Exploring the Neuro-Vascular Interface With Machine Learning

Matan Ben Tov, Nadav Gat, Eyal Ban

September 12, 2022

Contents

1 Abstract

In this paper we inspect the relations between neuronal activity and vascular activity in the brain. In particular, we try to estimate the Hemodynamic Response Function (HRF) - the mapping between neuronal spikes and blood flow variations, by using different Machine Learning approaches. In order to tackle this problem, we analyze the data, define naive baselines, engineer the data in various ways, and apply multiple machine learning models - each with a different approach to this task.

Our main contributions in this project are: (1) exploring the data and deriving important statistical observations, (2) establishing a baseline and quantifying the difficulty of the given datasets, (3) defining the machine learning tasks derived from this problem, (4) defining the suitable metrics for the task, and (5) testing, evaluating and ranking different models to solve it.

Our code can be found in a GitHub repository in order to reproduce all experiments, models and results. In addition, we publish a full log of our experiments[1] in W&B web interface.

2 Background

In this project, we worked with Dr. Pablo Blinder and Mr. Lior Golgher from Dr. Blinder's Neuron-Science lab, and joined the effort of estimating the HRF. The HRF maps between neuronal activity and vascular activity, specifically how blood vessels in the brain dilate or shrink in response to electrical signals from neuron cells. The domain knowledge in biology suggests that each blood vessel in the brain nourishes several different neurons, and vice versa (each neuron signals to a cluster of different blood vessels). However, the specific mapping between the clusters of blood vessels and neurons is unknown. Our first goal is achieving a good estimation of the HRF from the given dataset, but even more valuable is understanding what are the essential components and factors for such estimation.

Finding a good estimation of the HRF serves two main purposes. Firstly, it will allow to faithfully reproduce unmeasured vascular activity from measured neuron activity, in cases where measuring neuron activity is easier (meaning, apply the HRF in a straightforward way). Each measurement requires different equipment and has a different precision, and therefore an estimation of the HRF which is accurate enough may replace some measurements. Secondly, it may shed some light and help answer questions being actively researched today in biology. For example, what is the order in which neuron-blood-vessel interaction happens? Does the firing of a neuron cause blood to flow to it's area, or do the blood-vessels carry the blood there in anticipation of the neuron activating? Quantifying the weight given to neuron activity in the past and the future of a certain timestamp may reveal what is the chronological order between these events, and also what's the delay between them.

Previous work was done in order to find such an estimation (we shall review part of it in section 4), however applying modern Machine Learning approaches was not attempted yet. Our approach to the problem is, naturally, data-based. We have employed a variety of machine learning tools and models in order to estimate variations of the HRF. The dataset for the task consists of neural and vascular brain activities of a mouse, recorded in Dr. Blinder's lab (more in section 5).

[1] For each experiment reported, a link to the relevant *Weights&Biases*'s run it corresponds to is added.

3 Task Definition

To simplify describing the different approaches to this task, we define some useful mathematical notations, in order to formalize the problem as well as our suggested solutions.

$$
\begin{aligned}
T &:= \#\{\text{time units}\} \\
N &:= \#\{\text{neurons (neuronal energy demanding sites)}\} \\
M &:= \#\{\text{blood vessel segments}\}
\end{aligned}
$$

$$
\begin{aligned}
n^{(i)}, i \in [N] &: \text{neuron} \\
S_{it} &:= \text{spike of } n^{(i)} \text{ in time } t
\end{aligned}
$$

$$
\begin{aligned}
v^{(i)}, i \in [M] &: \text{blood vessel segment} \\
F_{it} &:= \text{blood flow in } v^{(i)} \text{ at time } t
\end{aligned}
$$

$$
D_{ij} := \text{eucledean distance between blood-vessel } v^{(i)} \text{ to neuron } n^{(j)}
$$

Now we can formally write our **problem's mathematical definition:** Given **S** predict **F**, while finding the the relations embodied by this prediction. This is the simplest definition of the problem, but we may also use the past vascular activity, distances and other factors in order to perform the prediction of **F**.

4 Previous Work

In his as yet unpublished thesis, Mr. Golgher has applied the following methods to the problem:

- **Simplistic Linear Transformation**:

$$F_i = \sum_{j,t} W_{it} \cdot S_{it}$$

  where W is the linear transformation learned by pseudo-inverting the Hankel matrix. This over-simplistic linear model resulted with extreme over-fitting of the training set. This attempt might imply that describing this mapping requires a slightly more complicated function.

- **Kalman Filtering:** Estimation of the HRF kernel using Kalman filtering. The result of this approach suggested that the estimation is feasible when a single neuron drives the hemodynamic response of a single blood vessel, meaning it resulted with good prediction on a subset of blood vessels. In our project we define specific metrics to capture such behaviors of our models.

- **Auto-Regression Model**: auto-regressive modeling of the vascular activity with neuronal activity and other additional variables. This approach models the vascular activity in a specific timestamp as a linear combination of the following: mean vascular activity, previous p timestamps[2] of vascular activity, and past p of neuronal activity, with an addition of normally distributed noise. The resulting formula is:

$$F_t = F_{mean} + \sum_{k=t-p}^{t-1} w_k^{(vascular)} \cdot F_k + \sum_{k=t-p}^{t-1} w_k^{(neuronal)} \cdot S_k + (\text{normal-noise})$$

  where **w** are the learnable scalars. In our project we took inspiration from this approach to design one of our models.

[2] p is a hyperparameter.

5 Data

5.1 Datasets

Two datasets were provided, both containing recordings of neural and vascular activity in the brain of a lab mouse. Both datasets[3] were recorded in similar conditions and contain the same neurons and blood vessels.

- The first dataset (will be referred as *Dataset-1*) is 19.3 minutes long (corresponds to 35K timestamps).
- The second one (*Dataset-2*) is 26.6 minutes long (corresponds to 48K timestamps).

As mentioned above, each dataset is divided to T timestamps, each including:

- The activity of 425 blood-vessels, F_t, measured by each vessel's diameter.
- The activity of 50 neurons activity, S_t, measured by each neuron's spike.

In addition, approximated coordinates of both the blood-vessels and neurons were provided, from which we derived the distances matrix, D. It's important to note that the coordinates were of a single point each, whereas blood-vessels and neurons are 3-dimensional and have varying shapes and sizes.

The coordinates approximate a "center" for each neuron, however the interesting part of it, for the purpose of prediction, may be at its edge (which is relatively far from the approximated "center"). Therefore, the coordinates are necessarily a very rough estimate, and possibly too inaccurate to extract useful information from.

The recordings provided were selected from a much longer and noisier recording, and some basic noise reduction methods (e.g. smoothing) were applied to them. Even so, the given datasets still suffers from noise, due to the measuring techniques used. It should be noted that the datasets may be considered "incomplete" from a biologically perspective, as some blood-vessels might be affected by neurons that were *not* recorded.

5.2 Data Analysis

In order to better understand the data, we visualized the activity time-series, analyzed each aspect of the data, and ran several statistical tests on it[4].

- **Vascular Activity Visualization, Neuronal Activity Visualization**: From the visualizations it is evident that for both the neuronal and the vascular cases, the activity changes sharply between timestamps, throughout the whole recording.

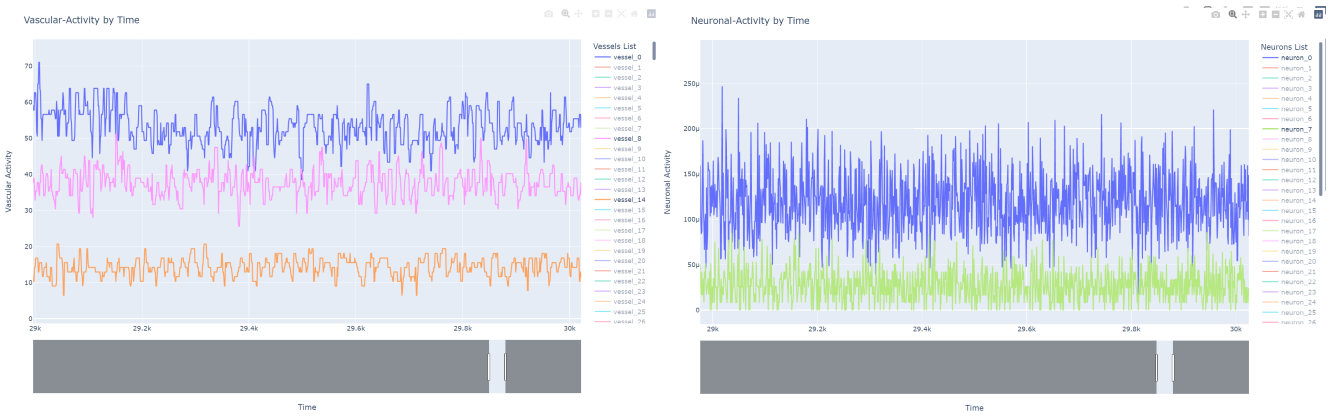

Figure 1: *Left:* Vascular activity of three randomly chosen blood vessels over 1000 timestamps. *Right:* Neuronal activity of two randomly chosen neurons over the same 1000 timestamp.

[3]The raw names of the *Dataset-1* and *Dataset-2* are 2021_02_01_18_45_51 and 2021_02_01_19_19_39 correspondingly.

[4]All visualization and tests shown in this section were done on *Dataset-1* (but most are identical or similar with *Dataset-2*)

- **Vascular Activity Values Distribution, Neuronal Activity Values Distribution**: Taking into account only the values of each blood vessel and neuron, we plot their distribution in Figure 2. From the visualization for blood-vessels it's clear there are many different centers of distributions (for different vessels), explained by their different sizes (e.g. arteries vs capillaries). This is an indication for the importance of the vascular mean activity of each blood vessel, discussed and utilized later.

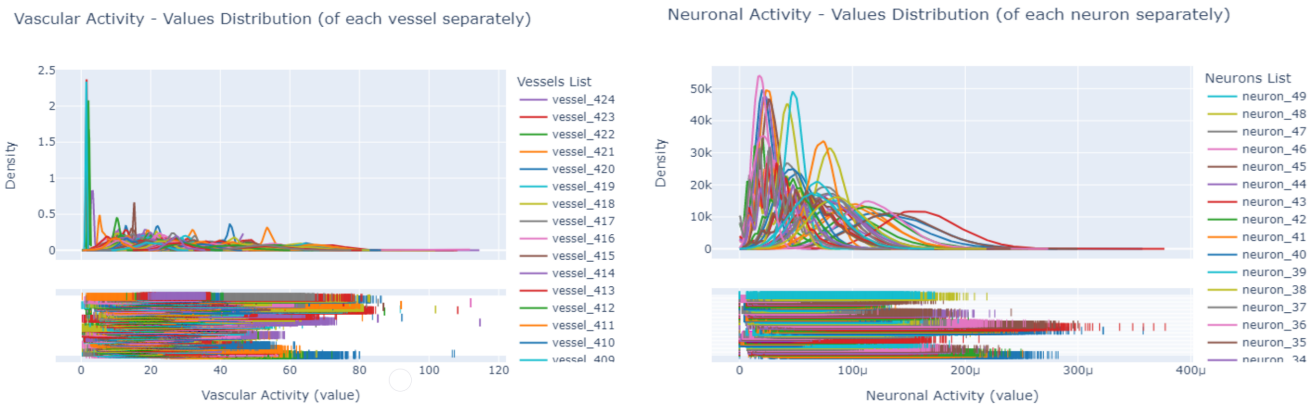

Figure 2: Distributions plots (by moving histogram of their values) of blood-vessels (on the left) and neurons (on the right).

- **Coordinates**: 3D visualization of the coordinates of the blood-vessels and neurons.
- **Vascular-Neuronal Cross-Correlation Per Timestamp:** For each timestamp, we take a mean of the vascular activity and neuronal activity, then plot these pairs to observe how well they correlate. If a neuronal activity in time t implies certain vascular activity in time t we'd expect (some) correlation between these values. However, as shown in Figure 3, the correlation is very low. This is not surprising biologically, as we should expect lags between the activities. Either way, this is another indication for us to build the HRF with *windows* of timestamps.

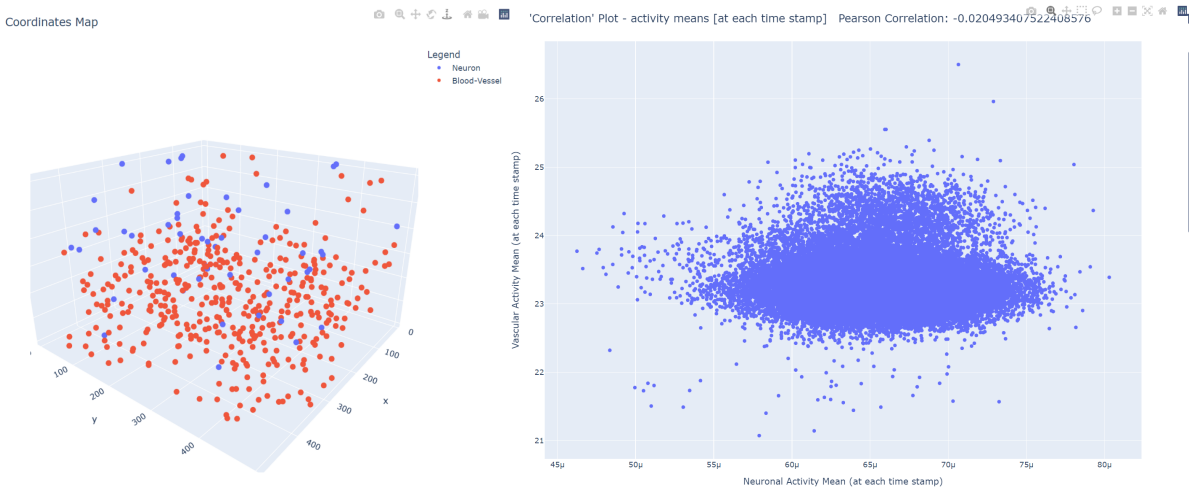

Figure 3: *Left:* 3D visualization of the recorded neurons and blood vessels coordinates. *Right:* combinations of mean neuron activity and mean blood-vessel activity that appear in the dataset together (each point is a timestamp).

- **Vascular-Neuronal Cross-Correlation:** For each *pair* of neuron and blood vessel, we calculate the correlation between the vascular-activity vector and the neuronal activity vector (after applying a sliding window to it) over time. We show these values as a heat-map of the cross (Pearson) correlations in Figure 4. We can observe that most blood vessels-neurons pairs hardly correlate, while some specific blood vessels seem to correlate well with many neurons. This is the main motivation for definition of metrics that will measure how well the model performs on a *subset* of blood vessels, discussed later (section 6.3).

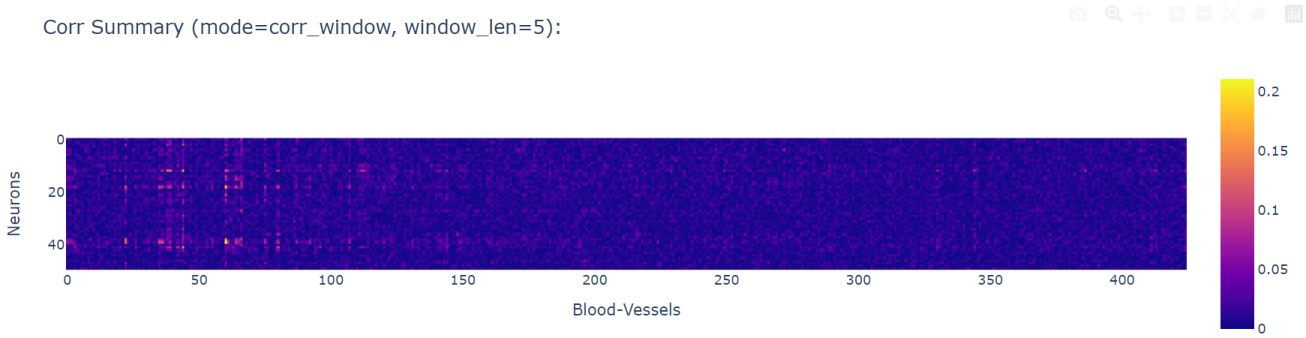

Figure 4: Heatmap of the correlations between activities for each neuron-blood vessel pair. A "hot" column might indicate a blood vessel that correlates well with many neurons.

- **Vascular Activity Auto-Correlation:** While the correlations between blood vessels and neurons are low, each blood vessel, naturally, highly correlates to its own past activity. That is, the vascular activity vector of some blood-vessel highly correlates to itself with a shift (in time). The smaller the shift, the higher the correlation, as shown in Figure 5. In particular, we observe that the auto-correlation from delay of about 5 roughly remains constant.

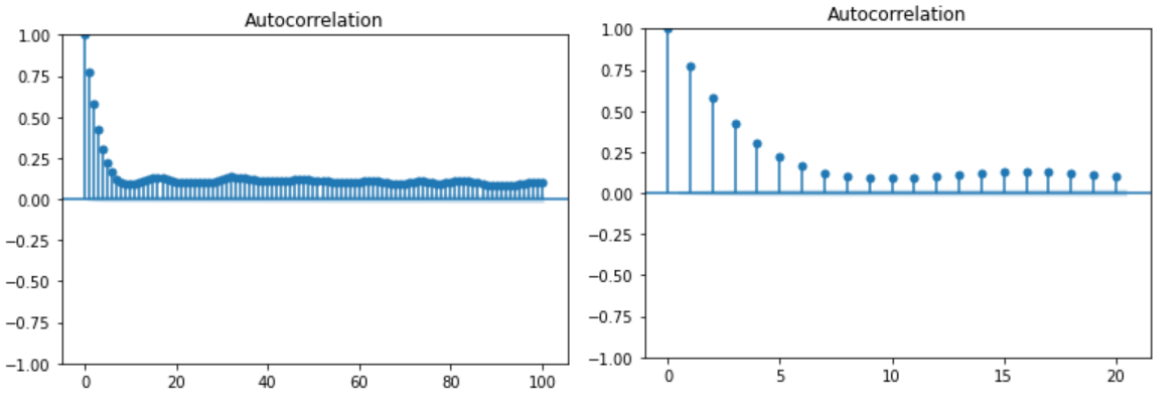

Figure 5: *Left:* Vascular activity auto-correlation on an arbitrary blood-vessel (#0, but results are similar for the rest of the blood-vessels). *Right:* a crop of the full plot on the left. X axis stands for the shift the auto-correlation was calculated with.

6 Our Approach

6.1 Introduction

To describe our Machine Learning approach we first describe our approach to the dataset, and what features we extracted from it. We then define the main loss function we used for our models as well as several additional metrics. We present naive baselines applied to the dataset and finally our models for estimating the HRF and their empirical results.

Our models can be roughly divided to 3 groups, all of which predict vascular activity at a certain timestamp t, each differentiated by the scope of the model's input (specifically the vascular input): models with **both** vascular activity input (up to timestamp $t-1$) and neuronal activity input, models with neuronal activity and ***delayed*** vascular activity input (up to timestamp $t-1-d$ with $d>0$) and models with neuronal activity input **only**. We don't compare models from different groups to each other as it is not a "fair" comparison, since a model receiving more varied information as input has an advantage.

We expect the first group to yield models with the best performance since they receive the most input information, but are also interested in the other two groups as those models can be applied in more settings (when there is only neural activity recording, or in real-time) and may also be informative.

While the first group of models seems the most natural, we shall further discuss the motivation for modifying the content given by the vascular activity in the following sections. In particular we were motivated by the analysis of the data, section 5.2, and the results of our linear regression baseline, section 6.5.

6.2 Dataset

6.2.1 Data Pre-Processing

In order to prepare the dataset as an input to our models, and in addition to the regular pre-processing done by the lab, we cleaned and organized the data using the following methods:

- **Missing Data:** There are some[5] missing points in the vascular activity data, in order to fill these data points we calculated the mean vascular activity of each blood vessel and replaced each vessel's missing data point with its mean activity.
- **Data Normalization:** In order for the data to consist of uniform zero-centered values we normalize the neuronal activity.
- **Dataset Split:** In order to both train on the data and evaluate our models, we split the data as follows: 85% for the training-set, 7.5% for the evaluation-set and 7.5% for the test-set. This split keeps the continuity of the time series, as we wish to avoid cross-contamination between the different sets. It should be noted that due to the smoothing performed on the dataset, there still may be a small cross-contamination in the timestamps closest (in time) to other datasets, but it's negligible.

6.2.2 Feature Engineering

Since we are given a time series, we need to define each sample and its label. In our approach we refer to this task as a Regression / Time-Series-Prediction problem. We treated the dataset in various ways, each variation suits to different machine-learning approach and model. For each timestamp t, our models' "X" is built of the following components:

- **Neuronal Activity:** In order to model a neuronal activity in **timestamp** t, we take a time-window of this activity. This window starts from a timestamp preceding t and ends in a timestamp postceding t (as we may use "future" neuronal activity to predict the HRF).

  Formally, we may write this part as $(S_{t-w_1}, S_{t-w_1+1}, \ldots, S_t, \ldots, S_{t+w_2})$, where w_1 and w_2 correspond to the past and future window sizes (these are hyper parameters we tune).

[5] Missing data of vascular activity concludes to less than 0.05% of the values.

- **Vascular Activity:** In order to model a neuronal activity in **timestamp** t, once again, we take a time-window of this activity. For vascular activity, the window starts from a timestamp preceding t, and ends in timestamp $t-1-d$, so as not to reveal vascular activity in timestamp t, nor the timestamps surrounds it. d is a hyper parameter of *delay* - the feature excludes d timestamps right before the timestamp being predicted[6].

  Formally, we may write this part as $(F_{t-w_3-d}, F_{t-w_3-d+1}, \ldots, F_{t-1-d})$, where w_3 is the past window size (a hyper parameter) and d is the *delay* parameter.

- **Spatial Proximity:** A matrix in which each coordinate i, j corresponds to a proximity score between blood vessel i and neuron j.

  This score is calculated by: $\exp(-normalized(D_{ij}))$, where the normalization of D_{ij} is done across all the distances values.

- **Mean Vascular Activity:** A vector the size of the blood vessels count, which contains the mean activity of each blood-vessel over the training set.

Note that the last two components, naturally, do not change between different timestamps (as opposed to the activities components).

Our models' Y (=label) is simply the vascular activity at **timestamp t**.

[6] Unless otherwise stated, the default for every model is $d = 0$.

6.3 Loss and Other Metrics

As we face a regression task, the most straight forward loss to define is the MSE of our predicted vascular activity. Indeed this is the loss we use to optimize all of our models. However, we might be interested in tracking additional metrics to compare between Y (ground truth) and the prediction $\hat{Y}$:

- **Mean Squared Error (MSE)**: The good old regression metric.
- **Normalized Root Mean Squared Error (NRMSE)**: A simple MSE might not be a "fair game", since the regression is done across different blood-vessel, one blood-vessel might be very big and another very small (e.g. arteries as opposed to veins). This means that one blood-vessel might yield larger MSE values (as its values are relatively large to begin with) while the other might yield smaller MSE values, since it's smaller. Hence the motivation to calculate the MSE with different weight for each blood vessel, that is, to normalize each MSE.

  In order to calculate this metric, we calculate the normalized RMSE of each blood-vessel i:

  $$NRMSE_i = \frac{\sqrt{\frac{\sum_{ij}\left(Y_{ij}-\hat{Y_{ij}}\right)^2}{N}}}{\overline{Y_i}}$$

  Where $\overline{Y_i}$ is the mean vascular activity of blood-vessel i. Then, in order to summarize this vector to a scalar we take its mean, that is:

  $$NRMSE = \frac{\sum_{i=1}^{M} NRMSE_i}{M}$$

- **Mean of Best K Mean-Squared-Errors (MBKMSE)**: Some models might perform very well on a subset of blood-vessels, and we would like to measure such behavior. Actually, previous work has already suggested this might be the case. Thus, we were motivated to define a metric that would summarize the performance of the most closely predicted blood-vessels by the model.

  In order to calculate this metric, we first calculate the NRMSE for each blood-vessel, as described in the previous metric. Then, we take the top $K = 50$ NRMSEs we found, these are the so-called "best predicted" blood-vessels, and take a mean of these values to the MBKMSE metric.

It should be noted that we optimize all our models with the classic MSE loss, and the additional metrics are only used for evaluation and comparison. We wish to minimize all the above mentioned metrics, as they grow when the error grows.

6.4 Naive Baselines - Control Models

In order to define a success criterion, we implemented several naive models as control models. If a model surpassed the naive models' performance on the evaluation-set, this might imply the model has "learned" something meaningful and not trivial. The following controls were tested:

- **Persistence Model**: For each timestamp t, naively "predict" the vascular activity to be F_{t-1} (the activity in the previous timestamp). Every model which has the vascular activity as an input may learn this mapping.
- **Delayed Persistence Model**: For each timestamp t, naively "predict" the vascular activity to be F_{t-1-d}. Every model which has the *delayed* vascular activity as an input may learn this mapping.
- **Mean Model**: For each timestamp t, naively "predict" the mean vascular activity of each blood-vessel over the training set. Every model that is optimized on the vascular-activity as Y of the training set, can learn the training-set's mean-vascular-activity, although this is less straight-forward.

As mentioned above, we can roughly divide our models to **3** groups, and each control model corresponds to a different group. For example, a model with both vascular and neuronal activity as an input is comparable to the Persistence Model.

6.4.1 Results

The main results of these models are detailed below:

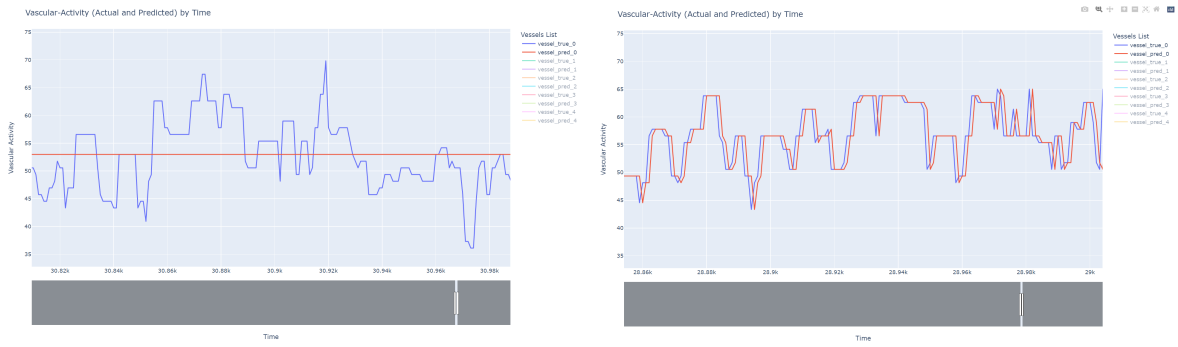

Figure 6: Errors of the Mean Model (left) and Persistence Model (right). Actual vascular activity is in *blue* and the predicted is in *red*. The persistence model produces a shifting effect.

Model	MSE	MBKMSE	NRMSE	Run ID
Persistence Model	7.145	0.0528	0.1207	1125n2kn
Delayed Persistence Model ($d = 1$)	13.41	0.07251	0.1655	amemyah5
Delayed Persistence Model ($d = 2$)	18.516	0.08511	0.1943	3b1rycb3
Delayed Persistence Model ($d = 5$)	26.644	0.104	0.2326	1qdmmino
Mean Vascular Model	15.226	0.07979	0.1773	ef1yzky2

Table 1: Control baselines performances on validation set of **Dataset-1**.

Model	**MSE**	**MBKMSE**	**NRMSE**	**Run ID**
Persistence Model	7.317	0.0568	0.122	i8cmp6x1
Delayed Persistence Model ($d = 1$)	13.781	0.07696	0.1677	243pozgy
Delayed Persistence Model ($d = 2$)	19.174	0.09021	0.1979	3v07ucpo
Delayed Persistence Model ($d = 5$)	27.8	0.1076	0.24	2eicizi6
Mean Vascular Model	15.251	0.0829	0.177	3j1xo4dk

Table 2: Control baselines performances on validation set of **Dataset-2**.

The main takeaway from these results is the strong relation between the vascular activities in adjacent timestamps (which was also shown in our statistical analysis of the data, section 5.2). This also means, as we'll show, that giving as input to the model vascular activity from timestamps immediately preceding the predicted one encourages it to simply ignore all other inputs, and in particular to ignore the neuronal activity. For this reason we will mostly avoid providing our models with past vascular-activity too close to the one predicted.

6.5 Baseline Model

As a first step, we attempted the most straightforward approach to the dataset - Linear Regression. We define X_t to be the flattened concatenation of the neuronal and vascular activity windows (we experimented with different windows sizes and *delay* parameters). Formally, the predicted vascular activity in timestamp t, $\hat{F}_t$, is given by:

$$\hat{F}_t = \mathbf{W} \cdot X_t + \mathbf{b}$$

where $\mathbf{W}$ is a matrix used to project the concatenated activities of the input to the output dimension, and $\mathbf{b}$ is an additive bias, with the same dimension as the output. We also used Ridge Regularization, which was essential to prevent over-fitting. As mentioned in the dataset section, the window sizes are hyper-parameters that can be tuned. Specifically, a window size of 0 means exclusion of the relevant activity (e.g. vascular) from the input X.

6.5.1 Experiments

We experimented with various window sizes[7], and in particular we experimented with excluding the **vascular activity** from the input, or modifying its vascular *delay* parameter. The results are detailed in Table 3.

Model	Windows Sizes	MSE	MBKMSE	NRMSE	Run ID
Lin. Regression	(1,1,1), 0	*7.180*	***0.05721***	*0.1201*	2dc4z09r
Lin. Regression	(1,50,50), 0	8.103	0.06169	0.1281	19lrogk9
Lin. Regression (with delay)	(1,50,50), 1	*12.878*	*0.0774*	*0.1615*	h5hnl936
Lin. Regression (with delay)	(1,50,50), 2	15.406	0.08439	0.1767	15m3c05l
Lin. Regression (vascular only)	(1,0,0), 0	***7.171***	*0.05767*	***0.1200***	wdzi13y4
Lin. Regression (vascular only)	(1,0,0), 1	11.449	0.07241	0.1516	z71yzdzq
Lin. Regression (neuronal only)	(0,1,1), 0	15.687	0.08657	0.1788	113gx5ri
Lin. Regression (neuronal only)	(0,5,5), 0	*15.292*	*0.08513*	*0.1758*	1l7xhr4x
Lin. Regression (neuronal only)	(0,50,50), 0	18.183	0.09424	0.1921	28ybmaj0

Table 3: Linear Regression baseline variations performance on the validation set of Dataset-2. Window sizes correspond to - *(Vascular past window size, Neuronal past window size, Neuronal future window size), Vascular Delay.* The best results for each subset of models is marked with *italicized* text, and the overall best is in **bold**.

6.5.2 Conclusions

- **Models with Full vascular and neuronal input:** This is the classic and most straight-forward variation of the Linear Regression baseline [8]. This variation is comparable to the Persistence model, since the model's input includes the vascular activity in the previous timestamp. A slight improvement compared to the Persistence model is noticeable.

  We can also notice that enlarging the neuronal windows (e.g. to size of 50 timestamps) hurts the performance. However, if we remove the neuronal activity completely we'd actually slightly improve the model's performance. This has lead us to probe the model's learned parameters.

  We probed the weights of the full-input models, in order to explain this phenomenon. We found that the model focuses on outputting the **previous** timestamp of each vascular activity. That is, when the model is given F_{t-1}, its best shot of minimizing the loss would be to predict F_t to be F_{t-1}. This high dependency can also be visualized by plotting the weights matrix[9], $\mathbf{W}$:

[7] The models presented here are differentiated merely by their *Window Sizes* parameters.
[8] Shown in first 2 rows in Table 3
[9] Weights of the model in the first row of Table 3.

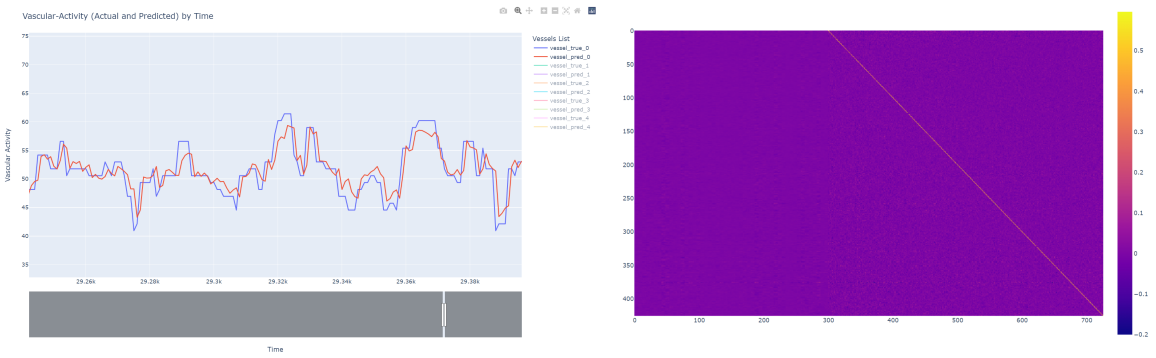

Figure 7: *Left:* A lag from the prediction (red) of the linear regression plot with the actual vascular activity (blue). It seems that the model learns the vascular activity with an offset. *Right:* A heat-map visualizing $\mathbf{W}$ learned from samples that included past vascular activity. The strong diagonal that can be easily spotted in the matrix, indicates the high weight given to the past vascular activity when it is projected to the output. In other words, this result with the prediction of F_{t-1} .

This has motivated us to exclude the vascular activity in timestamps immediately before the one we're predicting from most of the following models' inputs, as the models would be extremely biased towards them. Since our goal is to find the relation between neuronal activity and vascular activity, we are not interested in models that over-fit the vascular activity while ignoring the neuronal activity.

- **Models with Delayed Vascular Activity:** Since giving the model $t-1$ vascular activity is problematic, we tried adding a delay to the vascular-activity-input. These models are comparable to the *Delayed Persistence* control models.

  Firstly, we can notice slight improvement from the control model, which might be attributed to the model learning a certain pattern of vascular activity. A more interesting question could be - is that pattern dependent on the neuronal activity input. The answer is probably not, since ablating the neuronal activity from the input results with even *better* performance (shown in Table 3). This means that not only is the neuronal input not big contributor to these models but it actually hurts their performance. It should be noted that this negative impact of the neuronal activity might be caused by the Linear Regression being unsuitable modeling for this task.

- **Models with Neuronal Activity Only:** In these models we simply map the neuronal activity to the predicted vascular activity (that is, we don't insert the vascular activity as an input, in any form). These models are comparable to the Mean control model, since they can potentially learn the mean vascular activity of the training-set.

  These models perform mediocrely, even in comparison to the control model.

  From probing these models, we discovered that in most blood-vessels the model prediction is very close to predicting the mean-vascular-activity of each blood vessel. Moreover, we noticed that the resulted $\mathbf{W}$ contains values close to zero, and $\mathbf{b}$ values are very close to those of the mean vascular activity. Meaning, that model simply memorized the mean vascular activity over the training set.

In conclusion, we noticed the biggest improvement from the control models in the model[10] that maps $t-2$ vascular activity to t, this might indicate the potential of mapping vascular activity to itself, which we shall not focus on. In addition, the results, both including and excluding past vascular activity, might imply that simple Linear Regression is an over-simplistic model for this task.

[10]Shown in 3rd row from the end of Table 3.

6.6 Deep Linear Neural Networks

In recent years, several theoretical results ([AGCH19, ACHL19], for example) showed the possible advantages for optimization and generalization of using Deep Linear Neural Networks (LNNs) - networks consisting of a sequence of linear layers, without any non-linear activations in between them. Though the expressiveness of such networks, regardless of depth, is the same as simple linear regression, it was shown that optimizing these networks using Gradient Decent or any similar (gradient-based) algorithm may result in a better solution than the closed form solution derived from linear regression. A better solution in this context is one which generalizes better (i.e. lower error on test or validation set), since the closed form solution will achieve the minimal error on the training set.

Another advantage Deep LNNs have, which is very relevant to our task, is explainability - as mentioned above, similar to Linear Regression, any learned hypothesis may be expressed as a linear mapping from input to output. Even if the LNNs don't achieve a lower error compared to other, more complicated models, it may help to answer some of the unanswered questions mentioned in the background section (2).

In this model we shall focus mapping the neuronal activity (solely) to the vascular prediction.

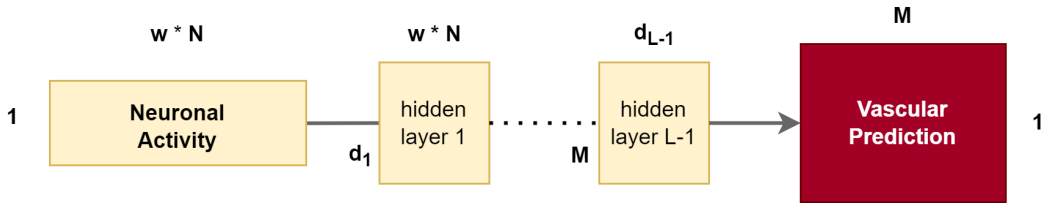

Figure 8: Architecture of a LNN with L layers. The input vector - flattened neural activity from N neurons over w timestamps - is multiplied by a sequence of matrices (with a bias added after each multiplication), and the final output is a prediction vector of dimension M - the number of blood-vessels being predicted. $d_1, ..., d_{L-1}$ are the dimensions of the hidden layers, and are hyperparameters we tune.

6.6.1 Experiments

The two important hyperparameters tuned in experiments were the depth of the network and the hidden widths. Specifically, the depth of the networks ranged from 2 (single hidden layer) to 10 (9 hidden layers), and the regimes tested for choosing the hidden widths were as follows:

1. Constant. The same width is used for every hidden layer. The values tested were between 100 and 450 - all between the dimensions of the input and the output (counting different timestamps of the same neuron as one). This is the most straight-forward architecture.

2. Monotonically increasing/decreasing. The first hidden layer has a relatively small/large size, and the size increases/decreases in each subsequent hidden layer.

3. Switching between increasing and decreasing sizes. The first hidden layer has a relatively small/large size, which is then increased/decreased in subsequent layers until the middle one, and then decreased/increased accordingly.

Every linear layer (except for the last one) was followed by a dropout layer, with dropout probability $p = 0.5$. In all experiments, the input given to the LNNs was the neural activity only, without any vascular activity. The best results for all of these settings are reported in Table 4.

We found that the LNN model performed worse than the Linear Regression baseline on Dataset-2 (therefore those results are not reported). The comparable Linear Regression baseline results corresponding to Table 4 (i.e. on **Dataset-1**, with only neural activity as input) are MSE=15.048, MBKMSE=0.08321, NRMSE=0.1759, meaning the LNN improves upon the baseline in two metrics.

Hidden widths	# Layers	Windows Sizes	MSE	MBKMSE	NRMSE	Run ID
200	4	1,1	16.81	0.1016	0.1849	11kj714r
450	9	1,1	15.364	0.08849	0.1777	1ixvq122
$50 \to 1000$	8	1,1	**14.969**	**0.08383**	**0.1756**	6zhfm5zw
$50 \to 1400$	10	1,2	15.001	0.08429	0.1758	2kf2wqqb
$100 \to 500$	10	1,1	16.943	0.1026	0.1854	1c54tt25
$1000 \to 50$	8	1,1	16.666	0.1009	0.1846	2pc5fm6v
$100 \to 800 \to 100$	8	1,1	15.594	0.09098	0.179	3g819a68
$800 \to 100 \to 800$	8	1,1	15.786	0.0926	0.1798	25feay0c

Table 4: Performance of different LNNs on the validation set of **Dataset-1**. The Hidden Widths column indicates the relevant regime from the above mentioned - the first and last values indicate the widths of first and last layers (the same value in regimes 1, 3) and $\to$ indicates monotonically increasing/decreasing values between layers. Window Sizes correspond to *Neuronal past window size, Neuronal future window*.

6.6.2 Conclusions

While the model does show improvement over the *Mean* control and Linear Regression baseline model, the fact that the model's performance was worse on **Dataset-2** suggests it's not a very robust model, probably due to the simplicity of the functions it represents. Even though, the results in Table 4 do seem to indicate there is potential in such modeling, and it can also provide meaningful and clear insights on the problem, exactly because of it's simplicity. Several conclusions may be deduced from these results, which can help further develop this model, when more data is collected:

- **Depth does improve performance**. As shown in theory, it's evident from these results that adding more linear layers improves upon the linear regression baseline. Furthermore, from the first rows of Table 4 it is clear that given the same widths, deeper networks achieved better results, and it's possible that networks deeper than the best ones reported may achieve even better results with the correct hyperparameters.
- **Small window sizes**. Although many window sizes were tested (ranging from 1 to 50), in most cases the smallest window achieved the best results. This may also be due to the simplicity of the model, causing more information (i.e. larger windows) to mislead it.
- **Monotonically increasing regime is best**. The best two results were achieved in the monotonically increasing regime, and the wide margin between these and the other regimes suggests this regime has an actual advantage - perhaps allowing the network to learn a more complicated structure in the deeper layers.

Another advantage of a Deep LNN is in the case that a new dataset will be supplied but will have different input or output dimensions (i.e. a recording of more neurons and/or more blood-vessels). In this case, the first and last layers of the LNN can be replaced, with all other layers either frozen or initialized with the learned parameters, and so some of the models understanding may be transferred from one dataset to another.

6.7 Engineered HRF (EHRF)

In this model we expand the straightforward Auto-Regressive modeling of this problem with additional learnable components, as well as utilizing new aspects of the data. Driven by our intuition and basic prior knowledge, we carefully engineered the vascular activity as a summation of a linear combination of the neuronal activity weighted by their distances from the blood vessels, a learnable term given by Fully Connected layer and the mean of the vascular activity taken over the training set.

Our motivation in this function is to build a model that is forced to have a certain logical structure in order for the learnable components to have interpretable aspect.

We present the full model architecture in Figure 8, and the function it represents can be written in a more formal way in Equation 1.

$$F_i^{(t)} = L_i(N^{(t-w,t+w)}) + \bar{F}_i + \sum_{j \in \{neurons\}} D_{ij} \cdot Conv(N_j^{(t-w,t+w)}) \tag{1}$$

Where:

- L_i is a latent term given by learnable Fully-Connected layers (applied on the neuronal activity window).
- $\bar{F}_i$ is the mean vascular activity of blood vessel i.
- D_{ij} is the proximity score (as defined in section 6.2.2) between blood vessel i to neuron j. In some experiments we learned these scores from random initialization (instead of using the measured distances).
- $Conv(N_j^{(t-w,t+w)})$ is a series of learnable 1D-convolution layers[KAA+21], applied on the neuronal activity window in order to extract a single feature (scalar) that will correspond to each neuron.

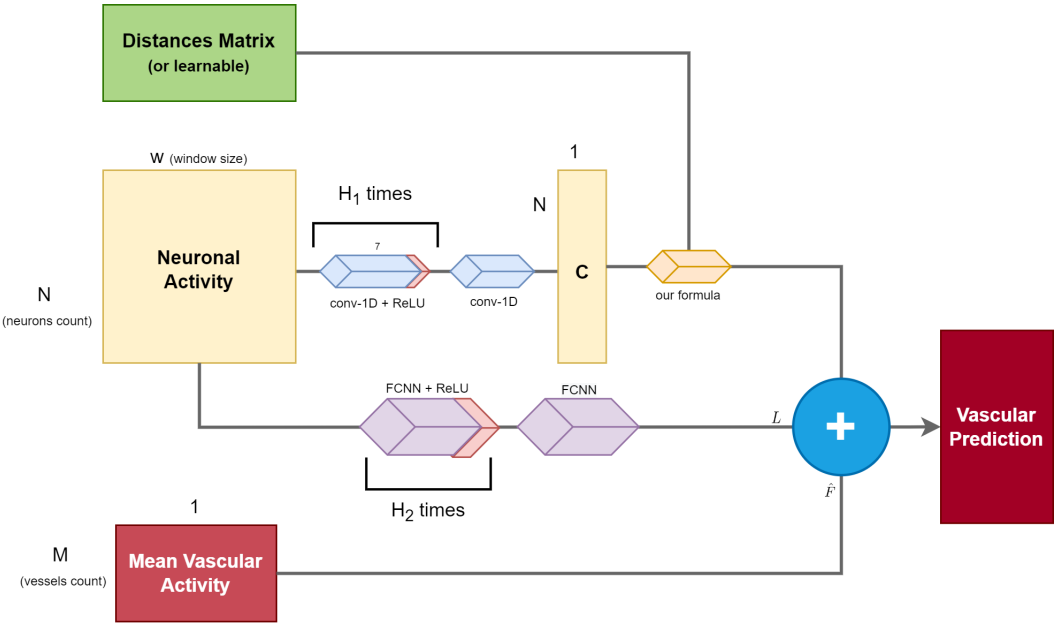

Figure 9: EHRF's model architecture, uses 1D-convolution layers to extract features from the neuronal activity which will then be fed to a weighted summation that uses the distances. It also uses Fully Connected layers to extract the latent vector L. The outputs of these components are then summed with the mean vascular activity to the prediction. H_1 and H_2 are hyper-parameters.

6.7.1 Experiments

We experimented with different variations of the model, in order to optimize the architecture, and also to evaluate the importance of certain components. In the following experiments we set the Conv-1D count (H_1) to 20 and the depth of the latent term's Fully Connected (H_2) to 2 layers of width 500. We present the best performing model (on the val-set) of each variation in Table 4:

Model	Window Sizes	MSE	MBKMSE	NRMSE	Run ID
Full Model	(0, 15, 10)	**15.009**	0.08328	0.1742	m9wa92gy
Full Model	(0, 1, 1)	15.617	0.08590	0.1783	2c1lyblb
Full Model	(0, 5, 5)	15.011	0.08375	0.1742	10iof1s4
Full Model	(0, 50, 50)	15.722	0.08557	0.1791	31cvbxe2
With Learnable Distances	(0, 15, 10)	15.012	**0.08302**	**0.1741**	16gxm97j
Without Latent Term	(0, 15, 10)	16.705	0.09142	0.1848	2sejb1aq
Without Mean	(0, 15, 10)	15.405	0.08640	0.1767	29md5jyy
Without Mean, Without Latent	(0, 15, 10)	142.943	0.15240	0.4589	2nxfyrjj

Table 5: EHRF variations performances on validation set of Dataset-2.

6.7.2 Conclusions

While the best results of this model do not improve significantly upon the *Mean* control model, we conducted some experiments to evaluate the importance of each component in the architecture.

- **Distances Impact:** We found that the original distance values had no performance impact, in comparison to the learnable distances. Meaning that, by our models' empirical results, the distances do not provide helpful information for the HRF. In fact, our results show that ablating the actual distances (and learning them instead) helps the model.
- **Latent Term Impact:** We also found that the latent term (introduced by a sequence of Fully Connected layers) is important for this architecture to perform better. In particular, this means that the weighted-convoluted-neuronal-activity and mean vascular activity terms are not sufficient for predicting the vascular activity. This is another indication that a simple modeling of the neuronal activity (e.g. by the convoluted neuronal activity term) is not sufficient for this task.
- **Window Sizes Impact:** The best performing model is given about 15 timestamps for the past and the future windows. However, this does not necessarily mean these are the optimal window sizes for modeling the problem, as they affect more factors of the model. For example, this might be attributed to the simplicity of the 1D-Conv modeling.
- **Residuals Structure is Better:** Even when ablating the mean-vascular-activity term, we noticed that similar values "moved" to a bias learned by another component (e.g. by the latent term's bias). This indicates that the focus of this model, whether it is given the mean-vascular-activity or not, is to learn the deviation from this mean. This might imply that if we insist on simple modeling of the HRF, it should include the mean vascular activity as a term.

6.8 Recurrent Neural Networks

Following their popularity in the NLP field in the last decade, RNNs have found their way to being applied in many sorts of tasks - in particular, for time series forecasting ([HBB19] and [Pet19], although it was suggested before [GES01]). It was also found to be the best performing model for such tasks in many use-cases, thanks to its recurrent architecture which allows feeding the model one timestamp at a time (preserving both spatial and temporal information). Looking for models that are able to preserve the original spatial structure of the input, rather than than flattening it, we chose to focus on this type of model.

In our case, we use a dual RNN[11] (specifically LSTM, [HS97]) model, where each RNN works in a "many-to-one" regime (Figure-10). We feed each RNN with a different activity (one for neuronal activity and the other for vascular activity) *timestamp by timestamp*, and eventually we take the last hidden states (outputs of the two nets), concatenate them, and apply a linear layer to predict the vascular activity from the concatenated vector. We are still in the "regression-setting", in particular - we do *not* feed the model with its own predictions[12].

While RNNs might not be as explainable as the models we presented so far, there are some meaningful components in the RNN architecture that can be probed (e.g. the Forget Gate, Update Gate, etc.). Moreover, thanks to the recurrence regime, window sizes do not affect the architecture (in particular, window sizes may change between training time and inference time).

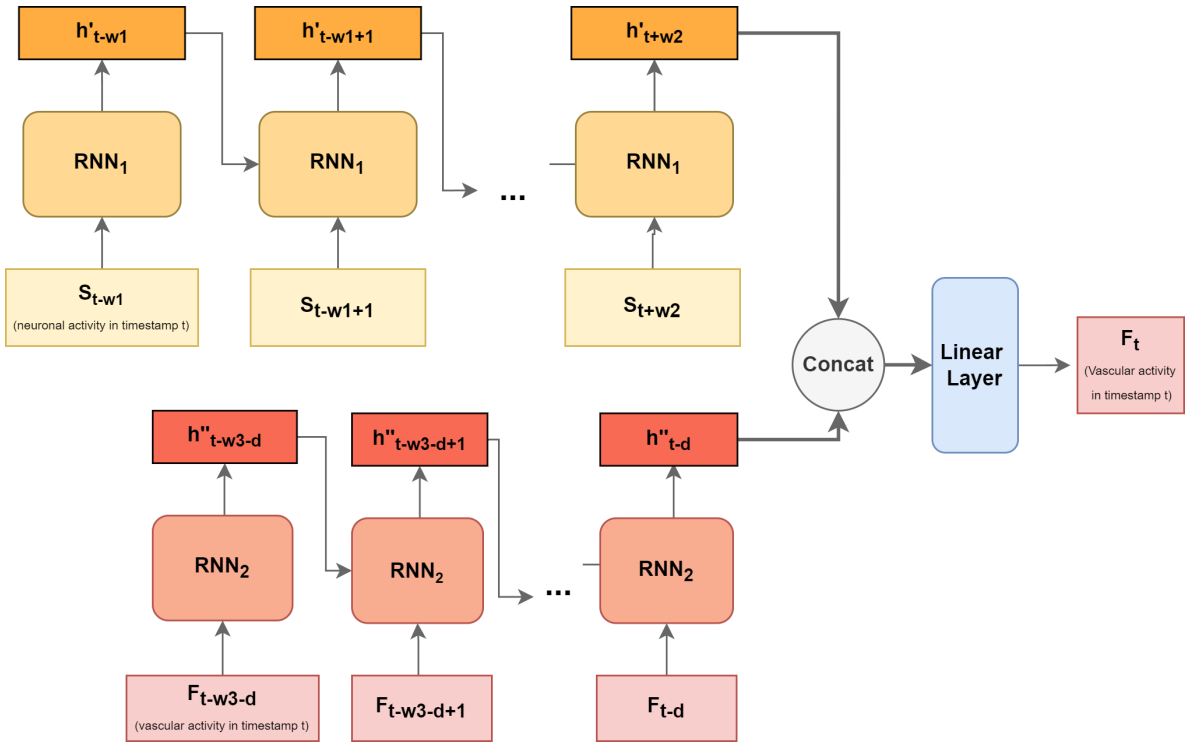

Figure 10: RNN's model architecture. The same RNN instances are recurrently fed with the activity (each time with the following timestamp) and the last output of the RNNs is projected as the prediction of the vascular activity in time t. Parameters w_1, w_2, w_3, d correspond to the past neuronal window, future neuronal window, vascular window and vascular delay accordingly.

[11] Hyperparameters for each RNN: Unidirectional LSTM, hidden-states-size of 500.

[12] Feeding the model with ground truth instead of its predictions, also called "Teacher Forcing", is a common training scheme in NLP applications of RNNs. We expand it to inference time as well.

6.8.1 Experiments

We've experimented[13] with two variations of this model: the first is the full architecture mentioned above, and the second completely ablates the vascular RNN component (that is, we exclude that RNN from the architecture, leaving only the RNN of neural activity). While the dual models are comparable to the *Delayed Vascular* Control, the models ablating the vascular RNN are comparable to the *Mean* Control.

Model	Wind Sizes	MSE	MBKMSE	NRMSE	Run ID
Full Model	(1,50,50), 1	**11.75**	**0.07823**	**0.1580**	14ayrfkf
Full Model	(50,50,50), 1	13.47	0.07902	0.1670	2hyool0r
Full Model	(100,500,200), 1	14.226	0.08027	0.1717	13zyintm
Full Model	(50,50,50), 2	14.255	0.07945	0.1711	2rwe6ska
Full Model	(50,50,50), 5	14.481	0.082	0.1734	isad2gr0
Neuronal Only	(0,1,1), -	14.973	0.08317	0.1755	b2eo0xt9
Neuronal Only	(0,50,50), -	14.970	0.08307	0.1752	l71b1tbm
Neuronal Only	(0,100,100), -	14.940	0.08238	0.1749	2h7tyuxj
Neuronal Only	(0,200,100), -	14.920	0.08186	0.1745	psp9vm64
Neuronal Only	(0,500,200), -	**14.852**	**0.08126**	**0.1742**	30ehsnd8

Table 6: RNN performances on validation set of Dataset-1. Window sizes correspond to - *(Vascular past window size, Neuronal past window size, Neuronal future window size), Vascular Delay.*

It should be noted that the RNN models took the most time to optimize, in particular the variations with large sequences as inputs, with some that were optimized over a week or more[14].

6.8.2 Conclusions

- **RNN is the most suitable model for the task:** not only does the RNN model outperform the control baselines, it also outperforms the rest of the models tested. This is no surprise since, as the papers we cited suggest, RNN's architecture is advantageous for sequence-oriented tasks. Nevertheless, RNN prediction of vascular activity merely from neuronal activity does not offer a significant improvement over other models.

- **Increasing the Neuronal Window Sizes:** As opposed to results in previous models, the RNN model responds well when increasing the window size. This is another indication that RNNs are more suitable for the task, since it handles long activity window lengths better. In addition, this might indicate that lengthy windows are essential as inputs to the HRF.

  However, as for the past vascular window sizes, we can observe from the results that a long vascular window hurts performance. This can be attributed to the fact that the most significant information in the vascular activity can be found in the very end of the window (that is, closer to the predicted timestamp), as can also be seen in the Vascular Auto-Correlation plot (5.2).

- **Delayed Vascular is helpful:** We can notice promising improvements over the Delayed Vascular Control and the rest of our models with vascular delay. This is another indication that the delayed vascular activity contains rich information and may serve as an important addition to the HRF input. Evidently, for vascular delay of value 1, the RNN (11.87 MSE) drastically improves the corresponding Delayed Control (13.41 MSE) and Linear Regression baseline (12.87 MSE). And as for delay value of 2 the RNN (14.2 MSE) improves its corresponding control (18.5 MSE) as well.

In conclusion, We believe that given the sufficient computing power required, scaling the input scope even more as well as training larger RNN models could yield even better results.

[13]Experiments were done on Dataset-1, as it is smaller then Dataset-2, and optimizing a RNN is a lengthy process.

[14]This slow training process is expected and due to fact that the optimization done by back-propagating through all the recurrent calculations.

6.9 XGBoost

Although Extreme Gradient Boosting (XGB) has proven to be a powerful machine learning tool [CG16] for many tasks, including regression, It failed to compete against even the control models defined in this paper. We attempted to design a tabular feature vector for this task without observing a remarkable improvement. The tested model was given only neural activity as input, and the best MSE achieved was > 20 (we don't detail the full results here).

6.10 Experiments Summary

To summarize the results, we will present best ones in the following tables. As mentioned above, our models can be divided to 3 groups. We shall compare between the control, baseline and best models of *each* group separately.

6.10.1 Full Models

For models that are given the classic input scope - neuronal activity and full vascular activity preceding the prediction, the best prediction strategy is to rely on the closest vascular activity, and as we've seen (section 6.5) - these models indeed do this. Since we wanted to focus on the neuro-vascular interface rather than the vascular-vascular interface we did not focus on this group.

Model	MSE	MBKMSE	NRMSE	Run ID
Persist. Control	7.145	**0.0528**	0.1207	1125n2kn
Lin. Regression	**6.681**	0.0561	**0.1175**	oeiebriu

Table 7: Performance of full Models on the validation-set of *Dataset-1*.

6.10.2 Neuronal-Input-Only Models

Setting the input to be the neural activity only presents the most difficult framework for vascular prediction. In order to decrease the difficulty slightly while still learning meaningful patterns, we give some of the models the mean vascular activity (taken over all timestamps, meaning a single value for each blood-vessel). Since the main goal is estimating neuronal to vascular activity mapping, we focused on these models.

Model	MSE	MBKMSE	NRMSE	Run ID
Mean Control	15.226	0.07979	0.1773	ef1yzky2
Lin. Regression	15.048	0.08321	0.1759	3nhb3p4u
LNN	14.969	0.08383	0.1756	6zhfm5zw
EHRF	15.012	0.08302	**0.1741**	16gxm97j
RNN	**14.852**	**0.08126**	0.1742	30ehsnd8

Table 8: Performance of Neuronal-Input-Only Models on validation-set of *Dataset-1*.

6.10.3 Delayed Vascular Models

On the one hand, providing the model with full vascular activity yields undesirable focus on the vascular interface. On the other hand, completely omitting vascular activity from the model's input yields relatively poor performance.There's a need to balance these two concerns, and therefore in these models we insert a *delayed* window of the vascular activity to the input.

Model	MSE	MBKMSE	NRMSE	Run ID
Delayed Persis. Control	13.410	**0.07251**	0.1655	amemyah5
Lin. Regression	12.878	0.07740	0.1615	h5hnl936
RNN	**11.750**	0.07823	**0.1580**	14ayrfkf

Table 9: Performance of "Delayed" Models (with lag value of $d = 1$) on validation-set of *Dataset-1*.

7 Summary

We started with providing in depth analysis of the data (5.2), from which we concluded that the neuronal and vascular activities are rapidly fluctuating time series that hardly cross-correlate[15] with each other, as opposed to the strong auto-correlation of the vascular activity. From our data exploration we derived a collection of suitable metrics to evaluate our models (6.3) as well as our feature modeling (6.2.2).

We review our straight-forward feature engineering method used across the models, and divide the models to 3 groups depending on their vascular activity input scope - models with full, delayed and no vascular activity (the last group includes solely neuronal activity input). We define naive *Control* baselines comparable to our proposed models (6.4). We experiment with different Machine Learning architectures to tackle this task, from Linear Regression to Recurrent Neural Networks. From our rich set of experiments we derive a few important conclusions:

- Simple modeling, such as Linear Regression, or even sophisticated linear combination, is **not expressive enough** to capture the complexity of the HRF. That is, we assume that the HRF includes relations between the vascular and neuronal activity that are much more complicated than those expressed by simple function (e.g. linear).
- While we didn't find a single **HRF input structure** which is optimal, when trying to challenge many of our models with neuronal activity alone, we observed the importance of combining the **vascular activity input** (either a whole vascular activity lag or at least by combining an activity summary, e.g. mean). Moreover, we showed that inserting a **delayed vascular activity** to the input doesn't encourage the model to simply predict the latest vascular activity as much as supplying the full vascular activity as input does. In addition we did notice slight improvement when **increasing the window sizes**, in models that are suitable for large inputs (e.g. RNNs). This might indicate that the HRF requires a long input window of neuronal activity (that is, at least 100 timestamps).
- We found **Recurrent Neural Networks** to be the best models for this task - they not only outperform the rest of our models, but they also respond better to scaling the length of the activity windows, which we found to be helpful.
- The second best model - **Deep Linear Neural Networks** - does significantly lag in performance behind the best, but its simplicity may allow easier exploration of the learned parameters and their meaning, in order to answer some open questions in the biological research.

In conclusion, we lay a wide foundation of machine learning results for the task of estimating the HRF which can be used immediately in research to explore an approximation of it. Additionally, these results, with the formal framework we presented, may be used in the long run as a basis to build more machine learning models upon, when larger and more varied datasets (from other specimen, different areas in the brain or different amount of neurons and/or blood-vessels) are available.

[15] By correlation we refer to the statistical *Pearson* correlation coefficient.

References

[ACHL19] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[AGCH19] Sanjeev Arora, Noah Golowich, Nadav Cohen, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. 2019. Publisher Copyright: © 7th International Conference on Learning Representations, ICLR 2019. All Rights Reserved.; null ; Conference date: 06-05-2019 Through 09-05-2019.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[GES01] Felix Gers, Douglas Eck, and Jürgen Schmidhuber. Applying lstm to time series predictable through time-window approaches. pages 669–676, 08 2001.

[HBB19] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *CoRR*, abs/1909.00590, 2019.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[KAA+21] Serkan Kiranyaz, Onur Avci, Osama Abdeljaber, Turker Ince, Moncef Gabbouj, and Daniel J. Inman. 1d convolutional neural networks and applications: A survey. *Mechanical Systems and Signal Processing*, 151:107398, 2021.

[Pet19] Gábor Petneházi. Recurrent neural networks for time series forecasting. *CoRR*, abs/1901.00069, 2019.