Using Machine Learning to Explore Hidden Behavioral States Encoded in Facial Movement Videos of Mice

Erin McCarthy *DRP 2019*

Abstract

The era of big data and machine learning has rapidly produced techniques to process and extract patterns from large amounts of data. Videos are the epitome of big data by being inherently high dimensional both spatially and temporally. In Neuroscience, murine experiments often include a camera capturing the behavior of head-fixed mice and a camera capturing the activity of the brain while performing the task. Recent work has shown the high explanatory power of linear models using the mouse's facial movements to predict brain activity while preforming a task. The goal of this work is to show the performance of linear models and recurrent deep learning models on predicting brain activity at different frame rates and using deep learning to extract a lower dimensional representation of the behavioral states of the mouse from the video for more efficient processing.

I. Introduction

From self-driving cars to augmented reality, video processing has benefited greatly from the advancements of GPUs. Imaging techniques have advanced with cheaper storage and computing power, allowing for videos to be collected and processed at a higher resolution. With increasing spatial and temporal dimensions, the data collect has grown larger and larger. The extraction of meaningful data has increased in difficulty with the growing datasets and require large amounts of computational power. It is unclear the necessary frame rate or the amount of data needed to efficiently find accurate patterns in the data. In Murine Neuroscience, the usage of machine learning techniques to process large amounts of data has only recently gained popularity. Common techniques include principle component analysis to reduce the dimensions of the videos [8], linear models to explain the variance of brain activity [4] and extracting information from the behavioral video using motion energy [8]. Where n is the number of pixels, motion energy is defined as

$$ME_i = \frac{(|frame_i - frame_{i-1}|)}{n}. \tag{1}$$

The brain is a complex organ studied across many species. Mice are popular for many neuroscience experiments because of the simplified structure that has fewer neurons and the easier access to the brain for higher resolution imaging. Although studying mice reduces the complexity of the structure of the brain, it adds additional challenges of discriminating what the mouse has learned as the task and the uninstructed behavioral aspect, such as running on the wheel or grooming, that contributes to brain activity. It is not possible to ask the mouse to stay as still as possible or to ensure that it understands the tasks. These are things that must be accounted for when extracting information from experiments. There have been recent advancements in using the behavior of the mouse to predict brain activity. Previous work has shown success in explaining 41% of variance in widefield two photon imaging using ridge regression. The work has shown that the behavior variables, such as whisker pad and the entire video motion energy, explain the greatest amount of variance and when visualized the beta weights and explained variance of some variables align with known areas of the brain, for example the beta weights of visual stimulus are highest in the visual cortex of the brain [4].

Although ridge regression has proven to be a useful for interpretability and explaining variance, linear models are known to have limits with dimensionality and extracting complex functions. There have been many recent advancements in neural networks and the accessibility of large-scale machine learning. Many types of models have shown success in extracting information from videos. LSTM networks have been known for their performance on time series data and have shown to be effective in video prediction [2] and interpretation [7]. In addition to convolutional networkings being able to interpret and classify images, it has been shown that convolutional autoencoders are capable of learning a manifold of human motion [3]

This work takes on a multidisciplinary data science problem that includes processing large datasets and designing models to explain variance in the activity of the brain. There are many challenges in working with mice data in addition to the difficulty of extracting information from videos and the size of the datasets. The experiments are run in sessions that contain a set of trials. These different sessions can have different lighting, camera angles and time length. On a biological level not all mice brains have the same structure, making datasets challenging to combine or compare. In addition to these problems, there is no ground truth. This creates a need to keep the the imaging of the brain activity at a high of resolution. This paper describes the initial work done with linear models on small datasets that inspired the continuation on larger datasets that compares the performance of ridge regression models (RRM) to LSTM networks on different frame rates, and the potential of linearly interpolated data. In addition, this work explores convolutional autoencoders as a method to compress and extract the behavioral state of the mouse from the video.

II. Methods

A. Data collection

Data for this work was collected during two different experiments. Experiment one was completed more than a year ago and experiment two started this past April and is ongoing. The first experiment collected the videos at 10hz, with the video of the mouse at a resolution of 360x412 pixels and the widefield (WF) video of the brain at a resolution of 172x130 pixels. This experiment includes 39 sessions. The second experiment has two sessions that have been collected at 30hz. The video of the mouse was expanded to capture the front paws, the snout and the jaw and was collected at a resolution of 948x700 pixel and the video of the brain was expanded to a resolution of 640x640 pixels and captures additional regions of the brain.

TABLE I: Summary of Experiment 1 and Experiment 2

	Experiment 1	Experiment 2
Number of Sessions	39	2
Pixel Dimensions of Behavior Video	360x412	948x700
Pixel Dimensions of WF Brain Video	172x130	640x640
Rate	10hz	30hz
Size of Behavior Video/ WF Brain Video	∼1GB/∼5GB	13,17GB/29,44GB

B. Data Alignment and Extraction

Fig. 1: Example image from the widefield two photon video of the brain. The front of the brain is to the right, with mainly the right hemisphere showing

Each dataset contains a widefield two photon video of the mouse's brain (Fig. 1), external variables about the task and a behavioral video of the movements of a head fixed mouse. The data collected in real time of the experiment are stored in a signal format using Spike2(Fig. 3, A). This information is necessary to align the data, including when each camera turned on, every frame collected, and the onset of the stimulus. This data also includes, when the mouse triggered the lick sensor, the pupil diameter, and wheel velocity. Events such as the onset of stimulus and the triggered lick sensor are extracted based on a threshold set by the experimentalist.
Post-processing of the behavioral video includes extracting the motion energy. In order to more efficiently extract motion energy from the videos, a script in Python was written that allows the user to draw boxes around multiple desired areas of video and draw the regions of interest in batches instead of processing one at a time(Fig. 3, B). The datasets includes a feature vector describing the state of the environment each time a frame was collected. In experiment one this includes every frame for two seconds after the onset of the stimulus and in experiment two it includes the entire session. Table 2 details the difference in variables of the experiments, and the type of the variable. The continuous variables are different frame to frame, the constant variables stay the same within trials but change over the session.

From the datasets in experiment two, three variations were created. One 10hz dataset by downsampling every 3rd frame of the original video, one 15hz dataset by taking every 2nd frame, and one linearly interpolated dataset from 15hz to 30hz. The 30hz dataset of the longer session contained 52,500 datapoints, and the shorter session contain 17,000 datapoints.

TABLE II: Variables available in each Experiment

		Exp. 1	Exp. 2
Last Stimulus Time	Continuous	X	X
Last Stimulus Type	Binary/Constant	X	X
Last Response Time	Constant	X	X
Last Response	Binary/Constant	X	X
Wheel Velocity	Continuous	X	X
Whisk ME	Continuous	X	X
Pupil Diameter	Continuous	X	X
Snout ME	Continuous		X
Jaw ME	Continuous		X
Full Video ME	Continuous	X	X

C. Training Linear Model on Experiment 1

The goal of this section of work was to create an additional data processing section with a linear model that enhanced the findings of the paper [6]. In order to fit with the rest of the work, the preprocessed video of the brain was used and was evaluated using explained variance. The video was smoothed spatially using a gaussian blur and the temporal smoothing was done using $\frac{df}{f}$(Fig. 2).

```
(length, y_dim, x_dim) = video.shape
# for spatial dimensions
for y_ind in range(y_dim):
    for x_ind in range(x_dim):
        # for each window in time
        for i in range(int(length/window)):
            video_window = video[i:(i*window+window), y_ind, x_ind]
            local_mean = mean(video_window)
            dff_video[i:(i*window+window),y_ind, x_ind] =
                (video_window-local_mean)/local_mean
return dff_video
```

Fig. 2: $\frac{df}{f}$ algorithm

These models were trained on an Intel i-7-7800X CPU with 64GB of ram. The RRM was implemented and cross validated to find the optimal penalty term, using the RidgeCV implementation in Sci-kit learn [5]. To account for overfitting and utilize the data most efficiently in the smaller datasets, 10-fold cross-validation was used to fit and evaluate the model. Additional features were created from the base features listed in Table 2. For each motion energy variable, pupil diameter and wheel velocity, an average of different time lengths was created. In order to find the most optimal features to include, a reduced model, where one of the features was shuffled throughout time, was used. The explained variance of the reduced model was subtracted from the full model's to get the contribution of the feature. For this experiment, across sessions, the optimal combination of features turned out to be an average motion energy going back three frames instead of the current values, as well as the rest of the features mentioned in Table 2. Both a label and one-hot encoding were tested out for the binary variables, and one-hot encoding ended up performing slightly better overall than encoding labels. These datasets ranged from 700 to 10000 usable data points. The method of reducing the model was used again with the final setup to determine the unique contribution of each feature. Models were also run with just a single feature, to determine the non-unique variance explained by the feature.

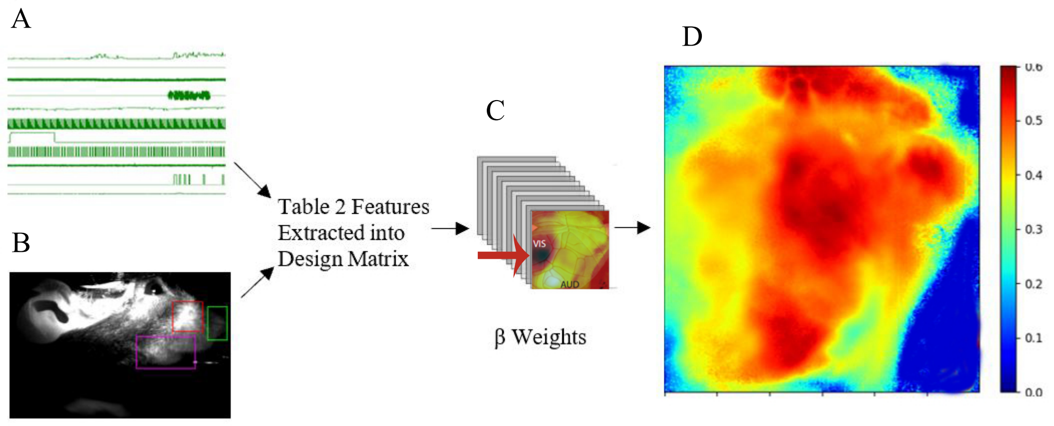

Fig. 3: Workflow of collecting data, extracting the features, fitting the RRM to get the beta weights and using the RRM to find the explained variance. A. Signal data from experiments captured in Spike2 B. Example behavioral video with regions of interest selected for ME. These inputs are used to create the design matrix to fit and test the RRM. C. Beta Weights of each feature from the model, the front is an example of the weights of the last stimulus type. The arrow highlights the visual cortex of the brain, in comparison to the bottom light spot which is the auditory cortex of the brain. D. The entire explained variance of the output of the model

D. Training Linear Model on Experiment 2

For the new experiment, the data was not preprocessed further than binning the video of the brain two by two, reducing the width and the height in half, and the other features were used in the raw form. The entire session, instead of two seconds after a trial was used for processing. In order to account for the large space requirement, the Ridge implementation was used and cross validated outside of the model's sci-kit implementation. This experiment had new features available with the larger behavioral video, including the motion energy of the snout, and jaw. To explore the impact these new features the linear model was run with out snout and jaw, and with snout and jaw. In order to be more comparable, the model was fit and evaluated with the same training and test data as the LSTM.

E. Designing Neural Networks

These models were trained on up to 3 Tesla V100s with 32GB of memory. All deep learning models were implemented using Tensorflow's implementation of Keras layers [1]. A variety of deep learning models have been shown to excel on time series data and imaging data. A few of different models that excel were implemented, including a GRU network, a LSTM network (LSTM), a convolutional neural network (CNN) and a recurrent convolutional neural network(RCNN). The CNN and RCNN took the entire binned behvioral video as input. Each model was tested with a different number of layers, batch sizes and when applicable, different sequence lengths. Although performance was considered, since there are many different hyperparameters to tune, other things were considered such as ease of use, memory usage and flexibility. The survey of these models found that LSTM and GRU networks performed similarly. The CNN showed promise but in order to get optimal results the video and experimental data would need to be combined in a way that is not trivial. The RCNN was large in memory and there was no flexibility with the design of layers, as anything over basic layers would run out of memory, even when using 3 Tesla V100s. This information led to the choice of LSTM neural networks. After the breadth examination of models, more depth went into designing the LSTM neural network model. The design choice was made to implement the model for the 30hz variation of the dataset and when comparing across the different variations the layers and hyperparameters would be kept the same. Different layers were tested by incrementally increasing the amount until the model was overfitting the data or performance stopped increasing. These layers were test with a different number of hidden states, starting at 16, and continuing with 32, 64 etc. After the optimal number along the sequence of powers of two, was found the number of states was decreased until before the model's performance decreased. This logic was to make a "fairer" model to the smaller datasets, having fewer parameters to train. Different dropout rates, 10%, 20%, 30% and 40% were tested. It was clear that as the model increase in the size, it over fit the training set which was identified by performance significantly increasing on the test set and decreasing on the validation set. The batch size was maxed out to memory constraints. The final design of the LSTM (Fig. 4) was two stacked LSTM layers with 13 and 26 states respectively, a dropout rate of .3 and a dense layer with 250 states and a dropout rate of .3 to the output.

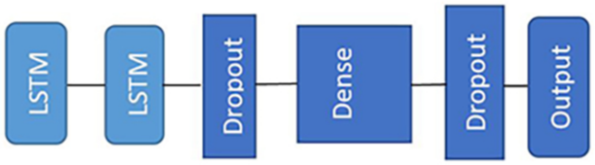

Fig. 4: Structure of the LSTM network

F. Training LSTM Networks on Experiment 2

```
import numpy as np
 ...
def random_batch_generator(batch_size ,sequence_length):
    #random batch generator that preserves sequence

    #infinite generation
    while True:
        #allocate a new array for batch
        x_batch = np.zeros(shape=(batch_size , sequence_length , num_featx),
                           dtype=np.float32)
        y_batch = np.zeros(shape=(batch_size , sequence_length , num_featy ,
                           dtype=np.float32)
        for i in range(batch_size):
            #get random start-index
            idx = np.random.randint(num_train-sequence_length)
            #x_train/y is training set defined outside of function
            x_batch[i]  = x_train[idx:idx+sequence_length]
            y_batch[i] = y_train[idx:idx+sequence_length]
        yield (x_batch , y_batch)
```

Fig. 5: Random batch generator for the LSTM

The model was trained and evaluated using MSE but results were collected using explained variance on the same train and test set as the RRM to be comparable to the previous results collected. The model was trained on 70% of the data, validated on 10% and tested on 20%. The test set was originally 10% of the data but this was not an accurate measurement of the performance on the data, and the change did not appear to impact the training of the model. The model was trained using a random batch generator, with a sequence length of 20 (Fig. 5). The batch size was 128 with 30 steps per epoch. There was a high epoch limit of 100 but the model stopped training early when the loss on the validation set stopped improving for more than 4 epochs, with a tolerance of .0001.

G. Interpolate

For two known points (x_0, y_0) and (x_1, y_1) an unknown point, y, can be solved for at x by using the equation:

$$y = \frac{(y_0(x_1 - x) + y_1(x - x_0))}{(x_1 - x_0)} \tag{2}$$

Linear interpolation between two known points was investigated as a method to enhance the performance of lower frame rate data and to more uniformly compare data processing techniques across experiments.

H. Autoencoder

Two autoencoders were surveyed as possibilities to encode the behavioral video. A traditional autoencoder made with a neural network and a convolutional autoencoder. Each model was implemented to encode the image in the same number of dimensions. The convolutional autoencoder was chosen since with little parameter tuning, performed better and took less memory than the neural network implementation of the autoencoder. A combination of the autoencoders was tested with a fully connected layer, between the encoding and decoding but there was no increase in performance. The layers of the autoencoder were first looked at using 64/32/16/8 filters and adjusted based on performance. Once the layers had been finalized, filters were subtracted until before the performance decreased. As demonstrated for the encoder portion in Fig. 6, the convolutional autoencoder (CAE) was implemented by alternating convolutional layers and max-pooling layers and decreasing the number of output filters in the convolution layers until the smallest encoding was reached. This encoding was expanded by alternating convolutional layers and up-sampling layers, increasing the number of output filters in the convolutional layer in the same way they were decreased. The autoencoder for the whisk and the full video were slightly different structures to account for different video dimensions and can be seen on Github (https://github.com/enmccarthy/drp). The data was preprocessed to remove the dark background and enhance the contrast of the whiskers and additional space was added to video on the width or height as needed to create even numbers suitable for the reduction without loss.

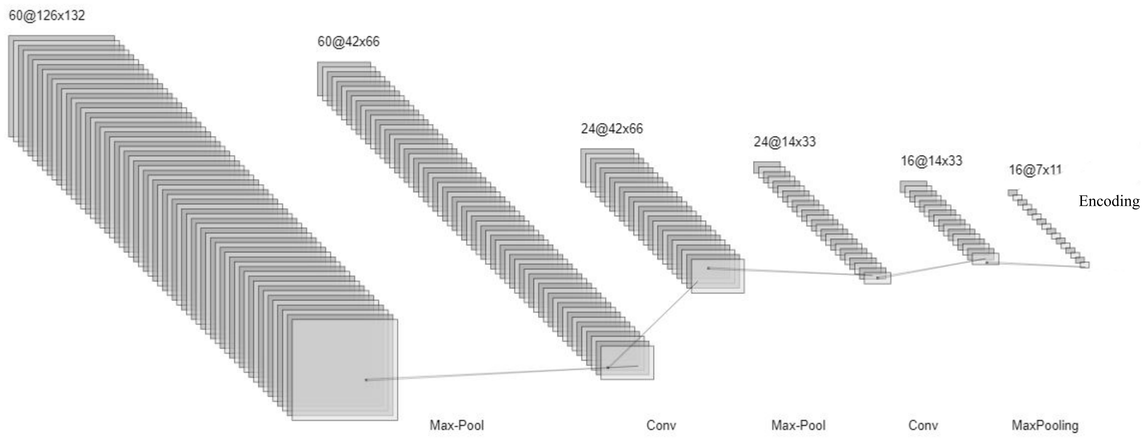

Fig. 6: The structure of the encoder portion of the whisk CAE

III. Results

In experiment 1, the RRM was able to explain 44.1 ± 10SD and on average took 20 minutes to run when cross-validated. Examining the individual explained variances of each feature, the top three features were all behavior related from the mouse. The β weights and explained variance agreed with the current definition of the areas of the brain, for example the explained variance was high for behavioral features in the motor cortex, and the visual stimulus type had large β weights in the visual cortex.

Using experiment 2 data, ridge regression worked nearly the same on the variations of frequencies. The RRM explained around 30% of variance in the large dataset and 34% of variance in the smaller dataset. When snout are jaw were left out both datasets saw a decrease in performance of around 2%. On both the large and small dataset, training at 10hz, 15hz and 30hz made no significant difference as seen in Table 3. The LSTM performed poorly on the 10hz dataset but scaled with the size of the dataset more than the performance of the RRM. The LSTM only performed slightly better than the RRM at 15hz and 30hz on the small dataset but explained about 4% more on the 15hz and 30hz of the large dataset. On the large dataset the RRM took 13 minutes to run and on the small dataset, 4 minutes. The LSTM ran for fewer epochs on the large dataset than the small taking 32 minutes and 40 minutes respectively.

TABLE III: Performance (explained variance %) of the LSTM and RRM in experiment 2

	LSTM SMALL EV	RRM SMALL EV	LSTM LARGE EV	RRM LARGE EV
10HZ	31.8	34.1	28.84	30.19
15HZ	34.5	34	34.67	30.2
30HZ	35.6	34.1	34.8	30.18
Interpolated	23.4	21.8	24.2	22.3

Linearly interpolated data was investigated to provide the deep learning models more data points to be trained on. The data was interpolated from 15hz to 30hz by adding two frames in between the real frame, similar with the features that had been already extracted from the behavioral video, such as motion energy. The models that were trained on the linearly interpolated data preformed worse than all the other models.

The whisk CAE encoded the whisker pad down to 7x11x14 from 126x132 dimensions with a binary cross-entropy loss of .4833 for the small dataset and .4672 on the large dataset. As seen in Fig. 9, the decoded image has lost a lot of fine detail. To look further into if useful data was encoded by the whisk CAE, and just fine detail was lost these encodings were used to predict motion in the whiskers. As seen in [4], a whisking event is defined as two standard deviations. This metric was used to transform the motion energy of the whisk to a binary dataset. The whisker pad of the entire video was encoded and used as input to a LSTM model. The encoding was then used to determine if the mouse was whisking or not. On the smaller dataset the model overfit the training set. On the larger dataset the model had 89.3% accuracy on the train set, 90.4% accuracy on the validation set, and 91.8% accuracy on the test set.

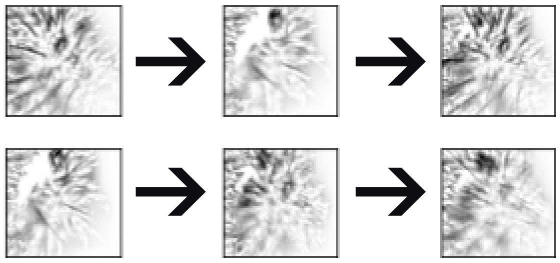
Fig. 7: Example of a sequence of motion in the whisker pad.

When the errors were examined it discovered that often it was an error in the metric used to define a whisking event. Although the definition of whisking is vague, Fig. 8 is an example of when the model would predict not whisking, which when compared to Fig. 7, the motion of whisk is clear. To compare the LSTM's probability output to the original raw motion energy, each was used as a training set alone to find total variance explained by whisking, similar to [4] and [6]. Here we saw the motion energy of the whisk was able to explain $4.9\% \pm .5$ SD when k-fold cross validated, and the probability output was able to predict $9.6\% \pm .4$ SD. The behavioral CAE encoded the behavior video from 328x424 to 41x53x8 with a binary cross-entropy loss of .2341 on the small dataset and .2290 on the large dataset. A LSTM model was trained on portions of the brain activity to predict the encoding produced by the behavioral

CAE, but the model overfit the data greatly on both datasets, with an MSE lower than .01 on both test sets and greater than 3 on the both validation and test set. This could be due to the high dimensionality of both the input and the output set.

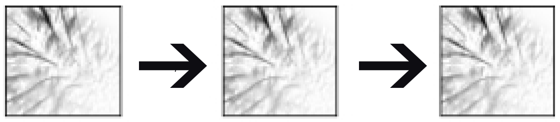

Fig. 8: Example of a sequence the model classified the encoding as not whisking but ME based whisk event indicated whisking.

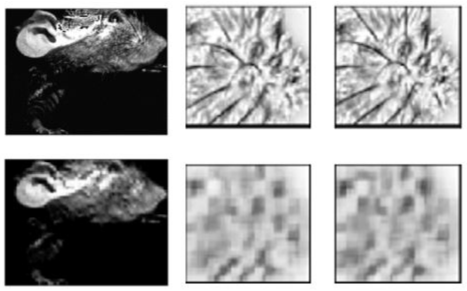

Fig. 9: Sample input (top) and output (bottom) of the whisk and full video CAE

IV. Future Work

This work has shown the potential of RRM and LSTM. RRM trains faster and needs less data than the LSTM, but the LSTM can outperform the RRM on larger datasets. It has demonstrated flaws in the popular representation of motion and has shown the potential to use machine learning to extract a more robust representation.

This work has left different combinations of variables and models, such as spatially aware models or state based models, for future work.

It has created a base line for further research of feature extraction and engineering. This includes the investigation of multimodal feature combination of signals, events and video data. In addition to feature engineering, there is a clear gap in tools and methods for automatic, unsupervised extraction and encoding motion from videos. There is the potential to increase the accuracy and represent more dimensions of movement such as direction and magnitude of the movement in videos. New representations of motion can push forward the understanding of the variability in brain activity but additionally from this new information advances in automatically aligning brain images based on the state and activity can be researched. This future work can be used to push forward the ability to use machine learning to extract information from videos.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Chelsea Finn, Ian J. Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. *CoRR*, abs/1605.07157, 2016.

[3] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, SA '15, pages 18:1–18:4, New York, NY, USA, 2015. ACM.

[4] Simon Musall, Matthew T. Kaufman, Steven Gluf, and Anne K. Churchland. Movement-related activity dominates cortex during sensory-guided decision making. *bioRxiv*, 2018.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] David Salkoff, Edward Zagha, Erin McCarthy, and David McCormick. Dissociating widespread cortical activity during visual detection task performance.

[7] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. *CoRR*, abs/1502.04681, 2015.

[8] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, and Kenneth D. Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 2019.