Supplementary Material

Paper ID: 3365

A Victim Population Settings

Collective learning frameworks where agents train on copies of the same environment fall under three lines of research [Parisotto *et al.*, 2019; Marthi *et al.*, 2005; Dimakopoulou *et al.*, 2018; Lupu and Precup, 2020; Yang *et al.*, 2020]. The first line aims to decrease the complexity of solving large Markov decision processes that model sizeable multi-agent systems [Marthi *et al.*, 2005]. They do so by breaking down the problem into tasks that are executed in parallel. Each task houses one or more agents and coordinates at run-time in order to push the agent population towards optimal behavior. The second line of research enables a single agent to better explore the given environment in a more efficient manner, by interacting with it in parallel using different policies [Dimakopoulou *et al.*, 2018]. In the third line of research, the agent aims to learn how to learn and strives to be able to perform efficiently across a family of tasks (Meta RL) [Parisotto *et al.*, 2019]. Herein, the agent trains on different tasks, in parallel. In all three domains, the agents exchange information in order to learn better strategies more quickly. [Yaman *et al.*, 2022] points out that an optimal balance between individual and social learning is critical for learning optimal behaviors in a population. Herein, individual learning is when each agent explores, interacts, and learns from the environment separately while social learning is when agents copy each other's behaviors. Therefore, individual learning leads to innovations which can then efficiently spread through the population via social learning. However, individual learning incurs high exploration cost. On the other hand, social learning is cost-effective but requires construction of a strategy by which the population chooses which behaviors (successful vs majority) to copy. The optimality of these strategies, akin to meta-control strategies in the human brain [Daw *et al.*, 2005], are sensitive to environment variability. In environments that change frequently, social learning might hamper the population's learning as the exchanged information can become invalid very quickly. This work proposes three learning settings namely; Implicit Collective, Collective, and Swarm Collective; wherein individual learning decreases while social learning increases progressively. In these settings, each agent trains to learn its individual task with the support of a separate reward signal while the attacker trains to push the complete (victim) populations towards a target behavior.

In the Implicit Collective setting, the innovation capability of the population is maximized via individual learning (at the cost of exploration), as agents practice Decentralized Training, Decentralized Execution (DTDE). This is achieved as a collection of learning agents individually experience a commonly parameterized environment, and interact only implicitly via observations of environment modifications. The environment observations serve as a medium of implicit interaction as the attacker takes a single attack action to modify/poison the blueprint of the victim environment, conditioned on the behaviors of all agents present in the population. Agents, therefore, have some influence on each other, since a failure or stubbornness of one of them has an effect on the next attack presented to the entire population. Drawing inspiration from [Dimakopoulou *et al.*, 2018], exploration is made more efficient during testing by assigning an independent, random number generation seed to each agent in the victim population. This increases diversity as well as commitment of the victim agents. The Implicit Collective setting can find application in the development of an adaptive single-player computer game engine wherein the adaptive game engine represents an attacker that strives to simultaneously push a complete population of victims (players) towards a target behavior (with maximum player stickiness) and each attack action represents a new release/version of the game.

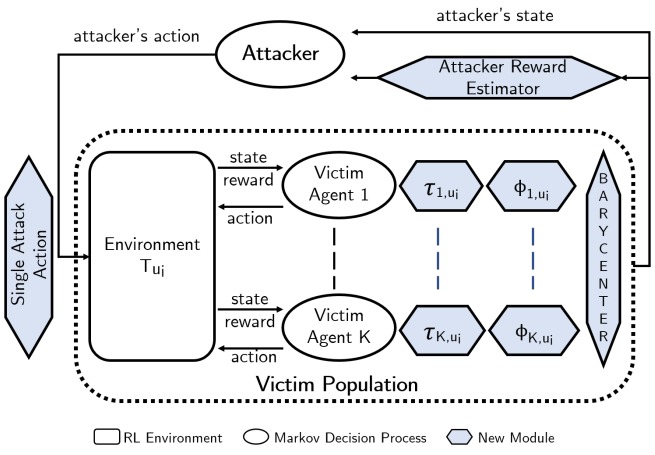

Figure 1: Bi-Level Attack Framework in Collective and Swarm Collective Settings

In Collective and Swarm Collective settings the agents occupy the same copy of the victim environment. Herein, each agent observes (anonymized version of) every other agent, and takes actions conditioned on this observation (along with its own position inside the victim environment). The agents practice Centralized Training, Decentralized Execution (CTDE) where a single network is trained using all victims' interactions. As the agents learn from each other's experiences, these settings support social learning. In Collective setting, individual learning is enhanced by inculcation of "agent indication" i.e. addition of an indication of the observing agent to its observations [Terry *et al.*, 2020]. This enables the same neural network to represent diverse victim behaviors. In addition, each agent is committed to a separate, diverse behavior by assigning it an independent, random number generation seed [Dimakopoulou *et al.*, 2018]. Collective setting therefore presents a balanced scenario that houses high support for both individual and social learning. On the other hand, Swarm Collective does not inculcate agent indication and independent random seeds, and therefore strongly promotes social learning. The centrally trained neural network in this setting learns a single optimal victim behavior using all victims' interactions. Lastly, a soft competition is injected into both settings by terminating victim-training episodes as soon as at least one victim reaches the goal state (or maximum training time has elapsed); to enable faster adoption of optimal behaviors inside the victim population.

B (Expanded) Related Work

This appendix helps to further position the current paper in terms of the proposed methodology as well as the overall framework through comparison with literature in related domains of Group-Behavior Modelling, Set Representation Learning, and Unsupervised Environment Design respectively.

B.1 Group-Behavior Modelling

The existing research works that strive to extract or model the behavior of a group of agents typically follow two approaches. In the first approach, a given agent strives to inculcate the effect of the behavior of other agents in its own learning, by directly feeding the other agents' observations, rewards, and/or behavior trajectories ((state, action, next action) tuples) into its own quality, value and/or policy network(s) [Zhang and Lesser, 2010; Foerster *et al.*, 2018]. In the second approach, a given agent first learns the single-agent behavior models of all the "other" agents [Raileanu *et al.*, 2018; Papoudakis and Albrecht, 2020; Rabinowitz *et al.*, 2018; Grover *et al.*, 2018; Tacchetti *et al.*, 2019; Grover *et al.*, 2018; Shum *et al.*, 2019; He *et al.*, 2016; Papoudakis *et al.*, 2021] and then utilizes the predictions of these models in its own decision making. To the best of our knowledge, this is the first work that creates a third category of group-behavior modeling wherein the complete group's behavior is modeled together to create a single latent representation that captures the distribution of behaviors present inside the agent population.

B.2 Set Representation Learning

Set-based machine learning tasks have recently come into focus, prompting several research works to propose deep modeling solutions to permutation-invariant collections of data. PointNet [Qi *et al.*, 2017] and Deep Sets [Zaheer *et al.*, 2017] learn a representation of an input set by first feeding each element of this set individually into a neural network and then aggregating the network outputs by using max-pooling (PointNet) and summation (Deep Sets) respectively. Rep the Set [Skianis *et al.*, 2020] computes the similarity of a given input set, to a pre-defined number (say k) of learnable reference sets and uses these similarity values (vector of length k) as the representation of the input set. Set Transformer [Lee *et al.*, 2019] is a permutation-invariant attention-based neural network developed by removing the positional encoding present in the original transformer and inculcating latent variables to reduce computational complexity. All these models are completely flexible wrt the size of the input set except Set Transformer which works with all set sizes smaller or equal to a pre-decided maximum. However, unlike the approach proposed in this paper, the aforementioned models do not support unsupervised learning. Optimal Transport Kernel Embedding (OTKE) [Mialon *et al.*, 2020] is the first set-based representation learning model that supports unsupervised learning. OTKE constructs a reference set, of a pre-defined length (say k), consisting of "k" cluster centroids or "k" Wasserstein barycenters of the input data. Thereafter, OTKE uses the Gaussian kernel to map all elements of the input and reference set to a non-linear space. Finally, the input set's representation constitutes a set of k-weighted sums of the input set elements wherein the j^{th} sum's weights are the similarity scores between all input elements and reference element j. The similarity score between the two elements is given by the transport map between them. OTKE is therefore a soft clustering technique based on the Wasserstein metric. The approach proposed in this work can be seen as a simplification of OTKE as it treats the individual behaviors of victims as a single behavior cluster. This simplified approach is justified as the victim populations considered in this work consist of homogeneous agents. Future works on attacks on heterogeneous victim populations can employ OTKE for constructing population behavior representations.

B.3 Unsupervised Environment Design

Design of RL environments takes a lot of time and effort, is error-prone, and is infused with designer bias. Unsupervised Environment Design (UED) is a recent paradigm that aims to not only automate this step but generate environment distributions that are conducive to emergent complexity, robustness, and efficient transfer learning in RL agents. Domain Randomization [Tobin *et al.*, 2017] creates a distribution of environments by assigning random values to the free parameters of the environment. Even though it exposes agents to a wide variety of environments, it does not create environments with complex structures and is not reactive to the capabilities of the learning agent. Minimax Adversarial Training [Morimoto and Doya, 2005] makes environment distribution generation reactive to the learning agent's capabilities by introducing an adversary that sequentially creates environments

that minimize the learning agent's rewards. This worst-case tendency has however shown to create unsolvable environments that hinder the learning agent's progress. PAIRED [Dennis *et al.*, 2020] does away with this worst-case tendency by introducing an additional "antagonist" agent which is smarter than the learning agent (protagonist). The adversary's objective is to maximize regret i.e. the difference between these two agents' rewards. This enables the generation of challenging but solvable environments. However, training the environment-creating adversary is challenging and is shown to produce weaker results than when main agents are trained on randomly-selected high-regret environments [Jiang *et al.*, 2021a]. This difficulty in training the adversary is due to sparse rewards and long-horizon credit assignment problems. Inspired by the Teacher-Student Curriculum Learning paradigm [Matiisen *et al.*, 2019], Prioritized Level Replay (PLR) [Jiang *et al.*, 2021b], and PLR+ [Jiang *et al.*, 2021a] do away with adversary-based mechanism and instead strategically sample the next environment by prioritizing environments that have larger future learning potential. However, they cannot create new challenging environments with complex structures. This limits the frequency with which the learning agent can be exposed to complex structures and can thus hinder its progress, especially in high-dimensional complex environments. ACCEL [Parker-Holder *et al.*, 2022] does away with both, adversary-training-based and sampling-based methodologies' drawbacks by utilizing an evolutionary environment generator and a regret-based curator. The generator makes small modifications (mutations) to high-regret environments present inside the replay buffer. These modified environments are added to the replay buffer if they result in high regret. Regret thus functions as the fitness function for evolution and helps develop challenging environments for the learning agent.

Training-time environment-poisoning attacks can be compared to hypothetical negative-UED methodologies whose aim could be automatic design of environment distributions that result in minimization of learning agent's (victim's) performance. However, a straightforward negative of existing UED methodologies is not equivalent to environment-poisoning attacks. First of all, adversary-based and evolution-based UED create challenging environments by aiming to minimize learning agent's rewards, while sample-based UED sample environments with high learning potential. All these mechanisms serve to improve the learning agent's performance. A straightforward opposite of these approaches will only lead to reduced victim rewards. Environment-poisoning attacks on the other hand strive to push the victim to adopt an attacker-desired target behavior employing minimal attacker effort, while the victim trains to optimize its own objective. Moreover, unlike UED, in environment-poisoning attacks, the sequence of modifications is critical. While the victim begins training in its environment, the attacker begins to periodically modify the victim's environment. Since the attacker intends to push the victim towards a target behavior that the victim has a low tendency of adopting by itself, in the original environment; the attacker must strategize modifications by carefully considering the current behaviors being learned by the victim. Moreover, the attack paradigm imposes constraints on the attacker in terms of the permissible magnitude of environment changes allowed at a single step. Any sizeable modification to the environment can therefore only be implemented through a sequence of small changes. The attack's sensitivity to current victim behaviors as well as magnitude constraints render the sequence of environment changes critical in the attack domain.

C Attacker's Reward

The aforementioned Implicit Collective, Collective and Swarm Collective settings enable the victim populations to practice different levels of individual vs social learning. To show that environment-poisoning attacks can be effective under different levels of uncertainty, Implicit Collective setting is set as a Blackbox setting for the adversary while Collective and Swarm Collective settings are set as Ultra-Blackbox settings. In both Blackbox and Ultra-Blackbox, the attacker must extract individual victim behaviors from across-policy, on-process behavior traces, and additionally, in Ultra-Blackbox setting, the attacker is also not supported with any extrinsic reward signal, during its training. The attacker must therefore construct an intrinsic reward and estimate the efficacy of its own attack strategy by observing the change in the victim population's behavior after each attack step, i.e. the victim population's behavior adaptation in response to the attacker's sequential attack.

This work extends the reward space design of [Xu *et al.*, 2021] to the multi-victim setting by treating the underlying environment dynamics coupled with each population member's behavior as a stochastic Markov process over state-action pairs $P_{k,u_i}(s_{j+1}, a_{j+1}|s_j, a_j)$. Herein, j is used to denote victim-level time step, just as i has been used to denote attacker-level time step. Construction of this victim process enables computation of the joint probability distribution of current environment dynamics and the current victim behavior which is critical as environment dynamics can impede certain victim tendencies while facilitating others. Kullback Leibler Divergence Rate, D_{klr} [Rached *et al.*, 2004]) between the ideal (in accordance with the attacker's objectives) and the current process of each member of the victim population is computed and their negative mean is taken as the attacker's reward for the given time-step. Herein the ideal process $P^*_{k,u_0}(s_{j+1}, a_{j+1}|s_j, a_j)$ is one where victim k adopts the attacker-desired behavior, τ^*_{k,u_i} aka the target behavior, without requiring any modifications to the underlying environment dynamics T_{u_0}. In this work, the target behavior is partial in nature:

$$\tau^*_{k,u_i}(s_n) = \begin{cases} a^*_n & s_n \in S^* \\ \tau_{k,u_i}(s_n) & s_n \notin S^* \end{cases} \quad (1)$$

Here S^* is the target state set, a^*_n is the target action for target state s_n, and $\tau_{k,u_i}(s_n)$ is the observed behavior of victim k as it trains in environment with transition dynamics T_{u_i}. τ^*_{k,u_i} is the partial target behavior of victim k. The partiality of this design allows the attacker to specify target behavior for only those states that it cares about. This design, unlike the complete target behavior design proposed in [Rabinovich *et*

al., 2010], is efficient in high-dimensional discrete and continuous state spaces.

The current process for victim k is defined using the current underlying (modified) dynamics T_{u_i} as well as victim k's behavior after training in this modified environment τ_{k,u_i}. In the Blackbox setting, the attacker is provided with an extrinsic reward signal that can access a victim k's Q function to compute its policy π_{k,u_i}; while in the Ultra-Blackbox setting, the attacker constructs its own intrinsic reward signal using the behavior traces of victim k. Herein, the attacker approximates π_{k,u_i} as a distribution using the last h actions taken by victim k in each state. The current process P_{k,u_i} of victim k in terms of current environment dynamics and victim k's current policy is defined a:

$$P_{k,u_i}(s_{j+1}, a_{j+1}|s_j, a_j) = T_{u_i}(s_{j+1}|s_j, a_j)\, \pi_{k,u_i}(a_{j+1}|s_{j+1}) \tag{2}$$

The target policy π^*_{k,u_i} is constructed by taking a copy of π_{k,u_i} and modifying it by assigning probability 1.0 to all target actions (and 0.0 to non-target actions) w.r.t. each target state. The ideal process P^*_{k,u_0} of victim k in terms of default dynamics and target policy is defined as:

$$P^*_{k,u_0}(s_{j+1}, a_{j+1}|s_j, a_j) = T_{u_0}(s_{j+1}|s_j, a_j)\, \pi^*_{k,u_i}(a_{j+1}|s_{j+1}) \tag{3}$$

The reward measuring the current performance of the attack strategy is then defined as negative of the average Kullback Leibler Divergence Rate between the ideal process and the current processes of all agents in the victim population. In equation 4 given below q_i is the stationary state distribution of the victim environment with dynamics T_{u_i}.

$$\begin{aligned}
r_{\alpha,i} &= -(1/K) \sum_k D_{klr}(P^*_{k,u_0}||P_{k,u_i}) \\
D_{klr}(P^*_{k,u_0}||P_{k,u_i}) &= \sum_j q_i(s_j)\pi_{k,u_i}(a_j|s_j) D^{kl}_i(s_j, a_j) \\
D^{kl}_i(s_j, a_j) &= \sum_{s_{j+1}, a_{j+1}} \Bigg(P_{k,u_i}(s_{j+1}, a_{j+1}|s_j, a_j) \\
&\qquad log \frac{P_{k,u_i}(s_{j+1}, a_{j+1}|s_j, a_j)}{P^*_{k,u_0}(s_{j+1}, a_{j+1}|s_j, a_j)} \Bigg) \\
q_i(s_{j+1}) &= \sum_j q_i(s_j)\pi_{k,u_i}(a_j|s_j) T_{u_i}(s_{j+1}|s_j, a_j)
\end{aligned} \tag{4}$$

D Experimental Setting - Victim Population

Each member of the victim population utilises independent Q Learning under Implicit Collective setting and DQN (Deep-Q Networks) under Collective and Swarm Collective settings. Under Implicit Collective setting, in each attack step, the victim population trains for 80 episodes using Q learning with discount factor $\gamma = 0.90$, and learning rate $\alpha = 0.100$. On the other hand, under Collective and Swarm Collective settings, the victim population trains for 40 episodes in each attack step, using DQN with discount factor $\gamma = 0.99$, and learning rate $\alpha = 0.001$. These values were chosen such that the victim population converges to optimal behavior (w.r.t victim population's objectives) in the default (unattacked) environment. Furthermore, it is important to note that the state-of-the-art single-agent environment poisoning work, TEPA [Xu *et al.*, 2021] whose multi-victim extension is constructed and utilized as baseline in this paper, tested their attack against an ϵ-Greedy victim that is heavily biased towards exploitation from the beginning of the training period. However, we test the barycenter-based attack against more-realistic SoftMax victims that spend substantial time exploring the environment when they begin their training.

E Experimental Setting - Attacker

The attacker in the baseline as well as proposed methodology is trained using Deep Deterministic Policy Gradient (DDPG) with discount factor $\gamma = 0.95$, and target network update rate $\tau = 0.005$. TEPA attacker collects data for 100 attack episodes before beginning to train while we reduce the initial data collection period to 30 attack episodes in the barycenter-based attacker. The policy function of the attacker is a fully connected, feedforward neural network with specification: INPUT(21)-FC(400)-ReLU-FC(300)-ReLU-FC(16)-Tanh.

TEPA attacker uses an auto-encoder to capture latest victim behaviors. Its encoder is a fully connected, feedforward neural network with specification: INPUT(12)-FC(36)-ReLU-FC(36)-ReLU-FC(5). The proposed barycenter-based attacker on the other hand utilises a variational auto-encoder (VAE) whose encoder is a fully connected, feedforward neural network with specification: INPUT(32)-FC(36)-ReLU-FC(36)-ReLU-FC(5). The learning rates of the two models are 0.001 and 0.00001 respectively. It is important to note here that unlike the pre-trained VAE employed in the proposed methodology, the auto-encoder in TEPA trains along with the attacker in order to ensure that the latest victim behaviors are mapped to the latent space with high accuracy. To enable evaluation of TEPA-based concatenation strategies, the auto-encoder parameters are saved during training, after every 20 attack episodes. At the time of evaluation of a particular strategy, the auto-encoder parameters saved closest to the given strategy are used by the attacker.

F Performance Metrics

The performance of the attacker is measured in terms of the accuracy of the victim population's adoption of the target behavior (Attack Accuracy and Attack SoftMax Accuracy) as well as the cumulative changes brought about in the victim environment by the attacker (Attacker Effort).

Attack Accuracy computes the level of adoption of the target behavior by the victim population in the given environment. Given that π_{k,u_i} is victim k's probabilistic policy in environment with dynamics T_{u_i}, K is the number of agents in the victim population and S^* is the set of target states, let N^* be the number of target states, a^*_n be the target action in target state s_n, and $f_a(s, \pi_{k,u_i})$ be a function that takes a target state and a victim policy as input and outputs 1 if the given

policy assigns highest probability to the target action in that state and 0 otherwise.

$$f_a(s_n, \pi_{k,u_i}) = \begin{cases} 1 & \pi_{k,u_i}(a_n^*|s_n) > \pi_{k,u_i}(a_n|s_n) \; \forall \; a_n \\ 0 & otherwise \end{cases} \tag{5}$$

$$\text{Attack Accuracy} = \frac{1}{K} \sum_k \left(\frac{1}{N^*} \sum_{s_n \in S^*} f_a(s_n, \pi_{k,u_i}) \right) \tag{6}$$

Attack SoftMax Accuracy computes the strength with which the target behavior is adopted by the victim population by calculating the probability assigned to the target path by the victim population.

$$\text{Attack SoftMax Accuracy} = \frac{1}{K} \sum_k \left(\frac{1}{N^*} \sum_{s_n \in S^*} \pi_{k,u_i}(a_n^*|s_n)) \right) \tag{7}$$

Attacker Effort computes the degree to which the attacker modifies the victim environment. Let $G = [g_{u_i}^1, g_{u_i}^2, g_{u_i}^3, ..., g_{u_i}^M]$ be the altitudes of the M grid cells in victim environment with dynamics T_{u_i}.

$$\text{Attacker Effort} = \frac{1}{M} \sum_{m=1}^{M} \mid \; g_{u_i}^m - g_{u_{i-1}}^m \mid \tag{8}$$

G Attack Strategy Selection

The attacker training episodes are 15-step sequential attacks on freshly initialized victim populations. Each episode begins with a slightly perturbed environment so that the attacker's reward that aims to minimize distance between current and default environment (along with minimizing distance between current and target population behavior) does not get stuck on the default environment without ensuring victim population's adoption of target behavior. The new population begins training to maximize its objectives on this slightly perturbed environment. At attack time step 1, the attacker makes its first modification of the victim environment. After each episode, the attack strategy employed in that episode is saved if it is better or equal to the best attack strategy found so far, with respect to last-timestep, mean or cumulative value of at least one strategy quality criterion. A given strategy's quality is approximated using 3 internal and 5 external quality criteria. Herein criteria that are approximated by the attacker are referred to as internal while criteria computed by the external system for the purpose of training the attacker are termed external. The 8 quality criteria are:

1. KLR-Full (Internal): KLR between current (current env * current victim behavior) and perfect process (default env * target victim behavior)
2. KLR-Environment (Internal): KLR between (current env * target victim behavior) and perfect process (default env * target victim behavior)
3. KLR-Behavior (Internal): KLR between (default env * current victim behavior) and perfect process (default env * target victim behavior)
4. Attack Accuracy (External): same as Attack Accuracy performance metric
5. Attack Softmax Accuracy (External): same as Attack SoftMax Accuracy performance metric
6. Attack Partial-Softmax Accuracy (External): unlike Attack SoftMax Accuracy where probability of attacker-desired actions in all attacker-desired states are added, here probability of attacker-desired actions in only those attacker-desired states are added where Attack Accuracy is 1.0 i.e. only probability of those attacker-desired actions are taken into account which are already assigned maximum probability by the victim. This quality criterion enables the attacker to identify (and thereby save) strategies that are capable of inducing strong adoption of target behavior but were not able to achieve this in all target states in the given trial.
7. Attacker Effort (External): same as Attacker Effort performance metric
8. Attack Time (External): computational time corresponding to each attack step (inclusive of time taken by the victims to train in the attacked/poisoned/modified environment).

The experiment graphs present in the main body of the paper demonstrate performance of the best attack strategies found by the different models. These best strategies are selected by prioritising Attack Accuracy, as the main goal of this work is to find strategies that push victim populations to adopt the target behavior. The strength of target behavior adoption (Attack Softmax Accuracy) and amount of changes made to the environment (Attacker Effort) in order to achieve this, demonstrate additional capabilities of the attack strategies. The selection procedure begins with a filtering process that shortlists only those strategies that acquire Attack Accuracy of 1.0 by the end of the attack. The shortlisted strategies are then arranged in decreasing order of mean Attack Accuracy. Thereafter, the top 20 + j strategies are selected where all strategies between ranks 20 and 20+j have the same mean Attack Accuracy. In case less than 20 strategies acquire Attack Accuracy 1.0 by the end of the attack, then top 20+j strategies are selected solely on the basis of last-step Attack Accuracy. Lastly, each of these 20+j strategies is utilised to attack 10 same-sized populations that utilize the same random seed (Implicit Collective) or utilize neural networks initialized with random numbers from the same range (Collective and Swarm Collective); as used during training.

H Experimental Data (Expanded)

This appendix includes results corresponding to experiments under the Swarm Collective setting as well as additional results corresponding to experiments under Implicit Collective and Collective settings.

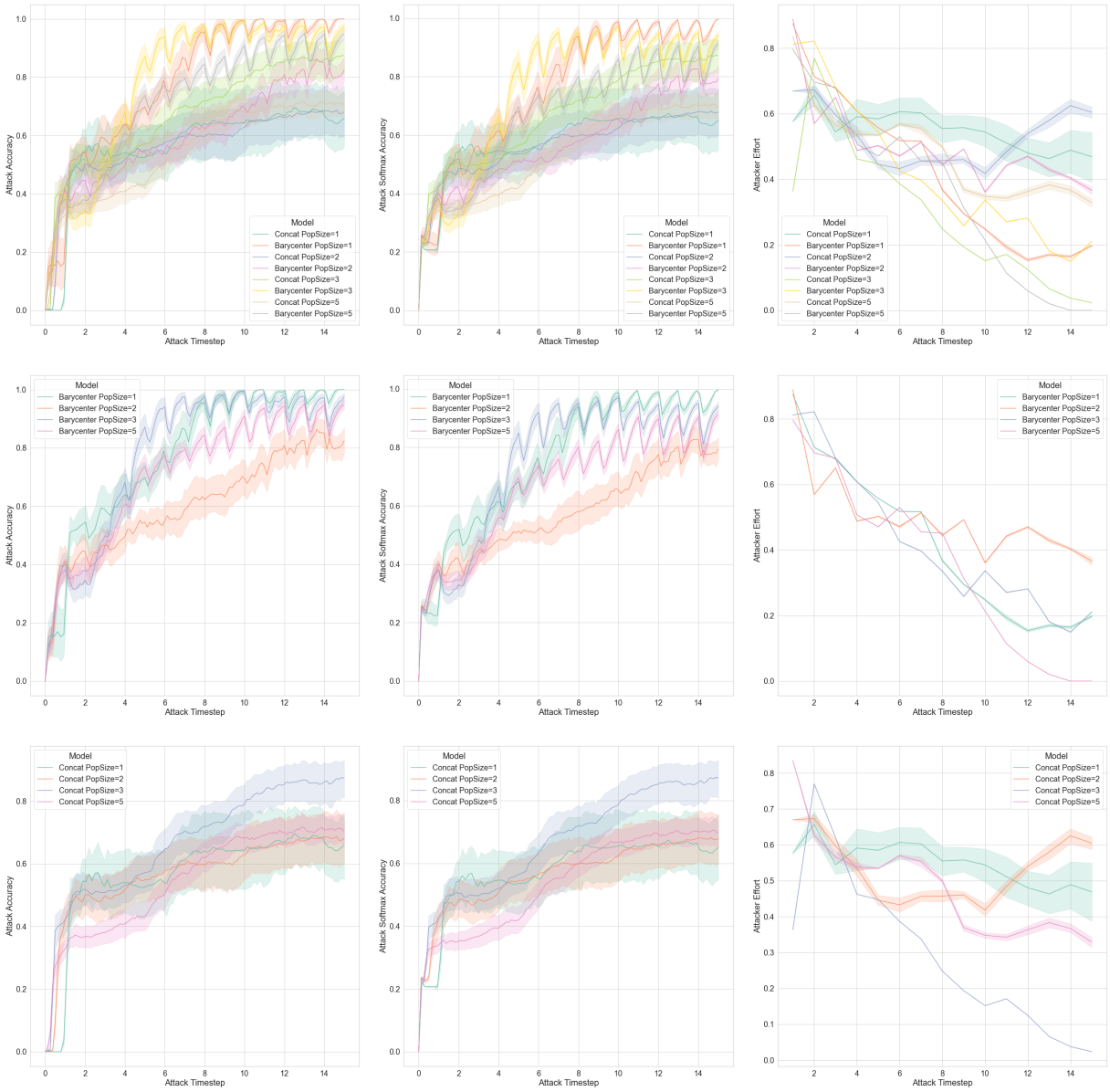

Figure 2: Accuracy, Softmax Accuracy and Effort of attack, trained and tested on same sized Swarm Collective victim

H.1 Swarm Collective Setting

Experiment A (see Figure 2) tests the capabilities of the proposed and baseline methods to attack Swarm Collectives of sizes 1, 2, 3 and 5. In this setting, majority of both concatenation and barycenter based strategies achieve better accuracies while exerting higher effort, compared to their Collective setting counterparts. All baseline strategies except the strategy trained on size-3 populations, achieve Attack Accuracy and Attack SoftMax Accuracy between 0.6 and 0.8, in contrast to the Collective setting, wherein they achieve both accuracies between 0.4 and 0.6 by the end of the attack. Size-3 baseline strategy achieves 0.9 Attack *and* Attack-SoftMax accuracy by the end of the attack in both Collective and Swarm Collective settings. Higher accuracies for Swarm Collectives are accompanied with higher Attacker Effort as all strategies except size-3 strategy exert effort between 0.3 and 0.6, in contrast to Collective setting wherein they reduce Attacker Effort to below 0.2 by the end of the attack. Size-3 strategy in both Collective and Swarm Collective settings converge towards 0.0 Attacker Effort by the end of the attack. This shows that the baseline method that uses concatenation of trajectory-based individual victims' behaviors to capture the population behavior is less capable of accurately understanding population behavior of populations of sizes $\neq 3$. Barycenter-based strategies that achieve high Attack Accuracy of above 0.8 and high Attack Softmax Accuracy of above 0.7 in Collective setting achieve even higher accuracies for Swarm Collectives with majority reaching above 0.9 by the end of the attack. This performance boost comes with slightly higher Attacker Effort which is between 0.2 and 0.4 for Swarm Collectives and between 0.0 and 0.4 in the Collective setting. Lastly, like Collective setting, barycenter-based attacks exhibit the climbing zig-zag behavior in terms of Attack and Attack-Softmax Accuracies for Swarm Collectives.

Experiment B, against Swarm Collectives, is presented in Figures 3 and 6. Figure 3 presents strategies trained on populations of sizes 3, 5, 10, and 20, and tested on populations of sizes 3, 5, 10, and 20 in the same graph in order to demonstrate that the proposed barycenter-based method showcases better size-agnosticity verus Swarm Collectives, than the Collective setting. On the other hand, figure 6 presents separate graphs for strategies trained on populations of sizes 3, 5, 10, and 20 respectively. Each strategy is tested on populations of sizes 1, 2, 3, 4, 5, 10, and 20 to facilitate understanding of how a particular strategy's performance changes with the test-population size. Last-step Attack Accuracy of size-3 strategy (strategy trained on size-3 populations) is highest when tested on size-3 populations, slightly lower for size 1 and 2 populations, and progressively lower for larger populations. Last-step Attack Accuracy of all tests carried using size-5 strategy is higher than their size-3 counterparts and show the same trend of decreasing with increasing population size. The climbing zig-zag behavior of size-10 strategy degrades with decreasing test-population size starting with size-5 test populations. Moreover, last-step Attack Accuracy of almost all tests carried using size-10 strategy is lower than their size-3 and size-5 counterparts. The climbing zig-zag behavior of tests carried out using size-20 strategy degrades further than their size-10 counterparts, on small populations (1-5). However, performance of size-20 strategy on tests carried out on large populations (10,20) is better than the performance of all other strategies. These results imply that strategies trained on smaller populations and tested on larger populations perfectly retain the climbing zig-zag behavior but this behavior slightly degrades for strategies trained on larger and tested on smaller populations. Attack SoftMax Accuracy shows similar trends to Attack Accuracy while Attacker Effort remains largely unchanged across tests on different sized test populations. Lastly, even while taking into account all the variations discussed above, all strategies showcase high level of size-agnosticity as all performance plots follow the same trend and remain within a certain range to the plot corresponding to attacks trained and tested on the same-sized populations.

H.2 Implicit Collective and Collective Settings

Figures 5 and 4 present Experiment B against Collectives and Implicit Collectives, respectively; for all strategies tested on populations of sizes 1-5, 10, and 20 instead of testing only on 3, 5, 10 and 20 as done in the main paper. Lesser populations were shown in the main paper to enable readers to catch overall trends quickly while more extensive results are presented here for completeness. Figure 4 shows that in contrast to other barycenter-based strategies that converge within 2 attack steps, size-1 strategy only converges by the end of the attack in Implicit Collective setting. However, even then, size-1 strategy is size-agnostic and achieves last-step Attack Accuracy of 1.0 when used to attack populations of sizes 1-5, 10, and 20. Other than this result, figures 4 and 5 are similar to their "clearer" counterparts presented in the main paper.

Figure 7 presents separate graphs for size 3, 5, 10, and 20 strategies, respectively. Each strategy is tested on populations of sizes 1-5, 10, and 20 to facilitate understanding of how a particular strategy's performance changes with test-population size. Last-step Attack Accuracy of size-3 strategy mostly decreases with increasing test-population size but stays above 0.6 until size-10 test populations and drops to 0.3 for size-20 test-populations. Last-step Attack Accuracy of size-5 strategy reaches 1.0 for small populations (1-5); above 0.8 for size-10 test-populations; and above 0.4 for size-20 test-populations. For size-10 strategy, last-step Attack Accuracy of small populations (1-5) drops to between 0.8 and 0.9; stays the same for size-10 test-populations and climbs to 0.6 for size-20 test-populations. Lastly, last-step Attack Accuracy of size-20 strategy climbs to above 0.9 for small and medium populations (1-10) and to 0.8 for size-20 large test-populations. Also, Attack and Attack SoftMax Accuracies show similar trends. These results imply that barycenter-based method against Collectives achieves better size-agnosticity w.r.t. accuracies when trained on larger populations. On the other hand, Attacker Effort reduces to below 0.2 for size 3 and 5 strategies; between 0.2 and 0.4 for size 10 strategy; and between 0.3 and 0.5 for size-20 strategy by the end of the attack. In addition, Attacker Effort shows higher variability across test-populations with increasing training-population size. Therefore, barycenter-based method achieves better size-agnosticity w.r.t. accuracies at the cost of higher and lesser size-agnostic effort with increasing training-population size.

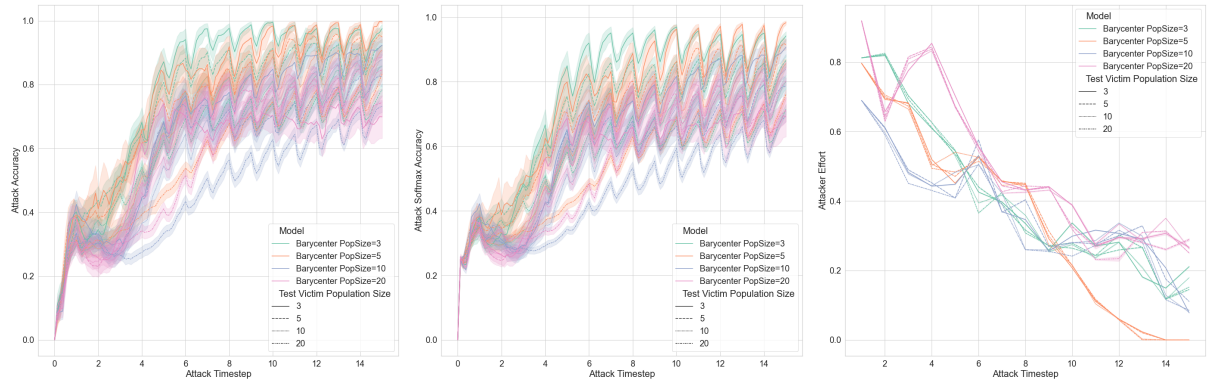

Figure 3: Accuracy, Softmax Accuracy and Effort of Barycenter attacks, tested on Swarm Collectives of sizes 3,5,10 and 20

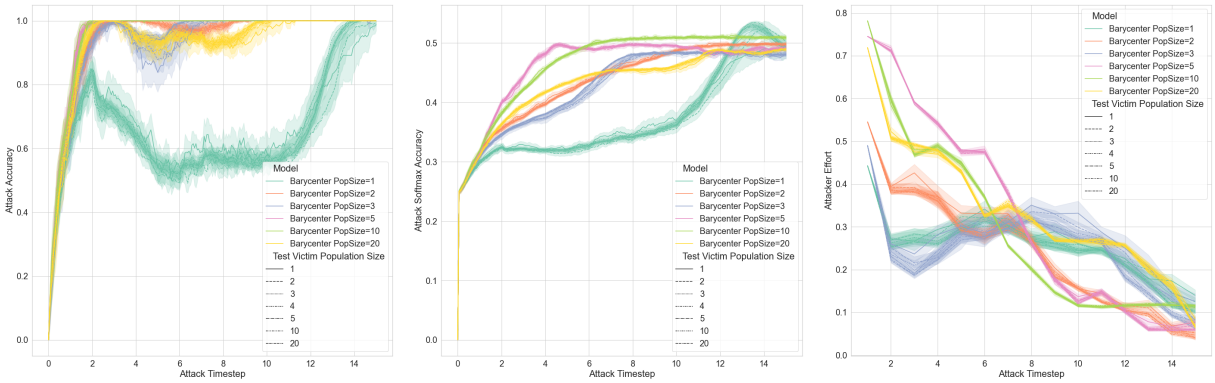

Figure 4: Accuracy, Softmax Accuracy and Effort of Barycenter attacks, tested on Implicit Collectives of sizes 1,2,3,4,5,10 and 20

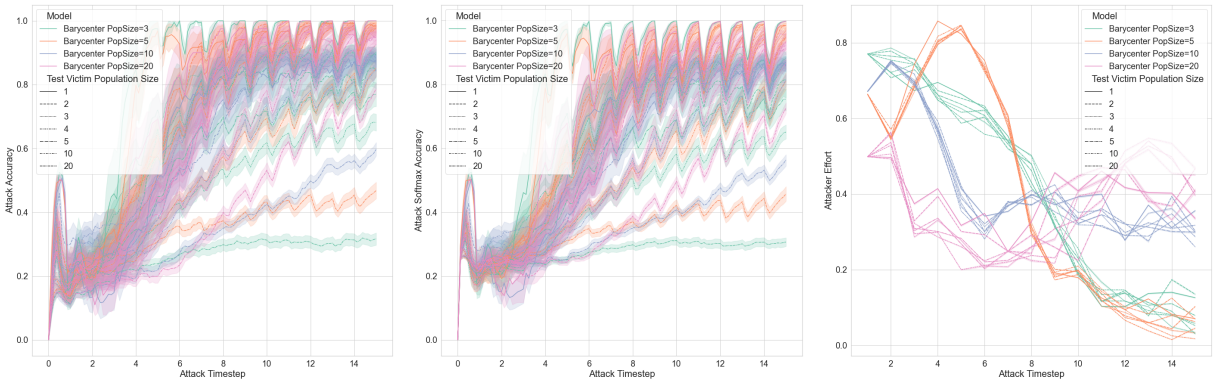

Figure 5: Accuracy, Softmax Accuracy and Effort of Barycenter attacks, tested on Collective victim populations of sizes 1,2,3,4,5,10 and 20

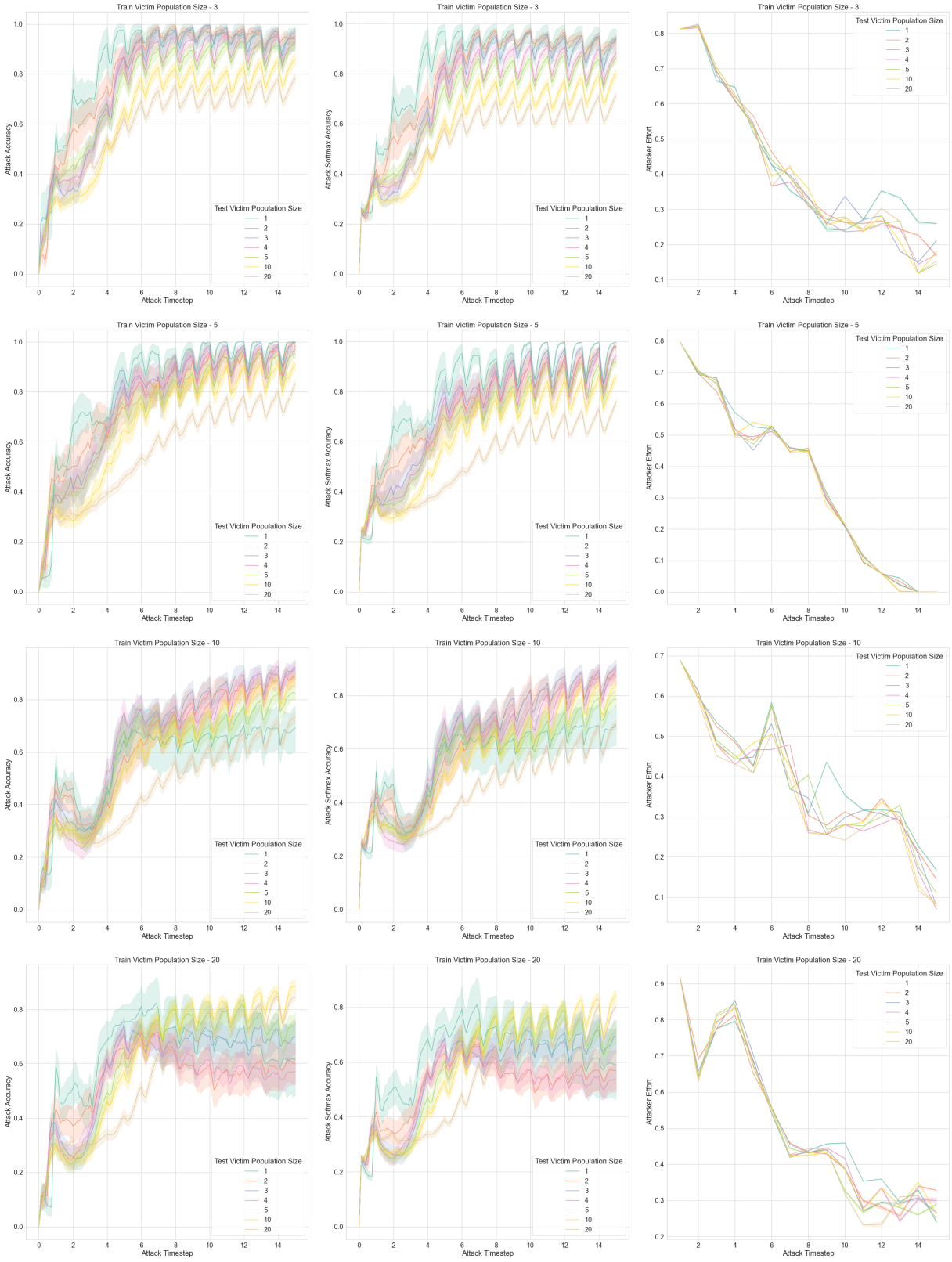

Figure 6: Accuracy, Softmax Accuracy and Effort of Barycenter attacks, tested on Swarm Collective victim populations of sizes 1,2,3,4,5,10 and 20, presented in separate graphs

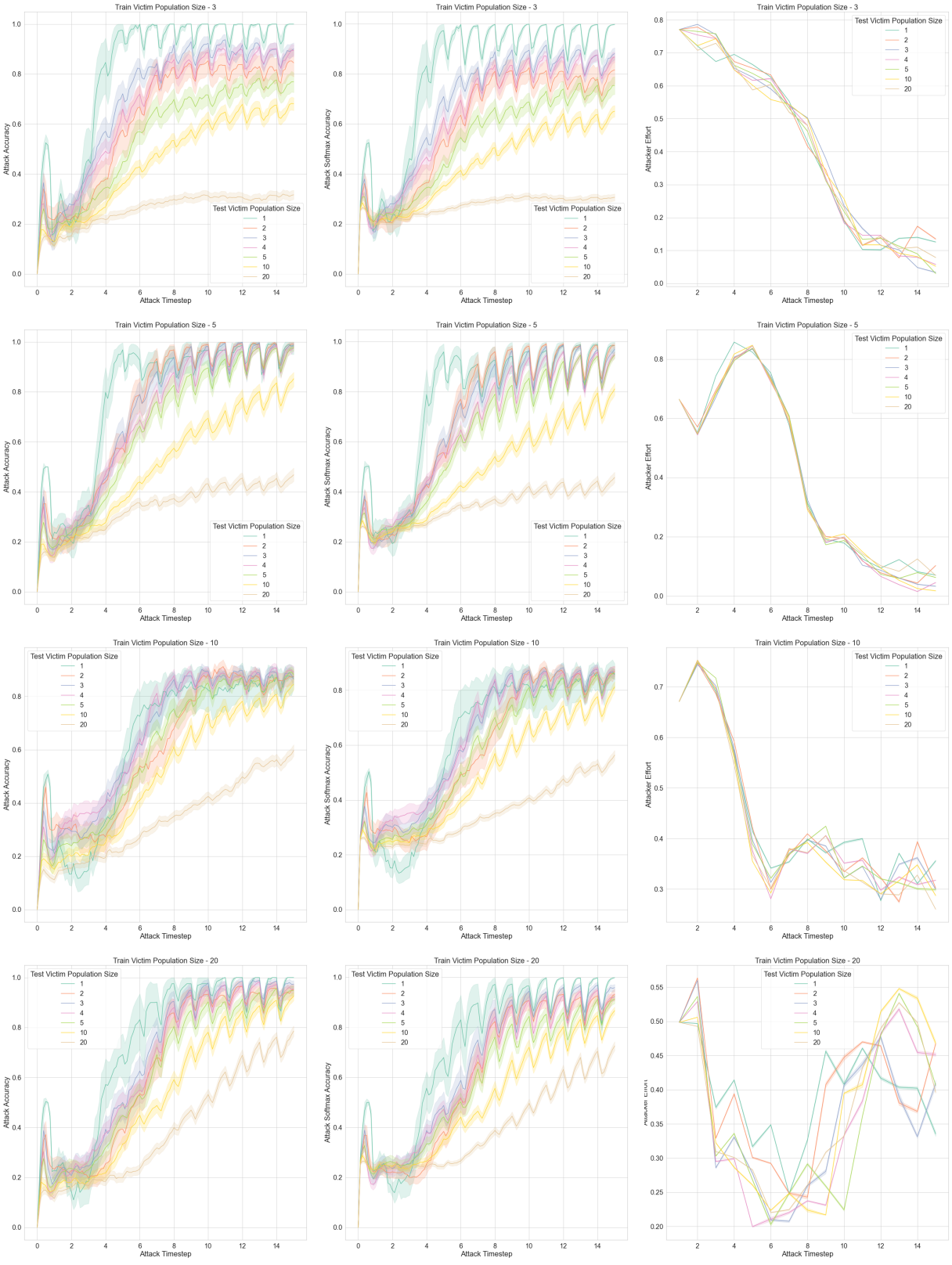

Figure 7: Accuracy, Softmax Accuracy and Effort of Barycenter attacks, tested on Collective victim populations of sizes 1,2,3,4,5,10 and 20, presented in separate graphs

References

[Daw *et al.*, 2005] Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711, 2005.

[Dennis *et al.*, 2020] M. Dennis, N. Jaques, E. Vinitsky, A. Bayen, S. Russell, A. Critch, and S. Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. In *NeurIPS*, pages 13049–13061, 2020.

[Dimakopoulou *et al.*, 2018] M. Dimakopoulou, I. Osband, and B. Van Roy. Scalable coordinated exploration in concurrent reinforcement learning. In *NeurIPS*, 2018.

[Foerster *et al.*, 2018] J. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch. Learning with opponent-learning awareness. In *AAMAS*, pages 122–130, 2018.

[Grover *et al.*, 2018] A. Grover, M. Al-Shedivat, J. Gupta, Y. Burda, and H. Edwards. Learning policy representations in multiagent systems. In *ICML*, pages 1802–1811, 2018.

[He *et al.*, 2016] H. He, J. Boyd-Graber, K. Kwok, and H. Daumé III. Opponent modeling in deep reinforcement learning. In *ICML*, pages 1804–1813, 2016.

[Jiang *et al.*, 2021a] M. Jiang, M. Dennis, J. Parker-Holder, J. Foerster, E. Grefenstette, and T. Rocktäschel. Replay-guided adversarial environment design. In *NeurIPS*, pages 1884–1897, 2021.

[Jiang *et al.*, 2021b] M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In *ICML*, pages 4940–4950, 2021.

[Lee *et al.*, 2019] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *ICML*, pages 3744–3753, 2019.

[Lupu and Precup, 2020] A. Lupu and D. Precup. Gifting in multi-agent reinforcement learning. In *AAMAS*, pages 789–797, 2020.

[Marthi *et al.*, 2005] B. Marthi, S. Russell, D. Latham, and C. Guestrin. Concurrent hierarchical reinforcement learning. In *IJCAI*, pages 779–785, 2005.

[Matiisen *et al.*, 2019] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[Mialon *et al.*, 2020] G. Mialon, D. Chen, A. d'Aspremont, and J. Mairal. A trainable optimal transport embedding for feature aggregation. In *ICLR*, 2020.

[Morimoto and Doya, 2005] J. Morimoto and K. Doya. Robust reinforcement learning. *Neural computation*, 17(2):335–359, 2005.

[Papoudakis and Albrecht, 2020] G. Papoudakis and S. V. Albrecht. Variational autoencoders for opponent modeling in multi-agent systems. In *AAAI WS on RL in Games*, 2020.

[Papoudakis *et al.*, 2021] G. Papoudakis, F. Christianos, and S. Albrecht. Agent modelling under partial observability for deep reinforcement learning. pages 19210–19222, 2021.

[Parisotto *et al.*, 2019] E. Parisotto, S. Ghosh, S. B. Yalamanchi, V. Chinnaobireddy, Y. Wu, and R. Salakhutdinov. Concurrent meta reinforcement learning. *arXiv preprint arXiv:1903.02710*, 2019.

[Parker-Holder *et al.*, 2022] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. Foerster, E. Grefenstette, and T. Rocktäschel. Evolving curricula with regret-based environment design. *arXiv preprint arXiv:2203.01302*, 2022.

[Qi *et al.*, 2017] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017.

[Rabinovich *et al.*, 2010] Z. Rabinovich, L. Dufton, K. Larson, and N. R. Jennings. Cultivating desired behaviour: Policy teaching via environment-dynamics tweaks. In *AAMAS*, pages 1097–1104, 2010.

[Rabinowitz *et al.*, 2018] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S.M. A. Eslami, and M. Botvinick. Machine theory of mind. In *ICML*, pages 4218–4227, 2018.

[Rached *et al.*, 2004] Z. Rached, F. Alajaji, and L. L. Campbell. The kullback-leibler divergence rate between markov sources. *IEEE Transactions on Information Theory*, 50(5):917–921, 2004.

[Raileanu *et al.*, 2018] R. Raileanu, E. Denton, A. Szlam, and R. Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *ICML*, pages 4257–4266, 2018.

[Shum *et al.*, 2019] M. Shum, M. Kleiman-Weiner, M. L. Littman, and J. B. Tenenbaum. Theory of minds: Understanding behavior in groups through inverse planning. In *AAAI*, pages 6163–6170, 2019.

[Skianis *et al.*, 2020] K. Skianis, G. Nikolentzos, S. Limnios, and M. Vazirgiannis. Rep the set: Neural networks for learning set representations. In *AIStats*, pages 1410–1420, 2020.

[Tacchetti *et al.*, 2019] A. Tacchetti, F. H. Song, P. A. M. Mediano, V. Zambaldi, J. Kramár, N. C. Rabinowitz, T. Graepel, M. Botvinick, and P. W. Battaglia. Relational forward models for multi-agent learning. In *ICLR*, 2019.

[Terry *et al.*, 2020] Justin K Terry, Nathaniel Grammel, Ananth Hari, Luis Santos, and Benjamin Black. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.

[Tobin *et al.*, 2017] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30, 2017.

[Xu *et al.*, 2021] H. Xu, R. Wang, L. Raizman, and Z. Rabinovich. Transferable environment poisoning: Training-time attack on reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1398–1406, 2021.

[Yaman *et al.*, 2022] Anil Yaman, Nicolas Bredeche, Onur Çaylak, Joel Z Leibo, and Sang Wan Lee. Meta-control of social learning strategies. *PLoS computational biology*, 18(2):e1009882, 2022.

[Yang *et al.*, 2020] J. Yang, A. Li, M. Farajtabar, P. Sunehag, E. Hughes, and H. Zha. Learning to incentivize other learning agents. In *NeurIPS*, pages 15208–15219, 2020.

[Zaheer *et al.*, 2017] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *NeurIPS*, 2017.

[Zhang and Lesser, 2010] C. Zhang and V. Lesser. Multi-agent learning with policy prediction. In *AAAI*, 2010.