*Group 39: Goh Chian Kai, Lam Kai Yi, Ferdinand Tonby-Strandborg*

# Neural Network Report

**Overview**

The neural network is configured to take in a scalar input from the OpenAI Gym in the previous milestone and output a vector of Q-values to the learning agent. A DQN agent is used as a placeholder for purpose of troubleshooting, and preparation for next milestone deliverable.

**Code Explanation**

Observation state from the OpenAI gym consists of the traffic light state (12 traffic lights; 12x1 array), lane occupancy time (4 direction, 3 lane, 5 sensors per lane; 4x3x5 array), queue length (12 lanes; 12x1 array), upstream status (single scalar) and downstream status (single scalar), totalling to 86 input features. While there are 4096 valid actions from a combination of 12 traffic lights, not all actions are logical. The actions are filtered and only 30 actions are found to be logical for implementation. As such, the action space is expected to be 30. The reasonable actions are documented and updated in the OpenAI gym. Components of the observation state can be represented in a scalar form, where it is flattened to a 1d array to be fed into the neural network. As the state space has a relatively low number of features and is fully observable, a feed-forward network with hidden fully connected layers will be used.

The structure of the neural network consists of 4 fully connected layers in the form of an expanding-contracting architecture. The output of each fully connected layers will be batch normalized (commented in the current code due to a setting conflict with the DQN agent). The size of each fully connected layer will be in powers of 2 (64, 128, 256, 512) as it is optimal for GPU training. The expanding-contractive architecture enables capacity for exploration of different combination of input features before condensing only the essential information as output. As the observation state has 86 features, the first hidden layer will have a size of 128 to allow for sufficient room for the network to learn useful features. A larger size can potentially result in overfitting or slow training. The second layer will have a size of 256 to provide more space such that the network is able to explore and learn more complex combinations between the different input features. The third and fourth layer will have a size of 128 and 64 respectively. The size of the fourth layer considers the size of the action space, where mapping of the learning outcome to the action space is smoother. After the features exploration of the second layer, the essential learning outcome is filtered and funnelled through the third and fourth layer such that only the key relevant information is sent as an output for the Q-values. ReLU is chosen as the activation function for all four layers, due to its computational efficiency, reduce possibility of vanishing gradient problem and allow for better gradient flow through the hidden layers.

To troubleshoot and test the features of the neural network, a DQN agent has been included although it is not required the current milestone deliverable. As the focus is on the neural network, minimal fine tuning is performed on the DQN agent. The DQN agent uses the Adam optimizer to update the network's weights (it implements the backwards passes), as it is most used in deep Q-learning due to faster convergence through adaptive learning rates, smoother optimization by combining momentum and adaptive learning rates and robustness in its hyperparameters. We are using a replay buffer to record previous states, actions and next states, which we are randomly sampling from when training. By ensuring that a valid DQN agent is used, this allows the team to focus on debugging the neural network.

**Existing Code/Libraries**

For this project, Pytorch is selected as the library to be used for the neural network. We chose it over Tensorflow for its ease of use and more intuitive syntax. Based on our current survey, it seems that Pytorch has a stronger community support, which will be helpful when there is a need for troubleshooting. Considering the short timeline of each deliverable in this project, it is preferred for the efforts to be directed on understanding of the architecture of the neural network, rather than navigating and debugging codes due to unfamiliar syntax. We are also using NumPy for its simplified and optimized matrix handling.

**Reflections**

When we started designing the neural network, the number of available options was overwhelming such as the different structures (Feed-forward, ResNet, CNN, RNN..) and the number of configurable options within each structure such as setting up a LSTM, sizes of each layer and the type of activation function to use. To get a better grasp of what will be the most suitable for our application, we looked at what will be the input and output required for the neural network. From there on, this allowed us to filter out and select what seems to be the suitable application. For our case, as our observative state does not need to be represented in a 2d array, we did not go with a CNN and chose a network with fully connected layers. This assignment allowed us to better appreciate the different types of neural network structures and layers out there and their respective applications.

During the proposal and the setting up of the OpenAI gym, we initially decided to have multiple components for the rewards function, attempting to cover different parts of the traffic light agent's action. However, after a discussion with Prof. Sartoretti and reviewing other reinforcement learning projects on related to traffic problems, we realised that the reward function can be simplified to just delta queue length and lane occupancy waiting time. This also made troubleshooting simpler and emphasized the value of keeping the code simple and more readable.

Initially, we planned for the traffic light agent to have full autonomy over control of each of the 12 traffic lights, or a combination of 4096 actions. However, during the training phase, we are unable to get convergence and one of the causes identified is the large action space and how long it will take before the agent can fully explore all the actions. Considering this, we decided to reduce the action space by only allowing for logical actions by the traffic agent.

**AI Tool Declaration:**

I (Ferdinand T-S) used Claude AI to get familiar with the PIL interface as used in generating GIFs. I am responsible for the content and quality of the submitted work.

I (Goh Chian Kai) used Claude AI to obtain information on the different type of neural network structure using pytorch and how to create a dqn agent. I am responsible for the content and quality of the submitted work.