Impact of Pre-training on Deep Reinforcement Learning Ramp Metering Systems

Callum Evans
Delft University of Technology
2628 CD Delft, The Netherlands
C.Evans@tudelft.nl

Marco Rinaldi
Delft University of Technology
2628 CD Delft, The Netherlands
M.Rinaldi@tudelft.nl

Henk Taale
Delft University of Technology & Rijkswaterstaat
2628 CD Delft, The Netherlands
H.Taale@tudelft.nl or henk.taale@rws.nl

Serge Hoogendoorn
Delft University of Technology
2628 CD Delft, The Netherlands
S.P.Hoogendoorn@tudelft.nl

Word count: 5299 words + 3 tables (250 words per table) = 6049 words

ABSTRACT

Pre-training is a process used to enhance the learning of deep reinforcement learning (RL) algorithms through initial guidance from an expert demonstrator. This involves training a neural network to replicate the outputs of the selected expert before allowing the RL agent to specialise and develop its own policy. This paper outlines a study that aims to analyse the impact of pre-training on deep RL algorithms used in ramp metering. Specifically, behaviour cloning is performed for increasing lengths of time (0-10,000 epochs), with ALINEA as the chosen expert algorithm guiding a proposed Proximal Policy Optimisation (PPO)-based system. The results confirm that, with the same length of training, some initial guidance through pre-training can significantly improve the system's effectiveness in reducing congestion compared to no pre-training. Otherwise, excessive pre-training may lead to overfitting and reduced generalisability. Design issues resulting in weak model convergence, however, limit the algorithm's overall performance in the chosen scenario.
Keywords: Network Management, Road Traffic Control, Ramp Metering, Reinforcement Learning

INTRODUCTION

Ramp metering is a method of traffic management that aims to reduce motorway congestion and increase throughput by using signals to regulate traffic flow at on-ramps based on current conditions. It has been used in various forms since the 1960s, initially in the US (*1*), but has since become much more common worldwide. Ramp metering can lead to more efficient use of road capacity when demand is high (*2*) as limiting inflow can enable the motorway to operate very close to critical density without exceeding it. This is also a critical method of preventing capacity drop (*3*), where the maximum outflow of a congested bottleneck is significantly lower than the maximum observable bottleneck flow. As a result, numerous studies have already demonstrated the benefits of ramp metering (*4*), such as a 2001 study in Minnesota (*5*) that found a 22% and 21% reduction in travel time and accidents respectively and an 8% and 16% increase in speed and throughput capacity when compared to a no-control scenario. This is supported by other studies in the Netherlands, although the improvements were somewhat smaller (*6*). One of the largest disadvantages of ramp metering is spill back. Depending on the storage capacity of the ramp, limiting flow too much can cause queuing vehicles to spill onto the local network (*1*).

The most common local ramp metering algorithm is Asservissement Linéaire d'Entrée Autoroutière (ALINEA), developed by Papageorgiou et al. (*7*), which is a simple but effective algorithm that is still very commonly used as a benchmark for the development of ramp metering systems. Whilst this and similar classical ramp metering algorithms have shown positive results, many learning-based algorithms have been proposed that can improve upon or match the performance of classical approaches, such as in (*8, 9*). Learning-based controllers, such as those that use reinforcement learning (RL) use knowledge learnt from historical traffic data or previous experiences to limit flow according to current conditions. This allows them to learn complex and effective policies that would be infeasible to create manually through explicit rulesets. RL algorithms also have the advantage over predictive approaches in that they do not require a model of their environment to operate, which can be computationally expensive in the context of traffic management. As a result, RL has become much more common in research surrounding traffic control systems, especially ramp metering. For example, Q-learning has been extensively researched, such as in (*10–12*), due to its robustness and simplicity.

Some of the largest disadvantages of RL surround the large amount of training required, which can be due to the sample efficiency of the model (*13*). This is a potential problem within model-free RL and refers to how efficiently a model can learn a useful policy from data or experiences. Particularly in environments that are computationally expensive to model, as is often the case with complex, nonlinear traffic flow dynamics, low sample efficiency of RL models can cause issues. One way to solve this is through pre-training, such as the technique of imitation learning (*14*). This aims to improve learning efficiency by training an RL model to imitate the actions of an expert demonstrator, before training further and allowing the model to specialise towards its own policy. Imitation learning has become more popular in recent years and has been shown to be beneficial in a variety of RL applications, such as robotics or computer vision (*15*). Approaches include behaviour cloning, inverse RL and imitation learning from observation.

This paper aims to analyse the impact of pre-training on the performance of deep RL-based ramp metering algorithms. Behaviour cloning is performed for increasing lengths of time (0-10,000 epochs), to pre-train a proposed Proximal Policy Optimisation (PPO)-based system. PPO, a policy gradient algorithm, is chosen as it is currently considered state-of-the-art in the field of RL (*16*), but has not yet been widely used within traffic control applications. ALINEA is used as the expert

ramp metering algorithm, as it is one of the most popular local ramp metering algorithms and is commonly used as a baseline. The paper is organised as follows; firstly some literature surrounding RL-based ramp metering systems is outlined. This is followed by the details of the PPO algorithm and the pre-training process of behaviour cloning, as well as details of the experiments performed to evaluate the system. Lastly, the results are analysed and plans for future research are presented.

LITERATURE REVIEW

RL is a paradigm of machine learning that aims to have an agent learn how to perform optimal actions by repeatedly interacting with its environment (*17*). This involves a training loop where an agent will measure the state of its environment, calculate and perform an optimal action based on this state, and evaluate its choice according to a predefined goal. Its learning performance is measured using a reward function, which can be used to reward or penalise the agent according to the impact of its chosen action. Experience gained through this iterative trial-and-error process can then be used to improve its action selection and maximise the reward function. In the context of ramp metering, the environment state represents the observed traffic conditions, the action set is the possible metering rates, and the state transition probabilities represent the traffic dynamics that convert the current state to its subsequent state. There is extensive literature surrounding RL-based ramp metering systems, particularly Q-learning. This is commonly used due to its simplicity and guaranteed optimal convergence. For example, a Q-learning algorithm is applied to local ramp metering in (*12*), where the proposed system was found to perform competitively with ALINEA by reducing total time spent (TTS) by 30.2% in comparison to a no-control scenario. However, as Q-learning uses a tabular RL approach with discretised inputs and outputs, the algorithm scales very poorly with the size of the state-action space. This tabular approach also limits Q-learning's applicability to complex nonlinear problems and to larger coordinated traffic management systems.

Alternatively, deep RL can perform much better in this type of environment by integrating neural networks into the RL framework. One such example is PPO, which, as a relatively new algorithm first proposed in 2017 (*18*), has not yet been extensively researched in its application to ramp metering. One example is the multi-agent PPO system proposed in (*9*), where on-ramps and corresponding motorway sections are assigned a PPO agent, and coordination is implemented through a shared observation space and reward. In this multi-ramp scenario, the proposed algorithm was able to achieve a higher average speed than ALINEA at both ramps and lower queue lengths. There was also less of a difference in average speed between the ramps, resulting in better homogeneity of traffic flow between their locations.

Pre-training is not often directly discussed within the literature, however, a deep RL ramp metering and perimeter control system is proposed in (*19*) that uses expert demonstration to guide the model. A Deep Q-network (DQN) is used, whose ramp metering policy is guided by a modified version of ALINEA during training. Their demonstration method is distinct from imitation learning and does not use a supervised learning approach, however, has a similar goal of improving the learning process. This is done by measuring the cross entropy between the "student" and "teacher" policies based on the likelihood of state-action pairs in each. The model is then optimised to minimise this error alongside its reward function, however, the importance of following the teacher is reduced over time to encourage the agent to develop its own policy. The algorithm reduced total travel time by 31.3% in a small-scale case study and increased the average trip speed by 6.7% in comparison to a no-control scenario. It also outperformed the demonstrator algorithms and scenarios using only ramp meters and perimeter control agents.

METHODOLOGY

Proximal Policy Optimisation (PPO)

The RL algorithm chosen for this analysis was Proximal Policy Optimisation (PPO), a relatively new policy gradient algorithm proposed in (*18*). PPO was proposed as a way to solve issues in the Trust Region Policy Optimisation (TRPO) algorithm, which had been deemed inefficient due to its second-order optimisation. TRPO itself had aimed to improve upon the stability of other deep RL methods by limiting the size of updates within a trusted region, implementing this using the Kullback–Leibler divergence between the old and new policy. PPO, instead, uses first-order optimisation and implements this update constraint with a clipping function, as well as penalisation of large deviations in the policy. As a result, PPO is much simpler and easier to implement than TRPO and is currently considered state-of-the-art (*16*).

PPO models use the Actor-Critic framework and consist of 2 neural networks; a policy network and a critic network. The policy network $\pi_\theta(s, a)$ directly represents the policy by mapping a given state s to its corresponding optimal action a, whilst the critic learns to estimate the state-value function $V(s)$ that represents the expected total return from a given state s.

The actor network π_θ is responsible for selecting an optimal action based on the current state in each iteration. Its weights θ are optimised using a stochastic gradient ascent algorithm that aims to maximise the clipped objective function L^{CLIP} in Equation (1). L^{CLIP} is implemented using an advantage function estimator $\hat{A}_t$ and the difference between the agent's old and current policies $r(\theta)$, calculated as in Equation (2). This ratio, $r_t(\theta)$ simply measures the divergence between an agent's previous policy and current policy, such that $r_t(\theta_{old}) = 1$. The clipping can then be implemented by discarding changes when this ratio falls outside the range $[1 - \epsilon, 1 + \epsilon]$, where ϵ is a tunable hyperparameter. This aims to increase the stability of the model and improve its robustness, as policy updates that make significant changes may be more likely to harm performance. However, setting ϵ too high, which allows larger policy divergences between updates, can lead to erratic behaviour.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \tag{1}$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{2}$$

The advantage function, $\hat{A}_t$ aims to estimate the advantage of performing a certain action, given the current state at time t, compared to the average benefit of all actions. This is calculated as in Equation (3). Here, a Generalised Advantage Estimator (GAE) is used as $\hat{A}_t$ where γ is the learning rate and λ is a factor denoting the trade-off between bias and variance in the estimator. $V(s)$ is the state-value function estimated by the critic network, meaning the expected total return from a policy given a state s. $\hat{A}_t$, therefore, runs the policy for T time steps and uses its estimated performance to update the current policy weights θ. This means that when $\hat{A}_t$ is positive, the chosen action will likely produce a higher-than-average reward, and its probability is increased.

$$\begin{aligned} &\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1} \\ &\qquad \text{where}\ \ \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{3}$$

In order to improve sample efficiency, the learning alternates between direct interactions with the environment and sampling of data collected during previous experiences. This means that, through repeated interaction with the environment, a dataset of previous state, action and reward tuples is collected. The model can then sample a minibatch from this dataset, using this to further optimise its networks' parameters.

Reinforcement Learning Pre-training

There are several approaches to pre-training, although the approach taken here is behaviour cloning, a type of imitation learning algorithm. Behaviour cloning is a process where a machine learning model is initially trained to develop a policy that mimics expert knowledge based on a demonstration dataset (*14*). This demonstration, or expert, dataset in the context of RL typically consists of a labelled dataset of state-action pairs, where the state is the observed environment conditions and the action is the optimal action generated by an expert. The model's parameters are then pre-trained to predict the state-action relationship in the dataset before training further by interacting with the environment. As a result, behaviour cloning is a supervised learning method and, in its simplest form, follows a process similar to standard data-driven machine learning.

First, a dataset $\mathcal{D}$ is created containing a set of optimal state-action pairs $\{(s_i, a_i)|i = 1, ..., N\}$ generated by the expert, where N is the size of the dataset. A policy π_θ, representing a neural network, is then initialised as a random approximator with parameters θ and optimised to minimise the prediction error on these state-action pairs in the dataset. For example, the mean squared loss can be used to produce an objective function as in Equation (4). This is repeated on each state-action pair in $\mathcal{D}$ for a number of epochs, where an epoch is one complete pass through the dataset.

$$\min_{\theta} \quad \frac{1}{n}\sum_{i=1}^{n}\left(a_i - \pi_\theta(s_i)\right)^2 \tag{4}$$

As previously mentioned, one of the largest disadvantages of RL is the required training time. This issue is particularly worsened in problems where the environment can be computationally expensive to model, as is the case with traffic modelling. Behaviour cloning, therefore, can be useful in speeding up the learning process through initial guidance to the model. Instead of learning a complete and optimal policy by itself, pre-trained models are provided a head start. During training, the model is then able to diverge from the policy of its expert demonstrator, exploring new policies with an established base. Importantly, the model is able to continue learning and improve upon the policy of its demonstrator at this stage, assuming the model can continue maximising its reward function. For example, an RL model could be pre-trained with a generic policy but can then further develop this into a much more specialised and effective policy through training.

However, although behaviour cloning is a relatively simple algorithm that can perform sufficiently well in situations similar to those found in the expert dataset, some of its main limitations surround overfitting and generalisability (*20*). Overfitting is a common problem within machine learning and refers to situations where an algorithm fits too closely to its training dataset and cannot generalise to unknown situations. This can occur if the expert dataset is not large enough or has insufficient coverage of possible situations. This coverage issue is called covariate shift, and there are a variety of solutions, including the DAgger algorithm (*21*) which aims to improve upon

behaviour cloning by changing how the dataset is sampled according to previous training iterations. However, DAgger must iteratively interact with the expert, which can be computationally expensive.

EXPERIMENTS

Scenario

The scenario chosen for training and evaluating the system uses a network introduced in (*22*), containing 3 on-ramps and 1 off-ramp along a 9.5km stretch of motorway and 24.5km of surrounding local roads. The scenario, which is simulated using SUMO (an open-source macroscopic traffic simulation package (*23*)), lasts 5 hours with on-peak flows active between around 0.5 and 1.5 hours. Figure 1 shows the network layout and Table 1 shows the corresponding demand profile.

As noted in (*22*), the scenario is designed to be difficult for local ramp metering systems due to drivers' routing choices, which cause a bottleneck around the furthest downstream on-ramp. Specifically, drivers travelling from A_1 to B_1 prefer to travel along the motorway using the furthest upstream on-ramp as this should be the fastest route. However, when demand exceeds capacity, they instead must reroute along the local roads and enter at the furthest downstream on-ramp to avoid congestion. Similar behaviour is found in those travelling between A_3 and B_2 who would prefer to use the middle section of the motorway but instead must reroute through the much slower local roads. This is replicated in SUMO as drivers calculate an optimal route based on conditions, done at the moment they are inserted into the simulation.

Therefore, whilst this is a difficult scenario, it still represents an interesting case study for a learning-based ramp metering system to attempt whilst analysing the impact of pre-training. This also leaves an opportunity for future works to more fully develop the proposed system into a coordinated approach.

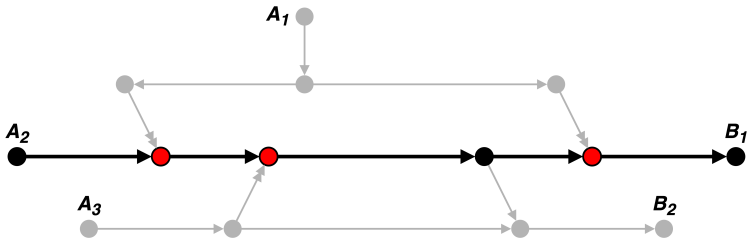

Figure 1 Scenario road network with the motorway in black and local roads in grey. The 3 metered on-ramp nodes are highlighted in red.

Table 1 Scenario OD demand profile (off-peak/on-peak).

O-D node	B_1	B_2
A_1	1250/**2250**	0/**0**
A_2	1750/**2550**	0/**0**
A_3	0/**0**	100/**1600**

PPO Ramp Metering

The ramp metering system used in this paper uses 3 locally operated ramp meters at each on-ramp, where each meter uses a singular PPO agent responsible for finding the optimal metering rate. However, 2 versions of this setup were tested. A purely local version was tested first, only operating based on local conditions, which will be referred to here as 2Do PPO. Then, due to the difficulty of the scenario for local algorithms, some coordination was tested with a second version using a shared observation space. This version is referred to as 6Do. All other elements of the two systems and their processes should be considered the same, except where specified. All the agents were modelled with Stable Baselines3 (*24*) using the default PPO parameters, the most important of which are outlined in Table 2. Both the policy and value networks are Multi-layer Perceptrons (MLPs) with 2 dense layers of 64.

Table 2 PPO hyperparameter settings.

Parameter	Value
Clipping range - ϵ	0.2
Discount factor - γ	0.99
GAE Bias vs variance factor - λ	0.95
Learning rate	0.0003
Minibatch size	64
Gradient ascent epochs	10

State

Each agent uses two main local, continuous variables, vehicle density ρ downstream of its on-ramp and the queue length l. ρ and l are measured as veh/km in the downstream section and as a percent of the total storage capacity of its on-ramp respectively. The 2Do PPO agents only use these variables as their inputs, resulting in a 2D state space. The 6Do PPO agents use a shared state space where all 3 agents' local state variables are concatenated to create this 6D shared state space.

Action

The output for each agent is its metering rate for the following control interval r_{t+1}, measured in veh/hr. r is discretised into 7 values evenly spaced between 200 and 1400, and one value of 2000 veh/hr representing deactivation. This results in an 8D action space of; 2000, 1400, 1200, 1000, 800, 600, 400 and 200.

Reward

The reward function used is in Equation (5) and has the main goal of reducing total travel time based on an optimal local density downstream of each ramp and the minimisation of queuing. It has also been designed to be continuous and very flexible through a set of tunable hyperparameters. When ρ is low, the controller receives the maximum reward as the meter should not be limiting flow when traffic conditions are in a free flow state. However, as ρ approaches a neighbourhood surrounding ρ_{cr}, the reward becomes dependent on the queue length l. The agent still receives the

maximum reward when no vehicles are queuing at the on-ramp, but it is penalised as the queue increases. This is done using α, representing $\mu l \rho_{cr}$, where μ regulates how heavily the agent is penalised. The bounds of this neighbourhood, represented by $[a-b]$ in the reward function, are tunable hyperparameters set to $[0.8-1.1]$. This means once ρ increases past 110% of ρ_{cr}, the agent is penalised much more heavily, according to the hyperparameter γ. The penalty is still dependent on l, however, this factor is regulated using β and reduced when l is below 10%.

$$R = \begin{cases} 0, & \text{if } \rho \leq \rho_{cr} \\ \alpha \cos \frac{2\pi(\rho - \rho_{cr}(b-2a))}{\rho_{cr}(b-a)} - \alpha, & \text{if } a\rho_{cr} < \rho < b\rho_{cr} \\ -\gamma \max(\beta l, 0.1)(\beta \rho_{cr} - \rho)^2, & \text{otherwise} \end{cases} \tag{5}$$

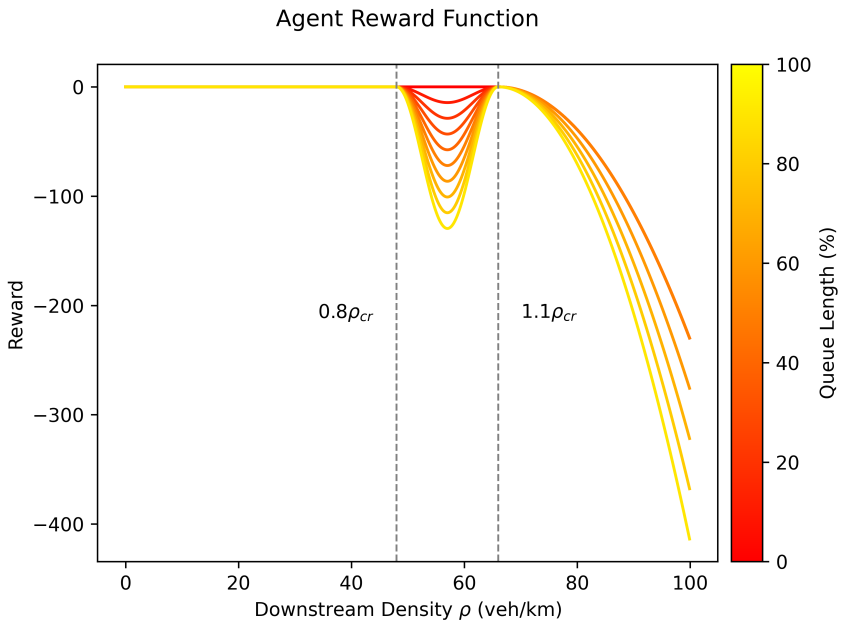

Figure 2 Local PPO agent reward function, based on downstream density and queue length.

Ideally, this reward function should encourage agents to push density into the neighbourhood of the critical value, but not exceed this by too much. Therefore, the agent is rewarded when density and traffic demand are low, but minimising queuing becomes more important as density approaches this neighbourhood. This is done to fully maximise efficient use of the available mainline capacity. However, when the downstream density is much higher than the critical density, the agent is heavily penalised as a result. Figure 2 demonstrates visually how the reward changes with respect to ρ and l.

PPO Training & Pre-training

ALINEA Dataset

ALINEA operates using Equation (6) which, for a time step k, outputs the metering rate r based on the difference between the desired occupancy $\hat{o}$ and measured downstream occupancy o_{dn}. A regulator parameter K_R is also used to change how sensitive the controller is to changes in input. ALINEA is based on the linear quadratic feedback law and is an example of a closed-loop feedback algorithm, referring to how the output from the previous time step, the metering rate, is used in the current step. Many studies have demonstrated the effectiveness of ALINEA over no-control scenarios, particularly with respect to its simplicity and low implementation cost, as only one mainstream measurement is required (*25*).

$$r(k) = r(k-1) + K_R[\hat{o} - o_{dn}(k)] \tag{6}$$

The expert dataset used for pre-training was, therefore, created by running the same scenario and demand profile as outlined previously, with ALINEA operating at each of the 3 on-ramps in the network. The necessary PPO model inputs and outputs (downstream density, queue length and metering rate) were collected from each controller with each update over 100 runs of the simulation. As ALINEA generates a continuous output, the metering rate was then discretised to mirror the PPO models' action spaces, as outlined previously, by taking the closest of the 8 values. With each simulation lasting 10000s and a short control interval of 60s, this resulted in a dataset containing 16664 state-action pairs that could be used for pre-training. Noise was added to the demand profile during each run in order to cover as wide a range as possible of inputs and outputs and aid the performance of the behaviour cloning.

Pre-training

The PPO model here uses MLPs as their actor and critic network, which are parameterised through their weights θ and ϕ respectively. During pre-training, these networks are trained on the expert dataset, ultimately producing a model that has learnt to predict the output of ALINEA based on the inputs outlined previously.

Three levels of pre-training were tested; a model without any pre-training and 2 models pretrained for 1000 and 10000 epochs. For the sake of brevity, these will be referred to as A0, A1000 and A10000 respectively. An example of system outputs after different levels of pre-training is shown to the left of Figure 3. As should be expected, it can be seen that the A0 model is acting close to random as the model weights have not been optimised at all since initialisation, whilst the A1000 and A10000 models are limiting inflow with more precision.

Training

For each of the 6 experiments, covering all combinations of the 2Do and 6Do PPO models and A0, A1000 and A10000 levels of pre-training, the 3 agents start with an identical copy of the pretrained PPO model. All 3 are then trained in parallel for 500 episodes, where an episode is 1 simulation run of 10000s with some random noise in how vehicles are inserted into the network. The training allows the agents to deviate and try to improve upon ALINEA, as well as specialise to their specific location in the network and the impact of queuing as an input. It should be noted that agents are only trained for 500 episodes to focus more fully on how pre-training changes per-

formance, as well as due to some convergence issues as discussed more in the following sections. An example of system outputs after training is shown to the right of Figure 3.

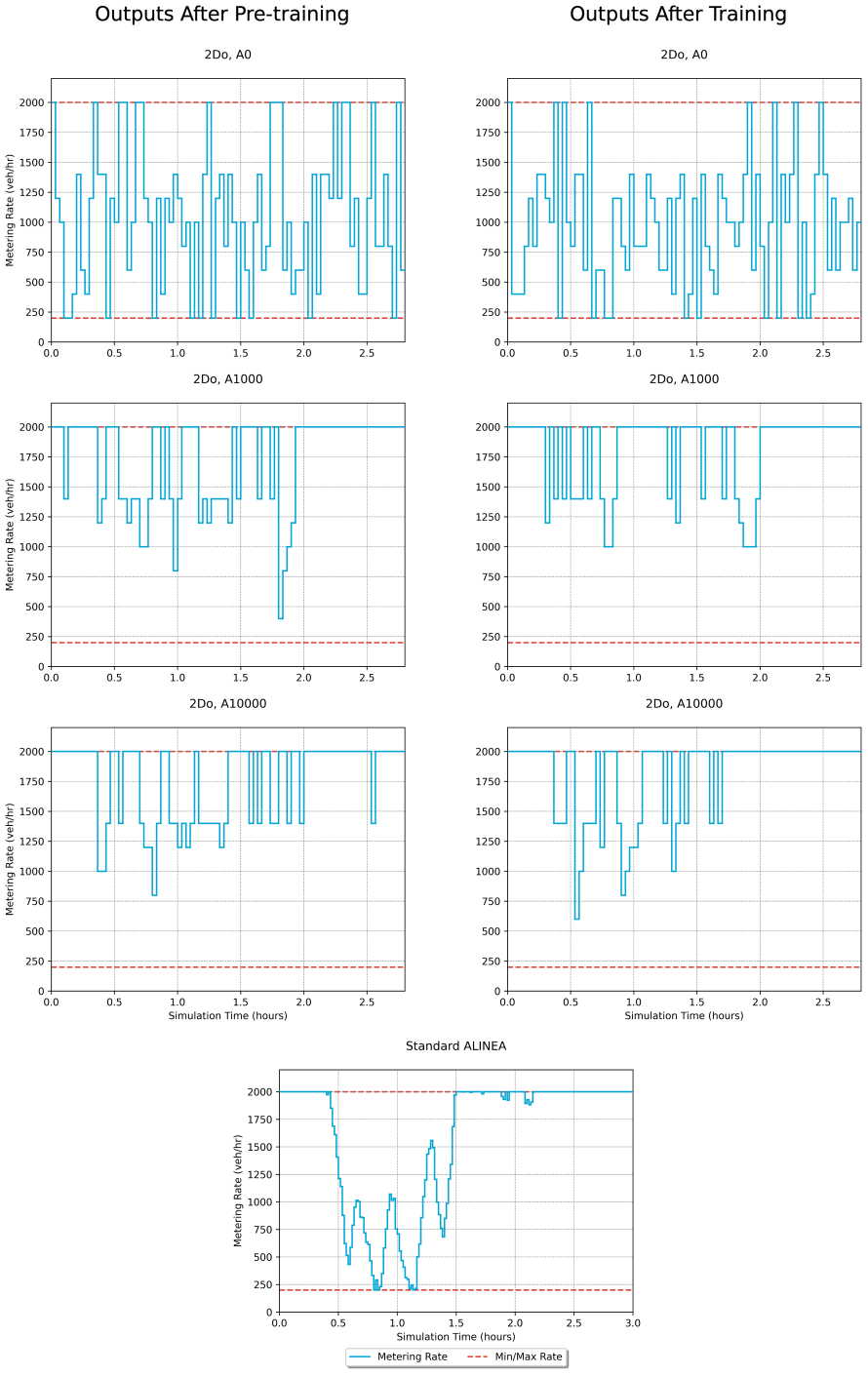

Figure 3 Metering rate outputs after pre-training and after 500 episodes of training. Outputs for the meter at node 3 are shown with increasing levels of pretraining, as well as the outputs of ALINEA for comparison.

RESULTS & EVALUATION

In order to evaluate the PPO controller and the impact of pre-training, all 8 control scenarios were run 20 times in SUMO. This includes all 6 combinations of 2Do, 6Do and A0-A10000 PPO models, one scenario where all ramps are controlled using standard ALINEA, and a no-control baseline scenario. All runs used an identical set of seeds and all data has been averaged across runs to draw valid comparisons between them. As discussed below, TTS throughout the whole network increased in comparison to no control for all scenarios, albeit by varying amounts, likely due to the difficulty of the scenario for local algorithms. Table 3 shows the TTS for ALINEA and all combinations of 2Do, 6Do and A0-A10000 models, with a comparison to the no control scenario.

It should also be noted that the proposed PPO ramp metering system showed a weak convergence during training, likely due to system design issues such as in the agents' reward function. This is demonstrated by no clear trend in the episode rewards in Figure 4. As a result of this, there were only limited improvements in the system's policy throughout training. Nonetheless, pre-training has been able to produce a significant and measurable difference in the performance of each model and so training was limited to 500 episodes. As less specialisation is introduced during training, the agents' policies should subsequently be much more dependent on the amount of pre-training. Instead, the design and performance of the proposed PPO system will be more fully addressed in later works.

Table 3 Total Time Spent for all scenarios and difference to no-control.

Scenario		TTS ($\times 10^6$s)	Difference (%)
No Control		8.523	-
ALINEA		9.092	6.7%
2Do	A0	9.796	14.9%
	A1000	8.613	1.1%
	A10000	8.561	0.4%
6Do	A0	9.766	14.6%
	A1000	8.571	0.6%
	A10000	8.594	0.8%

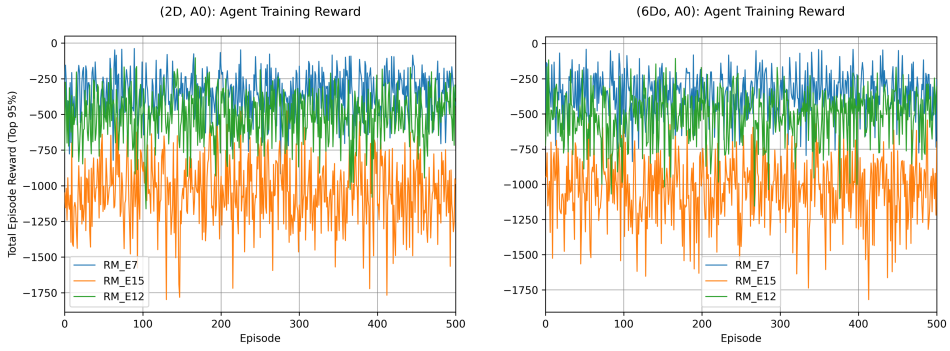

Figure 4 Total of the 95th percentile of episode rewards for the 2Do and 6Do A0 models.

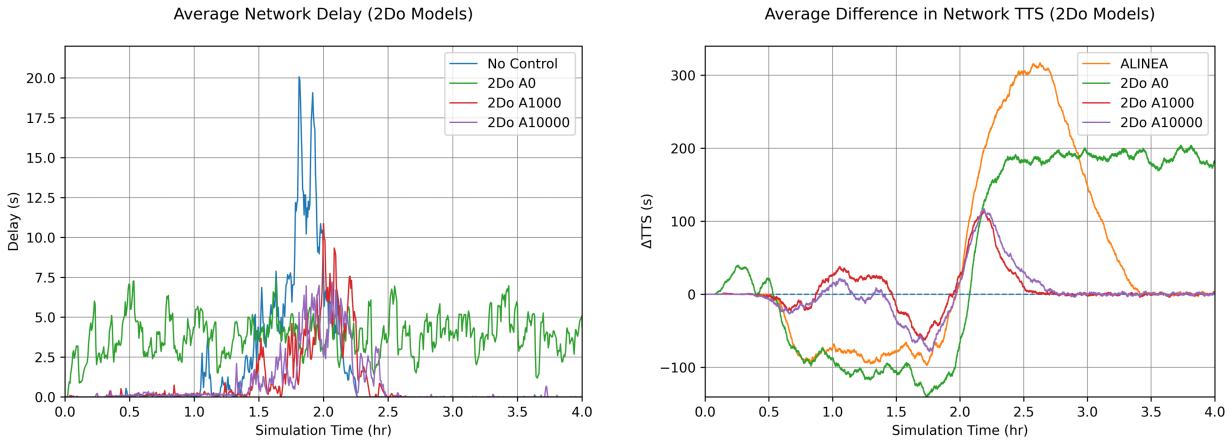

Figure 5 (Left) Average network-wide delay for the 2Do PPO models and no-control scenarios. (Right) Difference in network-wide TTS to the no-control scenario for all 2Do models and ALINEA averaged over 20 runs. A positive ΔTTS means the average TTS is higher than in the no-control scenario.

2Do PPO Agents

Firstly, as can be seen in Table 3, the A0 model clearly performs the worst of the scenarios tested. The model produced the largest increase in TTS compared to no-control, 14.9%, as it is still acting close to random. In Figure 5, which shows the network-wide delay on the left and the difference in TTS to no-control on the right, it can be seen that the model starts metering almost immediately which unnecessarily causes increasing TTS from delays. Performance is similar to ALINEA during peak times between 0.5 and 1.5 hours, if not slightly better as there is a greater reduction in TTS. However, its outputs are less stable as the A0 model is alternating between under and over-limiting on-ramp flow, as seen in the fluctuations of ΔTTS. This also results in a fairly consistent amount of network-wide delay throughout the simulation. Then, after around 2 hours, when the congestion should start to resolve, the agents have not learnt a policy that would release vehicles at the on-ramp. Vehicle delay, therefore, continues to increase at a steady rate until the end of the simulation whilst TTS tends very slowly towards no-control levels. Ultimately, this should be somewhat expected with the limited policy convergence during training, as the A0 has not been able to learn a complete policy on its own without any expert guidance through pre-training.

Conversely, the pre-trained models perform significantly better than the A0 model and ALINEA. For example, the TTS is reduced to no-control levels much faster, as seen in the right of Figure 5, doing so around half an hour before ALINEA. TTS is also maintained much closer overall to that in the no-control scenario, at 0.4% and 1.1% for the A10000 and A1000 models respectively. Whilst this is partially because the A1000 and A10000 models are not limiting flow enough at peak times, as TTS even increases past no-control between 1 and 1.5 hours, the delay is still kept much lower. As shown in Figure 5, the peak delay for the pre-trained PPO models is around half that of the no-control. The difference between the two levels of pre-training, on the other hand, is much smaller. Although the A10000 model performs slightly better and results in a smaller overall TTS, its outputs are somewhat less stable and produce larger fluctuations in ΔTTS during peak times. This could be a sign of overfitting during the longer pre-training, where the outputs of the neural

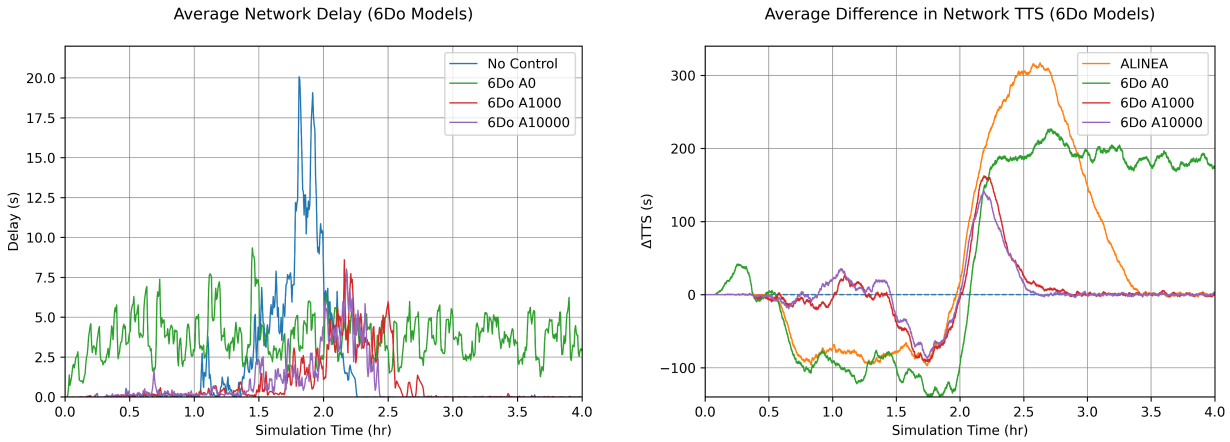

Figure 6 (Left) Average network-wide delay for the 6Do PPO models and no-control scenarios. (Right) Difference in network-wide TTS to the no-control scenario for all 6Do models and ALINEA averaged over 20 runs. A positive ΔTTS means the average TTS is higher than in the no-control scenario.

network become too dependent on replicating the values in the training dataset. This would lead to worse generalisability in unseen situations and convergence towards a suboptimal policy.

6Do PPO Agents

The 6Do A0 model performs similarly to the 2Do A0 model, although it is somewhat less stable during peak times, resulting in the larger fluctuations of ΔTTS in Figure 6 and an overall increase in TTS of 14.6% compared to no control. Again, the model is still acting close to random and causing consistent unnecessary delays throughout the simulation, as also seen in Figure 6. However, the difference between the pre-trained 6Do models is much smaller than between the 2Do models, with a 0.6% and 0.8% increase in TTS for the A1000 and A10000 models respectively. Both maintained TTS close to no control levels during peak times, again suggesting insufficient metering to some extent, however, both display similar levels of fluctuations in ΔTTS during this time. Therefore, whilst some pre-training has been beneficial as the A0 still clearly performed worst, the longer pre-training has only had a very marginal effect. The A10000 model produced a slightly lower total delay than the A1000 model and reduced TTS to no control levels faster, however, this difference is not significant. In comparison to the 2Do models, the peak delay using the 6Do pre-trained models was around 25% lower and occurred later on average as seen in the left of Figure 6, yet otherwise the shared state space has not shown any significant advantage.

As the 6Do models use more inputs than the 2Do models, more learnt parameters have to be prioritised, even though the agents' action space remains the same size. Therefore, whilst more training is always required as RL models become more complex, the partial reduction in performance could be due to the chosen expert algorithm. Specifically, one possible explanation for this is that issues could have arisen from pre-training agents with a shared state space based on a set of local expert algorithms operating in parallel. Whilst the 2Do models begin with a pre-trained neural network that has learnt to map a small set of local values to its input, the 6Do models must instead learn how to correctly prioritise these different inputs. This is alongside the

larger set of dynamics between each of these inputs and the single output that is not considered by separate ALINEA controllers in the expert dataset. In this instance, further training would help the agents to specialise towards their specific location and learn their relationship with other parts of the network, albeit without the aforementioned convergence issues. Nevertheless, this simple method of coordination highlights the challenge of designing coordinated ramp metering systems and the adaptations needed to convert a local system to one that combines multiple meters.

CONCLUSIONS

Pre-training can be a valuable tool in improving the learning and performance of deep RL ramp metering systems. For example, both A0 models, those without any pre-training, performed significantly worse than the pre-trained models, with a minimum 13.8% difference for the 2Do and 6Do PPO models. However, performing too much pre-training can be harmful as overfitting leads to poor generalisability to situations not found in the expert dataset used. Whilst the A10000 models performed similarly in terms of total TTS, their outputs were more unstable and produced larger fluctuations in TTS in comparison to the A1000 models. Careful attention should also be brought to the expert chosen based on the design of the ramp metering system itself, as can be seen in the difference between the 2Do and 6Do models.

Future works will primarily include changes to the design of the PPO system, such as around the reward function which could be adapted to give more guidance to the RL agents during training and improve convergence towards a more optimal policy. This should allow for a deeper analysis of how agents specialise after being pre-trained, particularly around how their policies diverge from that of the expert. Other design changes could include a different action space that allows for finer control, either through a different distribution of discrete values with more low values or an extension to a continuous action space.

ACKNOWLEDGEMENTS

This research is part of the project "AI in Network Management," funded by Rijkswaterstaat, grant agreement nr. 31179439, under the label of ITS Edulab.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design: C. Evans, M. Rinaldi; data collection: C. Evans; analysis and interpretation of results: C. Evans, M. Rinaldi, H. Taale, S. P. Hoogendoorn; draft manuscript preparation: C. Evans, M. Rinaldi, H. Taale, S. P. Hoogendoorn. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

1. E. D Arnold. *Ramp metering: a review of the literature.* Technical Report VTRC 99-TAR5. Virginia Dept. of Transportation, Virginia Transportation Research Council, Dec. 1998.

2. Joseph R. Scariza. "Evaluation of coordinated and local ramp metering algorithm using microscopic traffic simulation". ISSN: 5285-0447. Thesis. Cambridge, MA: Massachusetts Institute of Technology, 2003.

3. Yibing Wang et al. Freeway Traffic Control in Presence of Capacity Drop. *IEEE Transactions on Intelligent Transportation Systems* 22.3 (Mar. 2021), pp. 1497–1516. DOI: 10.1109/TITS.2020.2971663.

4. Habib Haj-Salem and Marcos Papageorgiou. Ramp metering impact on urban corridor traffic: Field results. *Transportation Research Part A: Policy and Practice* 29.4 (July 1, 1995), pp. 303–319. DOI: 10.1016/0965-8564(94)00034-8.

5. Cambridge Systematics. *Twin Cities Ramp Meter Evaluation: Final Report*. NTIS-PB2001107493. Minnesota Department of Transportation, Feb. 1, 2001.

6. Frans Middelham and Henk Taale. Ramp Metering in the Netherlands: An Overview. *IFAC Proceedings Volumes*. 11th IFAC Symposium on Control in Transportation Systems 39.12 (Jan. 1, 2006), pp. 267–272. DOI: 10.3182/20060829-3-NL-2908.00047.

7. Markos Papageorgiou, Habib Hadj-Salem, and Jean-Marc Blosseville. ALINEA: A Local Feedback Control Law for On-ramp Metering. *Transportation Research Record* 1320 (1991), pp. 58–64. ISSN: 0361-1981.

8. Francois Belletti et al. Expert Level Control of Ramp Metering Based on Multi-Task Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* 19.4 (Aug. 16, 2017), pp. 1198–1207. DOI: 10.1109/TITS.2017.2725912.

9. Fuwen Deng et al. Advanced Self-Improving Ramp Metering Algorithm based on Multi-Agent Deep Reinforcement Learning. *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. Auckland, New Zealand, Oct. 2019, pp. 3804–3809. DOI: 10.1109/ITSC.2019.8916903.

10. Kasra Rezaee, Baher Abdulhai, and Hossam Abdelgawad. Self-Learning Adaptive Ramp Metering: Analysis of Design Parameters on a Test Case in Toronto, Canada. *Transportation Research Record* 2396.1 (Jan. 1, 2013), pp. 10–18. DOI: 10.3141/2396-02.

11. T. Schmidt-Dumont and J.H. Van Vuuren. Decentralised reinforcement learning for ramp metering and variable speed limits on highways. *IEEE Transactions on Intelligent Transportation Systems* 14.8 (2015).

12. Chao Lu and Jie Huang. A self-learning system for local ramp metering with queue management. *Transportation Planning and Technology* 40.2 (Feb. 17, 2017), pp. 182–198. DOI: 10.1080/03081060.2016.1266166.

13. Zihan Ding and Hao Dong. "Challenges of Reinforcement Learning". In: *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Ed. by Hao Dong, Zihan Ding, and Shanghang Zhang. Springer Singapore, 2020, pp. 249–272. DOI: 10.1007/978-981-15-4095-0_7.

14. Zihan Ding. "Imitation Learning". In: *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Ed. by Hao Dong, Zihan Ding, and Shanghang Zhang. Springer Singapore, 2020, pp. 273–306. DOI: 10.1007/978-981-15-4095-0_8.

15. Zhihui Xie et al. *Pretraining in Deep Reinforcement Learning: A Survey*. 2022. arXiv: 2211.03959 [cs.LG]. URL: https://arxiv.org/abs/2211.03959.

16. Chao Yu et al. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Advances in Neural Information Processing Systems* 35 (Dec. 6, 2022), pp. 24611–24624.

17. Zihan Ding et al. "Introduction to Reinforcement Learning". In: *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Ed. by Hao Dong, Zihan Ding, and Shanghang Zhang. Springer Singapore, 2020, pp. 47–123. DOI: 10.1007/978-981-15-4095-0_2.

18. John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

19. Zijian Hu and Wei Ma. Demonstration-guided deep reinforcement learning for coordinated ramp metering and perimeter control in large scale networks. *Transportation Research Part C: Emerging Technologies* 159 (Feb. 1, 2024), p. 104461. DOI: 10.1016/j.trc.2023.104461.

20. Felipe Codevilla et al. Exploring the Limitations of Behavior Cloning for Autonomous Driving. *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9329–9338.

21. Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA, Apr. 11, 2011, pp. 627–635.

22. Marco Rinaldi et al. Centralized and decomposed anticipatory Model Predictive Control for network-wide Ramp Metering. *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. Oct. 2013, pp. 461–467. DOI: 10.1109/ITSC.2013.6728274.

23. Pablo Alvarez Lopez et al. Microscopic Traffic Simulation using SUMO. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2575–2582. DOI: 10.1109/ITSC.2018.8569938.

24. Antonin Raffin et al. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research* 22.268 (2021), pp. 1–8.

25. M. Papageorgiou, H. Hadj-Salem, and F. Middelham. ALINEA Local Ramp Metering: Summary of Field Results. *Transportation Research Record* 1603.1 (Jan. 1, 1997), pp. 90–98. DOI: 10.3141/1603-12.