university of groningen / faculty of science and engineering

# Dynamic coding in a neural model of activity-silent working memory

Bachelor's Project Thesis

Chiel Wijs, s3199886, c.wijs@student.rug.nl
Supervisor: Dr J.P. Borst

**Abstract:** Temporary strengthening of neural connections has been shown to be a mechanism capable of storing items in working memory without the need for persistent neural activity. While this type of activity-silent working memory does not depend on neural activity for the maintenance of a working memory item, it does for the encoding of that item. Moreover, the neural state, a snapshot of neural connectivity, rapidly changes during this encoding period, a phenomenon called dynamic coding. This study expanded on an existing neural model, that used short-term synaptic plasticity to model activity-silent working memory, by implementing a simplification of neural connectivity from the human visual system into the sensory part of the model. Analysis of the functional model shows that the resulting distributed response latency, in combination with the short-term synaptic plasticity mechanism, produces dynamic coding measures similar to those seen from analysis of human EEG data. Performance of the model decreases slightly compared to human performance when items are required to be remembered over a longer period of time. Implementation of attention control could reduce the differences we see in dynamic coding measures between the model and human subjects.

## 1 Introduction

Working memory (WM) is the cognitive process by which we retain information over a short span of time, keeping that information in such a state that it is readily available for use (Baddeley, 2003). The prevalent theory has long been that the retention of information in working memory is the result of persistent neural activity (Goldman-Rakic, 1995; Curtis and D'Esposito, 2003). Recent studies like Lundqvist, Herman, and Miller (2018) and Sreenivasan, Curtis, and D'Esposito (2014) show different results; they found that information maintenance in WM is not necessarily accompanied by persistent neural activity.

Pals et al. (2020) showed that such activity-silent WM could be modelled using the mechanism of short-term synaptic plasticity (STSP) (Zucker and Regehr, 2002). Their spiking-neuron model performed a retrospective cue (retro-cue) delayed-response task (Figure 1.1) (Wolff, Jochim, Akyürek, and Stokes, 2017). During such a task a participant is shown a stimulus, followed by a delay period and a subsequent probe to which the participant must compare the stimulus. Pals et al. used their model to simulate the experiments from Wolff et al.. Their model matched the human behavioural data from the article, making it a functional model.

STSP causes temporary changes to neural connectivity by facilitating (increasing) synaptic effi-

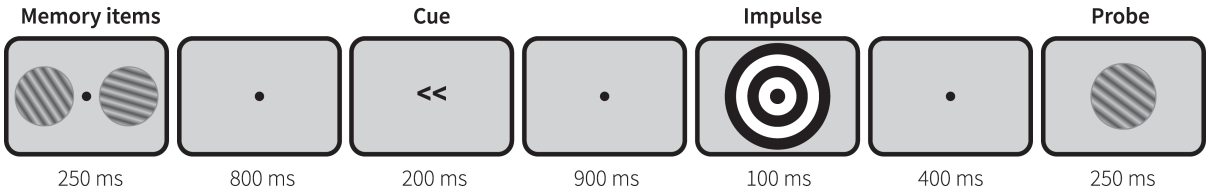


**Figure 1.1: Experiment 1; a retro-cue delayed-response task. The participant has to decide whether a probe shows a clockwise or counter-clockwise rotation compared to a stimulus. A cue indicates to the participant which of the two stimuli will be probed. Reprinted from Pals et al. (2020)**

cacy in areas of the brain related to, but not limited to, WM (Wang, Markram, Goodman, Berger, Ma, and Goldman-Rakic, 2006). This facilitation of connections is the net effect of a temporary facilitation of synaptic efficacy as a result of calcium buildup in the presynaptic terminal, and a shorter lasting depression of synaptic efficacy as a result of resource (read: neurotransmitter) depletion, both due to neural spiking. This mechanism maintains information in WM such that, when a novel stimulus is presented, the activity resulting from these facilitated connections represents the stimulus that was being retained during a delay.

A momentary capture of the state of these connections is called the neural state. In the context of activity-silent WM this is called a *hidden* neural state, because they are immeasurable during activity-silence, and can only be analyzed after excitation.

Wolff et al. (2017) performed a cross-temporal decoding analysis (CTDA) on their EEG data, a type of decoding analysis that uses a sliding temporal window, such that it can be used to determine the consistency of the activation of a neural population. From their analysis they found that decodability was high at a time point $t_1$ when the decoding classifier was trained on a time point $t_2$ that was close or the same as $t_1$. Poor decodability was found when $t_1$ and $t_2$ were different, non-neighboring time points.

This phenomenon is the focus point of this paper and is called dynamic coding: The neural state that stores a static WM item changes over time, while maintaining a representation of that item (Meyers, Freedman, Kreiman, Miller, and Poggio, 2008; Stokes, 2015). From a CTDA, where $t_1$ and $t_2$ are set as $x$- and $y$-axis, we would see high-decodability at the points on the graph where $t_1$ equals $t_2$. We would see this collection of points as a line across the diagonal of the graph. The counterpart to this phenomenon would be static, or sustained, coding of information where the neural state remains the same for the period that the item could be decoded. For this type of coding, a CTDA would show additional high decodability values at points where $t_1$ does not equal $t_2$. In the graph this would show as a square of high decodability values.

The model from Pals et al. (2020) does not explain the dynamic coding patterns as we them from human data. One goal of this paper was to extend the model such that it can explain these dynamic coding patters. Wolff et al. (2017, p.870) proposed that "Activity-dependent transformations in hidden states determine the temporary coding properties of memory networks, i.e., dynamic coding". One of the mechanism that causes such activity-dependent transformations is STSP, the mechanism implemented by Pals et al.. Therefore, this model was chosen as the basis for our own model.

## 1.1 Sensory connectivity

While Pals et al. (2020) their model behavior matched the results from Wolff et al. (2017), the decoding analysis of the period following stimulus presentation showed results different from analysis of the human data. This decoding analysis determines how we well we can decode the WM item from a neural signal [1]. For the model we saw an almost instantaneous drop in decodability after stimulus presentation. Conversely, decodability declined more gradually over time for the human subjects. The second goal of this paper was to extend the sensory part of the model from Pals et al. such that it can explain the gradual decline in activity we see from human data.

Connectivity in the sensory part of the model from Pals et al. (2020) was only in one direction and without any delay. While functional, this simple connection cannot produce the patterns seen from neural activity when it traverses across the human brain.

Lamme and Roelfsema (2000) studied the temporal and spatial characteristics of neural activity patterns caused by a visually presented stimuli in the brains of macaque monkeys. In the feedforward sweep information travels from low-level visual areas, through high-level areas into the rest of the brain. One of the areas that visual information will reach by this route is the prefrontal cortex (PFC), which is associated with WM (Wang et al., 2006). Lamme and Roelfsema identified three types of connections within the visual system: feedforward connections that provide input from lower level areas; parallel connections which receive inputs from neurons at the same level; and feedback connections that provide input from higher level neurons. As a

[1] This analysis also shows up as the diagonal of a CTDA, where the time of training and testing is at the same time point.

result of the various connections, pieces of stimulus-related information reach brain areas like the PFC across a latency distribution, where some pieces of information reach a brain area earlier than others.

### 1.2 Current study

In this study a simplified version of the connectivity of the human visual system was implemented into the sensory part of the neural model of activity-silent working memory made by Pals et al. (2020). The goal was to create a model that we can use to explain two neural phenomena as seen from human data presented by Wolff et al. (2017). The first phenomenon is the dynamic coding that we see during encoding of a WM item. We examined this dynamic coding using a cross-temporal decoding analysis (CTDA). The second is the gradual decrease in WM content decodability after stimulus presentation, which we examined using a standard decoding analysis. The model would have to produce these phenomena by a distributed latency of brain activity, and the facilitation of neural connection by short-term synaptic plasticity.

## 2 Model

The model was implemented using Nengo[2]: a python library that can be used to build and simulate neural models (Bekolay, Bergstra, Hunsberger, DeWolf, Stewart, Rasmussen, Choo, Voelker, and Eliasmith, 2014). Nengo consists of a front-end and a back-end.

The front-end is used to create a neural model to the users specifications. Models consist of a *Network* that contains different *Nengo objects*: *Nodes*, *Ensembles*, and *Neurons*. These objects can be connected to one another using *Connections* and data can be collected from them using *Probes*. A node is used to provide non-neural input to Nengo objects and an ensemble is a group of neurons. These neurons are of a specified type (e.g. a leaky integrate-and-fire neurons) and the neural activity of an ensemble, which represents information, is stored as a vector. A connection transfers these vectors between objects and allows for transformation of said vectors. Connections can change given certain inputs, and so the model can learn over time.

[2]https://www.nengo.ai/nengo

The back-end is a simulator that takes a network and builds data structures so that we can simulate the neural network, made with the front-end, following specified mathematical rules.

To improve the computational speed of the simulations we use a Nengo back-end called Nengo DL[3]. This simulator uses TensorFlow as its computational framework and allows us to run multiple simulations simultaneously as a batch. Because the model uses a custom neuron type that implements the calcium and resource dynamics, as well as a custom learning rule that uses these calcium and resource values to alter the strength of the connection weights, it is required that we add implementations for these to the Nengo DL back-end. All code discussed can be found on: https://github.com/ChielWijs/Dynamic-Coding-in-Working-Memory.

### 2.1 Model Architecture

Figure 2.1 shows the model architecture of the model used by Pals et al. (2020)[4], indicated by the "*Old Connection*", and the model from this paper.

For the model from Pals et al. (2020), the "Sensory" population transform the visual input into a vector representing neural activity. Activity in the "Memory" population increases the calcium level and depletes the resources of the neurons in this population. The recurrent memory connection is mediated based on these values, storing the neural response associated with the given input. When the probe is presented, the "Comparison" population integrates the information from the "Sensory" population, representing the probe, and the "Memory" population, representing the remembered stimulus. The "Decision" population interprets the difference between the probe and stimulus.

For an in-depth explanation of the methods for converting the visual imagery, calculating the calcium and resource levels in the memory population, and interpreting the activity from the "Comparison module" please see Pals et al. (2020).

The changes made to this model were implemented between the "Input" node and "Sensory" ensemble. Instead of a single connection between these two objects, a new ensemble called "Eye"

[3]https://www.nengo.ai/nengo-dl/

[4]https://github.com/Matthijspals/STSP

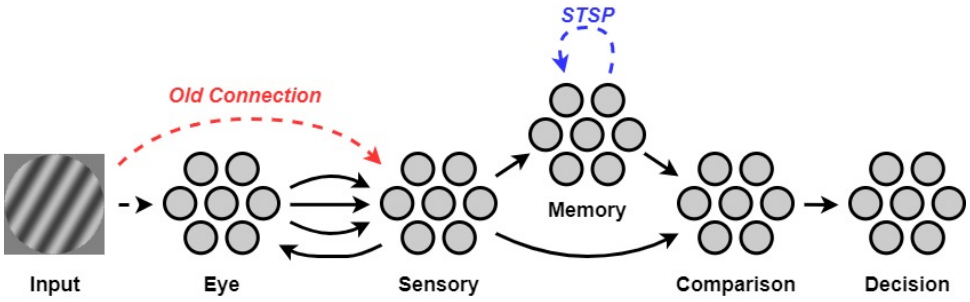


**Figure 2.1: Model architecture. The model is divided in two modules (only one is pictured) representing the two hemispheres. Temporary adaptation of the recurrent memory connection by the STSP mechanism represents storage of a WM item. The "Old connection" indicates the model from Pals et al. (2020). The multiple forward connections between the eye and sensory population indicate the multiple neuron-to-ensemble connections, implemented in the new model, which take their synaptic value from a distribution. A single recurrent connection allows the sensory population to excite the eye population.**

was placed between them. A single connection was made between the "Input" node and the "Eye" ensemble. This connection is the same as the connection between "Input" and "Sensory" in the old model and thus now it is the "Eye" ensemble that is responsible for transforming the input into a vector. The connection between "Eye" and "Sensory" is a more complex connection and the focus point of the new model.

Instead of connecting the "Eye" and "Sensory" *ensemble-to-ensemble*, individual neurons from the "Eye" population were connected to the "Sensory" ensemble using a *neuron-to-ensemble* connection. Each of these connections was given a synaptic delay from a distribution based on the findings from Lamme and Roelfsema (2000). The result is a distributed response latency across the rest of the model with a mean of $141ms$ and an onset of $51ms$. A single recurrent connection between the "Sensory" population and the "Eye" population, in combination with the feedforward *neuron-to-ensemble* connections, implemented the visual system organization as discussed in the introduction.

Two parameters we adjusted to fit the model are: the connection strength of the distributed feedforward connections (FF), the connection strength of the recurrent connection (REC). Changing the strength of a connection is done by multiplying the weights matrix with a scalar. We set these parameters as $FF = 1.50$ and $REC = 0.030$.

The third parameter we adjusted is the amount of noise for certain neural populations (NOISE). For the model from Pals et al. (2020), the tuning curves for the neurons of the "Eye", "Sensory", and "Memory" populations were set to a range such that no spiking occurs when no input is passed to the neural populations. This range is set using the *intercepts* parameter of an ensemble. From Wolff et al. (2017) we see that an EEG signal is not without noise. To implement this finding we added some white noise to the neural signal of these populations. White noise is a type of random noise, where the value added to the signal of a neuron is randomly taken from a distribution at each timestep. We chose a gaussian distribution with $mean = 0$ and $SD = 0.010$. The result is spontaneous spiking in about 15% of the neural population.

Adjusting these three parameters allowed us to create a great variety of results for the CTDA. It is important to note though, that not all combinations of these parameters produce a functional model. In Appendix A a more thorough discussion of these parameters is provided, as well as a figure showing the CTDA's from one of the parameter sweeps.

## 2.2 Nengo DL Implementation

This code discussed in this section can be found in the *stp_dl_implementation.py* file.

**Neuron.** The custom neuron type called *stpLIF*

is an extension of the spiking version of the leaky integrate-and-fire (LIF) neuron model that is part of the Nengo library. To the regular LIF neurons it adds a resource and calcium level, these were implemented using two parameters: the relaxation time constants for the resources and calcium, called *tau_x* and *tau_u* respectively, and the calcium base level *U*. The resource base level does not need to be added as a parameter as it has a value of one. The *step_math* method implements the calculations for the calcium and resource dynamics at each time step. A function called *build_stpLIF* is used to add the *stpLIF* neuron type to the Nengo build process so that we can use it when making the model.

When using the Nengo DL simulator we need an additional class, called a neuron builder, that allows us to specify how to simulate a neuron using TensorFlow. The Nengo DL library contains, in addition to the neuron builders for the regular neuron types, a generic builder that will work for any custom neuron type, like *stpLIF*. However, writing a custom neuron builder improves on build and simulation speed.

In contrast to the *stpLIF* class, which is a child class for a Nengo intrinsic neuron type, the *stpLIFBuilder* class does not inherit from a neuron builder from Nengo DL: The order of inheritance is different for the Nengo DL neuron builders than it is for the Nengo neuron types and this causes issues regarding the absence of certain parameters. To overcome this problem, instead of using inheritance, the relevant lines of code were copied from the neuron builder for a LIF neuron, a class called *LIFBuilder*, from the Nengo DL documentation. These lines of code are under the "### LIF ###" comments in the *stpLIFBuilder* class.

The computational framework used by Nengo DL, TensorFlow, uses what is called a tensor as the structure in which it stores information, similar to how regular Nengo uses NumPy arrays. The metadata of these data structures is represented in the Nengo (DL) simulator by a tensor signal and Nengo signal, respectively.

The *__init__* method of *stpLIFBuilder* creates tensors and tensor signals with the values from the parameters of the *stpLIF* neuron type. It makes use of the *combine* function that creates a tensor signal for each equivalent Nengo signal and then combines those into a single tensor signal. The *op_constant* function is used to build a tensor that represents a constant parameter. The *_step* method is equivalent to the *stpLIF.step_math* method and contains the calculations that need to be performed at each time step. The *build_step* method is responsible for gathering tensors from the tensor signal so that they can be used for calculations. It then sends the updated tensors back into the tensor signal, a process called scattering.

**Learning Rule.** The STP learning rule uses the presynaptic neuron resource and calcium values, initial calcium values (also called U), and initial connection weight values to determine the connection weigths between neurons.

The class *simSTP* is an *Operator*, which means it handles the calculations inside of a simulator. Like the *stpLIF* neuron class the *simSTP* class needs some additional code for us to be able to use it in the front-end. A *LearningRuleType* type class called *STP* is necessary so that the simulator knows which signals it should modify and which signals it can probe. The name of the *LearningRuleType* class is also the name we pass to a connection in the front-end to indicate which learning rule we want to use. The *build_stp* function is used to add the STP rule, calculated by *simSTP*, to the Nengo builder. The *SimSTP* operator receives the various signals it needs for the weight calculations in the *__init__* method. The class contains a number of methods so that it can read out the values from these signals and use them to calculate the new weights in the *make_step* method.

The custom neuron needed a custom neuron builder to be used with Nengo DL and likewise we need a builder for the custom learning rule. This custom builder is called *SimSTPBuilder*. The *__init__* method takes the parameters from *simSTP* and makes tensor signals and tensors out of them using the *combine* function, which we saw in the neuron builder, and the *tf.constant* function respectively. We can not use the *op_constant* function here because we are not reading a constant parameter but the initial value of a signal. However, the *tf.constant* function performs the the actual creating of the tensor inside of the *op_constant* function so the result can be used the same way. The *build_step* method gathers the tensors it needs from the various signals and uses these to calculate the connection weights according to the STP rule. The last thing it does is scatter the adjusted weight tensors back into the tensor signals.

# 3 Method

Experiment 1 (Experiment 1 from Wolff et al. (2017)) was a retro-cue delayed-response task. For this version of the task, participants had to store the angle of a visual grating in working memory, such that they could later determine whether a probe was oriented clockwise or counter-clockwise with respect to that stimulus (Figure 1.1).

Two stimuli were presented simultaneously, followed by a short delay period after which a cue, in the form of an arrow, indicated which one of the stimuli would be probed. The probe was presented after a second, longer, delay period. A generic impulse was presented during the delay period between cue and probe to examine the content of WM, as was done by Wolff et al. (2017).

The model simulates Experiment 1 for 30 participants that each perform 1344 trials, the same as human participants in Wolff et al..

In addition to Experiment 1, generalization of the model was evaluated by simulating a second WM experiment. Experiment 2 (Experiment 2 from Wolff et al. (2017)) was also a delayed-response task.

For this experiment, the participant had to compare the visual gratings of two stimulus-probe pairs (Figure 3.1). Both stimuli were presented simultaneously at the beginning of the experiment and the participant knew beforehand which of the stimuli would be probed first. This translates to the model as the late-tested stimulus being presented at 90% strength. The participant again had to decide whether a probe showed a clockwise or counter-clockwise rotation with respect to the original stimulus. A generic impulse, prior to both probes, was presented to examine the content of WM during the delay period.

The model simulates Experiment 2 for 19 participants that each perform 1728 trials, the same as human participants in Wolff et al..

For an in-depth explanation of the experimental simulation for both Experiments 1 and 2 please see Pals et al. (2020).

## 3.1 Analysis

To analyse the model behaviour, the proportion of clockwise responses was calculated for all angular differences between stimulus and probe. This was done for the single tested stimulus from Experiment 1 and separately for the early and late tested stimuli for Experiment 2.

A decoding analysis was used to determine the decodability of WM content from the signal of the model's memory population during, and shortly after, presentation of a WM item. For the first experiment, a single decodability score is calculated from the combined scores of both memory items. These items are combined because no distinction is made between the two before the cue. For the second experiment, separate decodability scores are calculated for the early and late tested module.

This analysis was also used to determine the decodability of WM content from the same signal around the presentation of the generic impulse, that was presented before each probe.

In addition, a CTDA was used to analyse whether the neural state was static or dynamic during encoding of a WM item. Figure 3.2 shows possible results for a CTDA. The axes represent the times at which a decoder is trained or tested. Decodability is indicated in red and no decodability is indicated in green.

In the upper-right graph of Figure 3.2 we see how a CTDA would look for a system where decoding is possible around (or at) only a single time point. The lower-left graph shows results of a CTDA of a system that shows static/sustained coding of information. For the entire time that the information is decodable, it does not matter on which time point we train or test the decodability classifier. The state

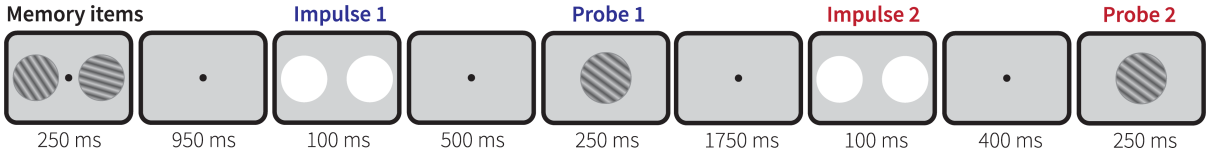


**Figure 3.1: Experiment 2. The participant knows which stimulus will be probed first. Reprinted from Pals et al. (2020)**

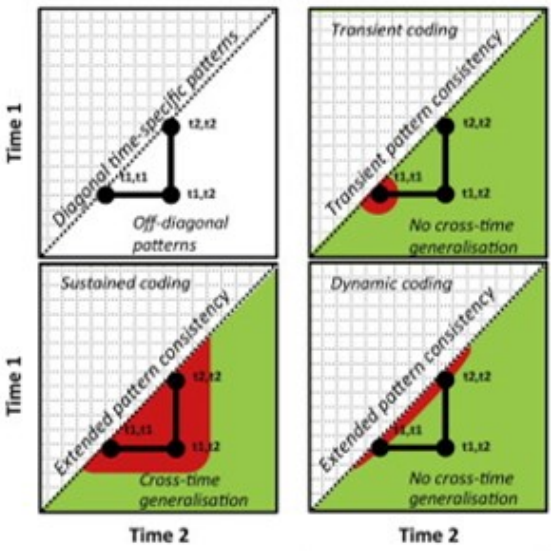


**Figure 3.2: Possible results for a cross-temporal decoding analysis. The bottom-right graph shows the results for a dynamic population code. Adapted from Stokes (2015)**

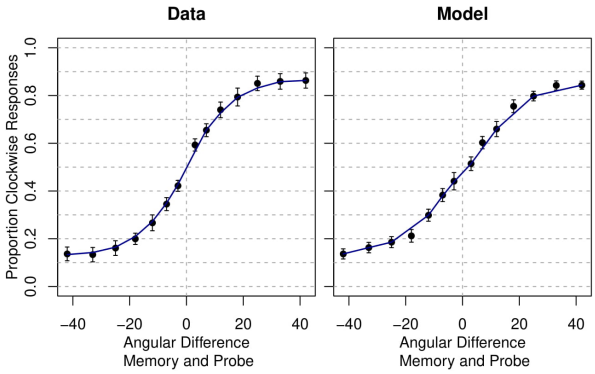


**Figure 4.1: A comparison between human performance and model performance for Experiment 1. Error bars indicate 95% CI of the mean.**

of decodable information does not change over time (cross-time generalization). When looking at the bottom-right graph we see a CTDA that would show the dynamic coding of information. The information is decodable over an extended period of time, as opposed to the upper-right graph, but it is only decodable when it is decoded using a classifier trained at the same temporal window as it is tested (no cross-time generalization). Or: the state the information is by changes over time.

For an excellent explanation of the method used for the decoding analysis and the CTDA please see the Bachelor's thesis written by Loran Knol (Knol, 2020).

The performance, WM content decoding analyses, and CTDA's were compared to human data from Wolff et al. (2017).

# 4 Results

## 4.1 Experiment 1

The behavioural results of Wolff et al. (2017) and the model are shown in Figure 4.1. The proportion of clockwise responses for all angular differences between memory and probe of the model shows a similar s-shape as human responses from Wolff et al.. The model performs slightly worse for the the small angular differences.

Figure 4.2a shows the decodability of WM content from the neural signal. We see a strong increase in decodability shortly after the onset of stimulus presentation. Then, for a short period, we see a persistent high decodability value. After this, the decodability value smoothly decreases towards its start value, reaching it before the end of the graph. The human data shows a more oscillating curve (Figure 4.2b). Instead of the stable high value period, we see two "peaks" connected by a "valley". These "peaks" are near the time point at which the stable period of the model analysis starts and ends. The decrease in value is a little bit less steep for the human participants. Also the decodability value is not back to its value at the start of the trial when the graph ends.

The CTDA of the model shows a high decodability at on-diagonal time points and a low decodability at off-diagonal time points (Figure 4.2d). This diagonal, with a similar slightly widening pattern, can also be seen in the CTDA of the human data (Figure 4.2c). The onset and duration of both diagonals are similar. Alongside the diagonal pattern is the presence of a set of "arms" in the model analysis; these extend from just after the onset of the diagonal, parallel to the axes, up to the same period where the diagonal ends.

A decoding analysis of the impulse response shows that the cued item can be decoded slightly

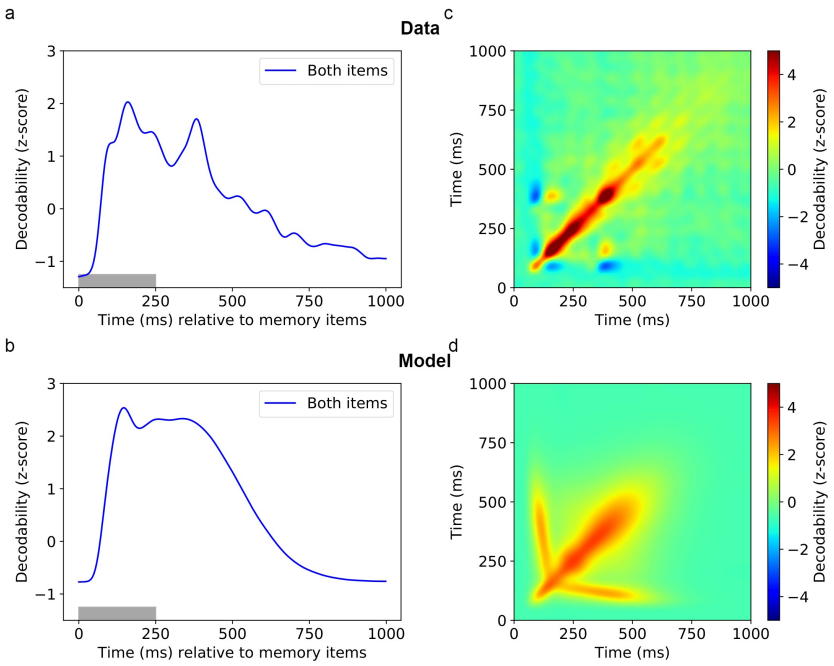


**Figure 4.2: A and b show how well a WM item could be decoded from the neural signal in the memory population for Experiment 1. C and d show the CTDA for the data and the model.**

better than the uncued item, but the difference is small (Figure 4.3). In comparison, from the human data we see a large difference in decodability. The cued item shows a strong decodability due to the impulse. Decodability of the uncued item does not seem to be affected by the impulse.

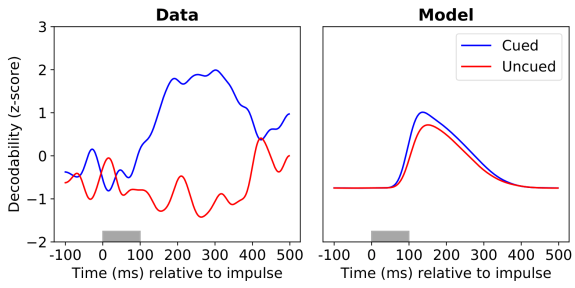


**Figure 4.3: Decodability of WM content due to the impulse presented during the delay period for Experiment 1. The analyses shows the decodability of the cued and uncued item from both the human data and the model.**

## 4.2 Experiment 2

The behavioural results of Wolff et al. (2017) and the model are shown in Figure 4.4. For the early-tested stimulus, the proportion of clockwise responses for all angular differences between memory and probe of the model shows a similar s-shape as human responses from Wolff et al.. The model performs slightly less for the the small angular differences. For the late-tested stimuli we see a overall flatter response curve, indicating less performance from the model for these late-tested items.

Figure 4.5a shows the decodability of WM content from the neural signal. We see a strong increase in decodability for both memory items shortly after the onset of stimulus presentation. Then, for a short period and also for both memory items, we see a persistent high decodability value. After this, the decodability value smoothly decreases towards its start value, both reaching it before the end of the graph. Onset of this decrease is slightly earlier for the late-tested item, but the shape of the curve is almost identical. The same figure for the human data shows a more oscillating curve (Figure 4.2b).

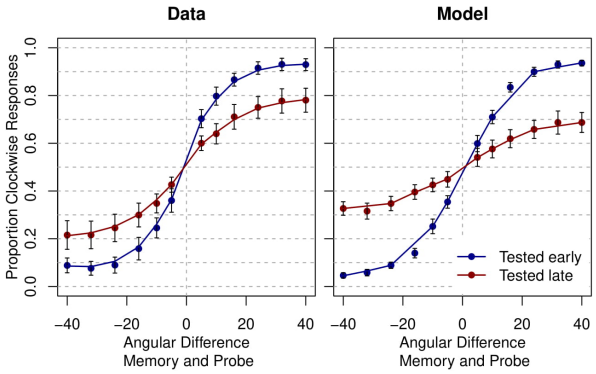


**Figure 4.4: A comparison between human performance and model performance for Experiment 2. Error bars indicate 95% CI of the mean.**

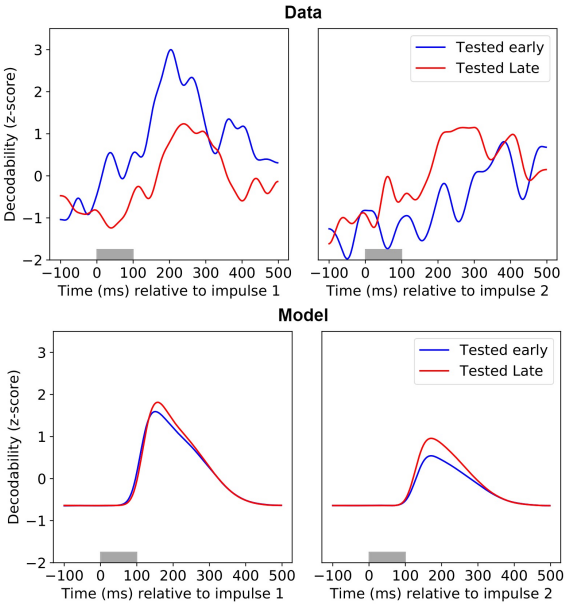


**Figure 4.6: Decodability of WM content due to both impulses presented during the delay periods of Experiment 2. The analyses shows the decodability of the late- and early-tested item from both the human data and the model.**

The decodability of the late-tested remains below that of the early-tested item throughout the graph. While the peak of the late-tested item is lower than that of the early-tested item, the rate of decrease is quite similar for the two. Both the early- and late-tested item do not show the second "peak" as we saw during Experiment 1 (Figure4.2a). The decrease in value for both items is again less steep in the data than in the model. The decodability value for the early-tested item is still well-above its start value when at the end of the graph.

The CTDA of the model for the early-tested stimulus shows a high decodability at on-diagonal time points and a low decodability at off-diagonal time points (Figure 4.5). This diagonal can also be seen in the CTDA of the early-tested stimulus from the human data. The onset of both diagonals is similar, however when the diagonal from the model CTDA ends, the human data shows a persisting, widening, decodability pattern. Alongside the diagonal pattern we see set of "arms" in both the model and human data analysis. the arms are however much more pronounced in the model.

The CTDA of the model for the late-tested stimulus shows results very similar to that of the early-tested stimulus, albeit slightly lower values. The diagonal of the human data for the late-tested item is skinnier than that of the model and this CTDA also does not show the "arms" seen with the early tested stimuli. The onset and duration of the model and human data diagonals are similar.

A decoding analysis of the model for the first impulse response shows that the late-tested item can be decoded slightly better than the early-tested item, but the difference is very small (Figure 4.3). Conversely, from the human data we see a larger difference in decodability following the first impulse. The early-tested item shows a definite higher decodability value than the late-tested item. Analysis of the second impulse shows a higher decodability value for the late-tested stimulus. The human data also shows a higher decodability for the late-tested item.

# 5 Discussion

We have extended a model of activity-silent working memory by implementation of a simplified version of the human visual system. This model is able to perform a WM task while producing dynamic coding as seen from human EEG analysis.

We first discuss the model behaviour and what, we think, caused the decreased performance for the late-tested stimulus from Experiment 2. We

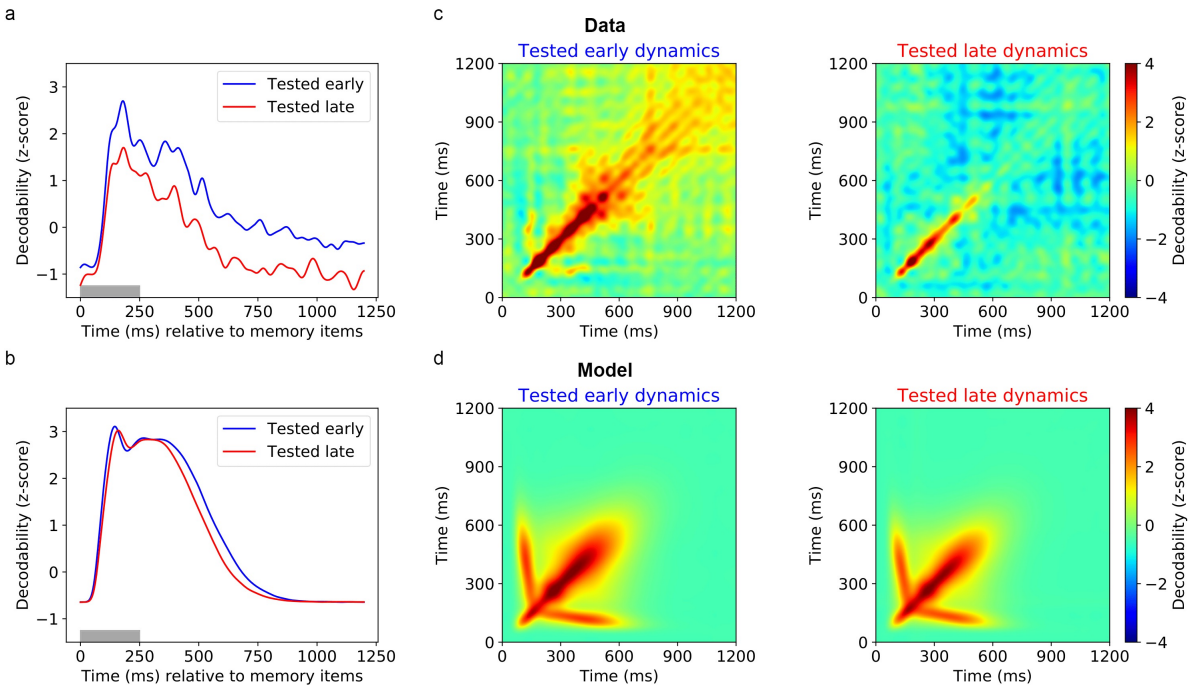


**Figure 4.5: A and b show how well a WM item could be decoded from the neural signals in the memory populations for Experiment 2. C and d show the CTDA for the data and the model.**

then discuss the decodability analysis for the period during/following stimulus presentation, which shows an overall good fit to the human data. Then we discuss the decodability analysis for the period around the generic impulse. The fit for this measure is the only one that shows surprisingly different result compared to the human data. We then discuss the results of the CTDA, which shows a very nice dynamic coding pattern. We provide suggestion for future research, as well as a comparison to a related model.

## 5.1 Behaviour

The decrease in model performance for the late-tested item from Experiment 2 (see Figure 4.4) can only be attributed to our implementation of the model's sensory system, as the original model did not show this decay in performance for the longer delay period. We do not think that the added noise is responsible for this decrease in performance. This is because we set the amount of noise below the value for which we saw any real decrease in decodability of the signal during the generic impulse, which we saw as a good indicator as to how well a stimulus was remembered.

We think that the connectivity implementation is responsible for the decrease in performance as it effectively splits up the information we get from the image. A simplified way to think about this is by imaging that we split the information we get from an image into two parts, *left* and *right*. The *left* and *right* information from the stimulus presented at a time point $t_1$ will not reach the memory population simultaneously. In addition, it can happen that two *left*s, one from $t_1$ and another from a different time point $t_2$, reach the memory population at the same time, while no *right* information is acquired.

The memory population does not seem to effectively integrate the *left* and *right* information from a stimulus at a single time point back into a whole. This is not a problem for the model when the delay period is kept short but it does influence performance for a longer delay period.

## 5.2 Neural activity following stimulus presentation

The decoding analysis of WM after stimulus presentation shows that we achieved one of the goals we had with the adaptation of the neural model. The decrease of decodability was very steep for the original model from Pals et al. (2020). The curve we have now resembles the curve from the human data quite well. That the peaks we see from the human data in Figure 4.2a match the start and end of the period of high decodability indicates that our implementation of the human visual system can produce the temporal characteristics we see from the human data. This also indicates that the values and distribution type used for the distributed response latency are not far from the true human distribution values. From the results of Experiment 2 for this analysis, we see that in the model there is little difference between the early and late-tested stimulus (Figure 4.5b).

This shows that the model is not able to capture the entire process by which two stimuli are processed, in the way required for Experiment 2. This was already seen for the model behaviour, and it is a conclusion often drawn throughout this discussion.

## 5.3 Revealing the hidden state

Eliciting a response by presenting a generic stimuli during activity-silence allows us to reveal the hidden neural state (Wolff et al. (2017)). This analysis is one for which our results show some significant difference from those of Wolff et al.. For Experiment 1 we can see that there is only a small difference between the cued and uncued item for the model while there is a large difference between these items for the human data. From this we can infer that the slow decay of the neural state that stores a memory item is not the only way by which we forget a working memory item, otherwise the model would have showed a larger difference between the cued and uncued module. It seems like humans can actively forget this memory. This does not mean that we think about forgetting something, as this make just very little sense. Rather, the brain can let go of the information once it knows that it is no longer relevant. This is obviously a useful asset for the brain, as it is a system of limited-capacity.

## 5.4 Dynamic coding and attention

One difference between the model and human CTDA is the longer lasting, strongly widening, diagonal seen in the early-tested dynamics of the human data (Figure 4.5).

We can infer that an additional mechanism is required to explain this difference. From the results of the parameter sweep (Appendix A) we know that the model is already able to produce something like this longer lasting, strongly widening, diagonal. However, this would mean different parameters setting for the two modules. We cannot have different parameters for the early- and late-module as this would result in a model where the early-tested would have to be presented to the same module, or brain hemisphere, for each trial. Human participant obviously do not have this limitation.

As participants that performed Experiment 2 knew which item would be tested first we can assume that the participants exert some kind of attentional control.In this light, we can think of the early-tested stimulus to be attended and the late-tested stimulus to be unattended. This control of attention is implemented in the model by a slightly weaker presentation of the stimulus that would be tested second.

From EEG studies we know that a visual stimulus elicits a larger response from neurons in the visual cortex when a subject directs its covert spatial attention to the location of that stimulus (McAdams and Maunsell, 1999). So far, this experimental implementation of attention seems very plausible. However, attentional control does not consist of only the scaling of sensory responses; Neuroimaging studies show that a network of frontal and parietal regions is also involved in the control of attention (Purves, Cabeza, Huettel, LaBar, Platt, Woldorff, and Brannon, 2008, p.213).

A second difference between the model and human CTDA are the “arms” (Figure 4.2 and Figure 4.5). While we also see these for the early-tested stimuli from human data, the “arms” are much more pronounced in the model. It could be that these “arms” seen in the human data are also the result of attentional control, as we do not see them for the unattended stimulus. The issue then is that we see them in the model without the explicit addition of an attentional control mechanism. The presence of these “arms” was persistent throughout a

parameter sweep and could already be seen from analysis of the original model by Pals et al. (2020) (see Appendix B).

But what knowledge can we infer from these "arms"? They indicate that a decoder trained on the signal around $150ms$ can be used to decode the signal between $150ms$ and $600ms$ (where these values are only a rough estimation for explanatory purposes). The reverse is also true, a decoder trained on the signal at any point between $150ms$ and $600ms$ can be used to decode the signal around $150ms$.

We cannot draw any proper theoretical conclusions from this but perhaps we can think of this signal around $150ms$ as containing information in such a way that it represents all possible states that the model uses to represent the stimulus.

## 5.5 Future research

For future research, the model could be expanded by the implementation of "reentrant" activity (Woldorff, Liotti, Seabolt, Busse, Lancaster, and Fox, 2002; Purves et al., 2008, p.190). "Reentrant" activity is seen as an increase in EEG signal in the lower-level cortical areas. This increase in signal is seen later than, and separate from, the increase in activity associated with the scaling of neural tuning curves as discussed earlier. This suggest a separation between attention-modulated enhancement during the early and late processing of visual stimuli. A strong argument for the use of "reentrant" activity comes from analysis of activity lateralization in the posterior regions during Experiment 2 (Wolff et al., 2017). The hemisphere that processes the attended item shows an increase in activity, during the time frame associated associated with "reentrant" activity, with respect to the other hemisphere. The difference in model and human data results are mostly seen in Experiment 2, so it might very well be possible that this implementation will improve the model over multiple measures.

Further model exploration presents itself in the form of two parameters that have not been examined for this study: the amount of neurons (in both the eye/sensory populations and the memory population) and the strength of stimulus presentation. While we have no results about the influence of the former, we do have some about the latter. The decoding values for the CTDA of the model show an overall slight decrease for the late-tested stimuli (4.5), where the stimulus was presented at 90% strength. Perhaps a further decrease in presentation strength could reduce the prominence of these "arms".

## 5.6 CTDA interpretation

One aspect of the CTDA that warrants caution is that the range at which we display decodability values for the CTDA has a large influence on the final figure. It can cause stronger or weaker coloring of parts of the diagonal, as well as stronger or weaker coloring of the "arms".

## 5.7 Related research

We have not been able to compare the dynamic coding measures of our model to that of other WM models, as we could not find any papers that analyzed such models using a CTDA or a similar analysis.

This leaves the question whether the mechanisms that cause dynamic coding in our model are the only mechanisms that cause these measures in the human brain. Obviously, this is very unlikely. Cognitive models are always a simplification and choices have to be made as to which promising findings from research will be implemented in a model.

One study that discusses neural dynamics within a modelling framework is Singh and Eliasmith (2006). This study implemented a type of neuron with a two-dimensional tuning curve. The response characteristic of this neuron is not only stimulus- but also time-dependent. This model however is not a model of activity-silent working memory.

With our model we have shown that there is no need for this type of complex tuning curve to produce dynamic coding . From our analysis it seems than any network with activity-dependant adaptivity of connection weights can show dynamic coding measures.

The short-term synaptic plasticity as implemented by Pals et al. (2020) is one possible mechanism responsible for the dynamic coding measures created in this fashion (see Appendix B). We have shown that the addition of a distributed response latency across their neural model produces an even more dynamic neural state.

### 5.8 Model parameters

The parameters for the distributed response latency were taken from a study using macaque monkeys, not human participants. Also the shape of the distribution was not based on literature but was chosen arbitrary as a gamma distribution with a shape parameter $k$, with $k = 2$.

The delay for the recurrent connection in the sensory system was also chosen arbitrarily. This value was set to just above twice the mean value of the latency distribution.

## 6 Conclusion

To conclude, our model shows that dynamic coding in working memory can be the explained by the combination of: the activity-dependent adaptation of neural connections by short-term synaptic plasticity; our simplified implementation of the human visual system, that results in a distributed latency of brain activity; and the addition of spontaneous spiking to the neural population responsible for storage of a working memory item.

In addition, we have provided arguments for the implementation of attentional control to further bridge the gap between the results from analysis of our model and that of human data.

As we have found no published literature that has provided similar results and insights, we proudly present to you the first activity-silent working memory model that shows dynamic coding.

## References


Alan Baddeley. Working memory: looking back and looking forward. *Nature reviews neuroscience*, 4 (10):829–839, 2003.

Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith. Nengo: a Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, 7(48):1–13, 2014. ISSN 1662-5196. doi: 10.3389/fninf.2013.00048.

Clayton E Curtis and Mark D'Esposito. Persistent activity in the prefrontal cortex during working memory. *Trends in cognitive sciences*, 7(9):415–423, 2003.

Patricia S Goldman-Rakic. Cellular basis of working memory. *Neuron*, 14(3):477–485, 1995.

Loran Knol. Dynamic coding in a large-scale, functional, spiking-neuron model. 2020.

Victor AF Lamme and Pieter R Roelfsema. The distinct modes of vision offered by feedforward and recurrent processing. *Trends in neurosciences*, 23(11):571–579, 2000.

Mikael Lundqvist, Pawel Herman, and Earl K Miller. Working memory: delay activity, yes! persistent activity? maybe not. *Journal of Neuroscience*, 38(32):7013–7019, 2018.

Carrie J McAdams and John HR Maunsell. Effects of attention on orientation-tuning functions of single neurons in macaque cortical area v4. *Journal of Neuroscience*, 19(1):431–441, 1999.

Ethan M Meyers, David J Freedman, Gabriel Kreiman, Earl K Miller, and Tomaso Poggio. Dynamic population coding of category information in inferior temporal and prefrontal cortex. *Journal of neurophysiology*, 100(3):1407–1419, 2008.

Matthijs Pals, Terrence C Stewart, Elkan G Akyürek, and Jelmer P Borst. A functional spiking-neuron model of activity-silent working memory in humans based on calcium-mediated short-term synaptic plasticity. *PLOS Computational Biology*, 16(6):e1007936, 2020.

Dale Purves, Roberto Cabeza, Scott A Huettel, Kevin S LaBar, Michael L Platt, Marty G Woldorff, and Elizabeth M Brannon. *Cognitive neuroscience*. Sunderland: Sinauer Associates, Inc, 2008.

Ray Singh and Chris Eliasmith. Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, 26(14):3667–3678, 2006.

Kartik K Sreenivasan, Clayton E Curtis, and Mark D'Esposito. Revisiting the role of persistent neural activity during working memory. *Trends in cognitive sciences*, 18(2):82–89, 2014.

Mark G Stokes. 'activity-silent'working memory in prefrontal cortex: a dynamic coding framework. *Trends in cognitive sciences*, 19(7):394–405, 2015.

Yun Wang, Henry Markram, Philip H Goodman, Thomas K Berger, Junying Ma, and Patricia S Goldman-Rakic. Heterogeneity in the pyramidal network of the medial prefrontal cortex. *Nature neuroscience*, 9(4):534–542, 2006.

MG Woldorff, M Liotti, M Seabolt, Laura Busse, JL Lancaster, and PT Fox. The temporal dynamics of the effects in occipital cortex of visual-spatial selective attention. *Cognitive Brain Research*, 15(1):1–15, 2002.

Michael J Wolff, Janina Jochim, Elkan G Akyürek, and Mark G Stokes. Dynamic hidden states underlying working-memory-guided behavior. *Nature Neuroscience*, 20(6):864, 2017.

Robert S Zucker and Wade G Regehr. Short-term synaptic plasticity. *Annual review of physiology*, 64(1):355–405, 2002.

# A Parameter influence on CTDA results

Figure A.1 shows the CTDA results for a variety of parameter combination, chosen from the results of an earlier sweep. Along the x-axis are various combination of the strengths of the distributed feedforward connections in the sensory part of the model and the recurrent connection in the sensory part of the model. This strength is the value by which we multiply the weights matrix of a connection. Along the y-axis the amount of noise added to the memory population is varied for all models, the noise increases as we progress downwards. This noise value is the standard deviation of a gaussian distribution with a mean value of zero. We can see that we can create a large variety of results. As interesting as this is, not all of the models result in a functional model. Especially, a high amount of noise causes a large decrease in overall decodability, and thus performance. The values of these parameters for the final model are shown in Table A.1.

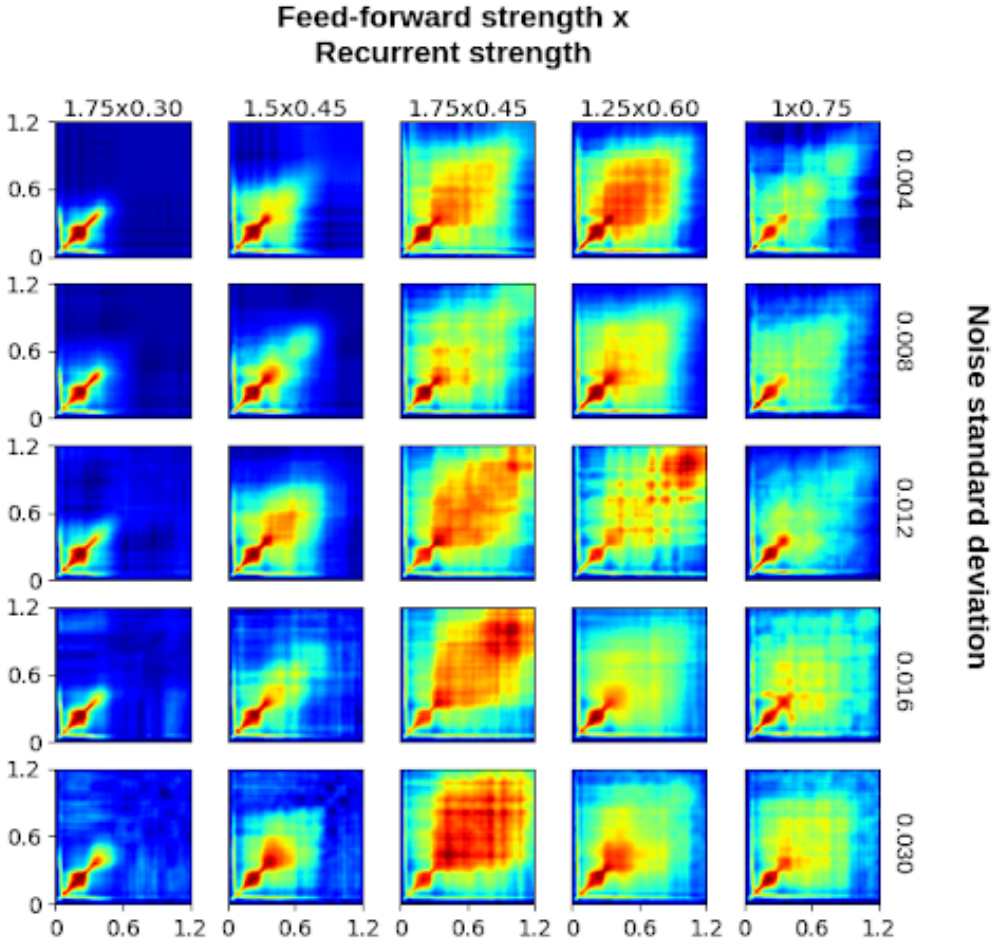


**Figure A.1: The CTDA results for a variety of models.**

| Parameter | value |
|---|---|
| Connection strength of the distributed feed-forward connections | 1.50 |
| Connection strength of the recurrent connection | 0.30 |
| Standard deviation of the noise distribution | 0.010 |

**Table A.1: Parameter setting for the sensory system of the model and the noise added to the memory population.**

# B Dynamic coding in the model from Pals et al. (2020) and the influence of the synaptic spike filter

The memory population of the model is read out with a synpatic spike filter. We can best think of this value as the effect that the skull has on readout of the EEG signal. Figure B.1 shows the effect that this parameter has on the result of the CTDA. The model used to produce this graph is the model form Pals et al. (2020). We see that this model already shows dynamic coding. The value used for the analyses in this study is 0.05; we judged this value to show the diagonal indicative of dynamic coding without too much off-diagonal decodability.

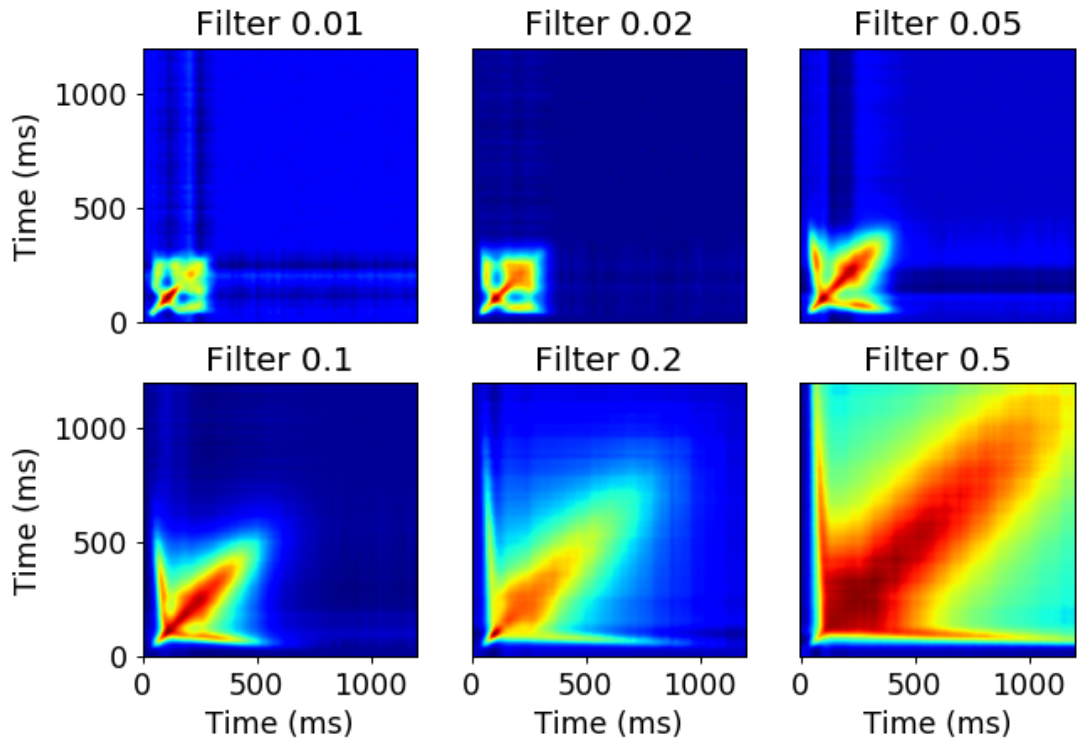


**Figure B.1: CTDA of the model from Pals et al. (2020) for various values for the synaptic spike filter.**