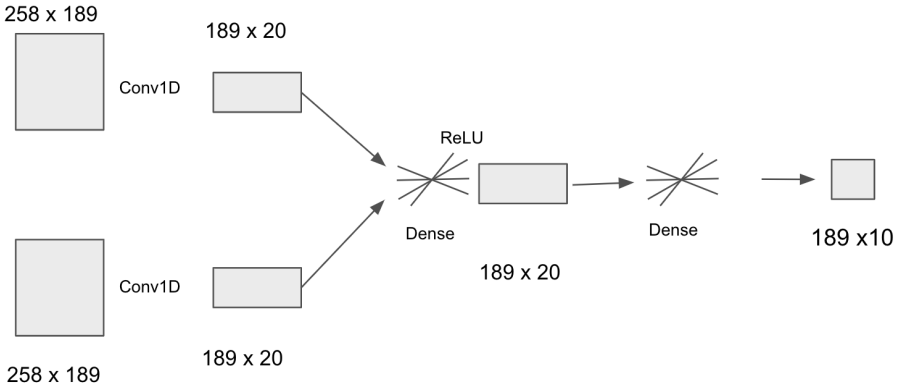


Figure 1: The Hierarchical convolutional decoding model has specialized conv weights for each session and then common general weights. It is important to note that data is passed into one of the input heads at a time, i.e., in a given pass through this model, either session 1 or 2 data is passed and those specific weights updated.

Following that, two common dense layers finally predict the behavioral latent time series of just ten dimensions compared to the higher input neural dimension.

At a given time, a batch of neural data is passed through just one of the input heads. The reason is that the target behavioral latent time series is separate for each batch so clearly, passing in two batches as input for a single target output is a mismatch.

2.2 Model Predictions

Figure 2 is a visual example of the target latent (blue) and predicted latent (black) time series of 10 dimensions. This particular batch corresponds to one with the lowest Mean Squared Error (MSE) summed across all dimensions among all test batches for the hierarchical convolutional decoder model on the session from 05-Dec. We do not display any more trace plot visualizations for brevity.

Figure 3b and 3a display the MSE for every model and each batch (a point represents a batch) from the test set from 2 sessions.

A visual inspection between the conv and LSTM model for both sessions indicates that the scatter points for the LSTM are concentrated at a lower value (hence better than conv). There seem to be a small (2 or 3) number of outliers with a higher MSE in the LSTM case.

Another comparison can be made between the single decoder (both conv and LSTM) and their hierarchical counterparts. This reveals that the hierarchical conv model does approximately the same as the conv single model. The hier-

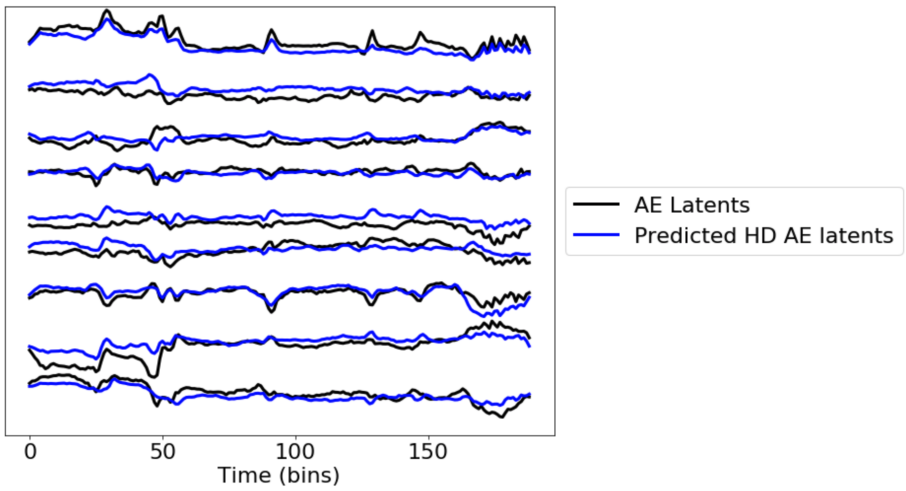
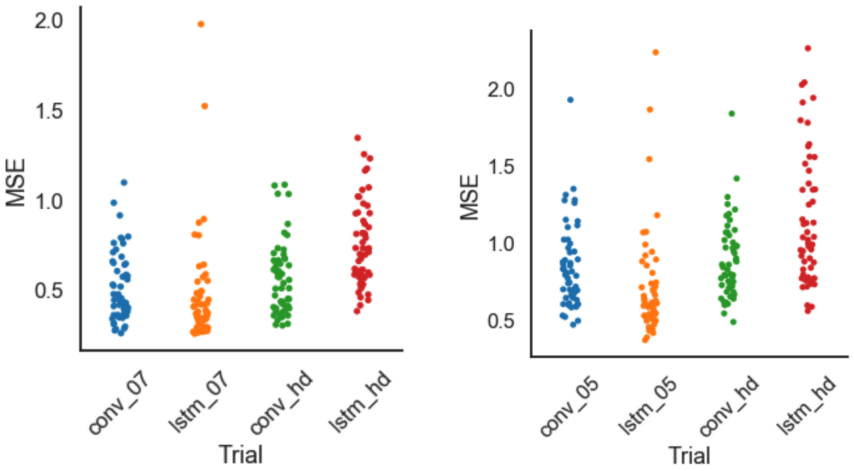


Figure 2: This is a traceplot of the true (black) and predicted (blue) behavioral latent time series. The true traceplot is obtained from a Convolutional Autoencoder (CAE) on mouse behavior videos. The blue traceplot is what our decoding model predicts.



(a) 07-Dec Session Results

(b) 05-Dec Session Results

Figure 3: MSE scatter plots for all the 4 models i.e. Hierarchical decoder Conv & LSTM, and single decoder Conv and LSTM. The left and right plots are for the 2 sessions.

archical LSTM, however, has a higher MSE by an offset than the single LSTM decoder.

Comparing the Hierarchical versions of the LSTM and hierarchical Conv decoders, we observe that conv does better, which is at odds with the observation that the LSTM did better in the single decoder case.

3 Methodology

3.1 Data

Neural Activity data: was collected from a head-fixed mouse that behaved freely (including spontaneous manipulation of a wheel with its forelimbs). Neural activity across multiple brain structures was electrically recorded using eight Neuropixels probes.

Behavioral video: data was recorded as part of the behavenet project using a single camera; gray-scale video frames were down-sampled to 112x192 pixels. Data consists of 96k frames (40 Hz frame-rate), and "trials" were arbitrarily defined as blocks of 1000 frames. Neural activity was binned at the video frame rate.

Behavioral latents: The aforementioned behavioral videos were compressed with a convolutional autoencoder (CAE), yielding a low-dimensional continuous representation of behavior. The CAE architecture is fixed for all datasets, except for the number of latent. The CAE was trained by minimizing the mean squared error (MSE) between original and reconstructed frames. Further training details are available in the behavenet paper (Batty et al. 2019).

Before going into the details of the hyper-parameters of each hierarchical model, here is a summary of the two broad models used in terms of the various layer dimensions:

MODEL	Neural Dim	MLP1 Output	Layers	Hidden Units	Target Latents
LSTM	258 or 259	50	2	20	10
Conv	258 or 259	-	2	20	10

3.2 Single session Conv1D decoding

Figure 4 represents the single session convolution-based model. The 1D convolution offers a fixed-sized window around neural dimensions for each time point. We use the following hyper-parameters in this case: $(N, C_{out}, L_{out}) = (1, 20, L_{out} = L_{in})$, kernel= $(9,)$, stride= $(1,)$, and padding= $(4,)$ although each hyper-parameter should be tuned, given access to larger compute and time resources. The interpretation of this is that we are learning 20 kernels of size $(9,)$ as features.

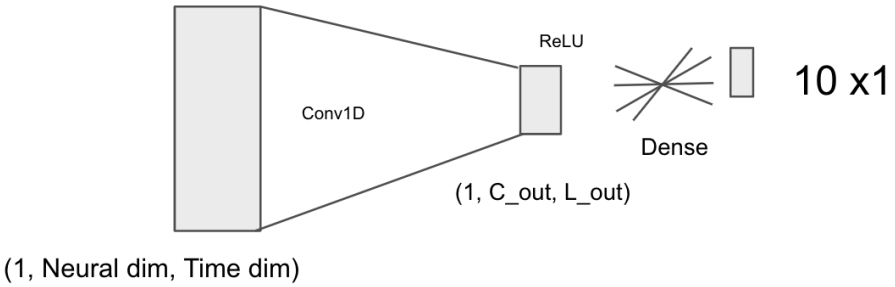


Figure 4: Single Session Conv 1D model

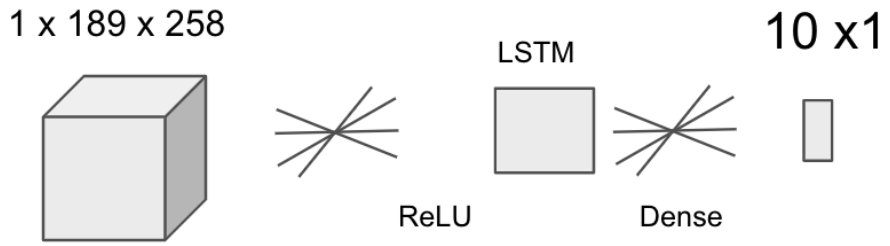


Figure 5: Single Session LSTM model:

3.3 Single session LSTM decoding

Figure 5 represents the single session LSTM based model. The LSTM consists of a cell state that carries relevant information from one time step to the next. This cell state gets regulated to preserve relevant memory and 'forget' the rest, with the help of gates. Each gate is essentially a single linear layer with a non-linear activation. A gate utilizes the previous hidden state, previous cell state, or the new sequence input. The equations governing the LSTM can be found in Hochreiter and Schmidhuber 1997.

We have a session-specific linear layer that reduces the dimensions from the session's neural input dimension to 50 dimensions. This is then fed into the LSTM. We use 20 dimensional hidden units for the LSTM, and stack 2 LSTMs i.e. have 2 layers where the second LSTM takes in the output from the first one.

3.4 Multi session Conv1D decoding

As discussed in the Results section before, Figure 1 represents the hierarchical version of the convolution based model. The major distinction between the sin-

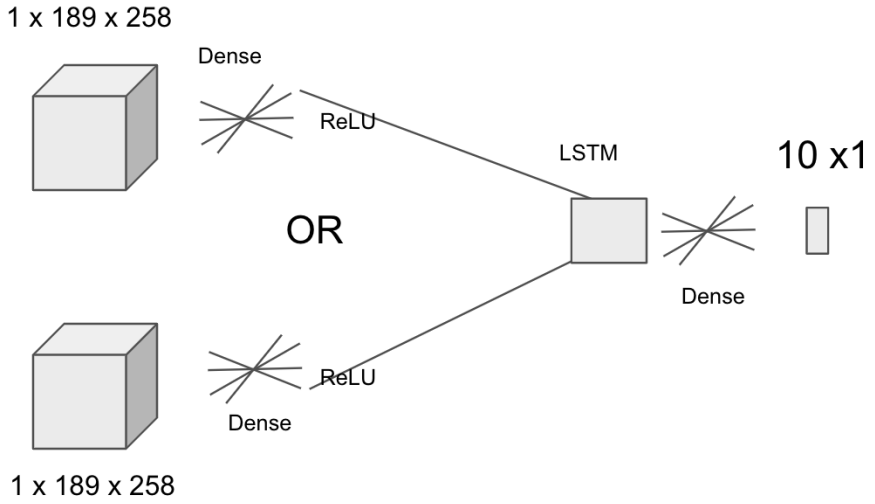


Figure 6: Hierarchical LSTM model for multi-session data.

gle session and multi session model is that there are session specific convolution layers followed by common MLP layers.

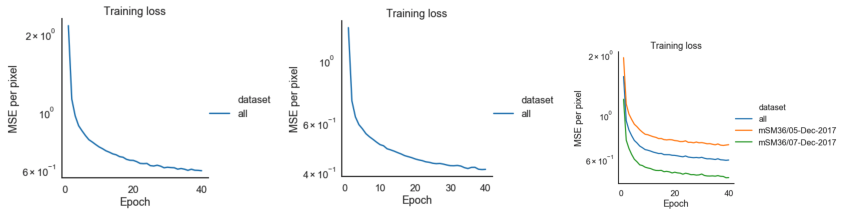
3.5 Multi session LSTM decoding

Finally, the hierarchical version of the LSTM model is shown in Figure 6. It is conceptually similar to the hierarchical conv model discussed before in that there are session specific layers, except except MLP instead of conv. The common layers involve the LSTM model discussed before and are followed by another MLP. The rationale is that the MLP that are session specific could learn how to make a smaller representation of the time-series that would be useful for the LSTM to learn common patterns from.

4 Discussion and Future Directions

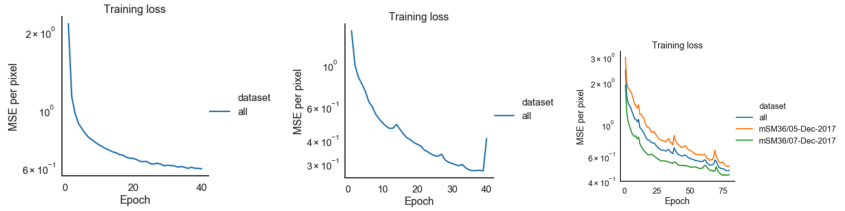
While the initial results in terms of reconstruction MSE in Figure 3b and 3a are not in favor of the hierarchical approach, this can partly be due to lack of resources to search the hyper-parameter space. One future task would be to search over the number of hidden units and the number of layers. We should then compare the best convolutional and LSTM models after finding the optimal hyper-parameters.

The single LSTM appeared to have done slightly better than its convolutional counterpart. We hope to explore a future direction to use the Sequence-to-Sequence LSTM topology, which involves an LSTM encoder and an LSTM



(a) 07-Dec (b) 05-Dec (c) 05-Dec

Figure 7: Training loss curves showing various degrees of convergence of the Conv models we trained.



(a) 07-Dec (b) 05-Dec (c) 05-Dec

Figure 8: Training loss curves showing various degrees of convergence of the LSTM models we trained.

decoder. This allows for variable length predictions and has outperformed simple LSTMs in tasks such as translation. They also provide the basis for utilizing the attention mechanism to allow the model to apply weights on the whole input sequence while decoding each time point of the output sequence. We have made an initial model of this mechanism for our time series data but due to lack of time have not concluded the results.

5 Supplementary Material

We show training loss curves for the LSTM and convolutional models in the single sessions - 05-Dec (Fig 7a, 8a) and 07-Dec (Fig 7b, 8b) as well as the hierarchical multi-session case (Fig 7c, 8c). We trained the LSTM for twice as many epochs (80 Vs 40) as compared to the rest of the models because its loss hadn't plateaued at 40 epochs. Training seems to have converged according to the plots.

References

[Bat+19] Eleanor Batty et al. "BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos". In: *NeurIPS*. 2019.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.