Department of Computer Science, University of Otago

Technical Report OUCS-2013-10

Response Fingerprinting: a probabilistic method for evaluating the network response to stimuli

Authors:

Mira Guise, Alistair Knott, Lubica Benuskova
Department of Computer Science, University of Otago, New Zealand

---

Department of Computer Science,
University of Otago, PO Box 56, Dunedin, Otago, New Zealand

http://www.cs.otago.ac.nz/research/techreports.php

Response Fingerprinting: a probabilistic method for evaluating the network response to stimuli

Mira Guise*, Alistair Knott, Lubica Benuskova

Dept of Computer Science, University of Otago, Dunedin

Abstract

Spiking neural networks that have variable connection delays have the interesting property that they are sensitive to both spatial and temporal patterns of input. Each neuron in the network receives input from spatially-distributed input neurons whose precise firing times interact with connection delays to determine whether the neuron can exceed the firing threshold and produce an output. The network is therefore most responsive to spatio-temporal stimuli whose temporal components match the delays in the connected structure of the network. Izhikevich (2006a) has shown that certain strongly connected groups of neurons known as polychronous neural groups (or PNGs) exist in large numbers within the network structure. The activation of these neural groups is stimulus-specific and produces unique firing signatures that are detectable in the firing data. Previous methods for detecting PNG activation have relied on a template matching technique that assumes a deterministic response to each stimulus presentation (Izhikevich, 2006a; Martinez and Paugam-Moisy, 2009; Guise et al., 2013a). Here we present an alternative probabilistic view of the stimulus response and demonstrate the application of this new detection method.

Keywords: spiking network, polychronous neural group, representation

Spiking neural networks, such as those in the brain, are connected networks of nodes that are able to produce computationally interesting transformations on their inputs. The network nodes (called *neurons*) exchange messages across their connections in the form of temporally discrete events called *spikes*. The significance of these spiking messages is determined by the *weight* of the connection bearing each message: messages borne on strong connections carry greater significance than messages borne on connections with lesser weight. The neurons in a spiking neural network receive spiking messages on their input connections, evaluate their significance and optionally produce a spike on their output connections. Message evaluation involves the integration of messages over a limited time frame, with the significance of individual messages decreasing exponentially the moment they are received. If the combined significance of recent messages reaches a threshold, the neuron *fires*, producing an output spike that provides an input message to other neurons.

*Corresponding author at: Department of Computer Science, University of Otago, PO Box 56, Dunedin 9054, New Zealand

Email address: mguise@cs.otago.ac.nz (Mira Guise)

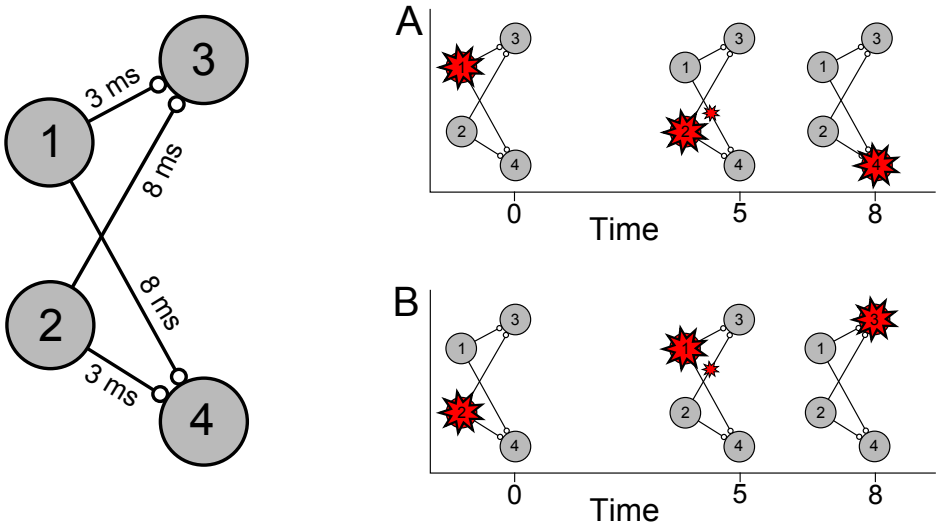


Figure 1: The significance of variable connection lengths. Each of the four neurons in this model network require at least two near simultaneous inputs in order to reach firing threshold. Neurons 1 and 2 are input neurons that provide synaptic input to output neurons 3 and 4. Due to the variable delays imposed by the different connection lengths, the precise firing times of each input neuron determine which of the two output neurons is likely to fire.

The pattern of messages flowing through the network has both spatial and temporal components: each neuron receives input messages from many neighboring neurons distributed across the network space, and these messages arrive at the neuron at discrete times. Each neuron therefore sees distinct spatio-temporal stimuli across its many input connections. The network parameters are typically chosen such that it requires at least two near simultaneous inputs for a neuron to fire. Incoming spikes from neighboring neurons must therefore converge to produce sufficient combined input for the neuron to exceed its firing threshold.

If the connections to neighboring neurons are of variable length, the precise firing time of each of the neighboring neurons interacts with the transit times of each of the resulting spikes to determine whether the neuron will fire. As shown in panel A of Fig. 1, if neuron 1 fires at $t = 0$ and neuron 2 fires at $t = 5$ then the combined input from two near simultaneous spikes is sufficient for neuron 4 to reach the firing threshold. While neuron 3 receives the same spike messages, they are well-spaced in time and the threshold is therefore not reached. In panel B, the input firing pattern is reversed i.e. neuron 2 fires at $t = 0$ and neuron 1 fires at $t = 5$. The result is that convergent input arrives at neuron 3, and only neuron 3 will fire. To summarize, a neuron

will only reach the firing threshold when the input connection lengths are congruent with the pattern of firing times of the input neurons. Each neuron therefore has the ability to selectively respond to spatio-temporal stimuli in which the precise timing of its input neurons matches the corresponding connection delays.

The connected structure of a spiking neural network can be viewed as a weighted digraph in which the graph vertices are replaced by neurons and the directed edges are replaced by connections. Izhikevich (2006a) has shown that certain strongly connected groups of neurons known as polychronous neural groups (or PNGs) exist in large numbers within this connection graph. A defining feature of polychronous groups is that the connected subgraph that distinguishes the group forms a broad pathway of congruent connections through the network that has the potential to sustain a causal cascade of neural firing. *Activation* of a PNG produces *polychronisation*, a reproducible and precisely timed sequence of firing events that is observable in the firing data generated by the network (Izhikevich, 2006a; Izhikevich et al., 2004).

Group activation requires an appropriate spatio-temporal *triggering* pattern that is able to match some portion of the PNG subgraph and produce convergent firing. If the resulting polychronisation is to be sustained, multiple group neurons must fire at precise times over the course of activation. The interaction of the convergent connections within the group with these precisely timed firing events allows individual group neurons to reach their firing thresholds, supporting further polychronisation. Although these pathways of converging connections have the potential to support polychronisation, PNG activation also requires that the connections between group members be sufficiently strong to allow the combined inputs to group neurons to pass the firing threshold. Training the network produces selective strengthening of group connection weights so that polychronisation is more likely to occur. The process of network training involves repeated exposure to a triggering stimulus in the presence of an appropriate learning rule such as Spike-timing Dependent Plasticity (STDP).

The unique pattern of firing events generated by each PNG activation provides a distinct activation signature that can be detected in the network firing data. In order to resolve these signatures within the massive flood of firing events generated by the network, PNGs that are triggered by parts of the stimulus are used as spatio-temporal templates that are matched against the firing data (Guise et al., 2013a). Alternatively, the triggering patterns that make up the stimulus are detected directly, on the assumption that the presence of the triggering pattern will entail PNG activation (Martinez and Paugam-Moisy, 2009).

One difficulty with these techniques is that in a network with recurrent

connections and random background firing, PNG activation is not deterministic: the same PNG can polychronise in different ways, with variation in both the neurons that participate in each activation and in the precise time of their firing. This variability is caused by perturbations in the internal dynamics of each neuron due to the integration of recent events. For example, if random or recurrent input to a PNG neuron has recently caused it to fire, the neuron may resist participating in the current activation for a small interval (the neural refractory period), or may fire with a small delay. In addition, Izhikevich et al. (2004) have noted that there is considerable competition between polychronous groups for the affiliation of individual neurons, causing the synaptic weights to be constantly adjusted to support the activation of first one group then the other. Methods for detecting group activation must take these variations into account, typically by reducing the number of firing events in a search template that must be matched, and by incorporating a term that accounts for an allowed jitter in the neural firing time.

Using techniques such as these, Izhikevich (2006a) has explored the idea that polychronous groups might provide a neural basis for representation and memory. According to this proposition, stimuli evoke specific mental representations by triggering PNG activation. However, for this to be true PNG activation must be both consistent and selective i.e. PNG activations must occur consistently on each stimulus presentation and must be specific to the stimulus. Our group has used a template matching technique to provide initial evidence for the consistency of PNG activation (Guise et al., 2013a). There is also some initial evidence for selectivity: using a network trained on two different stimuli, Izhikevich (2006a) has shown that different groups of PNGs were activated in response to each stimulus. However, similar work within our group has found that PNGs can be activated by different but overlapping stimuli and that individual PNG activations cannot therefore be selective for the input stimulus. Nevertheless, both our results and those of Izhikevich (2006a) are consistent with the idea that *sets* of PNG activations could be selective. Given this view, the response of the network to overlapping stimuli can be modeled by allowing two sets of PNG activations to overlap such that individual PNG activations occur in both sets. Thus, while the set as a whole is selective for the stimulus, the activation of any individual PNG within the set is not.

The PNG activations that comprise the network's response to a stimulus presentation together provide a stimulus-specific activation signature that will be referred to hereafter as the *stimulus response*. On first exposure to a stimulus, there may be few if any PNG activations making up the stimulus response, and any activations that occur may be partial. This initially weak response becomes increasingly well defined over the course of training

as the connection weights supporting PNG activation are strengthened. In previous research, the stimulus response has been studied on the assumption that PNG activation generates a fixed and deterministic sequence of firing events. However, both the variability between individual PNG activations and the changes in the stimulus response that occur with training are incompatible with this view. In the current paper we abandon this view in favor of a more probabilistic representation of the stimulus response. In order to explain this approach we will begin by creating a *Response Profile*, a profile of the firing behavior of each neuron that uniquely qualifies the network's response to the stimulus. Next, we will build upon this response profile to generate a stimulus-specific *Response Fingerprint* that supports a probabilistic interpretation of the stimulus response.

1. Response Profiling

A *firing event* records the firing of a specific neuron in the network at a specific time and can occur in direct response to an external stimulus or in response to input from recurrently connected neighboring neurons. An external stimulus is represented as a spatio-temporal pattern of firing events that causes the specified neurons to fire at the specified times. This input pattern may be repeatedly presented to the network at regular intervals, typically at a frequency of one hertz or greater. In addition, a random pattern of firing is generated within the network by causing each neuron in the network to fire at a random time in each second (the mean random background firing rate is typically set to one hertz).

Regular stimulation of a trained spiking network with a known pattern produces observable consistencies in the firing events occurring after each presentation of the stimulus. In response to an external stimulus the input pattern neurons fire, providing an activation trigger for one or more PNGs. If coherent external stimulation is provided to a network at regular intervals, the consistent firing of PNG neurons produces firing events whose firing times occur at a consistent interval following each stimulus presentation. In contrast, neurons that are not a part of any activated polychronous group will fire at random times over the same interval.

In this section we will characterize this consistent activation response by accumulating the firing times of each neuron over multiple stimulus presentations. The resulting histograms can be generated for every neuron in the network, creating a response profile that is unique to each stimulus.

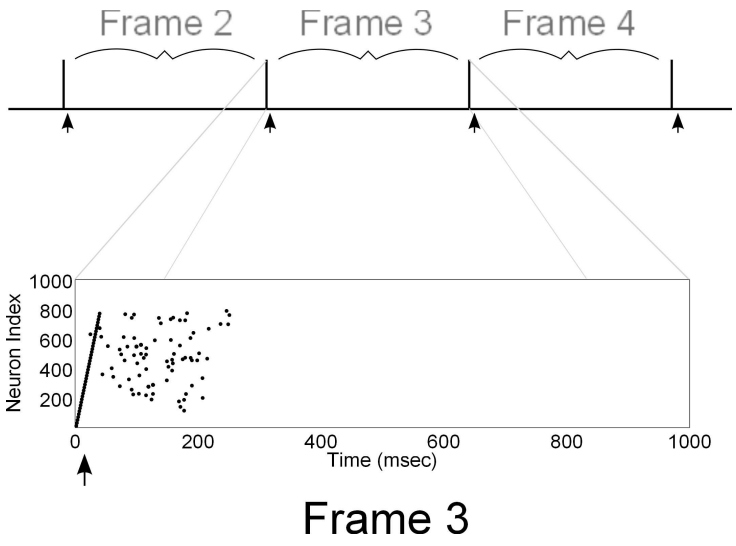


Figure 2: A schematic of a single response frame showing the presentation of a stimulus at the time indicated by the arrow, and the subsequent stimulus response caused by one or more PNG activations. Firing events are filtered so as to show just the stimulus events and those events that are part of the resulting stimulus response. This single response frame is assumed to be one of many in a long consecutive sequence of frames.

1.1. Qualifying the response to an input pattern

We start by defining a fixed interval called a *response frame* over which firing times are accumulated (see Fig. 2). The stimulus is presented at the start of each response frame and the stimulus response is then collected over the remainder of the interval.

A count of the number of firing events is accumulated for each fixed temporal offset relative to the start of the frame, producing a *response histogram* for each neuron. The collection of response histograms for each neuron in the network together define a unique profile of the response of the network to a specific input pattern (a *frame profile*).

Figure 3 shows the response histograms for eight selected neurons (neuron indices are indicated to the right of each plot). The temporal offset within the response frame is shown on the x-axis, with spike counts on the y-axis.

A low level of random background firing can be seen in each histogram, with firing times distributed throughout the frame. However, each histogram also shows one or more significant peaks in the spike counts, occurring within a small range of offsets relative to stimulus presentation. For the purposes of explaining these peaks, it is useful to view the stimulus response as a causal sequence of firing events as shown in Figure 4. The firing of neurons in the

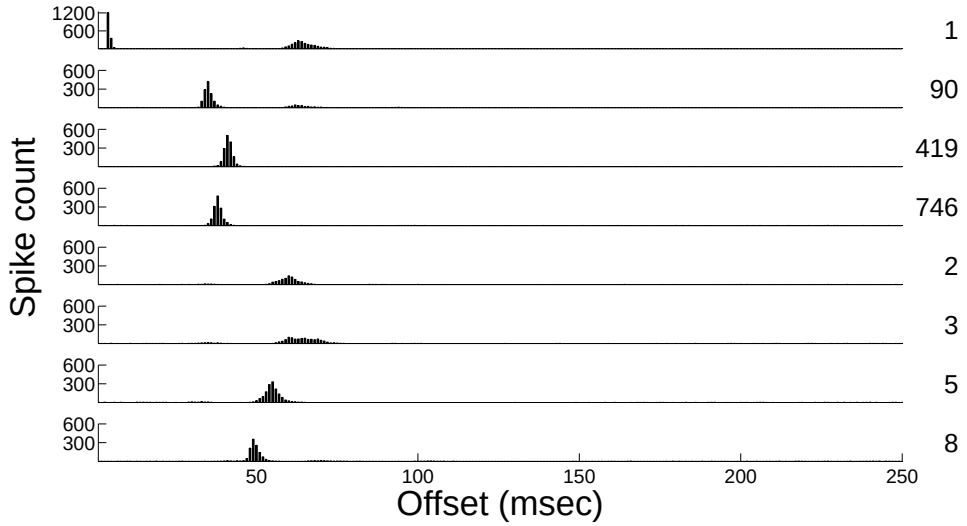


Figure 3: Histograms of spike counts for selected neurons over a 250 millisecond response frame. Spiking data was collected from a trained network following repeated stimulation with an ascending input pattern. Each neuron in the network averaged nearly 3000 firing events over the 400 second run. Some of these events were due to background firing, and some due to the direct or indirect effects of the firing of input pattern neurons. The histograms of just eight neurons (of 1000 total) are shown. The selected neurons are either directly or indirectly connected to neuron 1 (see index numbers on the right). They are as follows: neuron 1 is a neuron from the input pattern; neurons 90, 419 and 746 are neurons immediately post-synaptic to neuron 1; neurons 2, 3, 5 and 8 are not directly connected to any input pattern neurons. Both the directly connected and the indirectly connected neurons respond to only some of the 1600 stimuli. Note that neuron 1 is externally stimulated (as part of the input pattern) and responds to most presentations of the stimulus (for this reason, the y-axis for neuron 1 is approximately twice that of the remaining plots).

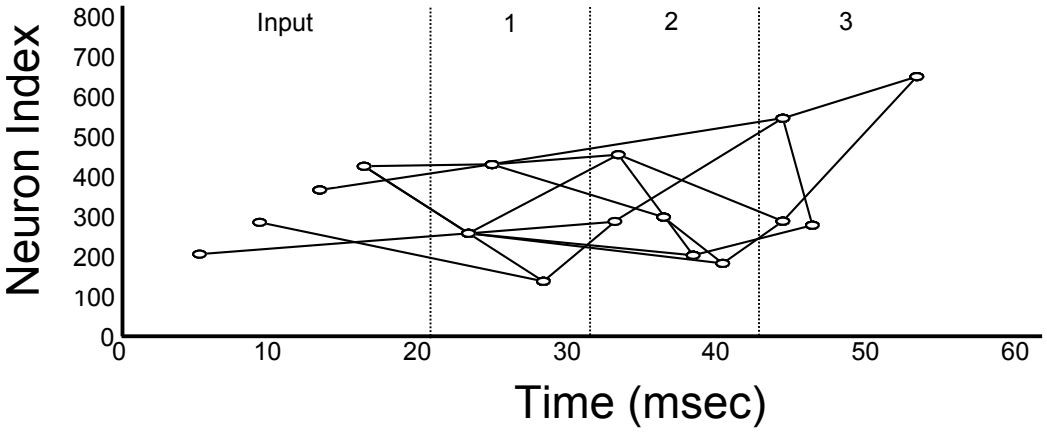


Figure 4: Activity propagation through multiple layers during activation of a PNG following stimulation of a trained network with an ascending input pattern. Nodes represent the firing of PNG neurons and lines represent causal links between firing events. For the purposes of explication most of the firing events and causal links have been removed and the timing of some firing events has been adjusted.

input layer produces subsequent firing of neurons in layer 1 that together produce firing in layer 2, and so on into deeper layers of the network. Given repeated stimuli then, the firing of input pattern neurons should be highly correlated with the subsequent firing of neurons in layer 1. We might also expect to see correlations between input pattern events and firing events in deeper layers of the network, although these correlations might get progressively weaker due to the non-deterministic nature of these firing events.[1]

Figure 3 shows the histograms for some of the more strongly responding neurons in the network. Other neurons produce a weaker response, or perhaps no response to the same input pattern. External input produces the first peak for neuron 1, an input pattern neuron (top histogram in Figure 3). The firing of neuron 1 and other input pattern neurons leads to subsequent firing of neurons immediately post-synaptic to the input neurons, including neurons 90, 419 and 746. These neurons show peaks at offsets that reflect the axonal delays between the input neurons and their post-synaptic neurons.[2]

[1]While it is useful to entertain a perspective on PNG activation that emphasizes a layered architecture, no such abstraction exists in a network with random connections. In such a network, the causal effects of firing events might be limited to a single layer in Figure 4, or might span across multiple layers in the diagram, limited only by the axonal lengths. Note the intra-layer and multi-layer links in Figure 4.

[2]The offsets also reflect the firing latencies for both sets of neurons.

Firing events in layer 1 then produce firing of the neurons in progressively deeper layers (neurons 2, 3, 5 and 8).

Secondary peaks are also observable in the figure: a strong secondary peak following external stimulation of neuron 1 suggests a long recurrent pathway producing self-stimulation of this neuron; a series of secondary peaks following the firing of neuron 90 might be due to a shorter recurrent pathway producing a sequence of progressively weaker inputs on each cycle.

These spike count peaks in the histograms of the strongly responding neurons define a spatio-temporal response signature that is unique to a given input pattern. If we assume that the effect of training is to reinforce the connections within one or more structural PNGs then this response signature describes the activation of these structural groups. More precisely, the response signature describes a statistical average over many activations of the connectivity subgraphs within the underlying groups. Each presentation of the pattern produces a stimulus response where the vast majority of the firing events are statistically likely to fall within temporal windows defined by the histogram peaks. It should therefore be possible to use this unique response signature to infer the presence of the input pattern, even after a single presentation of the pattern. We will explore this idea further in the next section.

2. The Response Fingerprinting method

In this section, a new method is presented that allows the presence of a particular input pattern to be inferred from the stimulus response using a pattern-specific fingerprint. A *pattern fingerprint* is a spatio-temporal pattern of *temporal windows* that together define the statistical response to the pattern. Each temporal window specifies a small range of temporal offsets within which the probability of a spike occurring is significantly greater than average. The temporal windows in the fingerprint are spatio-temporally arranged so as to capture the majority of the firing events in the stimulus response. If the majority of windows capture a spike, then it is probable that the input pattern associated with the fingerprint was present. On the other hand, if few windows capture a spike, then it is unlikely that the input pattern was present.

The process of generating a response fingerprint requires first generating a response profile and then analyzing the resulting histograms to determine the temporal windows that straddle the peak spike counts. Window start times and widths are mapped to the temporal offsets that optimally overlap

each peak.[3] Once a window is defined for each neuron, a thresholding step removes those that capture only a small proportion of the available spikes, leaving just the windows for strongly responding neurons. Although creating a response profile requires the gathering of statistics over many presentations of the input pattern, this process need only be performed once for each pattern. The fingerprint can then be used for pattern inference over multiple experiments, provided that the stimulus response does not diverge too far from the gathered statistical data.

The presence or absence of the input pattern associated with a fingerprint is inferred from the evidence provided by the stimulus response. Importantly, the response fingerprinting method expands on the response profiling method by imposing a probabilistic view on the response. Placing this evidence within a probabilistic framework usefully allows us to quantify the non-deterministic relationship between the input pattern and responding neurons. Ideally, we would like to make statements such as "given some spiking of selected neurons in these selected time ranges, the probability that the input pattern was presented is X". In order to support such statements we need to be able to compute the conditional probability that the input pattern was present, given a specific stimulus response.

2.1. A probabilistic view of the stimulus response

If we define the event of one or more spikes occurring within a temporal window as a *window activation*, then our objective is to compute the probability that a specified input pattern was presented, given the evidence of window activation (i.e. $p(InputPattern \,|\, WindowActivation)$). For convenience we will define $ip = InputPattern$, $act = WindowActivation$. Applying Bayes Theorem we get,

$$P(ip \,|\, act) = \frac{P(act \,|\, ip)\, P(ip)}{P(act)} \tag{1}$$

Given that each frame may include multiple items of evidence (i.e. multiple window activations), we need to compute: $P(ip_i \,|\, act_1 \wedge act_2 \wedge act_3 \ldots)$. For this we can use an extended form of Bayes Theorem that supports multiple events. Or more simply we can assume conditional independence of act_1, act_2, act_3 such that:

$$P(act_1 \wedge act_2 \wedge act_3 \ldots act_n |\, ip) = \prod_i P(act_i |\, ip) \tag{2}$$

[3]The window width is typically kept to a fixed value of 8 milliseconds.

allowing the more general equation:

$$P(ip_i \mid act_1 \wedge act_2 \wedge act_3 \ldots act_n) = \alpha \frac{P(ip_i) \prod_j P(act_j \mid ip_i)}{P(act_1 \wedge act_2 \wedge act_3 \ldots act_n)} \quad (3)$$

A naive Bayes classifier probabilistically selects the best hypothesis from a range of competing hypotheses based on evidence provided by a set of class features. For our purposes, we wish to decide on the presence or absence of a particular stimulus, given the set of coincident window activations in the frame. In equation 3 the denominator ($P(act_1 \wedge act_2 \wedge act_3 \ldots act_n)$) is constant and independent of the hypothesis on the input pattern allowing us to simplify[4]:

$$P(ip_i \mid act_1 \wedge act_2 \wedge act_3 \ldots act_n) = argmax(P(ip_i) \prod_j P(act_j \mid ip_i)) \quad (4)$$

We are now able to infer the presence of a particular stimulus by applying equation 4 to the evidence of coincident spiking in each frame. More generally, we can infer which of a number of potential input patterns was presented by applying equation 4 to multiple fingerprints and selecting the result with the largest probability. A strong correlation between the spiking of input neurons and the later firing of other neurons also provides evidence of a reproducible non-synchronous but nevertheless polychronous spatio-temporal firing pattern that is the hallmark of PNG activation. We can therefore use the response fingerprinting method for the detection of PNG activation within the firing data, despite the high variability in the stimulus response.

3. Examples of using the Response Fingerprinting method

Experiments that utilize the Response Fingerprinting method require that a test stimulus is presented in the context of a response frame, and that the firing data following each stimulus is collected for analysis. For some experiments the test stimuli will vary between frames, while in other experiments the same test stimulus will be presented in each response frame. In either case, a single test stimulus is presented in each frame so that the stimulus response can be analyzed in isolation. Where multiple test stimuli are used in the experiment, the stimulus selection for each frame is either randomly assigned or is selected from a pre-assigned stimulus-frame sequence. With the exception of experiments that study the evolution of PNG activation over

[4]The function, *argmax* selects the maximum value of its arguments i.e. the best hypothesis.

the course of training, the network will have been trained on the test stimuli prior to the start of each experiment. The test stimuli are therefore assumed to be either known training stimuli, or are generated from a blend of these known stimuli.

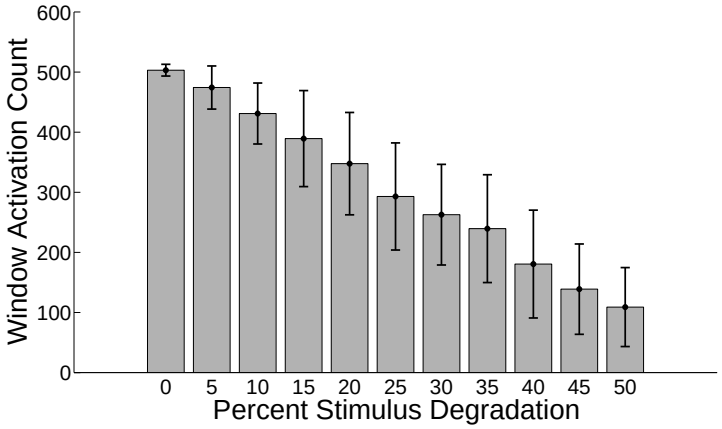
A *frame profile* of each experimental stimulus is provided to the Response Fingerprinting method, and the method then produces a probability distribution over the probabilities for each frame profile pattern. As a simple example consider the test for the presence or absence of a particular stimulus: in this case both the frame profile for the chosen stimulus *and* the frame profile for the null-pattern will be provided to the method, and a probability distribution over the presence or absence of the stimulus will be generated for each response frame in the experiment. The following sections provide some further simple examples of the use of the Response Fingerprinting method.

3.1. The effect of stimulus degradation on the PNG activation response

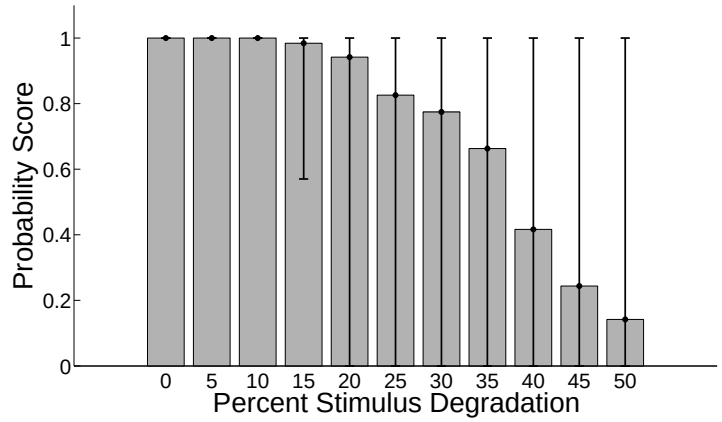
We have seen that the firing of neurons in a spiking network is not deterministic, due to the combined effects of random background firing and recurrent connections on firing times. Even the input pattern neurons that are directly stimulated by the external stimulus are affected by these processes, producing firing jitter and even occasional mis-firings. However, a small amount of degradation of the input stimulus appears to have little effect on the resulting PNG activations and is therefore likely to have minimal impact on probability scores.

In the following sample experiment we hypothesize that the Response Fingerprinting method will be resistant to small amounts of stimulus degradation but will produce diminished probability scores with substantial degradation. The script that performs this experiment can be found in Listing B.4 of the appendix (see *Method Scripts*). In brief the script loads a network that has been trained on a specific stimulus and tests the effects of progressively degrading the stimulus on the resulting probability scores. Each score represents the probability that the stimulus was present in each response frame. Probability scores, and the number of window activations, are averaged over multiple frames for each degraded stimulus sample, and multiple samples are tested for each level of stimulus degradation. The results are shown in Figure 5.

Increasing degradation of the stimulus produces a steady decline in window activation counts, although the variation in counts across multiple samples remains fairly constant throughout the decline. In contrast, the probability scores are unaffected by small amounts of stimulus degradation (0 - 10%), but then show a rapid decline beyond this range. The probability



(a) Window Activation Counts



(b) Probability Scores

Figure 5: The effect of progressively degrading the stimulus on the number of window activations (top) and the probability scores (bottom). The degree of degradation ranges from zero to fifty percent, increasing in five percent increments from left to right. The gray bars show mean values over multiple tests, while error bars show either the standard deviation (top) or the range between minimum and maximum values (bottom). Stimulus degradation is produced by randomly deleting a specified proportion of the firing events that constitute the spatio-temporal pattern. At each level of stimulus degradation, one hundred degraded samples were tested and the means, ranges and standard deviations were calculated over these samples. Each of these degraded test samples was independently generated, and the test scores and window activation counts for each sample were averaged over ten response frames.

scores vary wildly on degraded stimulus samples, unlike the window activation counts which show a constant variation with increasing degradation.

3.2. Visualization of PNG activation

The presentation of a stimulus to a network that has been previously trained on the stimulus triggers the activation of PNGs that are associated with the stimulus. In previous sections we have seen that firing events that occur within specific temporal windows in the activation response are sufficiently consistent to allow these window activations to be assessed probabilistically. Although the firing events that constitute the activated PNG can vary across response frames, the spatio-temporal pattern of the response in each frame is sufficiently consistent to produce a high probability of firing events occurring within these stimulus-specific temporal windows within the frame.

The Response Fingerprinting method provides a probabilistic evaluation of the stimulus response, but can also be used to *visualize* the response: the visualization technique involves filtering the firing events generated in each frame through the temporal windows defined by the fingerprint, leaving only those firing events that correspond to the response. The spatio-temporal causality between the remaining firing events can then be inferred using the underlying network connectivity: two firing events whose neurons are directly and strongly connected, and whose temporal spacing conforms with the axonal length of the underlying connection, are likely to be causally connected i.e. the earlier event is likely to be causal in the generation of the later event.

A script that uses this procedure to extract PNG activations from the firing data can be found in Listing B.5 in the *Method Scripts* section. The script generates ten frames of firing event data from a trained network and then extracts a PNG structural description from each frame. The PNG activation response in four selected frames is shown in Figure 6. The lines between nodes in the graph represent causal links between the firing events that constitute the group activation. For the purposes of this figure only connections with saturated weights are shown, although removing this restriction has very little effect on the resulting graph implying that most connections in the underlying structural group are saturated.

In order to prevent the evolution of the underlying structural PNG, synaptic plasticity was disabled during the collection of firing data for this experiment. The observable small variations between frames are therefore due to changes in the network dynamics. There are two types of variation: firstly, a firing event can occur in one frame but be absent in another; secondly, a firing event can occur in multiple frames but with variation in the precise firing time (i.e. temporal jitter). An example of jitter can be seen with the

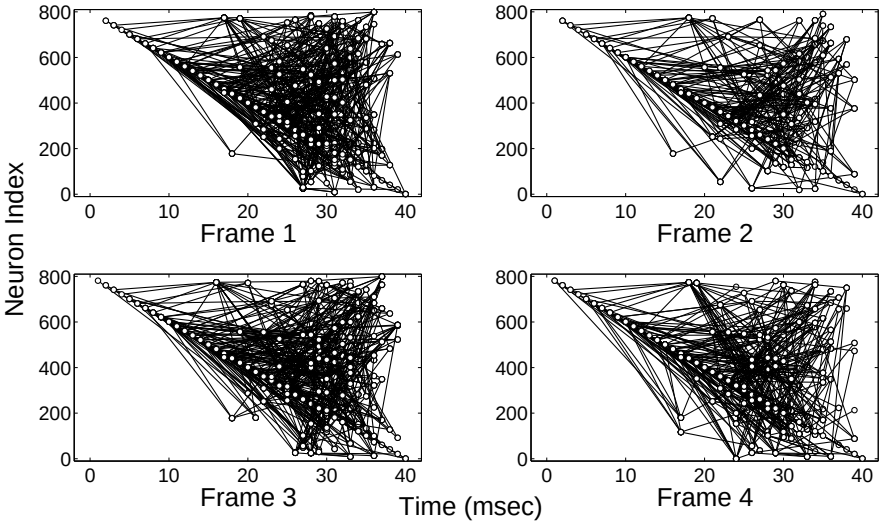


Figure 6: The PNG activations produced in four different frames in response to the descending input pattern. Nodes in the figure represent firing events, and the lines between nodes denote causal links inferred from the underlying network connectivity between event neurons. Only connections with saturated weights are shown (synaptic weight > 9.9). To produce this data, a network trained on the descending input pattern (thirty seconds at 5 Hz) was stimulated once per second using the descending pattern, producing a sequence of response frames.

firing event in the south-west region of each response: neuron 178 fires at $t = 16$ in frame 2 but $t = 18$ in frames 1 and 3 (it does not fire in frame 4). The same firing event is causal in producing either two, one or three subsequent firing events in frames 1, 2, and 3 respectively (observe the number of *out-going* lines from the node representing the event). This variation in the causality between the firing of neuron 178 and subsequent firing event might be due to the jitter in the firing of this neuron, or to jitter in the firing events of other neurons that are themselves causal in the firing of the same downstream neurons.

The network used for generating the data in Figure 6 is not a fully trained network having had only 150 presentations of the stimulus during the training period. The number of firing events and causal links in a structural diagram produced from a fully trained network is very dense, and a partially trained network was therefore chosen for clarity, to avoid obscuring the variability between frames.

The results shown in Figure 6 represent consecutive response frames generated from a single network state. However, the variability between frames limits the ability to compare between different network states, particularly in following changes in network response over the course of training. This variability can be reduced, and the generated structural diagrams made more accurate by averaging the stimulus response across multiple frames to produce a single response for each network state. Figure 7 demonstrates such a comparison by showing the evolution of the stimulus response as a network learns the ascending input pattern.

The forty input pattern events can be seen at each stage in Figure 7 as a straight ascending line of nodes. Any gaps between nodes are due to the removal of input pattern events that were not causal in the generation of later firing events. Nevertheless, almost all of the input pattern events participate in causal relationships, either with later input pattern events (these intra-pattern links are difficult to distinguish as they all occur on the same straight line), or with events that are a part of the subsequent PNG activation.

As learning proceeds, the size of the PNG activation resulting from each presentation of the stimulus increases. Most of this size increase occurs within one hundred seconds (500 stimulus presentations) although the probability scores suggest that the stimulus response is stable within the first minute of learning i.e. around 300 presentations (results not shown).

Figure 8 shows a different view of the data that focuses on just the firing events, removing the causal connections between events. The averaging method increases the selection accuracy for the filtering procedure that removes firing events unrelated to the stimulus response. In addition, averaging of the response across multiple frames also allows the empirical firing proba-

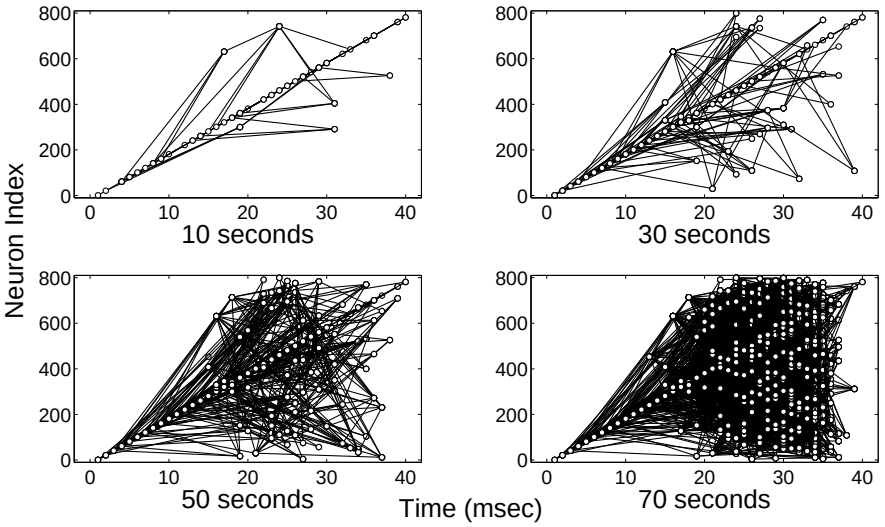


Figure 7: The evolution of the stimulus response over the course of training with the ascending input pattern. At four selected points (10, 30, 50 and 70 seconds) during training the network state was saved and the stimulus response to the ascending pattern was isolated. The stimulus response in each figure represents an average over one hundred response frames using a single stimulus presentation in each frame. The number of firing events and associated causal connections increases rapidly over the course of training.

bility of each firing event in the response to be calculated, generating a firing probability for each of the firing events that is consistently captured within a fingerprint window. Figure 8 shows changes in firing probability over the course of learning the ascending pattern. Firing events are colour-coded according to the empirical firing probability of the event, revealing a substantial increase in the firing probabilities as learning proceeds, particularly within the inhibitory neurons at the top of each diagram.

4. Discussion

The Response Fingerprinting method provides a new means of studying the response of a network to stimulus presentation and overcomes some of the difficulties of analyzing the stimulus response in terms of individual PNG activations. This new method views the stimulus response as a set of conditional firing probabilities for each of the neurons in the network. In the presence of a triggering stimulus, neurons that are involved in the resulting PNG activations have an elevated conditional probability of firing. The purpose of this technical report is to describe the background to the development of the method and also to provide some examples of the method in use. The first of the two provided examples tests the effect of stimulus degradation on the stimulus response, and the second example demonstrates the use of the Response Fingerprinting method to visualize PNG activation.

The effect of stimulus degradation was tested by progressively degrading a known stimulus and comparing the probability scores computed from each stimulus response. Both the window activation counts and the probability scores were found to decrease as the degree of stimulus degradation increased, although the probability scores were almost unaffected by small amounts of stimulus degradation. However, with heavy degradation of the stimulus, the probability scores became increasingly variable, suggesting that the score values were strongly influenced by which of the firing events were deleted in each iteration. Additional analyses support this hypothesis, with some firing events having a disproportionate impact on the probability score, at least on the single network used in this sample experiment. These early results suggest that some stimulus firing events (perhaps those that occur earlier) are more influential than others in producing PNG activation.

A particularly useful application of the Response Fingerprinting method is the visualization of PNG activation. Visualization supports the detailed study of significant features of PNG activation, such as the variability of the stimulus response and the role of inhibitory neurons in limiting activation. This visualization technique relies on the observation that the majority of firing events that constitute the PNG activations in each response frame

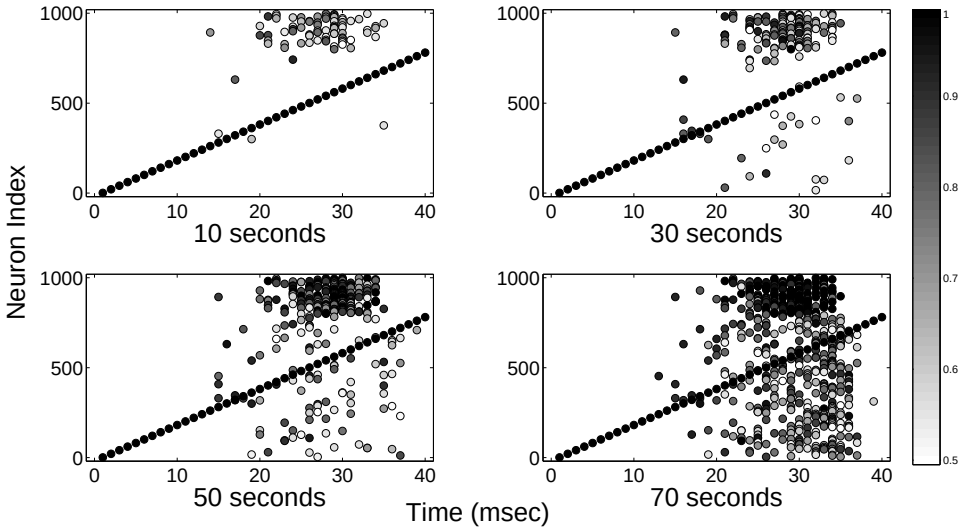


Figure 8: The evolution of the stimulus response over the course of training showing changes in firing probability. Each firing event is colour-coded according to the empirical firing probability of the event as measured over one hundred response frames (firing probability is encoded in the color bar on the right). The forty firing events that make up the ascending input pattern have a firing probability of 1.0 while the events in the resulting stimulus response often have significantly lower firing probabilities. Events with a firing probability lower than 0.5 were rejected. The firing probability of each event, and the number of firing events in the stimulus response both rapidly increase over the course of training. Inhibitory neurons (neurons 800-999) related to the stimulus response show a particularly large increase in both numbers and probabilities. The data shown in this figure and in Fig. 7 were produced in *separate* training runs of the same network and cannot therefore be compared. Note that inhibitory neurons (shown here) were not included in previous figures depicting PNG activation.

must occur within the temporal windows defined by the fingerprint because these firing events are strongly correlated with the stimulus. The fingerprint can therefore be used to filter the events in each response frame, assigning the firing events that are captured inside fingerprint windows to the stimulus response and rejecting the rest.

Analysis of the responses produced over multiple runs of the experiment found that the response of the first frame in each run was consistently muted when measured relative to subsequent frames. The first presentation of the stimulus occurs over the initial 40 milliseconds of the first one second frame and the subsequent stimulus response therefore occurs before the network has had the opportunity to come to equilibrium. The muted response in the first frame suggests that the firing events that make up the input pattern are not sufficient to trigger PNG activation, without the addition of the recurrent input produced by the underlying network dynamics of a network at equilibrium. To counter this reduced response in the first frame the network was first primed with ten frames of just random background stimulation, prior to presenting the first test stimulus.

The results from the visualization experiment show that the evolution of the stimulus response with training occurs very quickly, with a stable representation in as few as three hundred presentations of the stimulus. Almost all of the stimulus firing events were found to participate in causal relationships, although many of these causal connections were with later firing stimulus events at least initially (see Figure 7). Synapses between input pattern neurons are quick to learn the strong correlations between the individual firing events that make up the stimulus, and these intra-pattern links therefore tend to dominate the causal relationships in the early evolution of the stimulus response. As excitation builds within the PNG neurons, firing events that are not a part of the input pattern become increasingly involved in the PNG activation.

A significant feature of the mature stimulus response is that it does not extend beyond the last firing event of the input stimulus. The most likely explanation is that mounting inhibitory firing prevents any firing events that might occur later in the stimulus response, bringing the stimulus-driven PNG activations to an end (see Fig. 8). This activation dampening mechanism may also explain the apparently greater influence of earlier stimulus firing events on the successful stimulus response, as suggested by the stimulus degradation experiment. The ability to limit the temporal scope of PNG activation has implications for a representational system based on PNG activation and is worthy of further study.

References

Guise, M., Knott, A., Benuskova, L., 2013a. Evidence for response consistency supports polychronous neural groups as an underlying mechanism for representation and memory. In: Lecture Notes in Artificial Intelligence to appear.

Guise, M., Knott, A., Benuskova, L., 2013b. Spinula: software for simulation and analysis of spiking network models. Tech. rep., Dept of Computer Science, University of Otago, Dunedin, see http://www.cs.otago.ac.nz/research/techreports.php.

Izhikevich, E. M., Feb. 2006a. Polychronization: computation with spikes. Neural computation 18 (2), 245–82.

Izhikevich, E. M., 2006b. Reference software implementation for the Izhikevich model: minimal spiking network that can polychronize.
URL http://www.izhikevich.org/publications/spnet.htm

Izhikevich, E. M., Gally, J. a., Edelman, G. M., Aug. 2004. Spike-timing dynamics of neuronal groups. Cerebral cortex (New York, N.Y. : 1991) 14 (8), 933–44.

Martinez, R., Paugam-Moisy, H., 2009. Algorithms for structural and dynamical polychronous groups detection. Artificial Neural Networks ICANN 2009 5769, 75–84.

A. Methods

A.1. Networks

Sample experiments were conducted using networks composed of 1000 Izhikevich neurons (800 excitatory and 200 inhibitory) with parameters as described in Izhikevich (2006a). The networks were matured for two hours by exposure to 1 Hz random input generated by a Poisson process. Following maturation, the networks were trained on one of two input patterns or were left untrained.

A.2. Software

An implementation of the Response Fingerprinting method is available by downloading the Spinula software package (Guise et al., 2013b). Spinula is based on the reference software provided by Izhikevich (2006b) and is composed of a set of Microsoft Windows dynamic link libraries that provide functions for network construction, execution and analysis. These libraries can either be incorporated into a user program or instrumented using a Microsoft .Net scripting language such as F# Interactive. Some F# scripts are provided in later sections that allow a detailed reproduction of the example experiments.

A.3. Creating a response fingerprint

In order to employ this probabilistic approach, we must first gather some empirical data and generate a fingerprint. We need the following data: the frequency of presentation of each pattern during the data collection period; the temporal windows identified for each pattern; and for each temporal window of each pattern, the empirical probability of a window activation, both when the pattern is present and when the pattern is absent. Together these items provide the necessary pattern statistics that allow a naive Bayes classifier to compute the probability that a given input pattern was presented, given the evidence of coincident window activations within the current frame.

In the Spinula library, a response fingerprint is a type that exposes specialized data structures for the purpose of response classification. Of these, the most significant are the window map and the probability distribution map. The window map defines the layout of the temporal windows within the fingerprint while the probability distribution map holds an activation probability distribution for each temporal window in the fingerprint. Each of these probability distributions stores window activation frequencies as a conditional probability distribution over the presence or absence of the fingerprinted input pattern. The generation of a fingerprint therefore requires

that statistical data be gathered under two conditions: when the input pattern is present (i.e. the active pattern is the input pattern); and when the input pattern is absent (i.e. the active pattern is the *null-pattern*).

A fingerprint needs to store certain data that is needed to support its function, namely the input pattern, a frame profile, and a small set of parameters. Only these data items are saved to a file when a fingerprint is saved. The frame profile provides the core statistical data from which further statistics can be derived, while the parameter set provides arguments for the on demand generation of the window map and probability distribution map from this frame profile data. The parameter set specifies the window size, the frame size, the number of frames analyzed, and the initial and final threshold values. The primary task in the creation of a fingerprint is therefore the generation of a frame profile from the firing data produced by repeated stimulation of the network with the input pattern. By default, firing data is collected for each active pattern by running the network engine for 100 seconds in the presence of background stimulation. Typically, response profiles are only created for the excitatory neurons in the network.

A.3.1. Window selection

The primary window map is a map of window offset data keyed by neuron index and is generated on demand using the stored frame profile. For each neuron in the frame profile, temporal windows are selected by scanning a fixed sized typically 8 millisecond window along the time axis of the associated response histogram and selecting the temporal offset that produces the highest spike count within the sliding window. For each neuron a ratio is then computed of the window spike counts to the total spike count in the profile. The profiles for neurons in the input pattern typically produce spike count ratios approaching 100% while profiles for neurons further downstream produce progressively weaker ratios as the number of intervening causal links increases. Input pattern neurons are of course excluded from generating temporal windows.

Two levels of thresholding are applied over the course of window selection. Initial thresholding is applied to the spike count ratios with the effect of retaining only those profiles that show significant peaking.[5] Most neural profiles fail this initial thresholding step and therefore fail to produce a temporal window. A second thresholding step is applied to the ratio of the number of spikes in each window to the number of frames in the data (the stimulus

[5]The initial threshold defaults to 0.16, or five times the *expected fraction* assuming an equal distribution of spikes throughout the frame. Using a typical frame size of 250 msec and a window size of 8 msec, the expected fraction is 0.032.

response ratio). This final thresholding step requires that more than three quarters of the stimuli produce window activation and thus ensures that each of the selected windows is consistently activated by the input pattern.

A.3.2. Computing conditional probabilities of window activation

The probability distribution map stores the window activation probability for each temporal window in the fingerprint. The map is built from the stored frame profile by sampling the empirical probability of activation in both the presence and the absence of the input pattern. Background stimulation is provided in both *present* and *absent* sampling runs and there is therefore a non-zero probability of window activation within any of the selected temporal windows even when the input pattern is absent. The sampling procedure involves computing the proportion of spikes that fall within each selected window (the spike count ratio). For each temporal window, the resulting value is the conditional probability of window activation given the presence or absence of the pattern i.e. $p(WindowActivation \mid InputPattern)$, or $p(WindowActivation \mid \neg InputPattern)$.

A.3.3. Classification of the response

The fingerprint can now be used to probabilistically evaluate the firing data from a network and provide evidence that supports the presence or absence of a specific input pattern. The classification of the stimulus response from a network given a regular stimulus produces a probability score for each frame of the resulting data. The idea of scoring a frame is to compute the probability that the stimulus is present, given the coincident window spikes in the frame. The fingerprint provides the window layout and the conditional probability values for this process (the probability distribution map and the window map, respectively). The process begins by collecting firing data in the presence (or absence) of the test pattern and using the accumulated data to generate a map of temporal offsets for each firing event and each neuron in the frame. A list of the activated windows in each frame is then produced by using the window map as a mask and counting the spikes that fall both inside and outside each temporal window. The window is recorded as activated for each neuron if one or more spikes occur within the window.

Scores are generated as a probability distribution over the presence or absence of the input pattern as shown in Equation A.1.

$$P(ip_{present/absent} \mid act_{all}) \propto P(ip_{present}) \prod_{j} P(act_j \mid ip_{present/absent}) \qquad \text{(A.1)}$$

The conditional probabilities for each window are accumulated as follows: if the window was activated in the current frame, the values in the

probability distribution map are used directly (a probability distribution over $P(act \mid ip)$, assuming the pattern is present, or $P(act \mid \neg ip)$, assuming the pattern is absent); otherwise the probability distribution map values are subtracted from one (representing a probability distribution over $P(\neg act \mid ip)$ or $P(\neg act \mid \neg ip)$). The accumulated conditional probabilities are then multiplied with the input pattern frequency to produce two values that represent the probability distribution over the presence or absence of the pattern given the evidence from all window activations i.e. $P(ip \mid act_{all})$ and $P(\neg ip \mid act_{all})$.

B. Method Scripts

B.1. Response Histograms

Listing B.1 shows a script that generates and displays a frame profile for selected neurons, using firing data that has been accumulated from a network stimulated with the ascending input pattern. A trained network is stimulated 1600 times with the ascending input pattern and the resulting firing data is then folded into 250 millisecond frames to produce a firing response histogram for each neuron.

```
1  // Create and show response histograms for selected neurons
2  let CreateResponseHistograms pathToTrainedNetwork pathToOutputFolder =
3
4      let responseProfileTimeSlots = 250  // collect spike counts over this range
5      let runSeconds = 400
6      let backgroundFrequency = 1
7
8      // create an input pattern: the ascending input pattern at 4 Hz
9      let inputPattern = Stimulus.CreateLinearInputPattern(4, 1, 1, 40)
10
11     // load the state for a network trained on the ascending pattern
12     let network = CrossbarNetwork.CreateFromFile(
13             CrossbarNetworkSpecifier.N1000Network, pathToTrainedNetwork)
14
15     // select neurons for profiling
16     let selectedNeurons =
17         [
18             1;              // an input pattern neuron
19             90; 419; 746    // directly connected to neuron 1
20             2; 3; 5; 8;     // indirectly connected to input pattern neurons
21         ]
22
23     // run the network and collect firing data
24     network.RunWithDataCollection(runSeconds, backgroundFrequency, inputPattern)
25
26     // generate a response profile
27     let profile =
28         let firingData = network.OneSecondEventCollector.AllEventTriplets
29         let profileData = FrameProfile.GenerateFrameProfileData(responseProfileTimeSlots,
30                 network.TotalNeurons, selectedNeurons, firingData)
31         ResponseProfile(profileData)
32
33     // save the profile to a text file
34     let outputPathDescriptor = PathDescriptor.Create(pathToOutputFolder, "Profiles")
35     profile.Save(outputPathDescriptor)
36
37     // show the response histograms as a chart
38     let maxY = profile.GetMaxProfileSpikeCount(selectedNeurons)
39     SpikeProfileVisualisation.ShowReponseProfileHistograms("Counts for Selected Neurons",
40             outputPathDescriptor, profile, maxY, selectedNeurons)
```

Listing B.1

On line 9, a forty neuron ascending input pattern is generated at four hertz. The network state data from a network trained on the ascending pattern is then loaded into a newly created network at lines 12 and 13. Lines 16 to 21 define a list of eight indices for selected neurons that are indirectly or directly connected to neuron 1 (one of the input pattern neurons). The network is then stimulated 1600 times over 400 seconds with the ascending input pattern in the presence of 1 Hz background (line 24). Note that synaptic plasticity is disabled during data collection. The accumulated firing data

(around 1,700,000 firing events using a 4 Hz input pattern) is then passed to a method in lines 29 and 30 that generates response histograms for each selected neuron. The resulting frame profile is saved in lines 34 and 35, and then displayed in lines 39 and 40. The method call at line 38 determines the largest spike count in the response histograms which is used to set the maximum Y-value for the chart.

The process of creating a frame profile (lines 29 and 30) involves folding the collected firing data into fixed-sized frames to produce a firing response histogram for each neuron. Typically only excitatory neurons are profiled and the first step is therefore to filter the firing data to remove the firing events of inhibitory neurons. The event time is then remapped into the specified frame size and a list of firing response times is created for each neuron as an offset relative to the start of frame. Each response time list is then aggregated into a map of spike counts keyed by time, creating a response profile for each of the selected neurons. Some of these response histograms show distinctive peaks in the spike counts, producing a unique response signature when viewed over multiple neurons.

B.2. Network Training

Listing B.2 shows a script that trains each network in a folder of matured networks using a single input pattern (either the Ascending or Descending pattern) at 5 Hz. The trained networks are saved in separate folders, one for the Ascending pattern and one for the Descending pattern. The training period is typically 1200 seconds (twenty minutes) and the background firing rate is normally set to 1 Hz.

```
1  // Train each mature network in the specified folder
2  let Net20TrainWithCoherentStimulation runSeconds backgroundFiringRate
3          rootInputFolderPath rootOutputFolderPath basename saveState saveInterval =
4
5      // Run a mature network while providing coherent external stimulation
6      let TrainWithCoherentStimulation stateFilePath runSeconds patternType patternStimulationsPerSecond
7                        backgroundFiringRate pathToOutputFolder basename saveState saveInterval =
8
9          // create an input pattern composed from a 40 msec ascending or descending pattern
10         // continuously repeated over each second at a rate specified by patternStimulationsPerSecond
11         let inputPattern =
12             let patternStep = if patternType = InputPatternType.Ascending then 1 else -1
13             Stimulus.CreateLinearInputPattern(patternStimulationsPerSecond, 40, 1, patternStep, 40)
14
15         // load the untrained network
16         let network = CrossbarNetwork.CreateFromFile(stateFilePath)
17
18         // train the network with the input pattern
19         let saveParameters =
20                 new NetworkStateActionParameters(pathToOutputFolder, basename, saveState, saveInterval)
21         network.Train(inputPattern, runSeconds, backgroundFiringRate, saveParameters)
22
23
24     // train on both ascending and descending patterns at 5 Hz
25     let combinations = [ (InputPatternType.Ascending, 5); (InputPatternType.Descending, 5); ]
26
27     // get all the untrained networks
28     let filesMap = Span.GetFilesMap(rootInputFolderPath, basename, "txt")
29
30     for patternType, patternStimulationsPerSecond in combinations do
31         let outputFoldername =
32                 sprintf "Trained%s%d" (patternType.ToString()) patternStimulationsPerSecond
33         let pathToOutputFolder = Path.Combine(rootOutputFolderPath, outputFoldername)
34
35         for kvp in filesMap do
36             let key, stateFilePath = kvp.Key, kvp.Value
37
38             let outputBasename = sprintf "%s%d" basename key
39
40             TrainWithCoherentStimulation stateFilePath runSeconds patternType
41                     patternStimulationsPerSecond backgroundFiringRate
42                     pathToOutputFolder outputBasename saveState saveInterval
```

Listing B.2

Two parameters to this script (*saveState* and *saveInterval*) determine whether the network state will be saved over the course of training. Typically *saveState* is set to false and *saveInterval* to 0 so that only the final trained network is saved.

Lines 6 to 21 define a function (*TrainWithCoherentStimulation*) that performs the training for each network: lines 11 to 13 generate an input pattern (either Ascending or Descending) and line 16 loads the network state data of the current network. The *save parameters* defined in lines 19 and 20 determine the output folder and base filename used for saving the network state,

both over the course of training and for the final trained network.

The script begins on line 25 by setting the required training parameters; here they specify the Ascending and Descending patterns with a training frequency of 5 Hz for both patterns. Line 28 retrieves a list of the mature networks to be trained and lines 30 to 42 iterate through the training combinations (outer loop) and the network list (inner loop). The *TrainWithCoherentStimulation* function is called in lines 40 to 42 to perform the actual training procedure on each network.

B.3. Response Profiling

Listing B.3 shows a script that extracts and saves frame profiles for each network in a folder of trained networks. The frame profile specifications are defined by a list of pattern and pattern name pairs, with a unique frame profile generated for each combination of pattern and network.

```
1  // For each trained network in the specified folder, extract the frame profile statistics and save
2  let Net20ExtractProfiles backgroundFiringRate rootInputFolderPath rootOutputFolderPath basename =
3
4      let ExtractFrameProfile outputFolderPath basename (stateFilePath:string)
5              profileSpecs backgroundFiringRate =
6
7          let engine = new CrossbarNetwork(CrossbarNetworkSpecifier.N1000Network)
8          engine.LoadNetworkState(stateFilePath)
9
10         profileSpecs
11         |> Seq.iter (fun (pattern, patternName) ->
12                 let fp = new FrameProfile(engine, pattern, patternName, backgroundFiringRate, true)
13
14                 let fileName = sprintf "profile_%s_%s.txt" basename patternName
15                 fp.Save(Path.Combine(outputFolderPath, fileName))
16             )
17
18     // create profiles for these pattern/pattern name pairs
19     let profileSpecs =
20         let CreatePattern step =
21             let events = Pattern.CreateLinearPattern(1, step, 40)
22             new Pattern(events)
23         [ (1, "Ascending"); (-1, "Descending"); (0, "Null"); ]
24         |> Seq.map (fun (step, patternName) ->
25                 if step = 0 then
26                     None, patternName
27                 else
28                     Some(CreatePattern step), patternName
29             )
30         |> Seq.toList
31
32     // for both ascending and descending trained networks
33     let folderCombinations = [ (InputPatternType.Ascending, 5); (InputPatternType.Descending, 5); ]
34
35     for patternType, patternStimulationsPerSecond in folderCombinations do
36
37         let foldername = sprintf "Trained%s%d" (patternType.ToString()) patternStimulationsPerSecond
38         let inputFolderPath = Path.Combine(rootInputFolderPath, foldername)
39         let outputFolderPath = Path.Combine(rootOutputFolderPath, foldername)
40
41         // get all the trained networks in this folder
42         let filesMap = Span.GetFilesMap(inputFolderPath, basename, "txt")
43
44         for kvp in filesMap do
45             let key, stateFilePath = kvp.Key, kvp.Value
46
47             let outputBasename = sprintf "%s%d" basename key
48
49             ExtractFrameProfile outputFolderPath outputBasename stateFilePath
50                     profileSpecs backgroundFiringRate
```

Listing B.3

Lines 4 to 16 define a function (*ExtractFrameProfile*) that extracts a frame profile from the current network for each pattern in the profile specification. Lines 7 and 8 load the current network state data into a new network ready for profiling. Lines 10 to 16 iterate through the required patterns in the profile specification: for each pattern a frame profile is generated by repeatedly stimulating the network with the pattern (typically in the presence of 1 Hz

background firing) and analyzing the resulting firing data. The profile is then saved in line 15.

The main part of the script begins by defining the profile specification. Here the specification in lines 19 to 30 specifies the Ascending, Descending and null patterns for profiling, with each pattern paired with the corresponding pattern name. Line 33 defines the different sets of trained networks to be profiled: in this case trained networks stored in both the TrainedAscending5 and TrainedDescending5 subfolders will be profiled. Lines 35 to 50 then iterate through each subfolder, retrieving a list of trained networks (line 42) and iterating through each file (lines 44 to 50). For each file, the function *ExtractFrameProfile* is called to generate the required frame profiles.

B.4. The effect of stimulus degradation

Listing B.4 shows a script that examines the effect of stimulus degradation on the activation response. The script accepts a list of degradation values that determine the percentage of stimulus degradation that will be tested e.g. [10, 30, 50] where a 10% degradation value indicates that one tenth of the firing events will be randomly deleted. For each degradation value, a naive Bayes classifier computes the probability that the pattern is present. To allow for variations in the influence of deleted firing events in failing to trigger the stimulus response, the test is performed multiple times at each level with a different randomly selected set of deletions for each iteration (see the *iterations* parameter). The probability computation uses a matched pair of frame profiles (stimulus present/absent) as if the test were being performed on an undegraded stimulus.

```
1  // Compute pattern scores using progressively degraded input patterns
2  let ScoreDegradedPattern (percentDegradationList:seq<int>) iterations
3          (stateFilePath:string) (profilePair:MatchedProfilePair) (outputStream:StreamWriter) =
4
5      let engine = new CrossbarNetwork(CrossbarNetworkSpecifier.N1000Network)
6      engine.LoadNetworkState(stateFilePath)
7
8      let classifier = new NaiveBayesClassifier(engine)
9
10     for percentDegraded in percentDegradationList do
11
12         // score multiple instances of the degraded pattern
13         let scores, windowActivations =
14             seq {
15                 for sample in 1..iterations do
16                     let pattern =
17                         let undegradedPattern = profilePair.TestPatternProfile.Pattern.Value
18                         let proportionToKeep = 1.0 - (float percentDegraded / 100.0)
19                         undegradedPattern.CreateDegradedPattern(proportionToKeep)
20
21                     // average scores over ten frames
22                     yield classifier.ComputeProbabilityPresent(pattern, profilePair, 10)
23             }
24             |> Seq.toList
25             |> List.unzip
26
27         let scoreStrings, windowActivationStrings =
28             System.String.Join(" ", scores), System.String.Join(" ", windowActivations)
29
30         outputStream.WriteLine(percentDegraded)
31         outputStream.WriteLine(scoreStrings)
32         outputStream.WriteLine(windowActivationStrings)
33         outputStream.Flush()
```

Listing B.4

Lines 5 and 6 create a new network by loading the specified network state. The classifier is initialized in line 8. Lines 10 to 33 iterate over each level of degradation, while an inner loop at lines 15 to 22 performs multiple iterations of each test. An averaged probability score is generated at line 22 using a degraded version of the frame profile stimulus that is generated in lines 16 to 19. The pair of values generated by the test at line 22 is separated (unzipped) at line 25. Lines 27 to 33 aggregate the scores and save them to a file.

B.5. Visualizing PNG activation

Listing B.5 shows a script that allows PNG activation to be visualized by filtering the firing data in a response frame through the temporal windows defined by the pattern fingerprint. The script generates multiple structural diagrams, one for each response frame, and is therefore useful for showing variability in the stimulus response across consecutive frames.

```
1  // Show PNG activation over multiple frames
2  let ShowPNGActivation pathToNetworkStateFile pathToFrameProfileFile
3          pathToOutputFolder baseName =
4
5      let numberOfFrames = 10       // show this many frames
6      let maxPNGLength = 50         // maximum length (for image saving only)
7      let synapticThreshold = 9.9   // exclusive threshold: weights must be greater than this value
8      let saveGraph = true
9
10     let network = CrossbarNetwork.CreateFromFile(
11             CrossbarNetworkSpecifier.N1000Network, pathToNetworkStateFile)
12
13     // collect firing data in response to the selected pattern
14     // and create a list of activation-related firing events for each frame
15     let frameGroups =
16         let profile = FrameProfile.Load(pathToFrameProfileFile)
17         let classifier = NaiveBayesClassifier(network)
18         classifier.GetIndividualFrameGroups(numberOfFrames, profile)
19
20     // save each frame
21     for index in 0..numberOfFrames-1 do
22
23         let savePath =
24             let fileBase = sprintf "%s_%d" baseName index
25             PathDescriptor.Create(pathToOutputFolder, fileBase)
26
27         // generate a PNG structural description (saturated weights only)
28         let groupDescriptor =
29             network.GetPNGDescriptor(frameGroups, index, synapticThreshold)
30
31         // save the network graph data
32         groupDescriptor.Save(savePath)
33
34         // optionally save the image
35         if saveGraph then
36             PNGVisualisation.SaveGraph(savePath, maxPNGLength, network.TotalNeurons, groupDescriptor)
```

Listing B.5

In lines 10 and 11 of this script, a new network is initialized by loading the network state of a trained network. A frame profile containing response histograms for the required input pattern is then loaded at line 16. Using the input pattern associated with this profile, a Bayesian classifier then performs ten trials, generating ten frames of firing event data (line 18). Instead of scoring each frame, the classifier uses the pattern windows defined by the profile to select firing events related to group activation. Each of the ten sets of filtered firing events is then processed to extract a PNG structural description (lines 28 and 29) and the resulting graphs are saved (lines 32 and 36).

Listing B.5 generates multiple structural diagrams from a single network state. The next script, shown in Listing B.6 averages the stimulus response

over multiple frames producing a single structural diagram. Averaging increases the accuracy of the resulting structure and allows the empirical firing probability of each firing event in the response to be generated. The averaging method is useful for creating structural diagrams that compare different network states, such as the network states saved over the course of learning.

```
1  // Show PNG activation using an averaged stimulus response over multiple frames
2  // Generate an empirical probability for each window activation
3  let ShowAveragedPNGActivation stateFilePath trainedProfilePath outputFolder basename =
4
5      let numberOfFrames = 100         // average over this many frames
6      let maxPNGLength = 50           // maximum length (for image saving only)
7      let threshold = 0.5             // keep firing events with at least this firing probability
8      let synapticThreshold = 9.9     // exclusive threshold: weights must be greater than this value
9      let saveGraph = true
10
11     let network = CrossbarNetwork.CreateFromCompressedFile(
12             CrossbarNetworkSpecifier.N1000Network, stateFilePath)
13
14     // collect firing data in response to the selected pattern
15     // and create a list of group-related firing events averaged over multiple frames
16     let response =
17         let profile = FrameProfile.Load(trainedProfilePath)
18         let classifier = NaiveBayesClassifier(network)
19         classifier.GetAveragedResponse(numberOfFrames, threshold, profile)
20
21     // save the response
22     let responsePath =
23         let filename = sprintf "response_%s.txt" basename
24         Path.Combine(outputFolder, filename)
25     response.Save(responsePath)
26
27     // generate a PNG structural description (saturated weights only)
28     let groupDescriptor = network.GetPNGDescriptor(response.Data, synapticThreshold)
29
30     let savePath =
31         let filename = sprintf "descriptor_%s" basename
32         PathDescriptor.Create(outputFolder, filename)
33
34     // save the network graph data
35     groupDescriptor.Save(savePath)
36
37     // optionally save the image
38     if saveGraph then
39         PNGVisualisation.SaveGraph(savePath, maxPNGLength, network.TotalNeurons, groupDescriptor)
```

Listing B.6

The script is similar to the previous script except that only a single averaged response is generated rather than a response for each frame. The important difference between the two scripts is the call on line 19 to a classifier method that produces the averaged response. Internally, this method collects response frames and accumulates the spike times for each window across all frames. Typically one hundred frames are collected and windows that are only infrequently activated over the collected frames are rejected. Firing events are generated for each window by pairing the associated window neuron with the average spike time for the window. The empirical probability of activation is calculated for each window by dividing the number of activations by the total number of frames.