**Reinforcement Learning/Theoretical:**

* Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching (Lin, 1992)
* Markov games as a framework for multi-agent reinforcement learning (Littman, 1994)
* Long Short-Term Memory (Hochreiter, 1997)
* Evolving Neural Networks through Augmenting Topologies (Stanley, 2002)
* A comprehensive survey of Multiagent Reinforcement Learning (Busoniu, 2008)
* Policy Gradient Methods for Reinforcement Learning with Function Approximation (Sutton, 2010)
* Double Q Learning (Hasselt, 2010)
* Playing ATARI with Deep Reinforcement Learning (Mnih, 2013)
* Deterministic Policy Gradient Algorithms (Silver, 2013)
* Deep Reinforcement Learning with Double Q Learning (Hasselt, 2015)
* ADAM: A Method for Stochastic Optimization (Kingma, 2015)
* Human-Level control through deep reinforcement learning (Mnih, 2015)
* Continuous Control with Deep Reinforcement Learning (Lillicrap, 2015)
* Asynchronous Methods for Deep Reinforcement Learning (Mnih, 2016)
* Policy Distillation, (Rusu, 2016)
* Dueling Network Architectures for Deep Reinforcement Learning (Wang, 2016)
* Prioritized Experience Replay (Schaul, 2016)
* Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (Silver, 2017)
* Implementing the Deep Q-Network (Roderick, 2017)
* Evolving Deep Neural Networks (Miikkkulainen, 2017)
* Sample Efficient Actor-Critic with Experience Replay (Wang, 2017)
* An overview of gradient descent optimization algorithms (2017)
* Rainbow: Combining Improvements in Deep Reinforcement Learning (Hessel, 2017)
* Counterfactual Multi-Agent Policy Gradients (Foerster, 2018)
* Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents (Zhang, 2018)
* Implementation of Q-Learning and Deep Q Network for Controlling a Self Balancing Robot (Rahman, 2018)
* Self-Imitation Learning (Oh, 2018)
* Re-evaluating Evaluation (Balduzzi, 2018)
* Hierarchical Representations for Efficient Architecture Search (Liu, 2018)
* IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures (Espeholt, 2018)
* Bayesian Optimization in AlphaGo (Chen, 2018)
* Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement (Barreto, 2019)
* An investigation of model-free planning (Guez, 2019)

**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?
* Distributed Learning Agents in Urban Traffic Control (Camponogara, Kraus, 2003): Good definition of traffic control as a distributed stochastic game. It can also be easily cited to justify the election of temporal difference-based methods rather than Monte Carlo or Dynamic Programming. However, it does not investigate how to get the Nash joint policies for the system, but instead looks at the performance of the agents as they keep searching for them. This way they generate a method they call distributed Q-Learning.
  + **Architecture:** 2 traffic lights next to each other. Only 1 lane open at a time per intersection.
  + **Model:** Multi-Agent Q Learning
  + **State Space:** Not specified
  + **Action Space:** 4 phases per intersection. Choose phase.
  + **Reward Functions:** Not specified.
* Natural Actor-Critic for Road Traffic Optimisation (Richter, 2007): Argues for the use of Policy Gradient methods in traffic control. Well written and including the mathematical development of the model.
  + **Architecture:** 4 different experiments.
  + **Model:** Natural Actor-Critic (they develop it).
  + **State Space: Controller input:**  Cycle duration, current phase, phase duration. Detector active (loop simulation), detector history. Neighbour information, indicating where traffic is expected from.
  + **Action Space:** 4 phases, directly chosen per timestep. No specific order, but all should be activated within 16 time steps.
  + **Reward Functions: -**
  + **Comment:** Very good. Lots of expert knowledge in the experiments’ definition section. Details about hyperparameters and experimental setup.
* 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): Relatively early but well written RL. Uses an early version of DQN. Very well written and replicable, although with some modelling issues such as discretely modelling in matlab. Can be used as baseline.
  + **Architecture:** 5 intersection centrally connected traffic network.
  + **Model:** 2 algorithms. Longest Queue First (LQF) in the outbound intersections and Q-Learning using NN for approximations in the central one. Agents can communicate basic information.
  + **State Space:** 8 dimensional vector (1 per lane) containing the relative traffic flow, defined as the total delay of vehicles in a lane divided by the average delay at all lanes in the intersection. The central intersection has access to all the state vectors of its neighbors. Actions last 20 units but can be repeated in a row.
  + **Action Space:** 8 phases are available per intersection.
  + **Reward Functions:** Ranges from -1 to 1. R = Dlast – Dcurrent/max[Dlast, Dcurrent]
  + **Comment:** Results are good, beating the other method. Unsure about reward functions and some other details.
* An agent-based Learning Towards Decentralized and Coordinated Traffic Signal Control (Samah El-Tantawy, Baher Abdulhai, 2010): Acknowledges non-stationary nature of traffic and intends to tackle it. Very interesting in their descriptions and sources on MARL. Good ideas in terms of rewards and state.
  + **Architecture:** Single intersection.
  + **Model:** Q-Learning. Not very well explained.
  + **State Space:** 3 proposed. Queue length, cumulative delay, arrival of vehicles to current green direction and queue lengths at red directions.
  + **Action Space:** Next phase. If the current phase is chosen, add 1 second. Total of 4 phases.
  + **Reward Functions:** Delta cumulative delay (probably good idea)
  + **Comments:** E-greedy actions choice, annealed down. Odd, the application and experimentation only includes a single agent.
* 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.
* 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.
* 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.
* Distributed Learning and Multi-Objectivity in Traffic Light Control : Bad paper, avoid. (Brys, 2014)
  + **Architecture:** 2x2 grid of intersections
  + **Model:** Isoactuated and SARSA. Not current RL.
  + **State Space:** Not well explained.
  + **Action Space:** 2 phase. Simplistic.
  + **Reward Functions:** Not explained.
* Multi-Agent Deep Reinforcement Learning (Egorov, 2017)
* Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents (Zhang, 2018)

**Urban Traffic Control:**

* Reinforcement Learning with Function Approximation for Traffic Signal Control (Prashanth, 2010): No inter agent communication. Rigorous but aged badly. Mathematically complex since they build their own approximator, which deviates from the current state of the art. Not really usable. Prepared with actuated sensors in mind.
  + **Architecture:** 2x2, 3x3 grid networks. 8 junction corridor.
  + **Model:** Q-Learning with function approximation.
  + **State Space:** Discrete load (low, mid, high) per lane.
  + **Reward Functions:** Combination of queue length and delay.
  + **Comment:** Compared against fixed time, longest queue, SOTL and Qlearning.
* 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.
* Using a Deep Reinforcement Learning Agent for Traffic Signal Control (Genders, 2016): Interesting state space definition. Uses CNN in a single intersection. Good reward definition, consider normalizing it. Provides parameters, also for NN. Linear exploration decrease in exploration. Action lengths are 2 seconds.
  + **Architecture:** Single intersection
  + **Model:** Convolutional NN.
  + **State Space:** DSTE (grid of positions, speeds, and current phase)
  + **Action Space:** 4 non incompatible phases, seems no minimum cycle time.
  + **Reward Functions:** Change in cumulative delay between actions.
  + **Comment:** GOOD.
* Coordinated Deep Reinforcement Learners for Traffic Light Control (van der Pol, 2016)
  + **Architecture:**
  + **Model:**
  + **State Space:**
  + **Action Space:**
  + **Reward Functions:**
  + **Comment:**
* Traffic Signal Timing via Deep Reinforcement Learning (Li, 2016)
  + **Architecture:**
  + **Model:**
  + **State Space:**
  + **Action Space:**
  + **Reward Functions:**
  + **Comment:**
* Multi-Agent Reinforcement Learning (Egorov, 2016)
  + **Architecture:**
  + **Model:**
  + **State Space:**
  + **Action Space:**
  + **Reward Functions:**
  + **Comment:**
* Adaptive Traffic Signal Control: Deep Reinforcement Learning Algorithm with Experience Replay and Target Network (Gao, 2017)
  + **Architecture:**
  + **Model:**
  + **State Space:**
  + **Action Space:**
  + **Reward Functions:**
  + **Comment:**
* 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.
* An Efficient Deep Reinforcement Learning Model for Urban Traffic Control (Lin, 2018)
* 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.
* Deep Reinforcement Learning Adaptive Traffic Control (Genders, 2018)
  + **Architecture:**
  + **Model:**
  + **State Space:**
  + **Action Space:**
  + **Reward Functions:**
  + **Comment:**