

5-Load-in-NEST

January 4, 2020

0.1 Load-in-NEST

In this notebook we load some of the pretrained networks in NEST.

Experiment List: 1. DQN-Training (How to train a conventional DQN and a spiking DQN using Surrogate Gradients (DSQN).) 2. Load-DQN (How to load a previously saved D(S)QN and how to save a replay dataset.) 3. Train-Classifer (How to train a spiking or non-spiking classifier on the saved replay data set.) 4. SNN-Conversion (How to convert a DQN and a Classifier to a SNN.) 5. Load in NEST (How to load a converted or directly trained spiking network in NEST.) 6. Conversion in pyNN with NEST or SpyNNaker (How to load spiking network in pyNN using NEST or SpyNNaker as backend.)

```
[1]: import torch
import os
import sys
import random
import matplotlib.pyplot as plt
# hack to perform relative imports
sys.path.append('../..')
from Code import Nestwork, load_agent, FullyConnected, SQN

# set seeds
torch.manual_seed(1)
random.seed(1)
gym_seed = 1

# device: automatically runs on GPU, if a GPU is detected, else uses CPU
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# switch to the Result Directory
os.chdir('../..../Results/')
```

Detected PyNN version 0.9.5 and Neo version 0.6.1

To begin with, we load a network obtained by conversion from a classifier which was obtained in the previous tutorial (SNN-Conversion).

```
[2]: # specify the file, where the network is saved
file = '../CartPole-A/Classifier-Converted/model.pt'
```

```

# set hyperparameters of NEST:
# encoding/decoding methods are limited to constant input currents and potential
→ outputs.
# set correct architecture
architecture = [4,16,16,2]
# set simulation time in ms, changing the resolution is not supported in our code
simulation_time = 100
# the neuron type can be set to 'iaf_psc_delta' or 'pp_psc_delta' and determines
→ the neuron type in the hidden layers
# this also fixes the reset method to reset-to-zero and to reset-by-subtraction
→ respectively
# for converted DQNs the type pp_psc_delta should be chosen, for classifier it
→ makes not much difference.
neuron_type = 'iaf_psc_delta'

# set up network in NEST
network = Network(architecture,file,simulation_time,neuron_type=neuron_type)

```

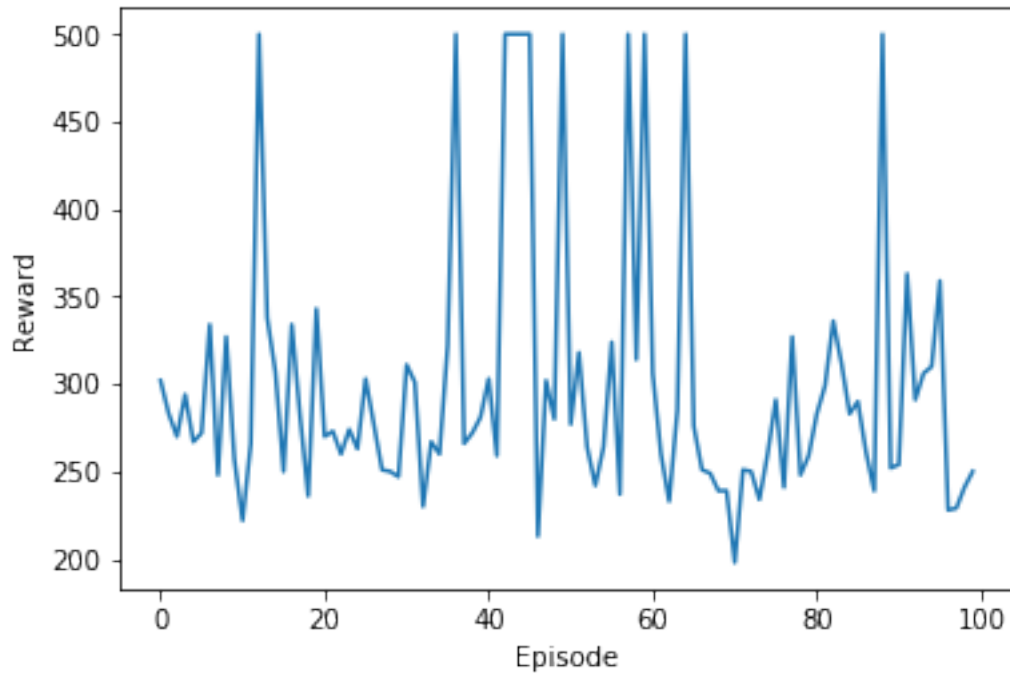
The NEST agent can be loaded equivalently to all other agents using the function `load_agent` (see tutorial 2: Load-DQN) as it implements the method 'forward'. We compare it to the original Classifier by setting `compare_against` to the original classifier agent.

```

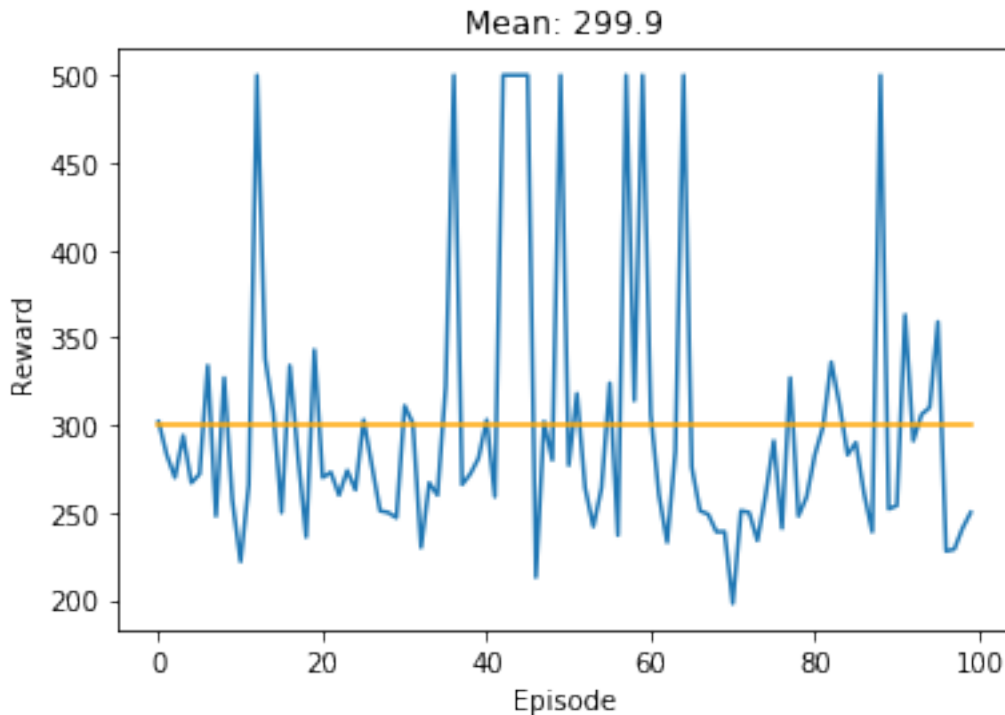
[3]: # Load the original classifier
# specify correct architecture
architecture = [4, 16, 16, 2]
# load weights
classifier = FullyConnected(architecture).to(device)
classifier.load_state_dict(torch.load('./CartPole-A/Classifier/trained/model.
→ pt'))

# run the NEST agent and compare against the original classifier
env = 'CartPole-v0'
load_agent(env,network,device,epsilon=0,gym_seed=gym_seed,save_replay=False,
            max_steps=500,num_episodes=100, render=True,
→ compare_against=classifier)

```



Similarity (Conversion Accuracy) after 29990 iterations: 88.62620873624542%
Complete
Mean: 299.9
Std: 77.74531068388198
Similarity (Conversion Accuracy) after 29990 iterations: 88.62620873624542%



Likewise, we can convert the DQN directly. Next, we also run a directly trained DSQN in NEST.

```
[2]: # specify the file, where the network is saved
file = './CartPole-E(SNN)/copy/model.pt'

# set hyperparameters of NEST:
# encoding/decoding methods are limited to constant input currents and potential
# → outputs.
# set correct architecture
architecture = [4,17,17,2]
# set simulation time in ms, changing the resolution is not supported in our code
simulation_time = 20
# this time both neuron types work similarly bad, we use pp_psc_delta
neuron_type = 'pp_psc_delta'

# set up network in NEST, as we load a SpyTorch trained network we have to add
# → the bias to each observation
# additionally we have to specify that the network has no biases, as this is a
# → special case when initializing Nestwork
nestwork = Nestwork(architecture,file,simulation_time,neuron_type=neuron_type,
# → add_bias_as_observation=True)

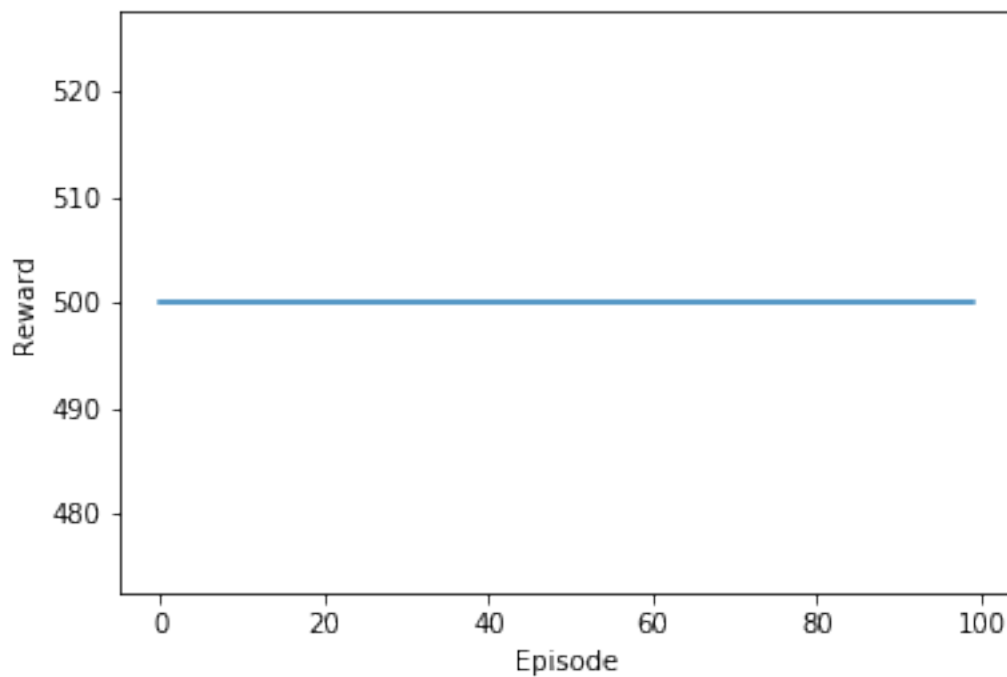
# Load the original DSQN to compare against
# specify correct architecture
```

```

architecture = [4, 17, 17, 2]
# load weights
dsqn = SQN(architecture, device=device, alpha=0, beta=1, simulation_time=20,
           add_bias_as_observation=True,
           encoding='constant', decoding='potential',
           reset='subtraction', threshold=1)
dsqn.load_state_dict(torch.load('./CartPole-E(SNN)/copy/model.pt'))

# run the NEST agent and compare against the original classifier
env = 'CartPole-v0'
load_agent(env, nestwork, device, epsilon=0, gym_seed=gym_seed, save_replay=False,
           max_steps=500, num_episodes=100, render=True, compare_against=dsqn)

```



Similarity (Conversion Accuracy) after 50000 iterations: 64.49%
 Complete
 Mean: 500.0
 Std: 0.0
 Similarity (Conversion Accuracy) after 50000 iterations: 64.49%

