SHUFFLING RECURRENT NEURAL NETWORKS

In this project i'm going to implement the SHUFFLING RECURRENT NEURAL NETWORKS (SRNN) a RNN architecture in wich the previous hidden state is multiplied by a fixed permutation matrix. With this mecchanism it is possible to obtain an efficient and simple RNN that does not suffer from vanishing and exploding gradients.
The hidden state of this model is defined as:

$$h_t = \sigma\left(W_p h_{t-1} + b(x_t)\right)\}$$

where:

σ is the activation function

h_{t-1} is the hidden state of the precious time step

x_t is the input state of the current time step

$W_p$is the fixed permutation matrix

and $b(x_t)$ is definied as:

$$b(x_t) = f_r(x_t) \odot sigmoid(W_s x_t + b_s)$$

where:

f_r is a MLP and W_s and b_s are the weight and the bias of the single affine layer.

I implemented this model with two subclass of the nn.Module.
In the first class i developped a single hidden state.
In the second class the forward function recall multiple time the forward function of the first class.

I tested the function with one of the datasets of the paper, The Adding Problem Dataset.
The input for this task consists of two sequences of length T.
The first sequence contains real numbers, the second sequence is a set of 0 except for two random entries that are set to 1.
The output is the sum of the two numbers in the firse sequence that correspond to the location of the 1s in the second.
I use two dataset one with T=200 and the other with T=750.
I used the MSE as loss function and I obteind a value of MSE = 0.016 for the network that works with T = 200, starting from a value of MSE = 2.5217, and a value of MSE = 0.12 for the second network starting from a value of MSE = 39.7789.