**Reinforcement Learning:**

* Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (Silver, 2017)
* Implementing the Deep Q-Network (Roderick, 2017)
* Asynchronous Methods for Deep Reinforcement Learning (Mnih, 2016)
* Implementation of Q-Learning and Deep Q Network for Controlling a Self Balancing Robot (Rahman, 2018)
* ADAM: A Method for Stochastic Optimization (Kingma, 2015)
* Human-Level control through deep reinforcement learning (Mnih, 2015)
* Evolving Neural Networks through Augmenting Topologies (Stanley, 2001)
* Evolving Deep Neural Networks (Miikkkulainen, 2017)

**Multi Agent Reinforcement Learning:**

* Multi-agent Reinforement Learning for Traffic Light Control (Wiering, 2000): Discretizes the places vehicles can take in the network. 8 traffic lights are controlled independently. Each light has the information car location and destination. Each node has 6 possible actions. The cars are also agents with access to the value functions, it differs meaningfully from pure RL. Makes bad use of the simulator, using refused cars into the simulation as metric. Not usable.
  + **Architecture:** 8 traffic lights per intersection (straight+right and left). 20 cars max capacity per lane. Controlled per intersection.
  + **Model:** Lookup tables. Calculate advantage of setting a light green. Local/Global information.
  + **State Space:** Location, place in queue and destination of cars in each queue.
  + **Action Space:** 6 potential phases (fixed time).
  + **Reward Functions:** Negative waiting times?
* Multi-agent Reinforement Learning for Traffic Signal Control (Prabuchandran, 2014): Uses RR phases in a MARL Q Learning setup. Discrete state and action spaces. One agent per junction as solution for dimensionality issues. Queue size is clustered into 3 different levels for the state. State vector is queues, plus the queue in the next phase that has to be set green. Actions involve setting the green time to one of 3 levels (H,M,L) for the current phase. Stationary policies. Reward is a function to the junction and neighbors. Implements counts of how many times an action has been taken in a state. Results on 2 roads of 9 and 20 junctions. Beats SAT and FST.
  + **Architecture:** 9 and 20 junctions. One agent per intersection.
  + **Model:** Markov Decision Process via Q-Learning.
  + **State Space:** Queue sizes discretized to 3 different levels. State vector is queue per lane plus an entry for the phase that is set to turn green.
  + **Action Space:** Decision of green times with RR arrangement. Also discretized in 3 different levels (10,20,30 seconds).
  + **Reward Functions:** Cost function is the queues in neighbouring intersections at time t+1, aiming to capture the effect of action a on the neighbouring junctions. It is not specified whether it accounts for all lanes or only lanes that are actually connected.
  + **Comment:** Results prove that this approach is good. The delay is reduced across the network for both 9 and 20 junctions. Interesting reward function on a network setup.
* Q-Learning Traffic Signal Optimization within Multiple Intersections Traffic Network (Kwong Chin, 2012): Discretizes Q-States to 4 levels of load per link, 2 intersections, for a total of 256 states. No valuable information for replication whatsoever.
  + **Architecture:** 2 adjacent intersections.
  + **Model:** Traditional Q learning. No coordination between intersections.
  + **State Space:** Queue length discretized to 4 levels of load.
  + **Action Space:** 4 potential phases per intersection.
  + **Reward Functions:** Not specified. A function of queue length with penalisation for every second of wasted green time.
  + **Comment:** Several issues make this paper non-replicable. The simulations and results section aren’t usable at all.
* Intelligent Traffic Light Control Using Distributed Multi-agent Q-Learning (Liu, 2017): Uses Q learning with agents exchanging information. Explores using a Boltzmann distribution. Actions are green phases (as which lanes are allowed forward instead of green times). Validation in SUMO. This is pretty much the paper I replicated during summer, in a simplification to Round Robin.
  + **Architecture:** 33 intersection network from California.
  + **Model:** Q-Learning with information exchange between intersections
  + **State Space:** Queue lengths for vehicles and pedestrians.
  + **Action Space:** 2, 4, 6 or 8 actions. Combination of green ways.
  + **Reward Functions:** Acknowledgementthat several can be used. Queues.
  + **Comment:** Results are good, mostly. They also include pedestrians. This is pretty much the paper I replicated during summer, in a simplification to Round Robin. It beats the paper from Pranabuchandran.
* A collaborative Reinforcement Learning Approach for Urban Traffic Control Optimization (Salkham, 2008): Lots of literature review, but mostly useless. Uses Adaptive Round Robin in a section of the centre of Dublin. Comparing against SCATS and Fixed Time. CRL shows the best results, with Q-Learning showing second best, but not by a lot of distance from the third.
  + **Architecture:** 64 junctions in the centre of Dublin.
  + **Model:** Decentralized Q-Learning and Collaborative Q-Learning (MARL).
  + **State Space:** Busy/Not busy for every lane. This is done comparing the number of vehicles within range for the next phase against a threshold. No more details given.
  + **Action Space:** Phase durations, including zero length (0,20,40 seconds).
  + **Reward Functions:** Net Present Value, allows exchange of rewards between agents (?). Function of number of vehicles clearing the junction and the number of vehicles still waiting (substracting them). In the CRL version, local rewards are sent to the neighbors who discount them and incorporate them to their own rewards.
  + **Comment:** The way of sharing rewards is quite sophisticated, but there are some doubts that it encapsules the needed information in a minimal way.
* Reinforcement learning-based multi-agent system for network traffic signal control (Arel, 2009)
* Distributed Learning and Multi-Objectivity in Traffic Light Control (Brys, 2014)
* Multi-Agent Deep Reinforcement Learning (Egorov, 2017)
* Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents (Zhang, 2018)

**Urban Traffic Control:**

* Intelligent Traffic Light Control of Isolated Intersections Using Machine Learning Methods (Araghi, 2013): Using a feed-forward NN, the queue lengths are taken as inputs and green times are estimated. Stochastic Annealing is used for optimizing the weights. Average delay is used as an indirect cost function. Training is done in an indirect manner. No details on cost function or action selection. Shows better performance than basic Q-Learning.
  + **Architecture:** Single intersection.
  + **Model:** Neural Network. Average delay is used as cost function for training.
  + **State Space:** Queue lengths.
  + **Action Space:** Choose green time for each phase. No limitation on cycle time. Comparing against Q-Learning with built-in restrictions.
  + **Reward Functions:** Inversely proportional to average length of queues, normalised to fall in [0,1].
  + **Comment:** Techniques used here have aged very badly. Unclear and not directly replicable. Useful to justify jumping to NN from Q-Learning.
* An Efficient Deep Reinforcement Learning Model for Urban Traffic Control (Lin, 2018)
* A reinforcement learning based traffic signal control algorithm in a connected vehicle environment (Yang, 2017): Basic level, introduces Q-Learning and DQN. Single intersection no turning allowed in SUMO. They fail to implement DQN. Discretization for Q-Learning.
  + **Architecture:** Single intersection no turning allowed. Probably ARR.
  + **Model:** Table based and early DQN (Neural Fitted Q-Learning, NFQ).
  + **State Space:** Sum of squared delays of NS and EW, current phase and current length.
  + **Action Space:** Phase lengths.
  + **Reward Functions:** Negative sum of squared delays in NS direction and EW direction.
  + **Comment:** Very bad experimental quality. Fail to implement DQN. Bugs in Q-Learning implementation. The agents are not trained for nearly enough time. Useless results. Some good ideas in terms of sate and reward, very bad implementation.
* Reinforcement Learning with Function Approximation for Traffic Signal Control (Prashanth, 2010)
* Deep Reinforcement Learning for Traffic Light Control in Vehicular Networks (Liang, 2018): Treats the problem as a computer vision one. The intersection is modelled as a series of empty or full cells, with a vehicles moving in a discrete manner. It includes some recent developments in terms of Deep Reinforcement Learning (Post AlphaGo).
  + **Architecture:** Single intersection. All turning allowed.
  + **Model:** CNN with Target Network, Dueling Architecture and Prioritized Experience Replay.
  + **State Space:** Grid, containing for each cell, a binary indication of whether there is a vehicle plus the speed in m/s.
  + **Action Space:** Select each phase’s duration in the next cycle. Small changes to this length. 5 seconds are added or substracted to one of the phases per cycle.
  + **Reward Functions:**  Negative cumulative waiting time between phases.
  + **Comment:** Very good and closest to state of the art. Still, given that uses a CNN on a vision imitation system, there is still space for our approach. Very replicable as it even gives the hyperparameters of the model.
* Using a Deep Reinforcement Learning Agent for Traffic Signal Control (Genders, 2016)