Transfer Learning for Efficient Iterative Safety Validation

Anthony Corso and Mykel J. Kochenderfer
Stanford University, Department of Aeronautics and Astronautics, 496 Lomita Mall, Stanford, CA 94305
{acorso, mykel}@stanford.edu

Abstract

Safety validation is important during the development of safety-critical autonomous systems but can require significant computational effort. Existing algorithms often start from scratch each time the system under test changes. We apply transfer learning to improve the efficiency of reinforcement learning based safety validation algorithms when applied to related systems. Knowledge from previous safety validation tasks is encoded through the action value function and transferred to future tasks with a learned set of attention weights. Including a learned state and action value transformation for each source task can improve performance even when systems have substantially different failure modes. We conduct experiments on safety validation tasks in gridworld and autonomous driving scenarios. We show that transfer learning can improve the initial and final performance of validation algorithms and reduce the number of training steps.

Introduction

Introducing autonomy into safety-critical domains, such as autonomous driving, aviation, and medicine, has the potential to improve both safety and efficiency. The consequences of operational errors of these systems include loss of property or human life, so extensive safety validation and testing is required before deployment. Black-box sampling approaches have emerged as a scalable safety validation tool for discovering failures in complex environments. Many algorithms for safety validation have been explored in the literature (Corso et al. 2020), often with the goal of finding failures of a system with fewer samples or less computational effort. There has been little focus on the potential efficiency of validating many related systems sequentially.

Designing and certifying safety-critical systems generally involves assessing a sequence of closely related systems. Safety validation of complex systems often requires significant computational effort and existing approaches generally start from scratch each time the system under test is changed. The need to frequently perform safety validation on related systems therefore imposes a large computational burden, but also an opportunity to improve safety validation efficiency.

To improve the efficiency of safety validation across related systems, we use knowledge from the validation of previous systems to inform the validation of the next system. We formulate iterative safety validation as a transfer learning problem by modeling each safety validation task as a Markov decision process. The previously solved tasks are used as the set of source tasks, and we transfer knowledge to future tasks in the form of action value functions. We use state-dependent attention weights to learn which previous solutions are applicable to the current problem.

Existing safety validation algorithms use approaches from optimization (Mathesen et al. 2019), path-planning (Zutshi et al. 2014), reinforcement-learning (Lee et al. 2020), and importance sampling (Huang et al. 2017), and often only address the validation of a single system. Uesato et al. (2019) use previous versions of a system to train a failure classifier that predicts which initial conditions of a system will lead to failure, but their approach is not applicable to sequential decision making problems of the type we consider. Wang, Nair, and Althoff (2020) alternately train an agent and perform safety validation on it to improve robustness. On each iteration, the safety validation algorithm starts with the parameters from the previous iteration to improve efficiency. In fact, any parametric safety validation algorithm (Koren et al. 2018; Akazaki et al. 2018; Kim and Kochenderfer 2016) could simply reuse parameters from previous tasks and then fine-tune them for better performance. We demonstrate in our experiments, however, that a fine-tuning approach often fails to reach the same performance as starting from scratch.

To investigate the effectiveness of transfer learning in the safety validation setting, we apply existing transfer learning algorithms to the problem. These include fine-tuning of past solutions, and the attend, adapt, and transfer (A2T) algorithm (Rajendran et al. 2017). When the systems we wish to validate have dissimilar behavior, however, we find that existing approaches can perform poorly. We propose a modification to A2T that transforms the state and action value spaces for each source task to increase knowledge transfer between dissimilar tasks. We evaluate the initial performance, the final performance, and the number of training steps required to reach the same performance as a no-transfer algorithm. We consider four iterative safety validation tasks in gridworld and autonomous driving scenarios and demonstrate that transfer learning has the potential to significantly improve the performance and sample efficiency of safety validation algorithms.

Background

In this section, we introduce Markov decision processes (MDPs), discuss knowledge sharing between related MDPs, and formulate safety validation as a sequence of MDPs.

Markov Decision Processes

A Markov decision process (MDP) (Kochenderfer 2015) is a model for sequential decision making problems defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. The state space $\mathcal{S}$ contains all possible states of the MDP and the action space $\mathcal{A}$ contains the possible actions of a decision making agent. At each step, the agent chooses an action and the MDP transitions to a new state s' with probability $P(s' \mid s, a)$ and receives a reward $r = R(s, a, s')$ discounted by a factor γ for each step. An agent's behavior is controlled by a policy π that maps states to actions such that $a = \pi(s)$. The optimal policy π^* maximizes the *action value function* $Q^\pi(s, a)$, which is the expected sum of discounted rewards by taking action a from state s, and then following policy π.

There are many approaches to solving for π^* (Sutton and Barto 2018) but we focus on deep Q-learning (DQN) (Mnih et al. 2015). In DQN, the optimal action value function is approximated by a deep neural network with parameters θ, $Q(s, a; \theta) \approx Q^*(s, a)$. The Q-network tries to minimize the loss with respect to a target network with parameters θ^-. The parameters are updated using gradient descent

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathbb{E}[L(y(s'; \theta^-), Q(s, a; \theta)] \tag{1}$$

where α is the learning rate and loss function L can be the squared error or the Huber loss (Huber 1992). The target is $y(s'; \theta^-) = r + \gamma \max_{a'} Q(s', a'; \theta^-)$ when s' is not terminal, and $y(s'; \theta^-) = r$ when s' is terminal. The target network parameters are periodically updated to θ after a specified number of training steps. During training, the expectation is computed from samples that are stored in an experience replay buffer that is prioritized by the temporal difference error of each sample (Schaul et al. 2016).

Transfer Learning

Transfer learning is concerned with using knowledge gained by solving one task to improve the learning process in another related task (Taylor and Stone 2009). In reinforcement learning, each task is an MDP and each solution is a policy. When tasks have different state and action spaces, we require task mappings that relate the states and actions between tasks. Task mappings may be provided by a human (Taylor, Stone, and Liu 2007) or learned from data (Taylor, Kuhlmann, and Stone 2008). If the state and action spaces are the same, then tasks can share a variety of low-level information such as experience samples (s, a, s', r), action value functions, policies, or models of the environment. High-level information, such as a set of options, shaping rewards or feature encodings, may also be transferred to improve learning on a new task. One challenge in transfer learning is *negative transfer* where knowledge from one or more source tasks impairs the performance on the current task. Negative transfer can be mitigated using an attention mechanism, or a human oracle that decides which tasks are relevant (Taylor and Stone 2009).

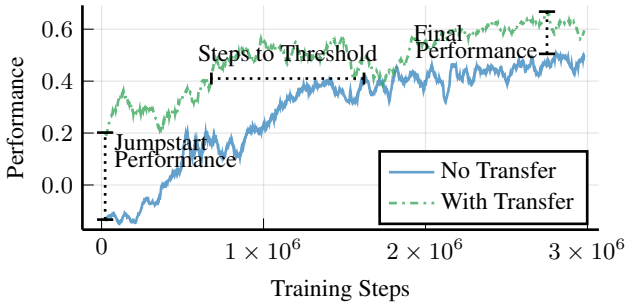


Figure 1: Metrics for evaluating transfer learning algorithms.

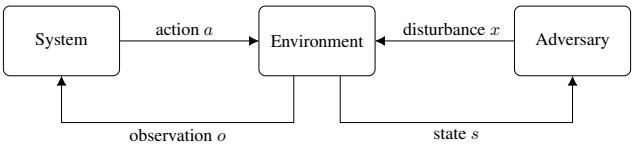


Figure 2: Model of the safety validation problem.

Transfer learning algorithms can be evaluated against a no-transfer alternative in a variety of ways (fig. 1). *Jumpstart* is the amount of improvement before any training has occurred, *final performance* is the difference between the best performances achieved, and *steps to threshold* is the difference between the number of training steps required to reach a specified threshold.

Safety Validation

Safety validation algorithms (fig. 2) search for sequences of disturbances in an environment that cause an autonomous system to fail (Kapinski et al. 2016; Corso et al. 2020). At each step, the autonomous agent under test (or *system*) makes an observation $o \in \mathcal{O}$ of the environment and decides to take action $a \in \mathcal{A}$. An adversary observes the state $s \in \mathcal{S}$, then applies a disturbance $x \in \mathcal{X}$ with the goal of causing the system to arrive in a set of failure states $E \subseteq \mathcal{S}$.

Safety validation can be modeled as an MDP defined by $(\mathcal{S}, \mathcal{X}, P, R, \gamma)$ where $\mathcal{S}$ represents the possible states of both the system and the environment, and $\mathcal{X}$ is the space of possible disturbances controlled by the adversary. The transition function $P(s' \mid s, x)$ includes the action of the system and the dynamics of the environment. The reward function depends on the safety validation goal, and in this work we solve for the most likely failure (Corso et al. 2020) with

$$R(s, x, s') = \lambda \log p(x \mid s) + \mathbb{1}\{s' \in E\} \tag{2}$$

where λ is a small positive constant and $p(x \mid s)$ is the probability of x occurring naturally in the environment.

Safety validation MDPs may differ in a variety of ways, which we can explain using an example based on autonomous driving. If validation is performed on two different road geometries (e.g. highway-driving and an intersection), then the state space and disturbance space may be different. If we fix the road geometry, then the transition model

may vary if we validate different driving policies. The reward function will vary if the disturbance model changes or we alter the set of failure states. In this work, we consider systems with different behavior operating in similar environments. We, therefore, assume that the state space, disturbance space, and reward function remain fixed while the transition model varies between tasks.

Proposed Approach

In this section, we first show how to formulate iterative safety validation as a sequence of tasks and specify two ways that knowledge transfer may occur. We then introduce A2T as our choice of transfer learning algorithm and propose a modification to it.

Problem Formulation

Suppose we are performing safety validation on a sequence of related systems and must validate each system before observing the next one. We model this problem as solving a sequence of MDPs (or tasks) $[T_1, T_2, \ldots]$ where the ith task is given by $T_i = (\mathcal{S}, \mathcal{X}, P_i, R, \gamma)$. Due to variations in system behavior, the transition model P_i is unique to each task, while $\mathcal{S}$, $\mathcal{X}$, R, and γ are shared across tasks. We wish to develop a learning algorithm L that solves task i given the $i-1$ previous solutions $[K_1, K_2, \ldots, K_{i-1}]$ such that

$$K_i = L(T_i; K_{1:i-1}). \tag{3}$$

The previous solutions may take the form of value functions or policies. The learning procedure is iterative because the new solution K_i can be added to the set of previous solutions when solving the next task T_{i+1}. Since all previous solutions are used, L must avoid negative transfer by learning which source task solutions are applicable to the current task.

In the context of safety validation, we hypothesize two qualitatively distinct ways that the tasks will be related. The first case is that of a *learning system* which is improving its performance from task to task. Each task is therefore more challenging for the adversary since failures that were present in previous tasks may no longer exist or may only occur due to a narrower range of disturbances. In this context, the most recent solutions are likely to provide the most relevant information and large parts of those solutions may be directly applicable to the new task. The second case is that of *comparable systems* where the systems have a similar level of competency but exhibit different behavior. Disturbance trajectories that lead to failure for one system may not cause failure in any other systems, although they might share similar conceptual failure modes. In this setting, direct transfer of solutions may be ineffective, and we need to rely on other types of knowledge.

Choice of Learning Algorithm

To accelerate safety validation we use the attend, adapt, and transfer (A2T) learning algorithm (Rajendran et al. 2017) with a minor modification. A2T accelerates learning on a new task by combining the solutions of k previous tasks using a learned set of state-dependent attention weights. To avoid negative transfer, A2T simultaneously learns a solution from scratch so there are a total of $k+1$ solutions and as many attention weights. A2T can be used to estimate optimal policies or optimal value functions and we found it most straightforward to estimate the optimal action value function. When Q^* is estimated by a neural network, called a Q-network, the input is a vector representing the state and the output is a vector representing the values of a discrete set of disturbances. To make this clear, if $\mathcal{X} \subseteq \mathbb{R}^m$, then $\hat{Q}(s) \in \mathbb{R}^m$, is a vector that represents the estimates of the optimal values for each disturbance. The action value function is estimated by the expression

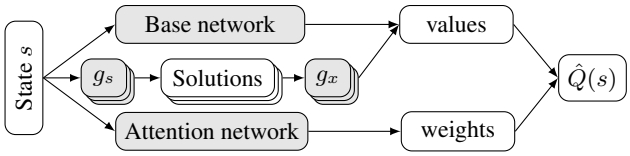


Figure 3: A2T network with state and action transformation.

$$\hat{Q}(s) = w_0(s)\hat{Q}_{\text{base}}(s) + \sum_{i=1}^{k} w_i(s)\hat{Q}_i(s) \tag{4}$$

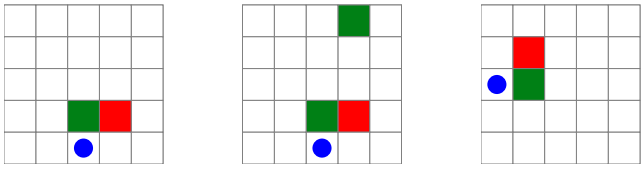
where $\hat{Q}_i$ comes from the ith source task, $\hat{Q}_{\text{base}}$ is learned from scratch and $w_i(s)$ is the ith attention weight, normalized so $\sum_{i=0}^{m} w_i(s) = 1$.

To handle substantially different system behaviors (as in the comparable systems setting), we propose a modification of A2T where we include a learned transformation of the state and action value function for each of the k previous solutions. If the state space $\mathcal{S} \subseteq \mathbb{R}^n$, then we define a *state transformation* as a function $g_s : \mathbb{R}^n \to \mathbb{R}^n$ and an *action value transformation* as a function $g_x : \mathbb{R}^m \to \mathbb{R}^m$. Applied to the A2T algorithm, the action value estimate with state and action space transformations is given by

$$\hat{Q}(s) = w_0(s)\hat{Q}_{\text{base}}(s) + \sum_{i=1}^{k} w_i(s) g_x^{(i)}(\hat{Q}_i(g_s^{(i)}(s))) \tag{5}$$

where $g_s^{(i)}$ and $g_x^{(i)}$ are the state and action value transformations for the ith source solution. In this work, we use linear transformations because they are effective and simple.

The A2T algorithm with state and action value transformations can be encoded as the network architecture shown in fig. 3. The state is used as input to the base network, the source solutions, and the attention network. The base network is a Q-network that learns from scratch. The k source solutions are the Q-networks that represent the optimal solutions of the source tasks. The source solutions are preceded by a state transformation and followed by an action value transformation. These transformations have the same output dimension as input dimension and are initialized to the identity transformation (with a small amount of noise for breaking symmetry). The attention network has $k+1$ output units with a softmax layer for normalization. The Q values of each source solution and the base network are weighted by the corresponding attention weights and summed together to get a final estimate. The network parameters from the base network, attention network, and state and action value transformations are trained using DQN. If the source solutions are



(a) Source MDP (b) Similar (c) Transformed

Figure 4: Knowledge transfer scenarios.

not differentiable with respect to the state then we would apply gradient free optimization (Kochenderfer and Wheeler 2019) to the state transformation.

Motivation. Here we provide some intuition for our choice of transfer learning algorithm and the reason for the state and action transformations. Suppose we are solving a sequence of tasks (fig. 4), each is a gridworld where an agent (blue circle) moves between adjacent squares attempting to achieve high reward (green squares) while avoiding states with low reward (red squares). Given the optimal action value function for a source MDP (fig. 4a), we wish to transfer it to two related tasks. In the first transfer problem (fig. 4b), the reward distribution is locally similar to the source MDP, and only differs in one state. In this case, a policy that works well in the source task will also work well in the new task, especially when the agent is in a state where the local reward landscape matches up (as depicted). A2T will work well in this setting because it can quickly learn attention weights that favor the source policy in most states, and only require our baseline solution when the agent is near the top right corner. In the second transfer problem (fig. 4c), A2T is likely to behave poorly because there are no states in which the source policy can be directly applied to achieve high reward. If, however, we could transform the state space by reflecting it across the diagonal and rotate the actions of the agent by $90°$, then we could directly apply the source policy. This is the motivation for applying transformations both before and after the source solutions.

Experiments

This section describes two safety validation problems: a gridworld scenario (GW) and an autonomous driving scenario (AD). Each scenario has two transfer learning problems, one for validating a learning system that improves over time, and the other for validating a set of comparable systems with different behaviors. We then describe the experimental setup and how we compute the evaluation metrics.

Gridworld with Adversary

The first safety validation problem we model is a gridworld with two agents, shown in fig. 5. The system is the agent in blue who is trying to arrive at one of the squares with positive reward while avoiding the adversary (orange agent). Both agents are initialized randomly and can move to any adjacent or diagonal state that is not a wall (marked as black). As a safety validation MDP, the state is the grid location of both agents and the disturbances are the actions of

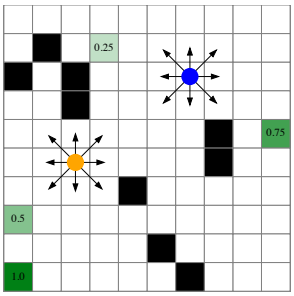


Figure 5: Gridworld with adversary scenario.

Figure 6: Autonomous driving scenario

the adversary: (up, down, left, right, up right, up left, down right, down left, stay). At each step, the blue agent chooses an action based on its policy and transitions to the appropriate state with probability 0.7 and transitions to a random feasible state with probability 0.3. Meanwhile, the orange agent transitions to the state specified by the disturbance x. The episode ends when the blue agent arrives at a square with positive reward or when the two agents collide in the same state, which is a failure of the system. We model each disturbance as equally likely to happen so we give a reward of 1 for finding a failure and 0 otherwise.

We design two sets of tasks that correspond to the learning system and comparable systems settings. For the learning system, the blue agent is trained using DQN against an orange agent that behaves randomly. Over 10^6 training steps, 10 versions of the system policy were stored, each with an increasing level of performance. Each safety validation task has the adversary validate an increasingly capable version of the learning system. For the comparable systems setting, each task has a different distribution of reward locations, reward values and location of walls. The system learns an optimal policy using dynamic programming (Kochenderfer 2015), assuming the adversary behaves randomly. Each system is therefore equally competent, but some configurations of the gridworld are more challenging than others.

Autonomous Vehicle

The second safety validation problem we model is an autonomous vehicle navigating an intersection with a crossing pedestrian, shown in fig. 6. The system is a vehicle controlled by the intelligent driver model (IDM) (Kesting, Treiber, and Helbing 2010), a rule-based driving policy that avoids collisions, and we add a blind spot with a specified direction and angular width. The vehicle tries to reach the end of the road while yielding to the pedestrian. Both agents are initialized randomly in a starting range of initial conditions.

As a safety validation MDP, the state is the position and velocity of the agents and the disturbances are accelerations

of the pedestrian. The pedestrian can accelerate up, down, left, or right by $1\,\mathrm{m/s^2}$ with probability 0.01 or have no acceleration with probability 0.96. The velocity of the pedestrian is limited to an absolute value of $3\,\mathrm{m/s}$. At each step, the vehicle chooses an acceleration based on the IDM rules and the location of the pedestrian. The pedestrian acceleration is specified by the choice of disturbance. Both agents have their position and velocity updated deterministically from their current state and acceleration. The episode terminates when the vehicle reaches the end of the road safely or a collision occurs between the pedestrian and the vehicle. The reward function is given by eq. (2) where the set E defined by any state where the vehicle and pedestrian overlap.

For the autonomous driving scenario, we also design two sets of tasks corresponding to a learning system setting and a comparable systems setting. Since the autonomous vehicle does not use machine learning, we simulate an improvement by progressively shrinking the blind spot of the vehicle. The blind spot remains in the same direction (20° from the horizontal), but reduces in width from 30° to 6° over 10 iterations. The vehicle therefore has a decreasing rate of failures over the tasks but there is some overlap in failure modes between adjacent tasks. For the comparable systems setting, each system has a blind spot sampled uniformly at random with a direction in the range $[-30^\circ, 30^\circ]$ and an angular width in the range $[3^\circ, 9^\circ]$. From a population of 30 tasks, we selected 9 tasks that differed substantially, to make the transfer problem as challenging as possible within our setting. Two tasks differed if the optimal safety validation policy of one performs poorly on the other.

Experimental Setup

The experimental procedure is as follows. For each task in a set of tasks, we solve for an optimal Q-network from scratch using DQN with prioritized replay, double Q-learning (Hasselt, Guez, and Silver 2016), and the Huber loss. We construct a learning curve during training by periodically storing the evaluation of the Q-network. For the second task onward, we then solve it using the same learning algorithm with the following Q-network architectures:

- **Fine-tune**: Train the last layer of the previous Q-network.
- **A2T**: A2T architecture with previous Q-networks as the source solutions.
- **A2T+SAVT**: A2T architecture augmented with linear state and action value transformations.

The networks are initialized with Xavier initialization (Glorot and Bengio 2010), while the transformations were initialized to the identity matrix with uniform random noise in the range [-1×10^{-3}, 1×10^{-3}] added to the parameters to break symmetry. Additional information on network architecture and hyperparameters is shown in table 1.

We filter each learning curve using a moving-average filter with a width of 20 evaluations steps to help remove the noise due to finite sample evaluation and any outlier evaluation points. For each learning curve we identify the *near-optimal* performance as $\mu - \sigma$, where μ and σ are the mean and standard deviation of the performance in a window with a width of 100 evaluation steps around the point of maximum performance. We use near-optimal performance because it is a more stable measure of how fast the learning took place than the point of maximum performance.

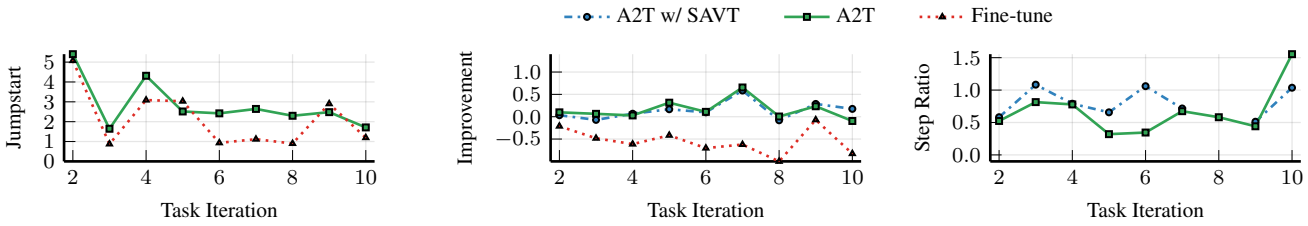
Parameter	**Value**
Base network	3 hidden layers, [64, 32, 16] relus
Attention network	1 hidden layer, 16 relus
Training steps	3×10^6
Batch size	64
Learning rate α	4×10^{-5} (GW), 5×10^{-5} (AD)
Target update frequency	2000 (GW), 3000 (AD)
Evaluation	300 episodes every 2000 steps
Exploration policy	ϵ-greedy with $\epsilon \in [1, 0.1]$

Table 1: Network architectures and hyperparameters.

The jumpstart is the difference in initial performance between a transfer and no-transfer learning algorithms. It can be computed from the first entries in the learning curves. When reporting jumpstart, we only include fine-tuning and A2T because A2T+SAVT has the same outputs as A2T until the transformations deviate from identity. The final performance is the difference in near-optimal performance between the transfer and no-transfer learning algorithms. The steps to threshold metric measures how many training steps are required for a transfer learning algorithm to reach the near-optimal performance of the no-transfer algorithm.

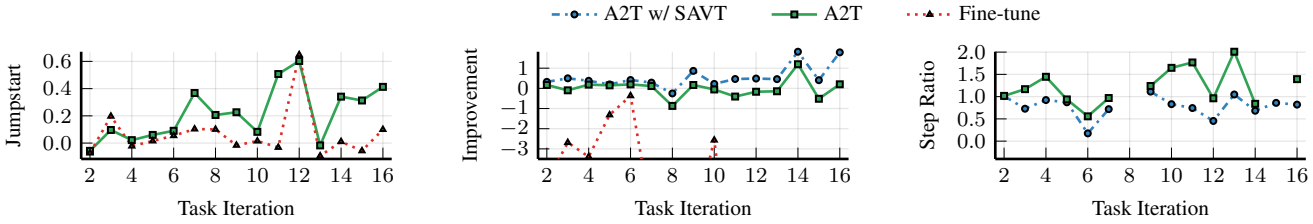
The metrics are normalized with reference to the learning curve of the no-transfer algorithm because the initial and near-optimal performance varies between tasks. Let y be the performance (initial or final) of a transfer learning algorithm and y_{ref} be the performance of the no-transfer learning algorithm, then we report the fractional difference in performance $(y - y_{\mathrm{ref}})/|y_{\mathrm{ref}}|$. Let t be the number of training steps required to reach a threshold for a transfer learning algorithm and t_{ref} be the same quantity for the no-transfer learning algorithm, then we report the ratio t/t_{ref}.

We use the number of training steps rather than the wall clock time because we assume that the cost of running the simulator is much larger than the cost of updating the parameters of the model. This is a good assumption for high-fidelity simulators that are often used for validating safety-critical systems. We also assume that the training time of previous source tasks is a sunk cost and it is not included in our efficiency metric. This assumption is valid in the case of iterative safety validation because the new version of the system must be validated regardless of the approach used. When using A2T in a real-world setting, we would not solve each task from scratch and therefore the source solutions would take the form of A2T networks. A practitioner may wish to compress the A2T network (Julian, Kochenderfer, and Owen 2019) into a traditional architecture before it is used as a source solution. We chose to use the networks trained from scratch for ease of implementation and to isolate the effects of transfer learning from other issues.



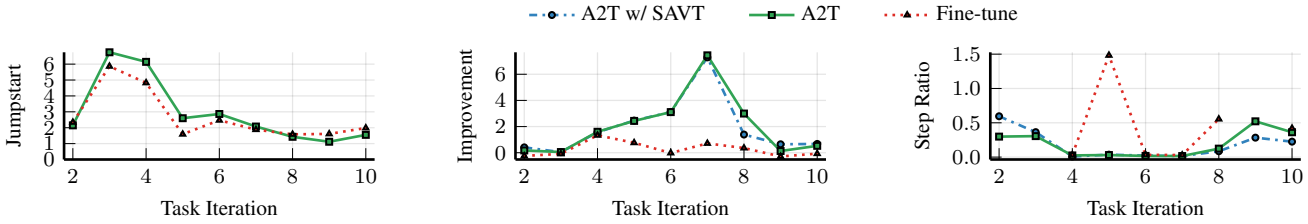
(a) Jumpstart improvement. (b) Final improvement. (c) Step ratio to threshold.

Figure 7: Evaluation metrics for the gridworld scenario with a learning system.



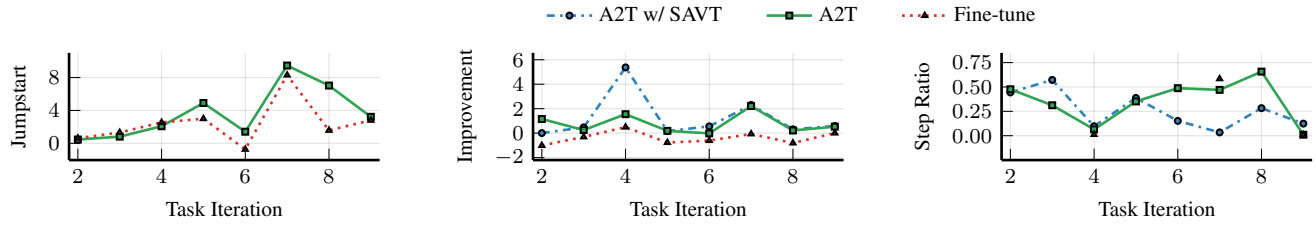
(a) Jumpstart improvement. (b) Final improvement. (c) Step ratio to threshold.

Figure 8: Evaluation metrics for the gridworld scenario with comparable systems.



(a) Jumpstart improvement. (b) Final improvement. (c) Step ratio to threshold.

Figure 9: Evaluation metrics for the autonomous driving scenario with a learning system.



(a) Jumpstart improvement. (b) Final improvement. (c) Step ratio to threshold.

Figure 10: Evaluation metrics for the autonomous driving scenario with comparable systems.

Results and Discussion

We solved each of the four safety validation tasks using 3 transfer learning algorithms and report the evaluation metrics against the task index in figs. 7 to 10. The discussion of the results is grouped by evaluation metric.

Jumpstart. Figures 7a, 8a, 9a and 10a show the jumpstart of the fine-tune and A2T architectures. Across all four safety validation problems, the transfer learning algorithms contributed a significant increase in initial performance. For most tasks, the A2T architecture had slightly better jumpstart than simply reusing the previous solution, especially in fig. 8a. The gridworld with comparable systems had substantially different failure modes between tasks and therefore had the least benefit in jumpstart. The safety validation problems with learning systems had the jumpstart decrease with the number of tasks observed, likely due to the increase in difficulty of the tasks. For the safety validation problems involving comparable systems, however, the jumpstart tended to increase with the number of source tasks. We hypothesize that with more source tasks, we are more likely to have a task that closely matches the current tasks, and can therefore immediately have reasonable performance.

Final Performance. Figures 7b, 8b, 9b and 10b show the final performance of each transfer learning algorithm. The final safety validation performance can be significantly improved by both A2T approaches, but not through fine-tuning. In all but fig. 9b, the fine-tuning approach was not able to match the no-transfer performance given the same number of iterations. The lack of performance could mean that the Q-network for one task is not learning a set of features that is useful for solving other tasks, so updating only the final layer does not provide enough capacity to solve the problem. The A2T networks, however, are able to achieve significantly improved final performance, which generally increases with the number of source tasks. The A2T network with state and action value transformations outperforms the basic A2T network in both safety validation problems with comparable systems, which are the problems it was designed for. Both of the safety validation problems with a learning system show the maximal gain in final performance in the middle of the sequence of tasks. A lower gain in early tasks may be due to those tasks being easy to solve, while a lower gain in the later tasks may be due to only having a few failure modes to exploit. The middle tasks may be challenging to solve but may have a diversity of failure modes that the previous policies can help identify. More experimentation is needed to fully understand these trends.

Steps to Threshold. Figures 7c, 8c, 9c and 10c show the number of training steps required to reach the near-optimal performance of the no-transfer algorithm. In some cases (and especially for the fine-tune approach), near-optimal performance is never reached so those data points are omitted from the plots. We observe that the number of training steps can be reduced by both A2T networks, but in different conditions. The basic A2T network performs well when validating a learning system because parts of previous solutions can be used directly. In fig. 7c, the number of training steps on some tasks could be reduced by 50% and in fig. 9c the number of training steps is reduced by more than an order of magnitude in some cases.

The A2T network with state and action transformations performs slightly worse than the basic A2T network in fig. 7c and has similar performance in fig. 9c, but significantly outperforms the A2T network for many tasks in the comparable systems setting, which is the setting it was designed for. In fig. 8c, the basic A2T network requires more steps than the no-transfer algorithm, which negates the utility of the more complex architecture, while the A2T+SAVT network was able to reduce the number of training steps by up to 50%. We note that generally, the fine-tune approach is unable to achieve the same performance as learning from scratch but when it does reach near-optimal performance, it requires fewer training steps than learning from scratch.

Summary. From our experiments we conclude that transfer learning can be an effective strategy for improving performance and efficiency of safety validation algorithms. Transfer through fine-tuning can give a significant increase in jumpstart but often fails to reach the level of performance of a Q-network trained from scratch. The A2T networks also provides an increase in jumpstart as well as an increase in final performance. The use of a small attention network allows for quick adaptation to new domains as evidenced by the reduction in the number of training steps required to reach near-optimal performance. When the tasks differ significantly from each other, however, the basic A2T network may take longer than the no-transfer algorithm to reach near-optimal performance. We fix this problem by introducing state and action value transformations for each source solution and demonstrate improved training efficiency over the no-transfer algorithm.

Conclusion

The validation of safety-critical autonomous systems is crucial for their safe deployment. Existing algorithms for validation often start from scratch each time the system changes. The nature of system design implies that safety validation will be performed iteratively on related systems, and should therefore benefit from past experience. We formulate iterative safety validation as a transfer learning problem and demonstrate improvements in both efficiency and performance of transfer learning algorithms compared to a no-transfer baseline. We augmented the attend, adapt, and transfer algorithm with state and action value transformations to allow for more transfer between disparate tasks. We evaluated jumpstart, final performance, and steps to threshold metrics on four iterative safety validation problems in gridworld and autonomous driving domains. Future work will include exploring the failure modes discovered by each algorithm to gain insights into how transfer is occurring. These insights may help us understand under what conditions we can expect performance and efficiency improvements.

References

Akazaki, T.; Liu, S.; Yamagata, Y.; Duan, Y.; and Hao, J. 2018. Falsification of Cyber-Physical Systems Using Deep Reinforcement Learning. In *International Symposium on Formal Methods (FM)*, 456–465. Springer International Publishing. ISBN 978-3-319-95582-7.

Corso, A.; Moss, R. J.; Koren, M.; Lee, R.; and Kochenderfer, M. J. 2020. A Survey of Algorithms for Black-Box Safety Validation. *arXiv e-prints* arXiv:2005.02979.

Glorot, X.; and Bengio, Y. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 249–256.

Hasselt, H. v.; Guez, A.; and Silver, D. 2016. Deep Reinforcement Learning with Double Q-Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2094–2100.

Huang, Z.; Lam, H.; LeBlanc, D. J.; and Zhao, D. 2017. Accelerated evaluation of automated vehicles using piecewise mixture models. *IEEE Transactions on Intelligent Transportation Systems* 19(9): 2845–2855.

Huber, P. J. 1992. Robust estimation of a location parameter. In *Breakthroughs in Statistics*, 492–518. Springer.

Julian, K. D.; Kochenderfer, M. J.; and Owen, M. P. 2019. Deep neural network compression for aircraft collision avoidance systems. *Journal of Guidance, Control, and Dynamics* 42(3): 598–608.

Kapinski, J.; Deshmukh, J. V.; Jin, X.; Ito, H.; and Butts, K. 2016. Simulation-Based Approaches for Verification of Embedded Control Systems: An Overview of Traditional and Advanced Modeling, Testing, and Verification Techniques. *IEEE Control Systems Magazine* 36(6): 45–64.

Kesting, A.; Treiber, M.; and Helbing, D. 2010. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 368(1928): 4585–4605.

Kim, Y.; and Kochenderfer, M. J. 2016. Improving aircraft collision risk estimation using the cross-entropy method. *Journal of Air Transportation* 24(2): 55–62.

Kochenderfer, M. J. 2015. *Decision Making Under Uncertainty: Theory and Application*. MIT Press.

Kochenderfer, M. J.; and Wheeler, T. A. 2019. *Algorithms for Optimization*. MIT Press.

Koren, M.; Alsaif, S.; Lee, R.; and Kochenderfer, M. J. 2018. Adaptive stress testing for autonomous vehicles. In *IEEE Intelligent Vehicles Symposium (IV)*, 1–7.

Lee, R.; Mengshoel, O. J.; Saksena, A.; Gardner, R. W.; Genin, D.; Silbermann, J.; Owen, M.; and Kochenderfer, M. J. 2020. Adaptive Stress Testing: Finding Likely Failure Events with Reinforcement Learning. *Journal of Artificial Intelligence Research* 69: 1165–1201.

Mathesen, L.; Yaghoubi, S.; Pedrielli, G.; and Fainekos, G. 2019. Falsification of cyber-physical systems with robustness uncertainty quantification through stochastic optimization with adaptive restart. In *International Conference on Automation Science and Engineering (CASE)*, 991–997.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540): 529–533.

Rajendran, J.; Lakshminarayanan, A. S.; Khapra, M. M.; Prasanna, P.; and Ravindran, B. 2017. Attend, Adapt and Transfer: Attentive Deep Architecture for Adaptive Transfer from multiple sources in the same domain. In *International Conference on Learning Representations*.

Schaul, T.; Quan, J.; Antonoglou, I.; and Silver, D. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations*.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press.

Taylor, M. E.; Kuhlmann, G.; and Stone, P. 2008. Autonomous transfer for reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 283–290.

Taylor, M. E.; and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10: 1633–1685.

Taylor, M. E.; Stone, P.; and Liu, Y. 2007. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research* 8: 2125–2167.

Uesato, J.; Kumar, A.; Szepesvári, C.; Erez, T.; Ruderman, A.; Anderson, K.; Dvijotham, K. D.; Heess, N.; and Kohli, P. 2019. Rigorous Agent Evaluation: An Adversarial Approach to Uncover Catastrophic Failures. In *International Conference on Learning Representations*.

Wang, X.; Nair, S.; and Althoff, M. 2020. Falsification-Based Robust Adversarial Reinforcement Learning. *arXiv e-prints* arXiv:2007.00691.

Zutshi, A.; Deshmukh, J. V.; Sankaranarayanan, S.; and Kapinski, J. 2014. Multiple shooting, CEGAR-based falsification for hybrid systems. In *International Conference on Embedded Software (ICESS)*, 1–10.