Project 1: Navigation report

Agent implementation

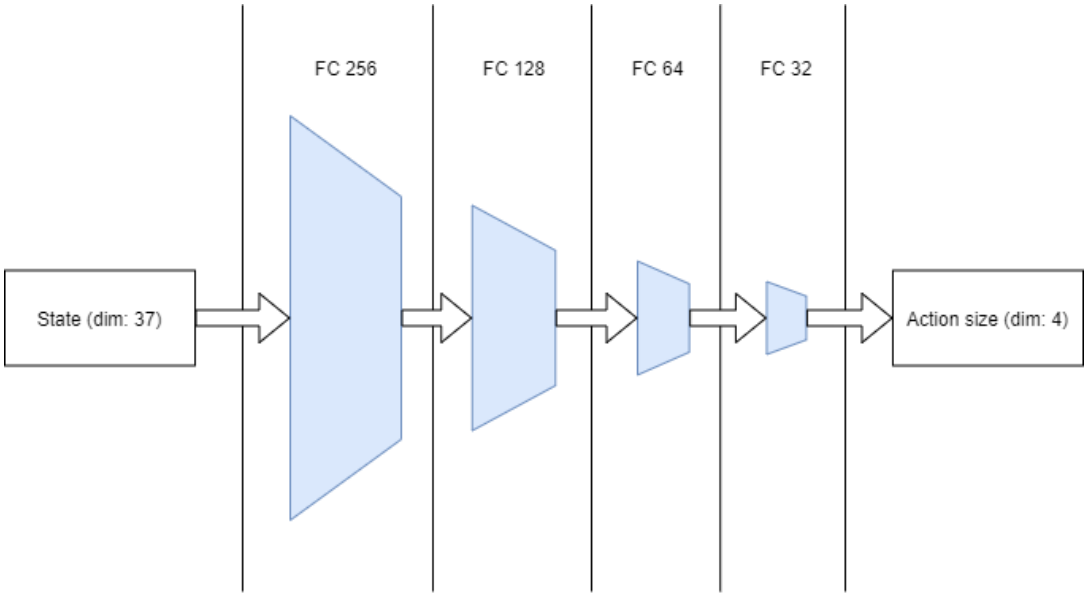
The solution implements the Deep Q-Networks, based on Google DeepMind's Research. This approach is a value-iteration method combining SARSAMAX (Q-learning) and encodes the Q-table as the underlying deep neural network (fully connected layers). The banana environment provides the state and action dimension for the problem (summarized in 1. Table).

1. Table: summary of environment-specific parameters (Banana)

Parameter name	Value
State space	37
Action space	4
Points for successful solution of the environment	13.0

The neural network architecture is composed of the following four fully connected neural network layers (depicted in 1. Figure):

- 256 neurons from the action space (input dimensionality: 37). (**feed_in**)
- 128 neurons in the intermediate layer (**feed_intermediate**)
- 64 neurons in the intermediate layer (**feed_intermediate2**)
- 32 neurons in the intermediate layer (**feed_intermediate3**)
- 32 neurons to the action space (output dimensionality: 4). (**out_act**)



1. Figure: Deployed DQN network architecture

The network was developed with the PyTorch framework. The implemented DeepQN network has to versions:

- The implementation (***DqnRelu*** class) has ReLU nonlienarities between the layers.
- The implementation (**DqnElu** class) has ELU nonlinearities between the layers.

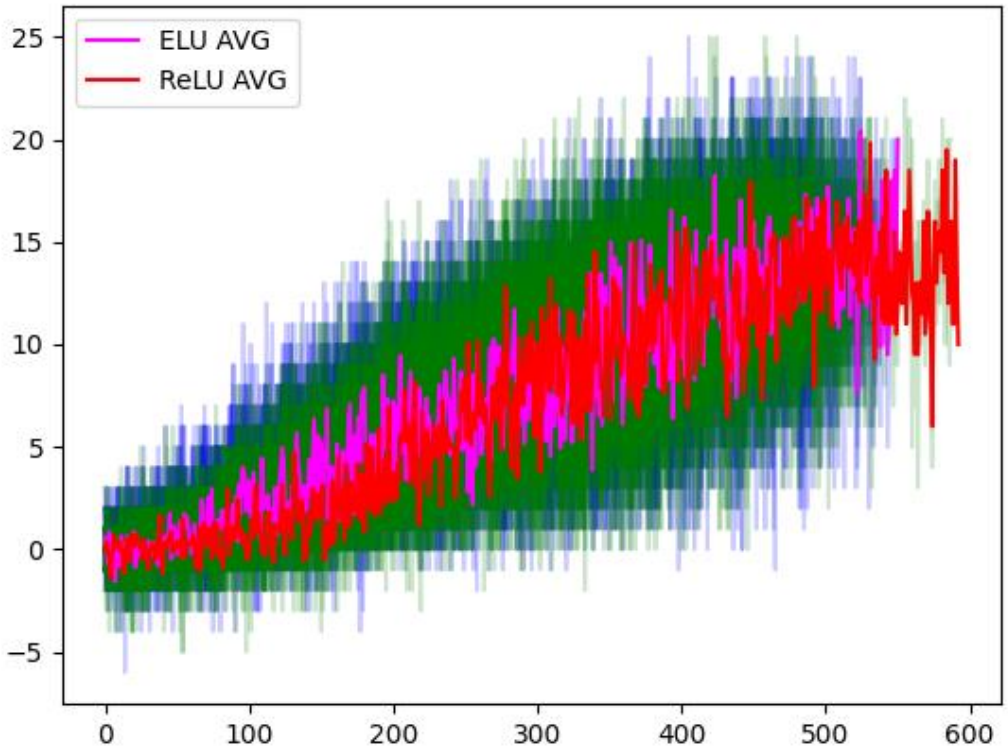
Besides these modifications, the DQN agent training-validation process basically follows the process presented in the *Udacity Deep Reinforcement Learning Nanodegree* learning material. The agent uses fixed update targets and an experience buffer during training.

The project structure is the following (with the corresponding file artifacts):

- *chkp_elu.pth*: successful training of the ELU activation function DQN (over 13 points).
- *chkp_relu.pth*: successful training of the ReLU activation function DQN (over 13 points).
- *models.py*: DQN PyTorch-based models for the two variation of the implemented network (ReLU and ELU activation functions).
- *dqn_agent.py:* the DQN agent and experience buffer implementation.
    - Experience buffer follows the ***list*** semantics
    - The agent steps based on $(S_t, A, R, S_{t+1}, done)$ tuple, learns based on an experience batch, and acts based on state
- *example.py*: standalone version of the agent training (independent of the jupyter notebook framework). Saves the training scores into text files for further analysis.
- *data_read.py*: reads and plots the previous training results.
- *Navigation.ipynb*: provided notebook, extended with the current training implementation (with ELU).
- *Report.pdf*: this report.

Results

In general, the training process lasted 7-9 minutes in average on a PC equipped with NVIDIA GTX 2070 GPU card, with both variations. Both agents always performed a satisfactory training under 700 iterations. In a random run, it seemed that the ELU variation increased performance slightly faster than the ReLU version, but the convergence stuck at some point. Running for 20 consecutive runs on both agents it seems that the iteration count has no significant difference for both versions (497 for ELU, 495 for ReLU in average). The training error is depicted in 2. Figure. The area shaded with blue depicts ELU runs, the green depicts ReLU runs.



2. Figure: Results of 20 consecutive training runs for both ReLU and ELU DQN versions (green depicts ReLU runs, blue depicts ELU runs)

The qualitative numbers for the iteration count for the 20 consecutive runs are listed in the following table (2. Table):

2. Table: quantiative results of the 20 consecutive runs

DQN version	Signle random run	Median iteration	Average iteration	Maximal iteration	Minimal iteration	Standard Deviation
ReLU	584	492	495	593	437	23.997
ELU	470	494	497	551	449	35.797

Hyperparameters

The following table depicts the hyperparameters of choice during training:

Parameter name	Value
Random seed	9090
Discount rate γ	0.99
Batch size	64
Experience buffer size	1e5
Update rate (fixed target update)	4
Tau (fixed target update blending)	1e-3
Learning rate α	5e-4