CONTINUAL SCENE DETECTION

Leon Li
Columbia University
New York
{al4263}@columbia.edu

ABSTRACT

In this paper, we propose a novel method for training fluid queues using direct backpropagation, capitalizing on their differentiable dynamics compared to traditional stochastic queueing networks. Fluid queues, which disregard the inherent stochasticity in arrival and service processes, offer a fully deterministic framework for modeling and optimization. By employing modern deep neural network architectures and leveraging the continuous nature of fluid queues, we can efficiently train a parametrized policy to minimize the cost of the system.

1 Problem Formulating

In this paper, we investigate the problem of continual scene detection. A *frame* $\boldsymbol{f}$ is defined as a vector $\mathbb{R}^n$ where each component is sampled from a Bernoulli distribution with probability $p = 0.5$ with possible outcomes in the set $\{-1, 1\}$. A *scene* s is composed of $n = 4$ frames. The first frame $\boldsymbol{f_1}$ being randomly generated and the subsequent frames being derived from the first frame with a degree of variation of 0.1. Specifically, this means we randomly flip the values of 10% of the components of $\boldsymbol{f_1}$ to create the subsequent frames. We continuously generate a stream of scenes. At a certain time point $t = R$, a decision is made to either generate a new scene or re-use the previous one to create a *familiar scene*. If a familiar scene is chosen, the scene from time $t = T - R$ is permuted randomly. However, if a familiar scene is generated at time $t = T$, it will not be generated again at $t = T + R$. The task at hand is to continuously identify whether a presented scene is familiar or novel.

2 Introduction to Memory Networks

Memory Networks comprise of different architectural designs, which include a Memory Layer acting as the initial hidden layer, and a variety of types of readout layers.

2.1 Memory Layer

The Memory Layer is a fully connected layer, endowed with either additive plasticity (A) or multiplicative plasticity (M). The weights and plasticity matrix at time t for this layer are represented by W_1 and $P_1(t)$ respectively.

The method of updating the hidden layer hinges on the type of plasticity as described below:

- **Multiplicative Plasticity (M):** The hidden layer $h_1(t)$ at time t is updated using the following equation:

$$\boldsymbol{h_1}(t) = \phi\left(\left(\boldsymbol{W_1} \cdot \boldsymbol{P_1}(t)\right)\boldsymbol{x}(t) + \boldsymbol{W_1}\boldsymbol{x}(t) + \boldsymbol{b_1}\right) \tag{1}$$

- **Additive Plasticity (A):** The hidden layer $h_1(t)$ at time t is updated as follows:

$$\boldsymbol{h_1}(t) = \phi\left(\left(\boldsymbol{W_1} + \boldsymbol{P_1}(t)\right)\boldsymbol{x}(t) + \boldsymbol{b_1}\right) \tag{2}$$

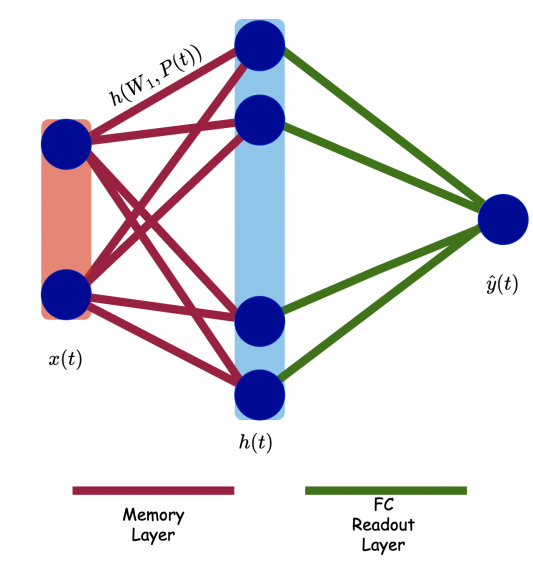
- **Stacked Plasticity (Stack):** The hidden layer $h_1(t)$ at time t is for the Stacked Plasticity is generated by stacking a **M** and a **A**. So the upper half of the network works as the **M** and the second half works as **A**.

In these equations, ϕ represents the $\tanh$ function. The plasticity matrix update rule is expressed as follows for both types:

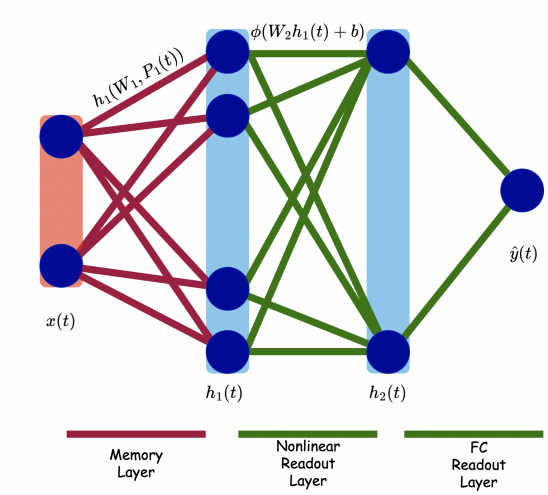
$$\boldsymbol{P}(t) = \lambda \boldsymbol{P}(t-1) + \boldsymbol{\eta} \odot \left(\boldsymbol{h_1}(t)\boldsymbol{x}(t)^T\right) \tag{3}$$

Observe that in this context, $\boldsymbol{\eta}$ is not a scalar. In fact, there exist three distinct forms of η, namely scalar, neuron-wise, and synaptic-wise. For scalar η, it retains its usual scalar nature. In the case of neuron-wise $\boldsymbol{\eta}$, it manifests as a vector of the same dimensions as $h_1(t)$, where each constituent of η individually controls each row of $\boldsymbol{h_1}(t)\boldsymbol{x}(t)^T$. The synaptic-wise $\boldsymbol{\eta}$ form is a matrix identical in size to $\boldsymbol{h_1}(t)\boldsymbol{x}(t)^T$, with each element acting as a heterogenous learning rate for its corresponding element in $\boldsymbol{h_1}(t)\boldsymbol{x}(t)^T$.

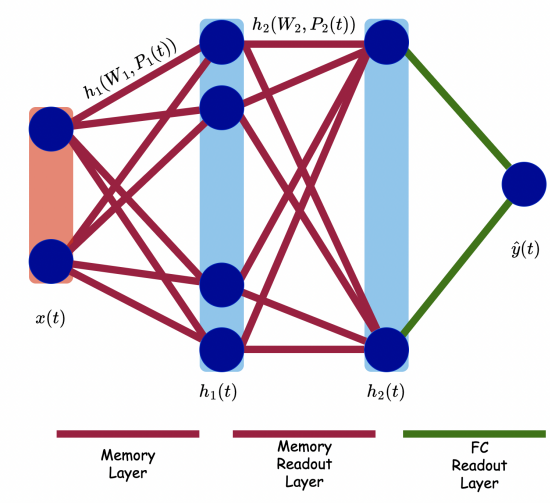
2.2 Readout Layer



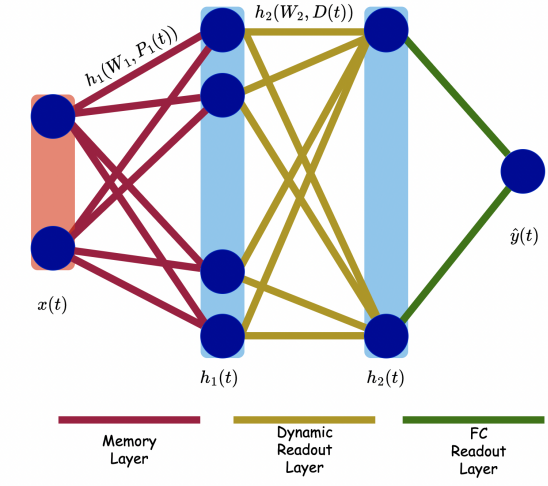
(a) Fully Connected Readout



(b) Nonlinear Readout



(c) Memory Based Readout



(d) Dynamic Read Out

Figure 1: The four types of readouts

For the readout layer, we present four different versions: Fully Connected readout, Nonlinear readout, Memory Based readout, and Dynamic readout:

- **Fully Connected readout (FC):** This version is a plain fully connected layer, represented as follows:

$$\sigma\left(\boldsymbol{W_f h_1}(t) + \boldsymbol{b_f}\right) \tag{4}$$

- **Nonlinear readout (Nonl):** This version transforms $\boldsymbol{h_1}(t)$ with a nonlinear layer before a FC transformation:

$$\sigma\left(\boldsymbol{W_2 h_1}(t) + \boldsymbol{b_2}\right), \tag{5}$$
$$\sigma\left(\boldsymbol{W_f h_2}(t) + \boldsymbol{b_f}\right) \tag{6}$$

- **Memory Based readout (Memo):** This version leverages the benefit of additional layers of memory for improved performance. This transformation, identical to the one performed by the first memory layer, is denoted by $h_2 = h(\boldsymbol{W_2}, \boldsymbol{P_2}(t))$. Subsequently, we apply the same FC layer transformation:

$$\sigma\left(\boldsymbol{W_f h_2}(t) + \boldsymbol{b_f}\right) \tag{7}$$

- **Dynamic readout (Dyn):** This version is inspired by the observation that the decay rate λ in **Memo** is extremely low, which signifies minimal memory retention. **Dyn** addresses this issue by modifying the update rule for the plasticity matrix as follows:

$$\boldsymbol{P}(t) = \boldsymbol{\eta} \odot \left(\boldsymbol{h_1}(t)\boldsymbol{x}(t)^T\right) \tag{8}$$

  This change results in a dynamic readout of the current time step. Apart from this modification, everything else in **Dyn** mirrors **Memo**. The dynamic transition is denoted by $h_2 = h(\boldsymbol{W_2}, \boldsymbol{D}(t))$. Following this, the same FC layer transformation is applied:

$$\sigma\left(\boldsymbol{W_f h_2}(t) + \boldsymbol{b_f}\right) \tag{9}$$

3 Results & Analysis

3.1 overall comparision

To maintain an equitable comparison, we constrain the number of dynamic neurons to be equivalent across all models. In our investigation, we considered models with dynamic neuron counts of 50 and 100. Their respective performance metrics are detailed in the table below:

Max R													
Size \ Net	**M**						**A**						**Stack**
	FC	**Nonl**	**Dyn**		**Memo**		**FC**	**Nonl**	**Dyn**		**Memo**		**FC**
			M	**A**	**M**	**A**			**M**	**A**	**M**	**A**	
50	5	7	9	9	8	9	6	4	4	4	4	4	6
100	9	10	12	13	9	14	9	7	5	5	5	5	7

Table 1: Performance metrics for different dynamic neuron sizes

Analyzing the table, we can deduce the following:

1. Across most types of readouts, **M** outperforms **A**.
2. **M** benefits from more intricate readouts and scaling. Conversely, the performance of **A** diminishes with more complex readouts.
3. The most noteworthy performance is achieved by the memory layer of **M**, combined with either **Dyn** or **Memo** readout layers.

A graph of performance is here:

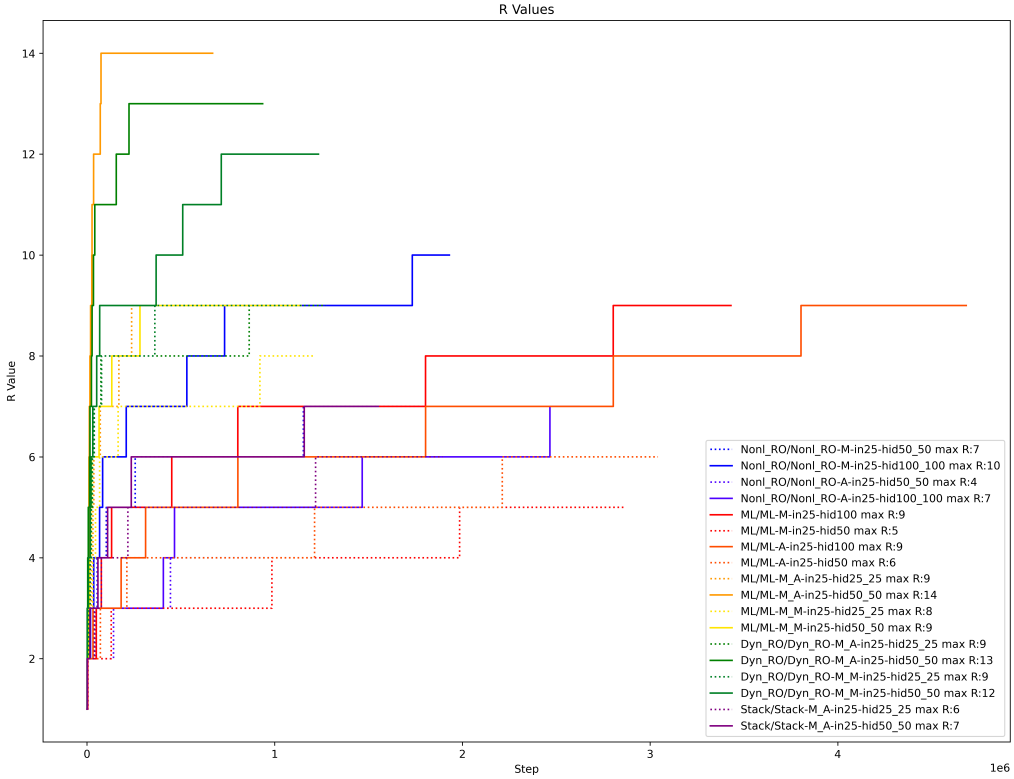


Figure 2: Performance across all networks and sizes

4 M vs A

In this section, we address the possible reasons why multiplicative plasticity matrix has a better performance. From the graph below, we can see that a complex readout hurts the performance of A networks, whereas a complex readout helps the performance of M networks.

To figure out why the network works, we plot the hidden layer activities across time, the eta matrix, the weight matrix, the actual weight/plasticity across time. Specifically, we choose the **Memo** Network with **M** memory and **A** readout:

We plot the actual weight and the plasticity matrix with t from 74-77, where at 74,75 the network output 0 and 76,77 the network output 1:

Notice that the **M** is dominated by several elements and rarely changes. So a hypothesis would be that most of the identification work were done by the dynamic readout, whereas **M** merely serves as a memory source.

References

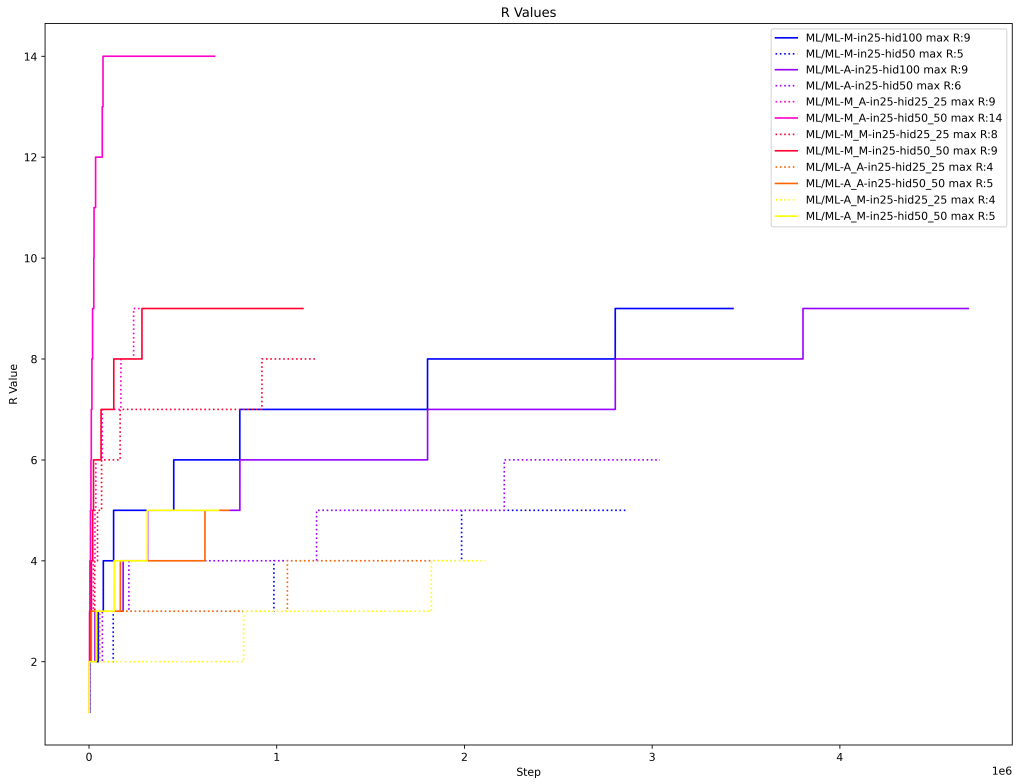


Figure 3: M vs A

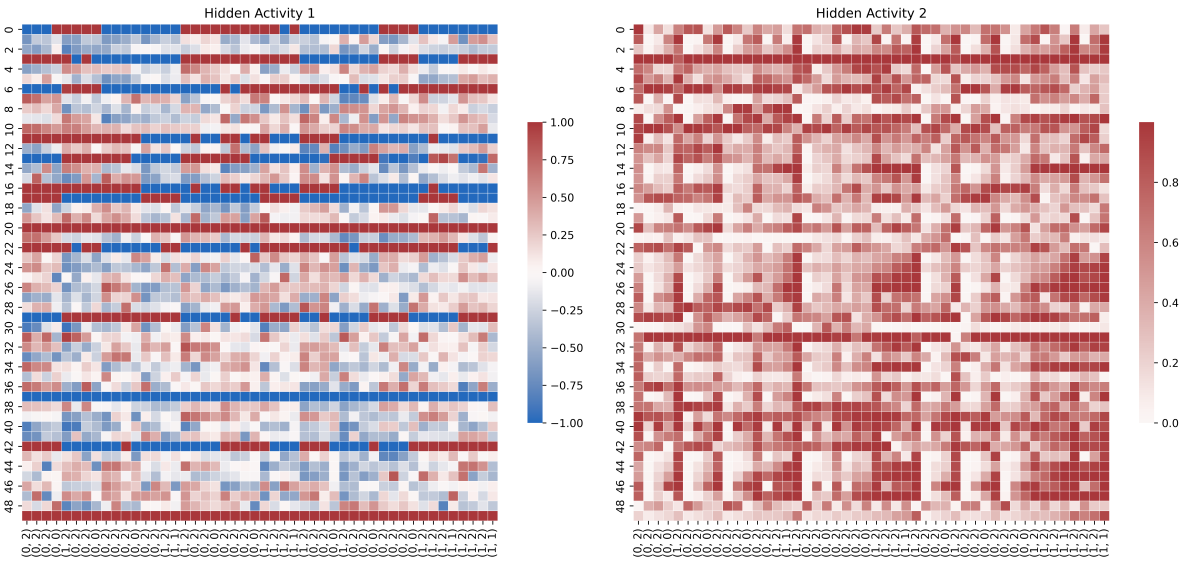
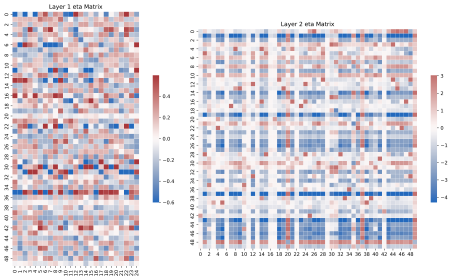
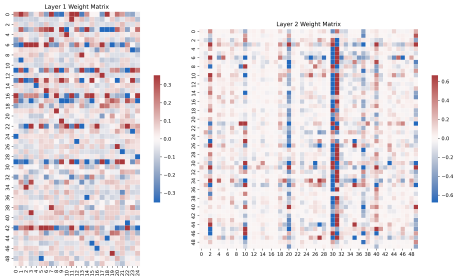


Figure 4: Hidden Layers Across Time

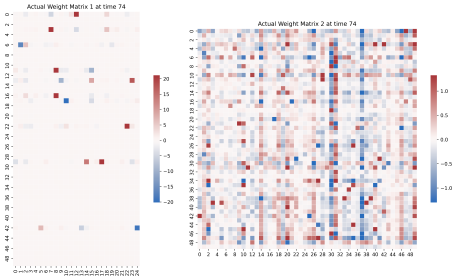


(a) eta matrix of **Memo M A** 50/50

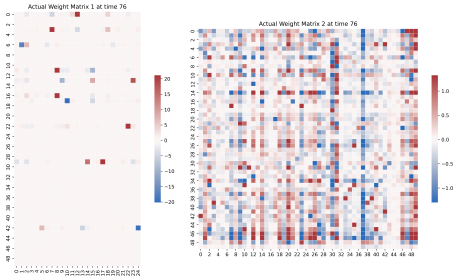


(b) weight matrix of **Memo M A** 50/50

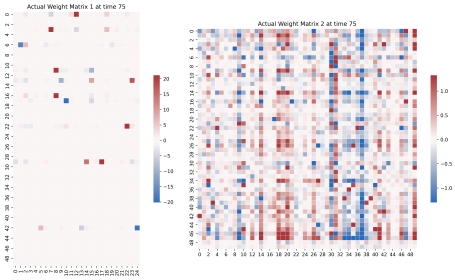
Figure 5: Dyn RO MA 50/50



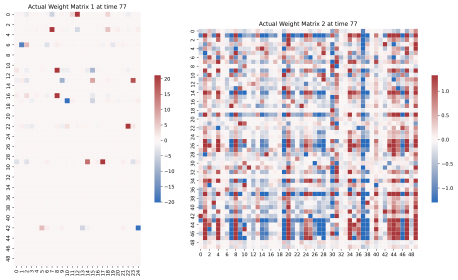
(a) T=74



(c) T=76

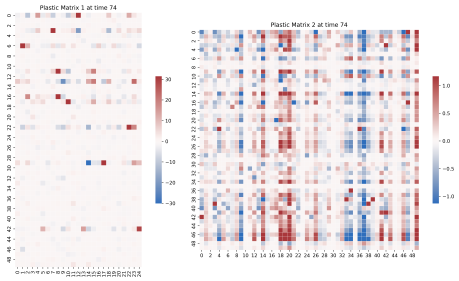


(b) T=75

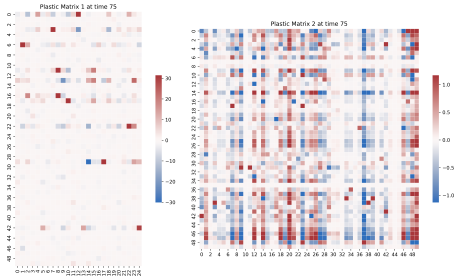


(d) T=77

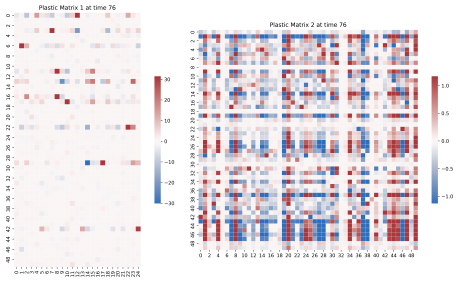
Figure 6: Actual weight



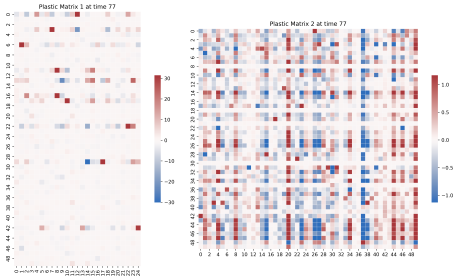
(a) T=74



(b) T=75



(c) T=76



(d) T=77

Figure 7: Plasticity