- Network Equations (updated):

$$
\begin{aligned}
z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \in \mathbb{R}^n \\
v_t &= (1 - z_t) v_{t-1} + z_t (W_v x + U_v h_{t-1} + \alpha U_{t-1}^{fast} h_{t-1} + b_v) \in \mathbb{R}^n \\
h_t &= [v_t]_+ \in \mathbb{R}^n \\
mod_t &= [W_{mod} h_t + b_{mod}]_+ \in \mathbb{R}^m, m < n
\end{aligned}
$$

  - The modulation signals (m_t, s_t) and learning rates (α, τ_U) are made scalar, which makes the derivation in the previous report valid, and also saves parameter. No drop in performance was found. This would make the fast weight perfectly skew-symmetric. The eligibility traces and fast weights are updated similarly to the last write-up.

$$
\begin{gathered}
\text{neuronal eligibility trace, related to the time window of STDP, controlled by r-gate} \\
r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \in \mathbb{R}^n \\
e_t = (1 - r_t) e_{t-1} + r_t h_t \in \mathbb{R}^n \\
\\
\text{synaptic eligibility trace, controlled by s-gate} \\
s_t = \sigma(w_s^\top mod_t + b_s) \in \mathbb{R} \\
E_t = (1 - s_t) E_{t-1} + s_t (h_t e_{t-1} + e_{t-1} h_t) \in \mathbb{R}^{n \times n} \\
\\
\text{fast weight, controlled by m, which can be positive or negative} \\
m_t = w_m^\top mod_t + b_m \in \mathbb{R} \\
U_t^{fast} = (1 - \tau_U) U_{t-1}^{fast} + \tau_U m_t E_t \in \mathbb{R}^{n \times n}
\end{gathered}
$$

  - alpha is the softplus transformation of a free parameter $\alpha = \ln(1 + \exp(\tilde{\alpha})) \in [0, \infty)$. tau_U is the sigmoid transformation of another free parameter $\tau_U = \sigma(\tilde{\tau}_U) \in [0, 1]$. Initializations of these parameters matter a lot.
  - The fast weight's magnitude is contrained otherwise the network's activity can be unstable. We want $|U_v + \alpha U^{fast}| < C$. So we enforce each entry of the fast weight to be $(-C - U_v)/\alpha < U^{fast} < (C - U_v)/\alpha$
- Omniglot Few Shot Classification
  - In each episode, the network is given a set of support images, their corresponding labels, and a test image. The task is to provide the label of the test image, which should be of the same category with one or more of the support images.
  - The is a hard version of the Associative Retrieval Task: the network needs to learn the association between image and label, and retrieve the correct label.
  - The image input to the network is encoded by a four layer CNN (3x3 conv->2x2 max pool->ReLU->BatchNorm). The label is encoded with a embedding layer. They are concatenated and passed to the recurrent network as input.
  - Each pair is presented at least 2 times (**presenting more times actually lead to faster convergence in experiments I tried, as long as τ_U is initialized properly to account for longer sequence length**). Then the test image (only one) is presented for a few times, concatenated with a novel label (for example, if there are 20 categories in each episode, the test image is concatenated with an embedding of 21).

- Maze with switching reward
    - A small T-maze like the one below (1 are walls, 0 are legal positions)

$$
\begin{matrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & l & b & r & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & x & 1 & 0 & 1 \\
1 & 0 & 0 & d & 0 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1
\end{matrix}
$$

    - Episode Structure
        - Two phases
            - In the Approach phase, b=0, d=1. The agent starts at x. It can only move forward. After it arrives at point marked with b, it either moves left (to l) or right (to r). Only one will be rewarded. After it leaves b, it moves into the next phase.
            - In the Return phase, b=1, d=0. The agent should return back to x. Then the next trial starts.
        - The agent receives a negative reward for running into wall. It also receives a negative reward at each time step to encourage shorter paths.
        - The agent receives a large positive reward for arriving at the correct goal. In each trial, the reward has a probability p of being at l, and $1-p$ at r. This reward probability remains the same for a few trials, then it switches.
        - The best strategy is to constantly visit the site which leads to more reward, until a switch happens.
    - Model Fitting
        - adapted from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4588166/
        - To see if the model is performing implicit reinforcement learning, we fit a simple RL algorithm to the model's behavior
        - Three parameters to fit
            - α: learning rate
            - β: inverse temperature of the softmax
            - ϵ: "lapse rate", stochasticity in the model's choice
        - $Q:\{L,R\}\to\mathbb{R}$ is a value function that maps action (going left or right) to a real value
        - Q is updated in the following equation, as in the Rescorla-Wagner model

$$Q_t(a) = Q_{t-1}(a) + \alpha(r_{t-1} - Q_{t-1}(a))$$

        - Action is selected with the probabilities:

$$

```
P_t(L) = \epsilon+(1-2\epsilon)\cdot \frac{\exp(-\beta Q_t(L))}{\exp(-\beta Q_t(L))+\exp(-\beta Q_t(R))}\\ \\
```

```
= \epsilon+\frac{1-2\epsilon}{1+\exp(-\beta(Q_t(R)-Q_t(L)))}\\\ \\
P_t(R) = 1-P_t(L)
```

$$

$$
\begin{aligned}
P_t(L) &= \epsilon + (1-2\epsilon)\cdot\frac{\exp(-\beta Q_t(L))}{\exp(-\beta Q_t(L)) + \exp(-\beta Q_t(R))} \\
&= \epsilon + \frac{1-\epsilon}{1+\exp(-\beta(Q_t(R)-Q_t(L)))} \\
P_t(R) &= 1 - P_t(L) \\
P_t(L) &= \epsilon + (1-2\epsilon)\cdot\frac{\exp(-\beta Q_t(L))}{\exp(-\beta Q_t(L)) + \exp(-\beta Q_t(R))} \\
&= \epsilon + \frac{1-2\epsilon}{1+\exp(-\beta(Q_t(R)-Q_t(L)))} \\
P_t(R) &= 1 - P_t(L)
\end{aligned}
$$

- Given a set of choices made by the network $C_t, \forall t$, we can fit the parameters using maximum likelihood

$$(\alpha, \beta, \epsilon) = \arg\max \sum_t \log P_t(C_t)$$