Neural net as an inverse solution to MEG signal

August 9, 2022

1 Basic idea

Neural nets in the form of multi-layer perceptrons or more intricate versions have been shown to be capable of performing many tasks, which would be very hard to program a computer to specifically do. While neural nets fall into a broad category of machine learning algorithms, what makes them interesting is that they are designed to imitate the brain. This raises the (surprisingly obvious) question of whether the neuron connections and locations in the brain could be reverse engineered by "placing" a neural net into a mesh representing the outer surface of the brain. Basic principle would thus be to train a neural net to do the same task as is asked from subjects in an experiment. During the experiment MEG data can be measured from the subjects. MEG is known to have excellent temporal resolution, but very limited spatial resolution. This is because MEG measures the magnetic field caused by action and postsynaptic potentials from thousands of simultaneously firing neurons from a relatively large region. And even though it creates an ill-posed inverse problem, as we will see in following sections, a solution should be possible to find.

2 Simulation of synaptic responses

2.1 Electric currents in neurons

The electric signal associated with the firing of a neuron consist of the presynaptic action potential as well as postsynaptic inhibitory and excitatory potentials. Out of these the action potentials with their rapid change of membrane potential do contribute to the high frequency parts in the measurements, but the postsynaptic dendrite current is the main source for the measured magnetic field [1][2]. It also happens to be very easy to model as it diminishes exponentially as a function of distance [1]:

$$I(x) = I_0 e^{-x/\lambda} \tag{1}$$

where x is the distance from source (synapse), I_0 is the initial current and λ is the length constant. Most importantly for our use we can make the assumptions that $a_i^{(l)} \propto I_0$ and $d_{i,j}^{(l)} \propto \lambda$, where $a_i^{(l)}$ is the activation of neuron i of the multi-layer perceptron (MLP) on layer l and $d_{i,j}^{(l)}$ is the distance between neuron i on layer l and neuron j of layer $l-1$ (of the MLP). The speed at which the signal propagates through the neuron can vary from 1 to 100 m/s, but moves at mostly constant speed [3]. Thus also holds that for each current $x \propto t$. This thinking is used in the current model, but it would probably be more biologically "accurate" if λ was tied to the weight or some other implicit quality of the neuron (or neuron connection).

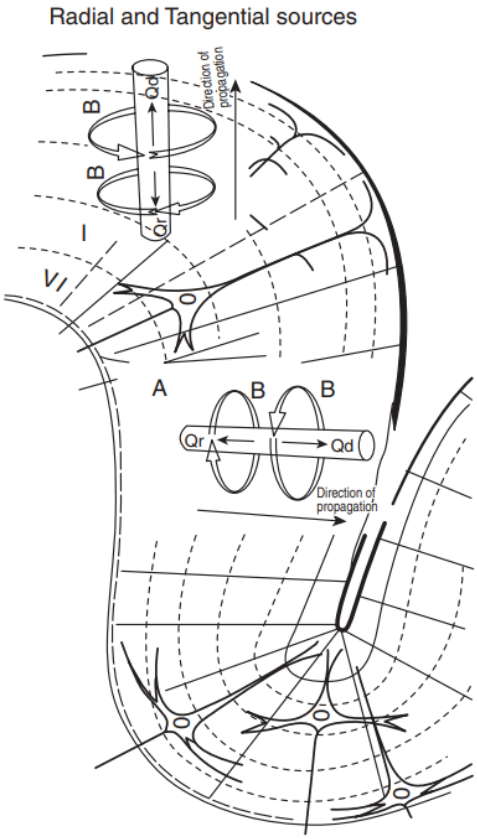

Figure 1: Schematic drawing showing the alignment of neurons and by extension the currents and magnetic fields. From [5]

The calculated distance would then simply shift the current onset time-wise as the signal should of course first travel to the synapse. All of this is of course a huge simplification and considers only the postsynaptic potential in it's most simple form, but one that should yield a usable initial approximation that can be improved later on. For a more thorough model see [4].

2.2 Resulting magnetic fields

The magnetic field caused by a current is a loop perpendicular to the conductor in accordance to Biot-Savart law. Thus the alignment of the neuron affects the measured magnetic field (or whether it gets measured at all). Interestingly most neurons align themselves perpendicular to the cortex [1][2]. In contrast the dendrite branches can be aligned arbitrarily. In this model the dendrite connections are modeled running on the surface of the mesh (or should be modeled, but as calculating the distance between two points on a surface is surprisingly challenging the connection is actually just a straight line, and the distance is calculated using Euclidean distance). This means that the neurons actually contributing to the MEG signals are located on the wall of the sulcus (see Figure 1). The accurate and actually physically correct magnetic field caused by a changing current isn't computationally trivial to calculate so in the initial implementation we will use Biot-Savart law:

$$B(r,d) = \frac{\mu_0 I_{i,j}^{(l)}(d)}{2\pi r} \tag{2}$$

where r is the distance from line conductors point d, μ_0 is the permeability of free space, $I_{i,j}^{(l)}$ is the current function for neuron i on layer l of the MLP caused by neuron j on layer $l-1$ of the MLP. Biot-Savart law (in this form) is really only applicable to infinite line conductors with a steady current, neither of which is the case here. More accurate magnetic fields described in [1][2] can be used in future models. Note, that equation 2 doesn't actually provide any vector coordinates. The location of the magnetic field strength should hence be deduced from the coordinates and the alignment of the neuron. That is if we know the coordinates of the wanted neuron $\vec{x}$ as well as d and r, the point at which the magnetic field strength, as given by equation 2, is measured is:

$$\vec{x_0} = \vec{x} + d\vec{n_1} + r\vec{n_2} \tag{3}$$

where $\vec{n_1}$ is a unit vector normal to the mesh surface at point $\vec{x}$ and $\vec{n_2}$ is a unit vector normal to vector $\vec{n_1}$ at point $\vec{x} + d\vec{n_1}$. Note, as $\vec{n_1}$ is just a vector rather than a plane the direction of $\vec{n_2}$ is ambiguous as it could be anywhere in a circle around $\vec{n_1}$. Thus a better way to calculate it would be:

$$\theta = \frac{\pi}{2} - \arccos\left(\frac{(\vec{x} + d\vec{n}_1) \cdot (\vec{x} + d\vec{n}_1 + \vec{x}_0)}{|\vec{x} + d\vec{n}_1||\vec{x} + d\vec{n}_1 + \vec{x}_0|}\right) < \varepsilon \tag{4}$$

where $\varepsilon \approx 0$ can be considered as a margin of error as it's very unlikely that any point where the magnetic field strength is actually calculated would be at an exactly 90 degree angle from the normal vector of any given neuron. In the computer model this is further simplified by assuming that the current is found at $\vec{x}$ rather than offset from it by distance d.

2.3 SQUID-sensors

From [6] we know that the magnetic field measured by a SQUID-sensor i would be:

$$b_i = \int_A \mathbf{B}(\mathbf{r}) \cdot d\mathbf{A} \approx \sum_{k=1}^{N} w_k \mathbf{B}(\mathbf{r}_k) \cdot \mathbf{n}_k \tag{5}$$

where in the approximate solution N is the number of points in the sensor area at which magnetic field strength is calculated, $\mathbf{B}(\mathbf{r}_k)$ is the magnetic field strength at $\mathbf{r}_k$, w_k is the surface area of the point k and $\mathbf{n}_k$ a unit normal vector, although normal to what I'm not quite sure. As I'm not quite sure what exactly is calculated in the approximate sum (and as it involves vector operations which would increase computational complexity) I chose to use a very rough approximation:

$$b_i \approx \sum_{k=1}^{N} B(r, d) \tag{6}$$

where $B(r,d)$ is calculated using equation 2 over multiple points k in the sensor area. There is no need for constant like w_k to increase surface area as from equation 4 we already allow for wanted margin of error.

2.4 Modeling neurons

Neural nets are defined by a few equations out of which the most interesting for us is:

$$\vec{a}^{(l+1)} = \sigma(\mathbf{W}\vec{a}^{(l)} + \vec{b}) \tag{7}$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function that we can use to scale a real number to $[0, 1]$, $\mathbf{W}$ is a matrix of weights, $\vec{a}^{(l)}$ is a vector of activations of layer l and and $\vec{b}$ is vector of biases. We can simplify this to the case of a single neuron rather than a whole layer:

$$a_i^{(l+1)} = \sigma(\sum_{j=0}^{n-1} w_{l,j} a_j^{(l)} + b_l) \tag{8}$$

The activation of any one neuron is dependent on the activations and weights of all of the neurons from previous layer. However, unlike in the case of McCulloch-Pitts neurons, which fire once if the combined inputs from previous layer exceed some threshold, in this implementation we allow all of the neurons from the previous layer to contribute by summing up individual currents caused by them. The activation onto the next layer is released only after the sum impulse from previous layer has completely died out. This is consciously not a biologically accurate simplification, but allows for far more interesting and complicated currents and thus magnetic fields. When the neural nets are scaled up in size this simplification would be removed.

3 Optimization of neuron positions

3.1 Basic premise

The optimization is really what (might) yield the interesting results. By shifting the locations of the neurons we can not only change how the magnetic field is measured (if at all), but at which moment in time. This should provide a more or less infinite amount of possible simulated signals with already a relatively limited number of neurons in the network. Thus if we are given some signal, more specifically the MEG measurement, the simulated signal could be made to match it by maximizing the correlation between the signals. However, optimizing the neural net to match a single measurement wouldn't provide accurate positioning to all possible inputs. Thus the more precise objective of our optimization is to maximize the average correlation between selected inputs and also multiple sensors if more than one is used.

3.2 Problem formulation

More officially the problem is to maximize the average correlation, by changing the coordinates of the neurons subject to some mesh that represents the cortex. Correlation can be calculated from:

$$\rho(\vec{b},\vec{s}) = \frac{n\sum_{i=0}^{n-1} b_i s_i - \sum_{i=0}^{n-1} b_i \sum_{i=0}^{n-1} s_i}{\sqrt{n\sum_{i=0}^{n-1} b_i^2 - (\sum_{i=0}^{n-1} b_i)^2}\sqrt{n\sum_{i=0}^{n-1} s_i^2 - (\sum_{i=0}^{n-1} s_i)^2}} \tag{9}$$

where $\vec{b}$ is the simulated signal and $\vec{s}$ the measured signal. Knowing this the problem is:

$$\begin{aligned} \text{max.} \quad & \frac{\sum_{i=1}^{n}\sum_{j=1}^{l} \rho(\mathbf{B}_{i\cdot l+j},\mathbf{S}_{i\cdot l+j})}{n} \\ \text{s.t.:} \quad & d_k(\mathbf{X}_k) = 0 \ , \ k = 0, 1, ..., m-1 \end{aligned} \tag{10}$$

where $\mathbf{B}$ is a matrix where row $i \cdot l + j$ contains the simulated signal from input i and sensor j (same for $\mathbf{S}$ but for measured signals), n is the number of inputs, l the number of sensors, $d(\vec{x})$ is a function that calculates the distance from brain mesh

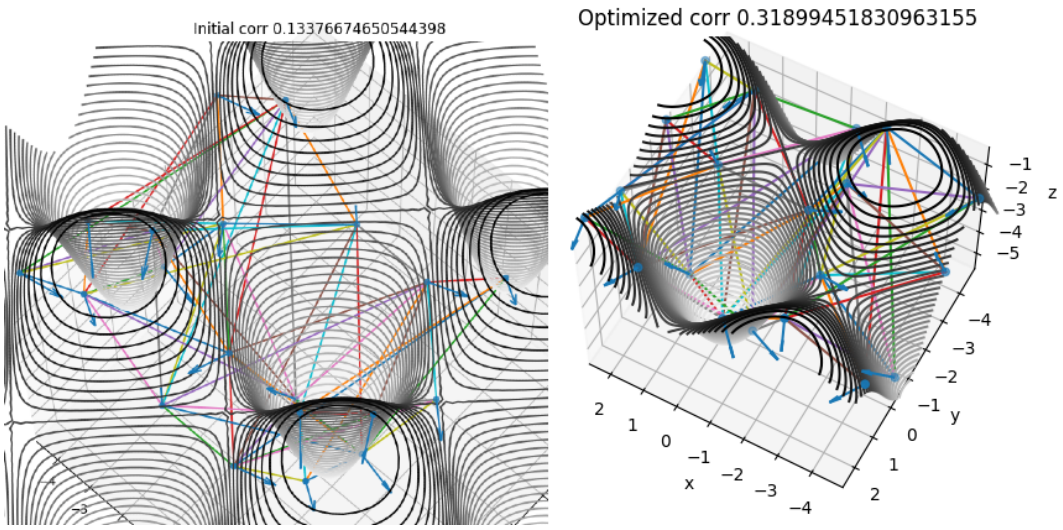

Figure 2: The initial and optimized neuron placements for 5x4 network

and finally $\mathbf{X}$ is a matrix where each column corresponds to the coordinates of some neuron.

4 Results

4.1 Basic premise

The code used to run the simulations can be found at my github (https://github.com/krantamaki/mlp_meg_inverse). There is some clear limitations to it currently. Firstly instead of a mesh of the brain the surface is defined by the mathematical function:

$$z(x, y) = -\frac{3}{2} \cdot (\cos^2(\frac{x}{2}) + \cos^2(\frac{y}{2}))^2 \tag{11}$$

This function does provide pretty decent height variations and thus walls for the neurons, from where magnetic field is measurable. Secondly the "real" signal to which neurons where fitted was just a sine curve with some added noise to it. This is otherwise fine, but as I took the average of five sensors in a cross pattern, each with the same signal, it was clear that some were better fitted than other. Thirdly the coefficients used in calculating the directly proportional values were not grounded in reality. This is understandable as the "real" signal had a duration of 1000 ms, but actually accurate signal duration for a neuron would be in range of tens of milliseconds and thus no matter how they are fitted the result would ne useless. The duration of the neuron signals was "stretched" by altering λ and the speed of propagation. Finally the neural net is not actually trained to do anything as the signal it's fitted to is also arbitrary. Thus the weights and biases were just randomized, which might mean that in the case that neural net was actually trained to do something some specific layers or neurons might be more active than in this version.

4.2 Results for 5x4 neural net

In figure 2 we can see the initial and optimized neuron locations of a 5 layer neural net where each layer had 4 neurons. The first layer doesn't actually contribute to

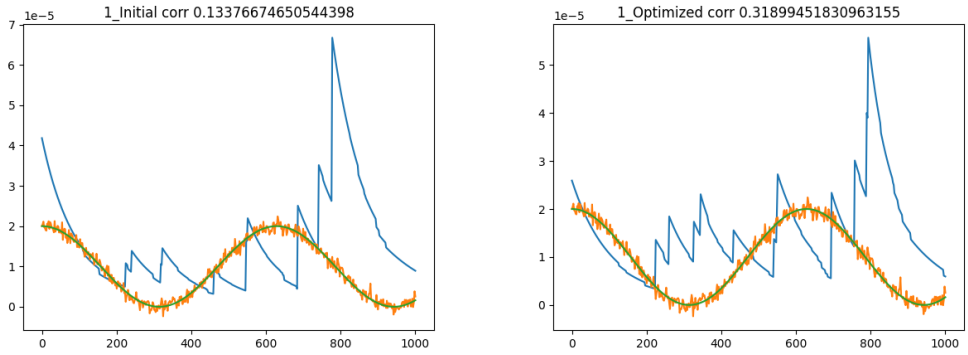

Figure 3: The initial and optimized signals on sensor 1. Blue curve is simulated signal, green clean signal and orange noisy signal. The correlation in the title of the figures is the average correlation and not specific to the sensor.

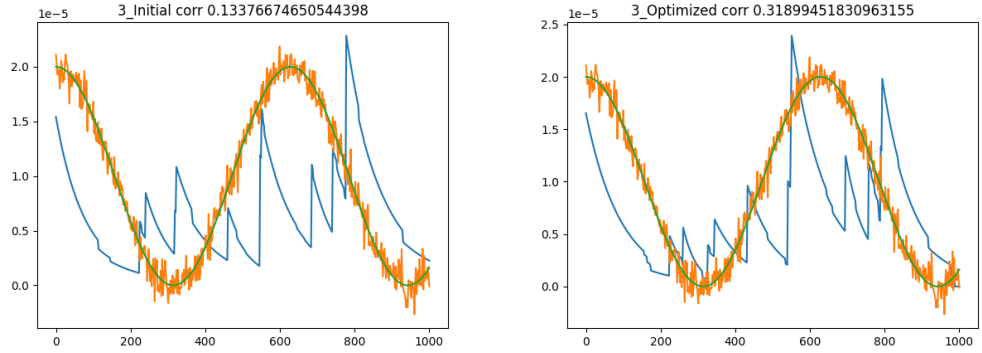

Figure 4: The initial and optimized signals on sensor 3

the magnetic field as the currents are calculated using previous layers activations (as discussed in section 2.4). In the figure arrows describe the neuron alignment, points the neuron position on surface and lines the connections between neurons. However the figure still remains hard to read, but what's important is the fact that optimized average correlation is almost three times higher than the initial one. This by no means means that it's good as we'll see in other figures, but it is improvement none the less. In figures 3 and 4 we can see the initial and optimized signals for two of the sensors. And while there are some slight improvements made the optimized signal is by no means good.

4.3 Results for 6x5 neural net

One of the reasons why the fitting results for the 5x4 neural net had limited success was the low number of layers and neurons in general, which ends up leaving just a few very prominent spikes in the signal. The slightly larger neural net of six layers of five neurons should do a little better then. And I would say this is the case although the correlation value doesn't reflect it. Still the optimized correlation is nearly double that of the initial one. But as we can see from figures 6-10 the fitting seems to be much better and consistent across all sensors. In the case of figure 6 the optimized signal seems pretty well fitted to the second "bump" of the sine curve while the initial one

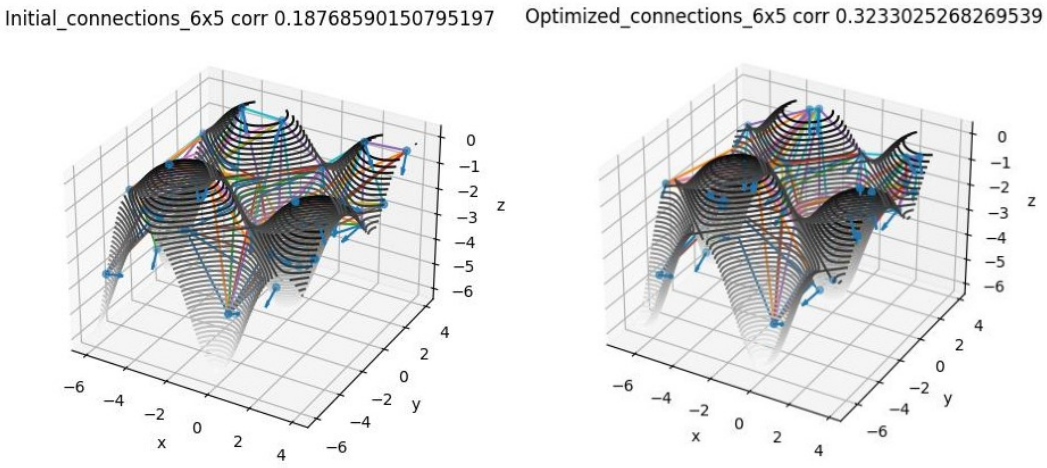

Figure 5: The initial and optimized neuron placements for 6x5 network

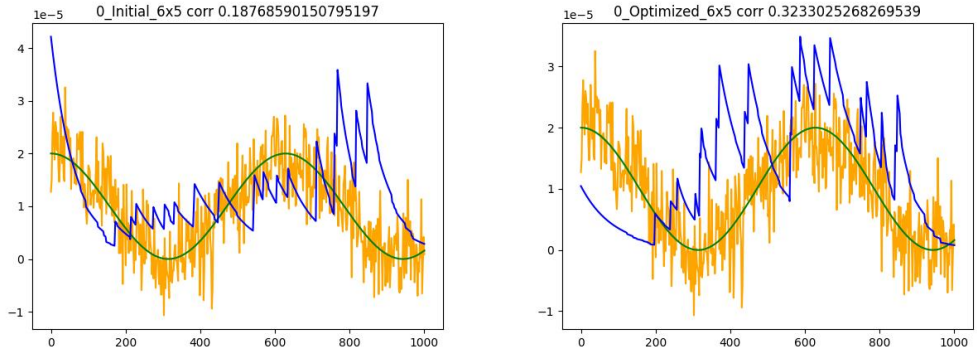

Figure 6: The initial and optimized signals on sensor 0

had pretty steady rate of growth and then sudden spike at the end. However figure 10 seems to show the best improvement going from probably a relatively high negative correlation to a much more reasonable shape.

4.4 Discussion

It seems that with increase in layers and neurons we get far better fitting quality. And I have confidence that if the number of layers (and neurons) was increased to a large enough level the results could be very good as it could fix the main issue currently harming the signal, that of the very prominent spikes caused by stretching the signal. As there are more layers we can get longer total signal while keeping the individual signals very shortlived. In figure 11 we can see the initial signal measured by sensor 4, but in "squashed" form accomplished by changing the speed and length coefficients. This signal seems to consist of just two more unified spikes rather than multiple individual ones. With more layers similarly united signal could be extended to a far longer duration and could possibly lead to a far better fitting result. Of course increasing the number of layers and neurons increases the computational costs as well, and as optimizing the 6x5 neural net took my computer around 12 hours (and ran out of iterations rather than finding a actual optimum) the time and computational resources needed for optimizing locations of thousands

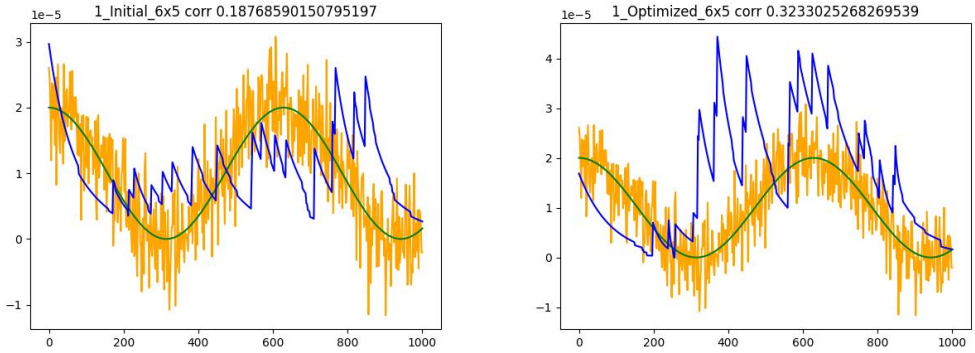

Figure 7: The initial and optimized signals on sensor 1

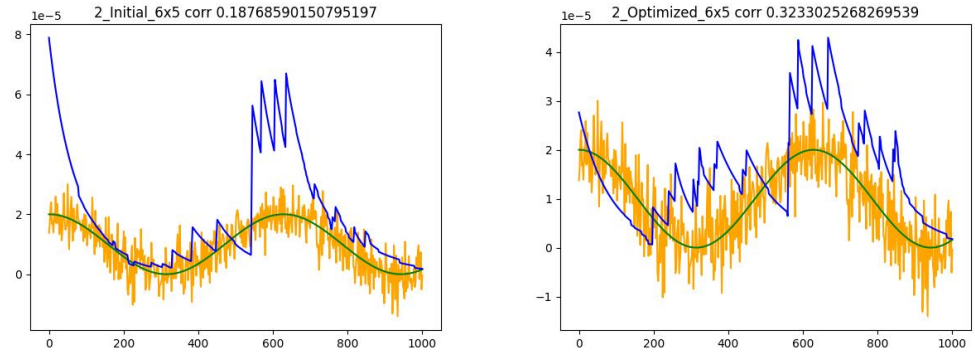

Figure 8: The initial and optimized signals on sensor 2

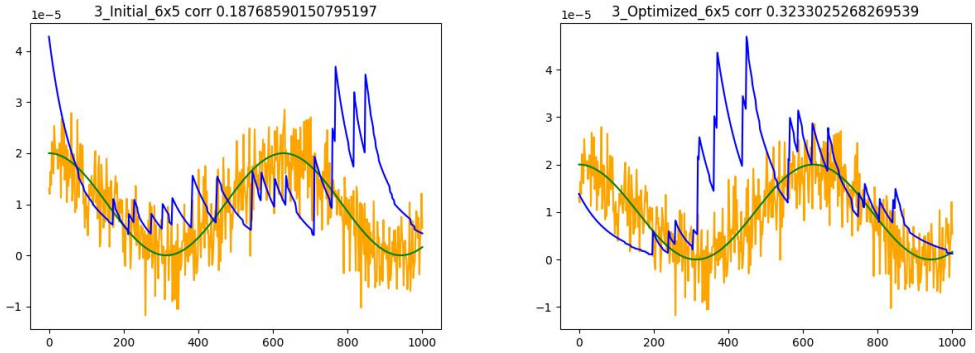

Figure 9: The initial and optimized signals on sensor 3

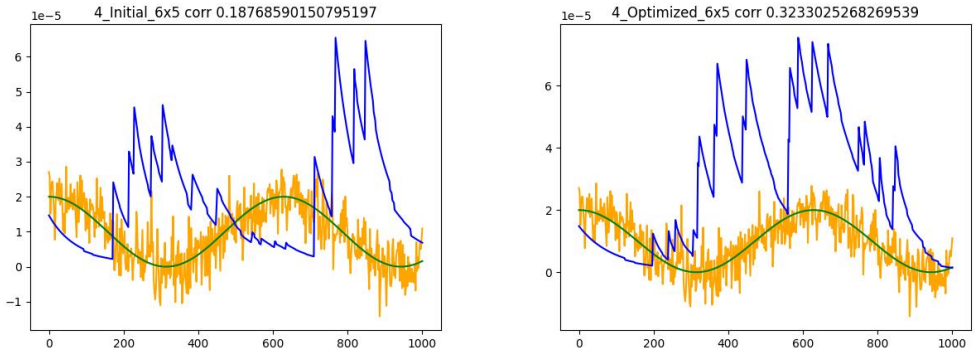

Figure 10: The initial and optimized signals on sensor 4

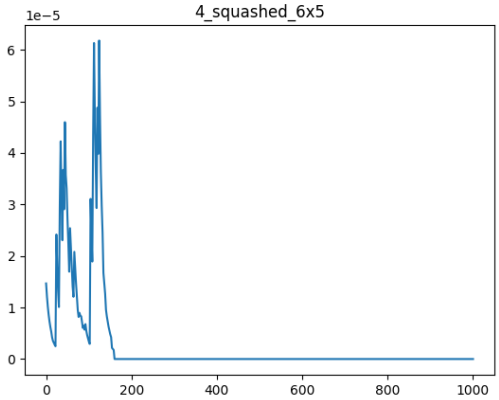

Figure 11: Shorter signal measured by sensor 4

if not tens of thousands of neurons would be very vast indeed. But if the code was well optimized and in parallelizable form the optimization should be doable by large computer clusters.

References

[1] Brain Signals: Physics and Mathematics of MEG and EEG

[2] MEG: An Introduction to Methods, section 1

[3] Principles of Neural Science, section 2

[4] Fundamentals of Computational Neuroscience, section 2

[5] MEG: An Introduction to Methods, Figure 1-3

[6] MEG: An Introduction to Methods, section 2