Scaling Laws for Cooperation in Multi-Agent Reinforcement Learning

Alexandre Larouche
Département d'informatique et de génie logiciel
Université Laval
Québec, QC G1V 0A6
alexandre.larouche.7@ulaval.ca

Ali Chams Eddine Touil
Département d'informatique
Université de Montréal
Montréal, QC, 2920, chemin de la Tour
ali.chams.eddine.touil@umontreal.ca

Abstract

Emergent behaviors, commonly defined as any behavior that an agent exhibits that it wasn't explicitly trained to achieve, are an interesting phenomenon which will likely be crucial to understand for the future of artificial intelligence. In this work, we aim to study the effect on scaling on such behavior. However, these behaviors are usually difficult to measure and obtain. As such, we focus on cooperation in multi-agent reinforcement learning as cooperation is measurable via Convergent Cross-Mappings and previous work already discussed guidelines to induce new behaviors. By scaling models in depth and width, we find a scaling law on the Simple Tag environment which holds no matter whether the model is scaled in depth and width. Furthermore, we find that agents who receive more rewards in a cooperative setting do not necessarily cooperate more, which corroborates previous findings in the field.

1 Introduction

Emergent behaviors are a more or less pleasant surprise in several reinforcement learning (RL) settings. For example, the emergence of tool usage, like in [1], can both be a blessing and a curse. Indeed, new behaviors can be interesting and relate with the human understanding of environments, but can also stem from inaccuracies in environment modeling [1]. Indeed, while interesting and certainly useful to the agents, certain emergent behaviors in [1], such as box surfing, stem directly from the limitations of the environment. While the debate over a "good" and "bad" emergent behavior is philosophical in nature and is thus subjective, it remains interesting to see if emergent behaviors could follow a scaling law. Such a law could enable researchers to better understand the minimal requirements needed to achieve emergent behaviors in multi-agent games.

In this project, we define an emergent behavior as any behavior that an agent exhibits that it wasn't explicitly trained to achieve. With this definition, cooperation comes as an obvious proxy in order to measure emergent behaviors, as games focusing on an objective reward rarely explicitly give a reward based on cooperation. Furthermore, [1] mentions that the autocurricula induced by self-play is crucial in achieving the emergence of new strategies in the Hide and Seek game. Therefore, in order to study scaling laws for cooperation in RL we focus our attention on multi-agent competitive games in order to induce cooperation.

Preprint. Under review.

Therefore, the aim of this work is to answer the three following research questions:

- Do bigger models cooperate more?
- What is the minimal amount of computation required in order to achieve a certain level of collaboration?
- Do scaling laws exist between CCMs, compute and model size?

In order to measure answer these questions, we opt to measure collaboration on top of the regular performance metrics used for learning, in order to avoid embedding the cooperation measure in the reward signal. We use convergent cross mappings (CCMs) [14], which has been shown to be an adequate collaboration measure in multi-agent settings empirically in [2]. Although the results from [2] indicate that high performance in collaborative settings do not necessarily yield collaborative agents, it remains an interesting research direction to study the effect of scale on cooperation. As such, we propose to scale up the experiments from [2] to obtain a scaling law of cooperation in multi-agent RL agents.

2 Related Works

2.1 Scaling in RL

The study of scaling in RL is not new, however, to the best of our knowledge, no work focuses on cooperation or emergent behavior in RL. Indeed, most work focuses on scaling laws for a specific performance metric which the agent implicitly or explicitly tries to optimize, like the reward signal. [1] explores some aspects of scaling by varying batch sizes and its influence on performance. Furthermore, [10, 11] focuses on scaling laws in multi-agent RL (MARL) and found that agents evaluated with Elo-like rating systems have performance scaling as a power law of the model size and compute with AlphaZero agents. Although the Elo rating is an adequate performance metric in multi-agent competitive games, and can be used to optimize policies, it fails to measure cooperation.

2.2 Cooperation in RL

Many works study the cooperative nature of RL. Indeed, a lot of work has been done to get multi-agent RL methods to cooperate together [12] or with humans [3, 4]. Oftentimes, encouraging a cooperative behavior is done by shaping the reward so it is big when desirable interactions happen and less so when the interactions are undesirable. [6] uses the influence over other agents as part of the reward signal for a given agent's actions. However, as the notion of cooperation guides the reward signal, it cannot be considered an emergent behavior as it is explicitly optimized. Interestingly, the fact that we aim to not explicitly optimize cooperation also means that reward is not the target indicator of cooperation. Cooperation must therefore be estimated independently, which is not common practice in RL. To the best of our knowledge, only [2] propose a method to measure cooperation without embedding the measure in the reward signal. Indeed, they propose using CCMs in order to measure to causal influence one agent's action has over the another agent, which can be measured after training agents.

3 Preliminaries

In this work, we consider a multi-agent reinforcement learning setting in which we train multiple agents and then measure their cooperation using CCMs. We therefore introduce these two notions here.

3.1 Multi-Agent Reinforcement Learning

Single-agent RL is typically modeled as a game between an agent and an environment inside a Markov Decision Process $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \mathcal{T}_0, \gamma\}$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function, $\mathcal{T}(s'|s, a)$ is the probability distribution of transitioning to state s' given state s and action a, $\mathcal{T}_0 : \mathcal{S} \mapsto [0, 1]$ is the probability distribution over initial states and γ is the discount factor.

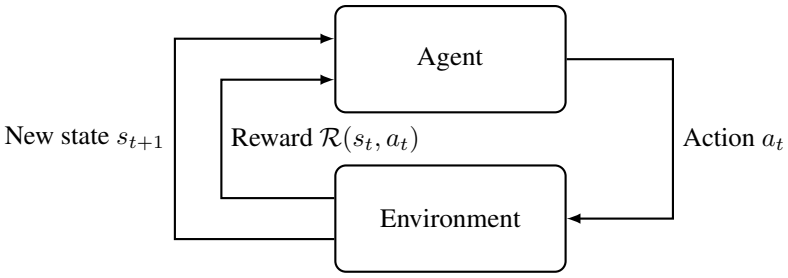


Figure 1: The basic game loop for the single agent in the RL setting.

At each time step $t \in \{0, 1, ..., T\}$ (typically, $T < \infty$) of the game loop, the agent observes a state s_t and selects an action $a_t \sim \pi(\cdot|s_t)$ to play in response to s_t. The function $\pi(\cdot|s_t)$ is called a *policy* and may sometimes be used as a probability distribution over actions. Once the agent plays their action, the environment returns a reward $\mathcal{R}(s_t, a_t)$ and a state s_{t+1} to the agent. Figure 1 shows the game loop of RL in general.

The reward aims to guide the agent towards some desirable behavior by tuning the π to favor actions which yield high rewards. The goal of the agent in such an MDP is to maximize its sum of rewards, that is,

$$\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t). \tag{1}$$

To achieve this goal, the agent must find a π which maximizes

$$\mathbb{E}\left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(s_t, a_t) \middle| a_t \sim \pi(\cdot|s_t), s_0\right]. \tag{2}$$

In multi-agent RL (MARL), the principle remains exactly the same, except there are multiple agents, and thus multiple policies. Furthermore, each agent may have different goals and thus different rewards, state sets and action sets for the same environment. In the MARL setting, performance is quite easy to measure as Equation 1 is directly available for each agent. Therefore, if the goal was for the agents to cooperate, it would be straightforward to embed a cooperation measure in the reward signal as to encourage this behavior. However, as we aim to discover scaling laws for the emergence of cooperation in MARL, we must avoid doing so, as the resulting cooperation would not be emergent anymore. Therefore, we next introduce CCMs, a measure of cooperation which can be applied after training the agents.

3.2 Convergent Cross-Mappings

To answer the three research questions posed above, we endeavor to observe the effect of scale on some measure of cooperation. We decide to use CCMs in order to quantify cooperation, a method first proposed by [14] and implemented as a measure of cooperation in [2] in the multi-agent RL setting.

Convergent cross mappings [14] are a useful tool for one seeking to measure cause-and-effect relationships between two time-series variables. Given observational data over a period of time of a set of variables $\mathcal{S}$ we seek to measure the causal link between two variables from this set. In a dynamical system, like in RL, two variables are causally linked if the state of one variable can be inferred from the other. Furthermore, if X causes Y, then Y contains information about X. If this causation is unidirectional (i.e., Y does not affect X), then the X does not contain information about Y.

Consider the manifold $\mathbf{M}$ of the dynamical system and a timestep t. We say $\mathbf{M}(t)$ is the state of the dynamical system at time t. For example, if $\mathbf{M}$ tracks three variables X, Y and Z, then $\mathbf{M}(t) = [X(t), Y(t), Z(t)]$ is the vector of values of these variables at time t. According to Taken's

Theorem, one can reconstruct $\mathbf{M}$ from only one variable by constructing a shadow manifold built from an embedding of time-lagged values of the variable. More formally, to reconstruct $\mathbf{M}$ from variable X, one can construct the shadow manifold $\mathbf{M}_X$ with.

$$x(t) \triangleq \mathbf{M}_X(t) = [X(t), X(t-\tau), X(t-2\tau), ..., X(t-(E-1)\tau)]$$

.

The parameter τ is a factor determining the gap between time steps and E determines the size of the embedding. An example of the construction of such a shadow manifold is displayed in Figure 2. In this figure, we consider variable X taken from a Lorenz system (i.e., generated from a Lorenz Attractor) and observe it is sufficient to keep a single time-lagged variable in order to built a shadow manifold, mimicking the original Lorenz Attractor.

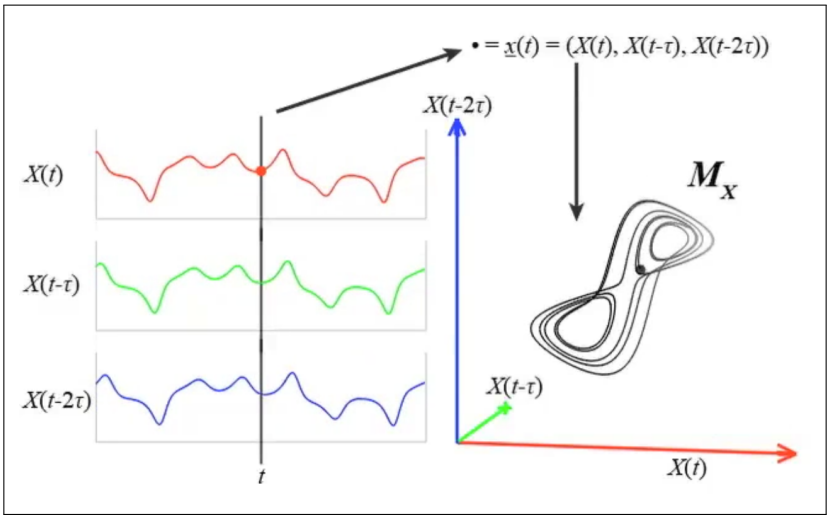


Figure 2: Example of the shadow manifold $\mathbf{M}_X$ produced from the single time-lagged variable X extracted from a Lorenz Attractor taken from [14]. Although the reconstruction of the original manifold is imperfect, it maintains its shape properties.

To measure the causal link between two time series of size N, $X = \{X(1), X(2), ..., X(N)\}$ and $Y = \{Y(1), Y(2), ..., Y(N)\}$, we first construct the shadow manifolds $\mathbf{M}_X$. For a given point $x(t) \in \mathbf{M}_X$, we can find its $E+1$-nearest-neighbors on $\mathbf{M}_X$ and the time step at which they occur, $t_1, ..., t_{E+1}$. We can then predict $Y(t)$ from $\mathbf{M}_X$ like so:

$$\hat{Y}(t)|\mathbf{M}_X = \sum_{i=1}^{E+1} w_i Y(t_i) \qquad \text{where} \tag{3}$$

$$w_i = \frac{\exp\left(\frac{||x(t)-x(t_i)||}{||x(t)-x(t_1)||}\right)}{\sum_{j=1}^{E+1} \exp\left(\frac{||x(t)-x(t_j)||}{||x(t)-x(t_1)||}\right)}. \tag{4}$$

The prediction $\hat{Y}(t)|\mathbf{M}_X$ is the so-called cross mapping from X to Y and is thus a weighted average of the estimated nearest neighbor values in Y. By repeating this process for many points across $\mathbf{M}_X$ (i.e., $x(t), \forall t \in [N]$), we obtain a series of prediction upon which to measure accuracy or correlation with respects to the ground truth. This measure is the measure of causal effect of Y on X. The measurement of the causal effect of X on Y is analogous. Figure 3 shows graphically the intuition of this procedure. In this figure, consider $\mathbf{M}_X$ and $\mathbf{M}_Y$, the shadow manifolds of X and Y respectively. Since both shadow manifolds map points one-to-one to the original manifold, then every point from $\mathbf{M}_X$ maps one-to-one to $\mathbf{M}_Y$. Therefore, if the nearest neighbors of a point $x(t)$ taken from $\mathbf{M}_X$ can predict well a point $Y(t)$ on the original manifold, then these neighbors should be the same in $\mathbf{M}_Y$, i.e., the points should be close together across shadow manifolds. When this happens, we say

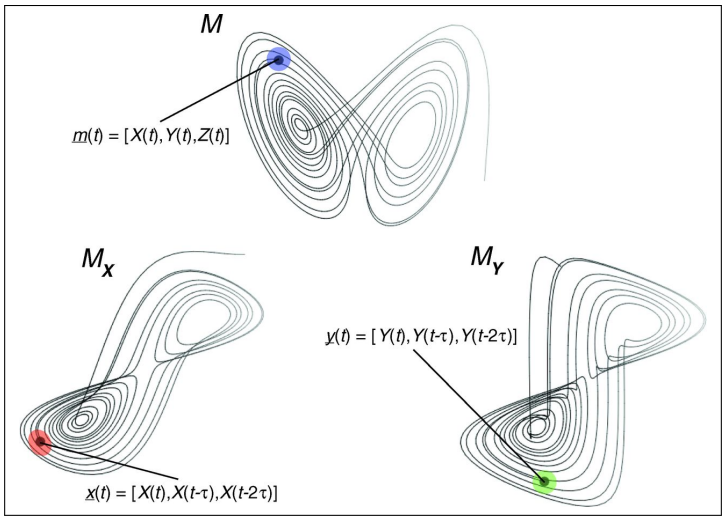


Figure 3: From a point $x(t)$ taken from the shadow manifold $\mathbf{M}_X$, we can use its neighborhood in order to predict a point $Y(t)$ on the original manifold $\mathbf{M}$. Therefore, if Y has an effect on X, then the neighborhood of $x(t)$ on $\mathbf{M}_X$ should map to a relatively close neighborhood on $\mathbf{M}_Y$.

that Y has an effect on X. CCMs can take values in the range of [0, 1], 0 meaning a variable has no causal influence over the other and 1 meaning a variable has total causal influence over the other.

We borrow the method from [2] that empirically validated that CCMs is an appropriate measure of cooperation by showing that agents with a higher CCMs indeed have more statistical dependence between their movements. Since cooperation should translate to statistical dependence in movement between agents, we deem this measure appropriate for our experiments. Therefore, the time series used with CCMs in this report are simply the actions taken by the agents. Thus, a *variable* in the CCMs is formally an agent and the observed values for this variable are the actions taken by the agent.

4 Experimental Setup

To conduct our experimentation, we use PettingZoo, a Python library designed for conducting research in MARL environments [16]. The library's initial goal was to speed up research in the field by providing a collection of benchmark environments that could be accessed by all researchers, along with a standardized application programming interface (API). This is similar to the approach taken by OpenAI's Gym library, which aimed to make single reinforcement learning more accessible. Furthermore, we use Tianshou [17], a general-purpose RL library to implement our agents.

4.1 Agents

In this work, we consider Actor-Critic Proximal Policy Optimization (PPO) [13] as a template for the agents we train. We use this algorithm as it is well known, easy to implement and has been shown to work well in practice. Thus, the actor and critic both consist of a simple Multi-Layer Perceptron (MLP) and the number of parameters we report in this work are the sum of parameters of both MLPs. The weights of the MLPs are optimized using Adam [7] with default hyperparameters presented in the Tianshou PPO example.

4.2 Environments

To train our agents, two environments taken from the Multi Particle Environments (MPE) [8, 9] suite. Specifically, *Simple Tag* and *Simple Adversary*.

Simple Tag A Predator-Prey environment which consists of a variable number of predators and preys as well as multiple obstacles which block agents. In our setting, we use three predators, a single

prey and no obstacle. The prey has to avoid colliding with the slower-moving predators, while the predators have to capture the prey. Each collision of the prey with a predator will result in a -10 penalty to the prey and a +10 bonus to the predators. Additionally, the preys are penalized for moving away from a predefined area, which is a one-unit square. At every time step, each agent observes its own position, its own velocity as well as the velocity and position of every other agent relative to itself. Furthermore, each agent can choose between five actions, that is, moving in one of the four cardinal directions or not moving, since we consider the discrete version of the environment.

Simple Adversary In contrast, Simple Adversary is an environment consisting of a team of *N* agents, *N* landmarks and an adversary agent. In our case, we consider the environment using $N = 2$ which results in two teamed agents and two landmarks. All agents observe the position of landmarks and other agents. One landmark is designed as the 'target landmark' colored in green. The teamed agents are rewarded equally based on the distance of the closest agent to the target and the negative distance of the adversary to the target. Meanwhile, the adversary is also rewarded based on its distance to the target, but it must infer the target location based on the cooperative agents observed locations. All rewards are the Euclidean distance. Thus, the cooperative agents have to learn to work together by splitting up question to distract the adversary from the target while trying to approach the target as much as possible. As with Simple Tag, each agent can choose between five actions discussed earlier that will be applied on a discrete version of the environment. The Simple Adversary environment is also considered general-sum since multiple agents with different objectives are involved.

4.3 Scaling

In order to study scaling laws, we scale the MLPs of the Actor-Critic networks at the same time and in width and depth separately. Table 1 shows the considered architecture for fixed-depth and fixed-width experiments. For fixed-depth experiments, we only consider MLPs with three hidden layers and vary their hidden layer sizes uniformly, going from 64 neurons per hidden layer to 1024 neurons. In contrast, for fixed-width experiments, we only consider MLPs with 64 neurons per hidden layers, going from two to six hidden layers. Furthermore, for every considered architecture, we consider multiple values of embedding size E in the CCMs calculation as this parameter can have a big impact on the observed CCMs. Specifically, we consider the range $E = \{2, 3, 4, 5\}$.

Architectures considered for scaling experiments	
Fixed-depth	***Fixed-width***
$64 \times 64 \times 64$	64×64
$128 \times 128 \times 128$	$64 \times 64 \times 64$
$256 \times 256 \times 256$	$64 \times 64 \times 64 \times 64$
$512 \times 512 \times 512$	$64 \times 64 \times 64 \times 64 \times 64$
$1024 \times 1024 \times 1024$	$64 \times 64 \times 64 \times 64 \times 64 \times 64$

Table 1: Architectures considered for fixed-depth and fixed-width scaling experiments.

4.4 Implementation details

Environment As mentioned previously, we opt to remove obstacles from Simple Tag. We do so in order to accelerate training progress, as we assume the obstacles would only prevent the predators from reaching the prey, thus slowing down cooperation emergence by slowing down reward accumulation. Furthermore, we opt to extend the episode duration to 100 steps instead of the default 25 steps for both environments. This is done in order to encourage the observation of cooperative behavior, as we felt 25 steps would probably be too short to observe cooperation. Finally, no preprocessing is performed on the observations.

Training We train agents on both environment where we set several hyperparameters for the model. Specifically, the agents play for a total of 1,000,000 time steps, totaling 10,000 episodes and store their experience in a replay buffer of size 200,000. Every 10 episodes, each agent's networks are updated and each batch of data is learned over twice. Every agent is initialized in the same way and optimized with Adam with gradient clipping at 0.5, a learning rate of 0.001 and $\beta = (0.9, 0.999)$.

These hyperparameters are not tuned to the task and were taken directly from Tianshou's PPO example. Therefore, PPO does not use an entropy bonus in our experiments and the advantage values are normalized. Finally, we repeat the experiments on 5 different seeds and save a checkpoint of every agent every 100,000 steps in order to compute the CCMs.

Post-Training Once the model training is complete, we proceed to retrieve the agents and randomly assign one of the teamed agents a random policy, as proposed in [2]. We then calculate the CCMs of this random policy over the other agents on the cooperative side. We repeat this process across every seed and permutation of random policy/trained policies, with the average CCMs value calculated for every checkpointed time step. To generate the time series used to compute CCMs, the agents play 1,000 steps of their respective game each.

5 Results and Discussion

In this section, we present and discuss results we obtained from fixed-depth and fixed-width experiments for the Simple Tag and Simple Adversary environments. We report results in FLOPs instead of the number of parameters, as suggested in previous work [5] as the number of parameters may be a confounder in measuring model performance as it does not take into account the arithmetic intensity of a model and thus may not allow easy comparison across model architectures (e.g. for a variable number of hidden layers). We also report the 66% confidence intervals along with the recorded mean across seeds[1]. Furthermore, in the following sections, the points used to fit the scaling law curves only consider the best performing checkpoint in terms of CCMs for a given actor-critic model.

5.1 Simple Tag environment

In Simple Tag, since the predators are three, we can set each single predator to a random policy and compute the resulting CCMs for the other two agents. Thus, for each seed and for each checkpoint (10 checkpoints total), we obtain 6 CCMs.

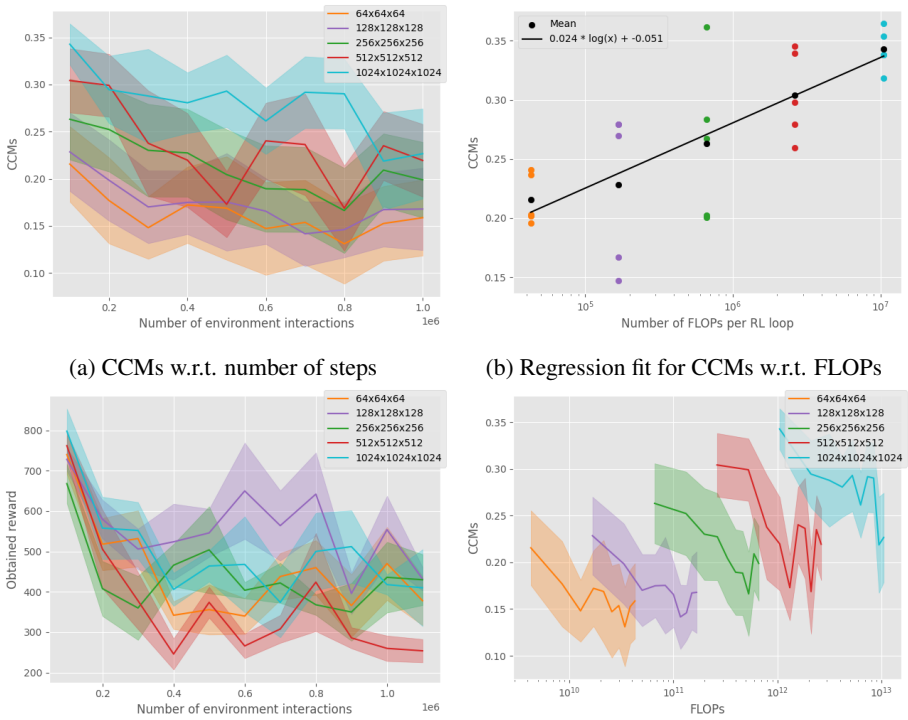
5.1.1 Fixed-depth scaling

We first report the results for the fixed-depth scaling experiments, shown in Figure 4. For these results, we report only what was obtained with a CCMs embedding size $E = 4$, as this gave the highest overall CCMs, however, additional results are in Appendix A.1.

Firstly, we notice that the rewards during test time in Figure 4c seems to decrease first and then increase or fluctuate up and down. The initial decrease is most likely due to the fact that the prey also becomes better as the training progresses, which leads to less rewards given to the predators. Furthermore, when comparing Figures 4a and 4c, we notice that agents that obtain more rewards during test time do not necessarily cooperate more, which corroborate the findings of [2]. This fact is highlighted specifically by the purple curve (128 neurons, 3 layers MLP), which obtains more rewards than the other architectures but still remains low in terms of CCMs. However, it is worth noting that the blue curve (1024 neurons, 3 layers MLP) seems to have a reward somewhat correlated with the CCMs while the red curve (512 neurons, 3 layers MLP) obtains rewards which are anti-correlated with the CCMs, indicating that perhaps the contribution of cooperation to the sum of rewards obtained is model-dependent.

Interestingly, although cooperation seems to have varied effects on the rewards obtained by the agents, we find that agents do not cooperate much in this experiment, only obtaining CCMs values of around 0.35. However, when looking at Figure 4b, showing the CCMs w.r.t. to the number of FLOPs per RL loop, we find that agents with more FLOPs per RL loop seem to cooperate more, although to a limited extent. Indeed, the curve in Figure 4b shows that CCMs scale in $\log$ w.r.t. the number of FLOPs per RL loop, which tends to saturate fairly quickly. Finally, Figure 4d, showing the CCMs w.r.t. to the total number of FLOPs during training, we see that more FLOPs do not necessarily lead to more cooperation. However, it seems that bigger architectures is a determining factor in the level of cooperation achieved by agents. Interestingly, we find that high CCMs are achieved early on by

[1]The choice of a 66% confidence interval is not intentional. Rather, we realized too late that we missed a multiplicative constant in order to measure the confidence intervals properly. This could easily be fixed in the future.



(a) CCMs w.r.t. number of steps

(b) Regression fit for CCMs w.r.t. FLOPs

(c) Test rewards w.r.t number of steps

(d) CCMs w.r.t. to total number of FLOPs

Figure 4: (a, b, d) CCMs and (c) test rewards for checkpointed agents with the CCMs embedding size $E = 4$ in fixed-depth experiments on Simple Tag.

the agents, which indicates that agents seem to *learn* to cooperate less. Nonetheless, it is interesting to observe that agents with bigger architectures seem to cooperate more overall, at least up to the number of steps considered in this work.

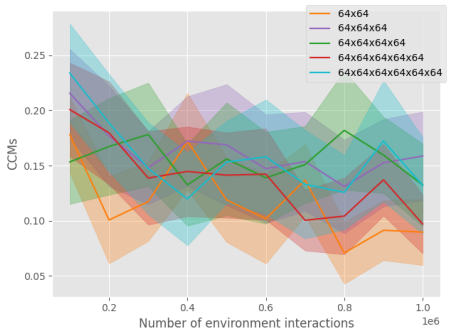
5.1.2 Fixed-width scaling

We then study the effect of scaling the number of layers instead of the number of neurons per layer, focusing on architectures with 64 neurons per hidden layers. Figure 5 shows the results obtained with CCMs embedding size $E = 4$, as this gives the higher overall CCMs across architectures. However, results for other embedding sizes are shown in Appendix A.2.

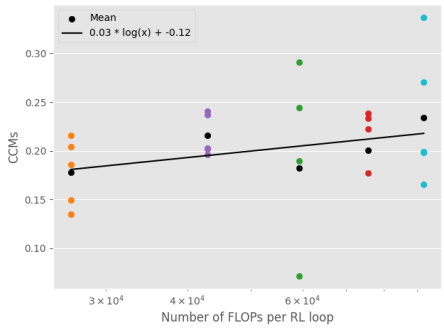
Here again, in Figure 5c we observe that rewards start by decreasing and then fluctuating for all agents, for the same reasons highlighted previously. Comparing Figures 5a and 5c gives no particular insight as most architectures obtain rewards and CCMs similar. This is most likely due to the fact that, although the architectures do add more and more layers, the numbers of parameters vary much less than in the fixed-depth experiments. By inspecting at Figure 5d, it is unclear in this experiment whether depth-wise scaling has an effect on CCMs. However, the curve fitted in Figure 5b has very similar coefficients to the curve fitted in Figure 4b. Furthermore, when comparing the curves on the number of parameters instead of the number of FLOPs per RL loop (Figure 6), we notice that the multiplicative coefficients are almost identical, which seems to indicate that the scaling law holds whether the models are scaling in depth or width.

5.2 Simple Adversary

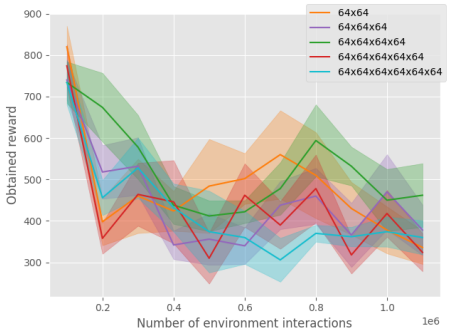
Next, we present the results for the Simple Adversary environment. As only two agents are teamed together, we only obtain two CCMs values per checkpoint per seed, resulting in slightly less data than in the previous Simple Tag experiment. Furthermore, many trained agents result in almost constant policies (i.e. policies that play a single action), leading to fewer valid CCMs. Indeed, constant policies generate constant trajectories with no variance. However, variance is needed in order to



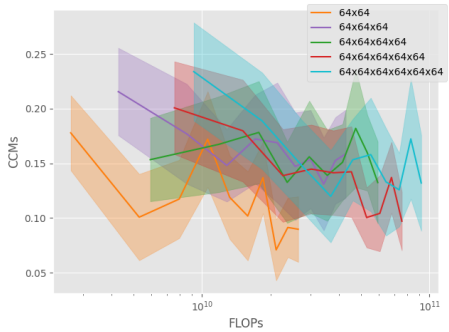
(a) CCMs w.r.t. number of steps



(b) Regression fit for CCMs w.r.t. FLOPs

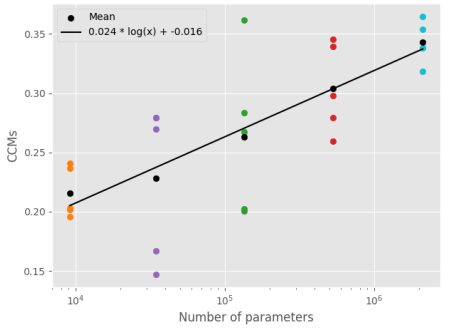


(c) Test rewards w.r.t number of steps

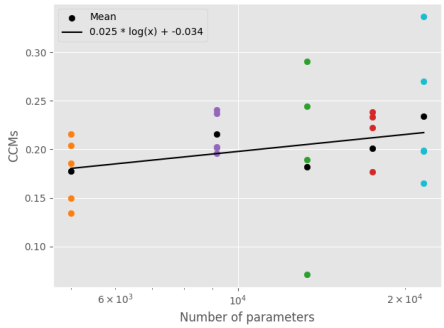


(d) CCMs w.r.t. to total number of FLOPs

Figure 5: (a, b, d) CCMs and (c) test rewards for checkpointed agents with the CCMs embedding size $E = 4$ in fixed-width experiments on Simple Tag.



(a) Fixed-depth experiment



(b) Fixed-width experiment

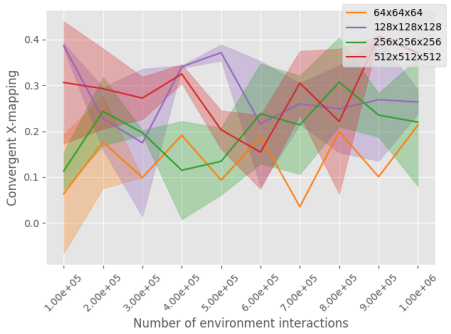
Figure 6: Regression fits for CCMs w.r.t. the number of parameters of the agent models in (a) fixed-depth experiment and (b) fixed-width experiment. We notice the multiplicative coefficient for both curves are almost identical.

compute the correlation and thus the CCMs. Therefore, when variance is zero, no CCMs can be computed. The results presented in this section should be considered with care, as much less data is actually present than in the previous Simple Tag experiments. Furthermore, constant policies usually indicate a failure in learning in agents.

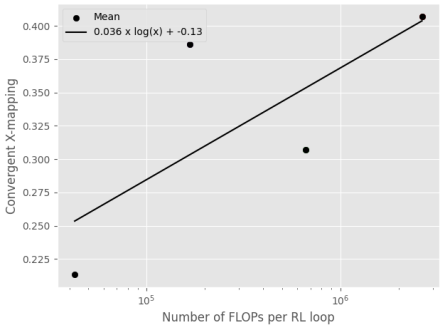
5.2.1 Fixed-depth scaling

As with Simple Tag, the following analysis is based on experimental results obtained for a single embedding size, $E = 4$, as this gave the highest CCMs values overall. Results with other embedding sizes can be found in the Appendix A.3. The largest architecture of $1024 \times 1024 \times 1024$ is not

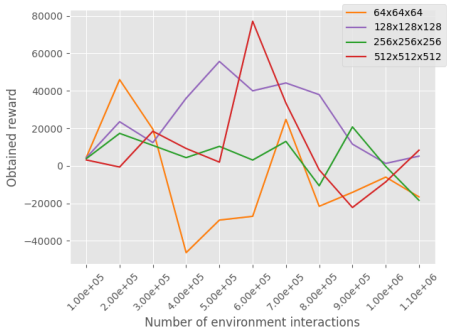
included in the results due to technical issues. Indeed, the program crashed several times toward the end of the training because the neural network would only output "NaN" values, perhaps because of gradient explosion. To address this issue, one could consider a smaller value of gradient clipping or normalize the input and output values in future works.



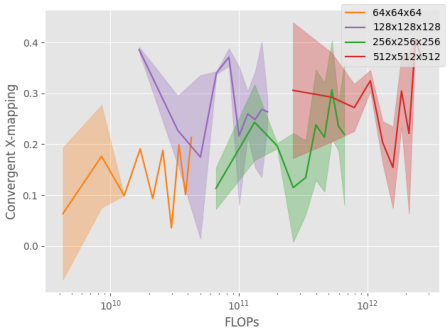
(a) CCMs w.r.t. number of steps



(b) Regression fit for CCMs w.r.t. FLOPs



(c) Test rewards w.r.t number of steps



(d) CCMs w.r.t. to total number of FLOPs

Figure 7: (a, b, d) CCMs and (c) test rewards for checkpointed agents with the CCMs embedding size $E = 4$ in fixed-depth experiments on Simple Adversary.

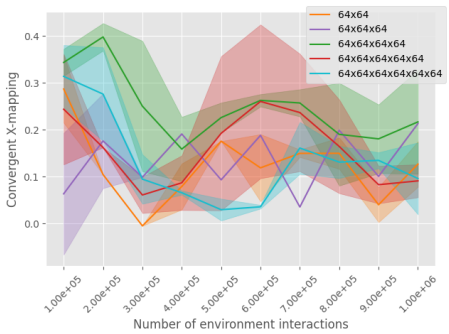
Figure 7a shows the CCMs values w.r.t. the number of environment interactions, ranging from0.005 to 0.4. Since most of the recorded CCMs for every architecture overlap during training, it is difficult to determine whether any interesting behavior stems from scaling. We notice, however, that the smallest architecture considered here (3 layers of 64 neurons) struggles to achieve the CCMs levels of other architectures. Similar behavior is observed for the reward, shown in Figure 7c, as they fluctuate a lot during training. However, two interesting observations can be made from this figure. First, the 3 layers of 64 (yellow line) neurons achieve the lowest amount of rewards and CCMs simultaneously, which means that perhaps this architecture was too small for the problem, as it never performed well on the task. Second, the highest achieved reward for the 3 layers of 512 neurons (red line) happens at the same time as its lowest recorded cooperation (at time step 6×10^5). However, we also notice that CCMs from this point onward keep increasing on this architecture, while the reward drops sharply, indicating that more cooperative agents are worse at the game. This behavior can be observed in all other architectures at a lesser extent. The downward trend in the reward could be explained by the fact that as the number of steps increases, both agents and the adversary get better, leading to games totaling a reward near zero. If this is the case, then the higher cooperation observed as training progresses may be indicative of an emergent behavior appearing in order to beat the adversary.

Figure 7b, shows the number of FLOPs per RL loop for each architecture. We notice that few points are plotted in this figure as opposed to previous results. This is due to the constant trajectories, mentioned previously. From the plot, however, the results suggest a correlation between the number of FLOPs and CCMs. As the number of FLOPs per RL loop grows, the agents demonstrate more cooperation, as shown by the positive slope. Furthermore, opposite to the previous findings in Simple Tag, it seems that CCMs do not directly decrease with the total number of FLOPs in this experiment, although they do not increase monotonically either, as shown in Figure7d. However, as the model architecture gets bigger, it seems that agents cooperate more, though the relationship between model

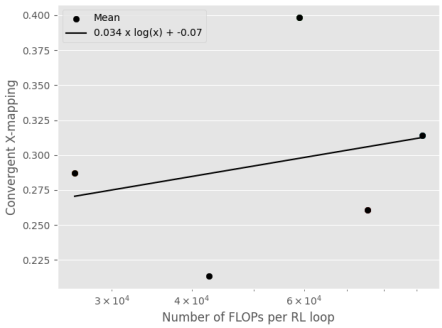
size and CCMs in this figure is not monotonically increasing either as there is a lot of overlap between curves. Furthermore, as the computed CCMs are limited in this experiment, it is difficult to be certain whether this holds true in general.

5.2.2 Fixed-width scaling

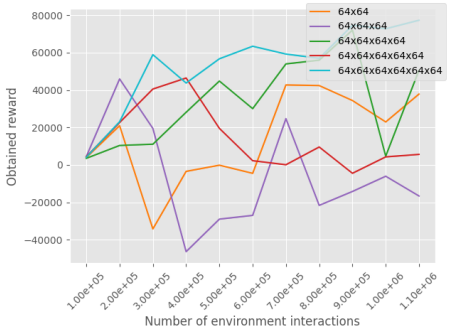
As previously, we also report results of fixed-width scaling in Simple Adversary, as shown in Figure 8. An embedding size of $E = 4$ is also used here as this gave the highest CCMs overall. Additional results are shown in Appendix A.2.



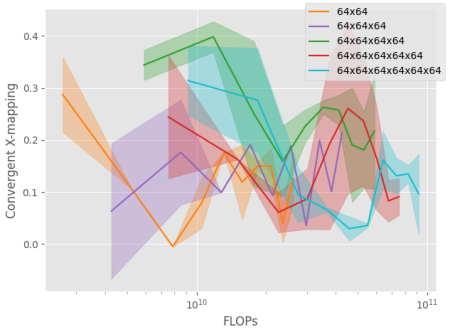
(a) CCMs w.r.t. number of steps



(b) Regression fit for CCMs w.r.t. FLOPs



(c) Test rewards w.r.t number of steps



(d) CCMs w.r.t. to total number of FLOPs

Figure 8: (a, b, d) CCMs and (c) test rewards for checkpointed agents with the CCMs embedding size $E = 4$ in fixed-width experiments on Simple Adversary.

Again, the results for the fixed-width experiments are less straightforward than the results in the fixed-depth experiment. Results from Figure 8a, which shows CCMs w.r.t. to the number of environment interactions, indicate that bigger models do not cooperate more. In Figure 8c, we observe that more cooperation does not always lead to more rewards, as highlighted by the largest architecture in blue (6 layers of 64 neurons). Furthermore, some architectures that obtain higher CCMs (e.g., 4 and 5 layers of 64 neurons) either obtain more or less rewards when comparing at the same number of environment interactions.

In Figure 8d, we observe that as the number of FLOPs increases, CCMs follow a downward trend across architectures, much like previous results in Simple Tag. Unlike previous experiments, however, the curve obtained from the CCMs in Figure 8b is much less straightforward as the recorded CCMs fluctuate heavily and non-monotonically with the number of FLOPs per RL loop.

6 Conclusion

In this study, we found a scaling law for cooperation in MARL in the Simple Tag environment. Furthermore, this scaling law held whether the models were scaled in depth or width. Additionally, we found that agents that obtained more rewards did not collaborate more, which corroborates the findings of [2]. Because of limitations such as frequent crashes and invalid CCMs, our results in

Simple Adversary are much less convincing and would require more work to achieve significant insights. However, this work, to the best of our knowledge, was the first step towards understanding scaling of emergent behaviors.

We acknowledge some room for improvement on multiple aspects of our study. First, we did not perform any hyperparameter tuning in the training process. Although we have selected reasonable default parameters for the MPE environments, it is possible that the agents' performance could be further improved with hyperparameter tuning. Additionally, we recognize that our study lacked the necessary compute to fully investigate the scaling laws for cooperation in MARL. While we were able to scale up the models to a certain extent, we were limited by our available computational resources. Finally, our choice of baseline models is somewhat arbitrary and could stand to be better justified in the future. Some work could also be done in order to further study scaling laws for cooperation in different settings. Despite these limitations, we believe that our study provides valuable insights into the potential scaling laws for cooperation in multi-agent RL. Encouragingly, these early results bring forth multiple avenues for potential research. Specifically, by trying other learning algorithms (i.e., alternatives to PPO) or environments, one may observe different behaviors. Furthermore, recent work proposed multi-task pretraining [15], which could have a big impact on the RL landscape in general and thus the results we found.

References

[1] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*, 2020.

[2] S. L. Barton, N. R. Waytowich, E. Zaroukian, and D. E. Asher. Measuring collaborative emergent behavior in multi-agent reinforcement learning. In *IHSED2018: Future Trends and Applications, October 25-27, 2018, CHU-Université de Reims Champagne-Ardenne, France 1*, pages 422–427. Springer, 2019.

[3] M. Bouton, A. Nakhaei, K. Fujimura, and M. J. Kochenderfer. Cooperation-aware reinforcement learning for merging in dense traffic. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3441–3447. IEEE, 2019.

[4] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan. Cooperative inverse reinforcement learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[5] J. Hilton, J. Tang, and J. Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint arXiv:2301.13442*, 2023.

[6] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas. Intrinsic social motivation via causal influence in multi-agent RL, 2019.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[9] I. Mordatch and P. Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[10] O. Neumann and C. Gros. Scaling laws for a multi-agent reinforcement learning model. In *Deep Reinforcement Learning Workshop NeurIPS 2022*, 2022.

[11] O. Neumann and C. Gros. Size scaling in self-play reinforcement learning. ESANN, 2022.

[12] A. Oroojlooy and D. Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *Applied Intelligence*, pages 1–46, 2022.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[14] G. Sugihara, R. May, H. Ye, C.-h. Hsieh, E. Deyle, M. Fogarty, and S. Munch. Detecting causality in complex ecosystems. *science*, 338(6106):496–500, 2012.

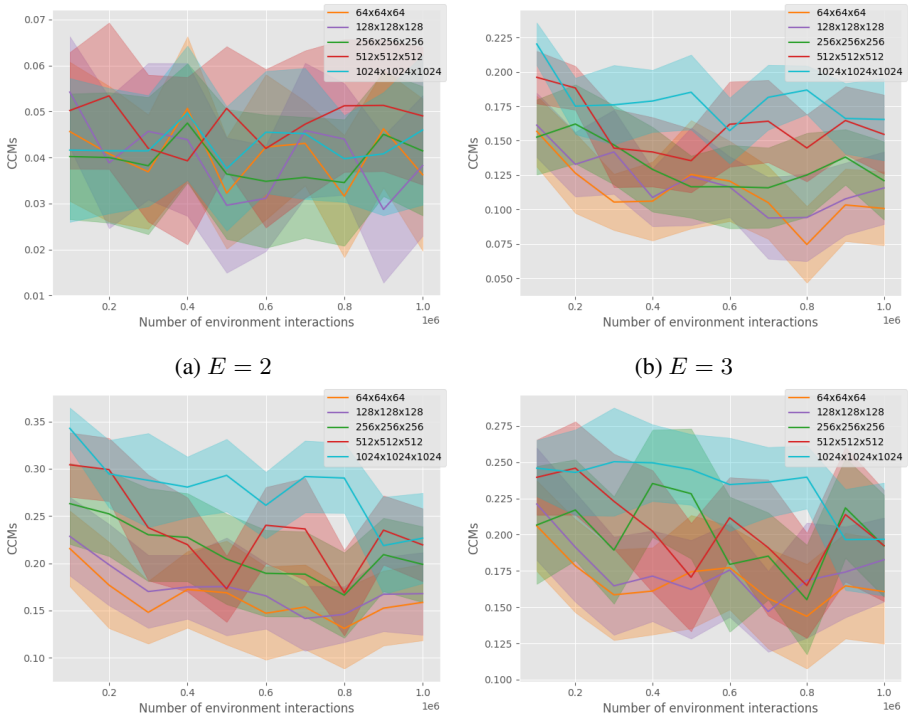
[15] Y. Sun, S. Ma, R. Madaan, R. Bonatti, F. Huang, and A. Kapoor. SMART: Self-supervised multi-task pretraining with control transformers. In *The Eleventh International Conference on Learning Representations*, 2023.

[16] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.

[17] J. Weng, H. Chen, D. Yan, K. You, A. Duburcq, M. Zhang, Y. Su, H. Su, and J. Zhu. Tianshou: A highly modularized deep reinforcement learning library. *Journal of Machine Learning Research*, 23(267):1–6, 2022.

A Additional results

A.1 Fixed-depth experiments on Simple Tag



(a) $E = 2$ (b) $E = 3$

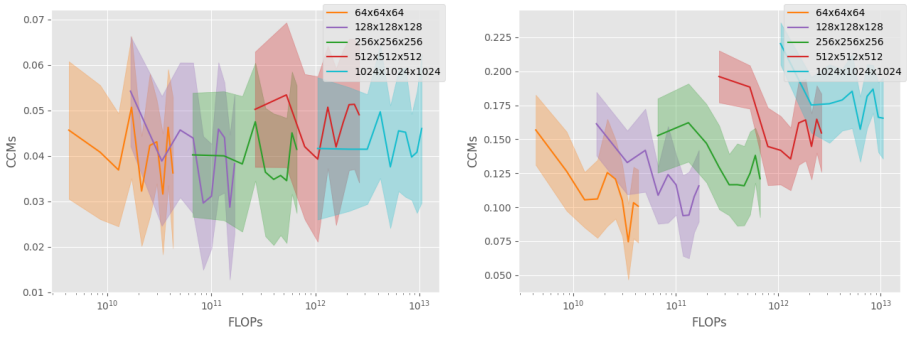
(c) $E = 4$ (d) $E = 5$

Figure 9: CCMs for varying values of E in fixed-depth experiments on Simple Tag

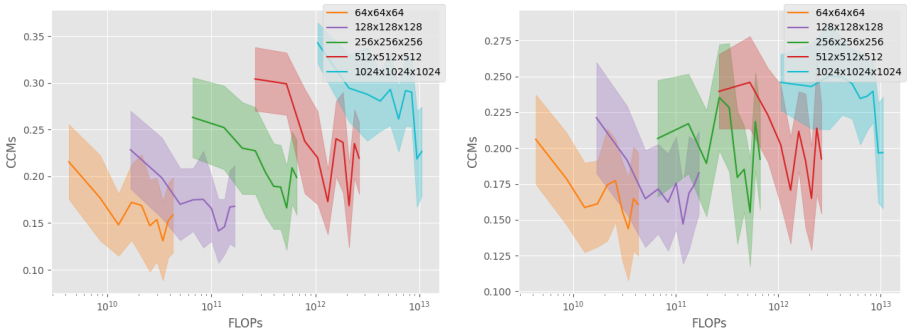
A.2 Fixed-width experiments on Simple Tag

A.3 Fixed-depth experiments on Simple Adversary

A.4 Fixed-width experiments on Simple Adversary

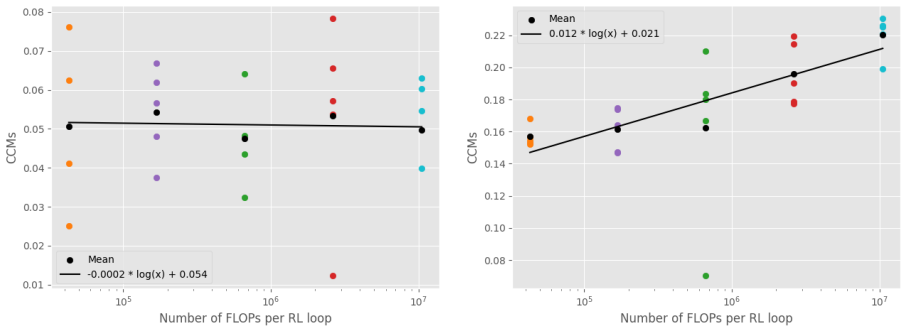


(a) $E = 2$ (b) $E = 3$

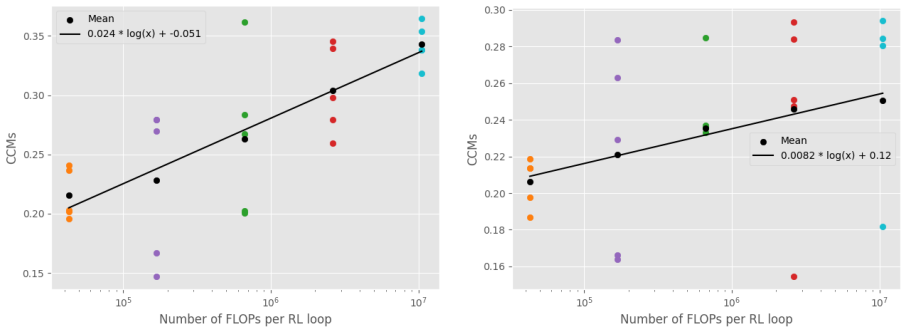


(c) $E = 4$ (d) $E = 5$

Figure 10: CCMs w.r.t. FLOPs for varying values of E in fixed-depth experiments on Simple Tag

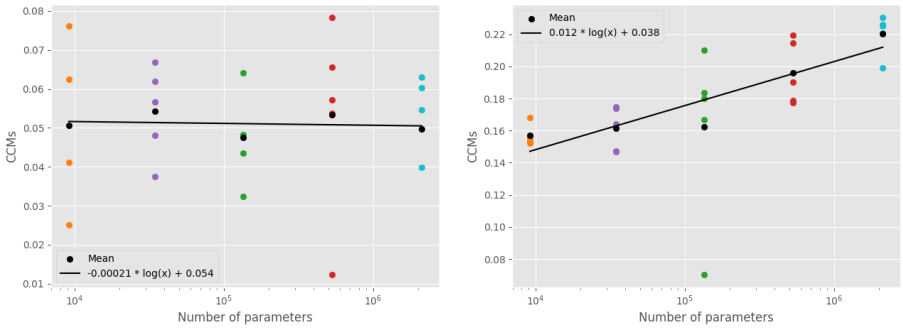


(a) $E = 2$ (b) $E = 3$



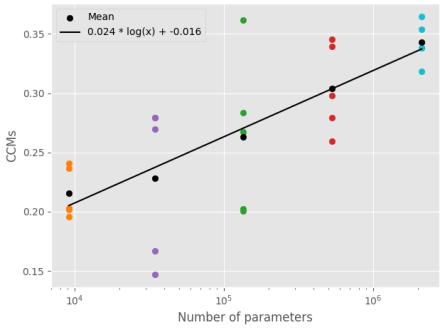
(c) $E = 4$ (d) $E = 5$

Figure 11: Regression fit for best CCMs across checkpoints w.r.t. FLOPs for varying values of E in fixed-depth experiments on Simple Tag

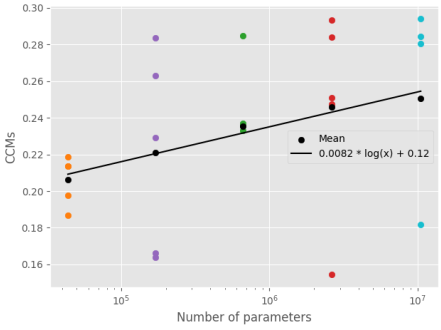


(a) $E = 2$

(b) $E = 3$

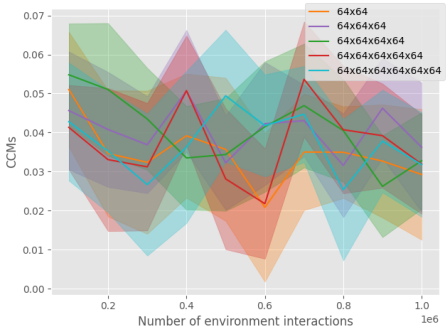


(c) $E = 4$

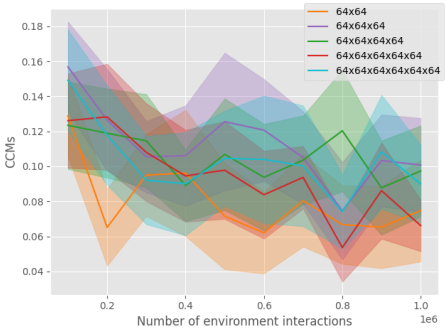


(d) $E = 5$

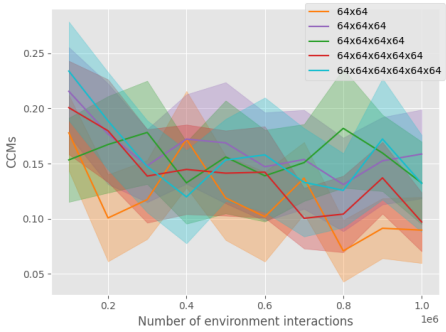
Figure 12: Regression fit for best CCMs across checkpoints w.r.t. number of parameters for varying values of E in fixed-depth experiments on Simple Tag



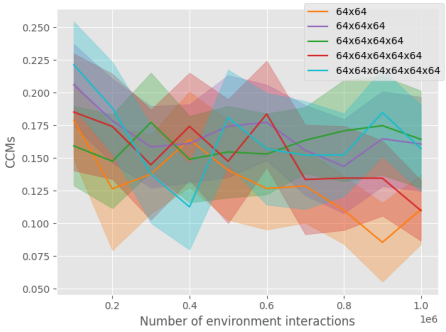
(a) $E = 2$



(b) $E = 3$

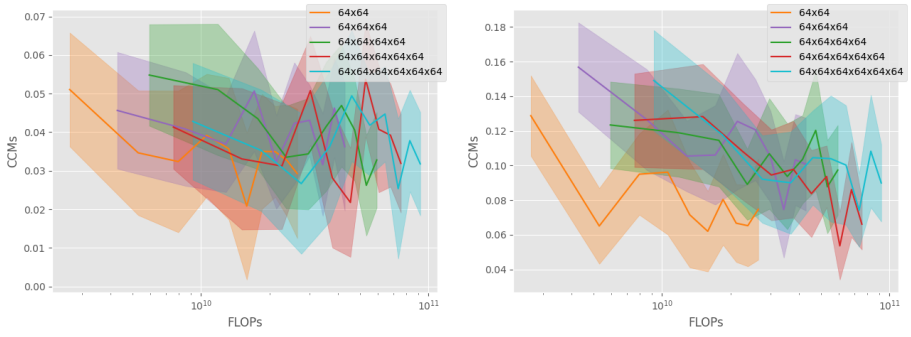


(c) $E = 4$

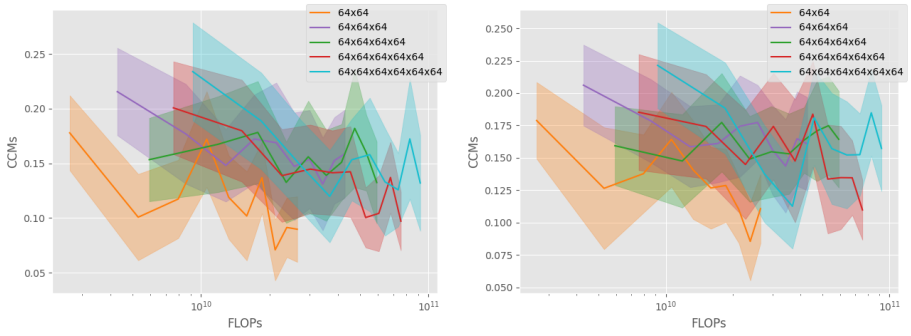


(d) $E = 5$

Figure 13: CCMs for varying values of E in fixed-width experiments on Simple Tag

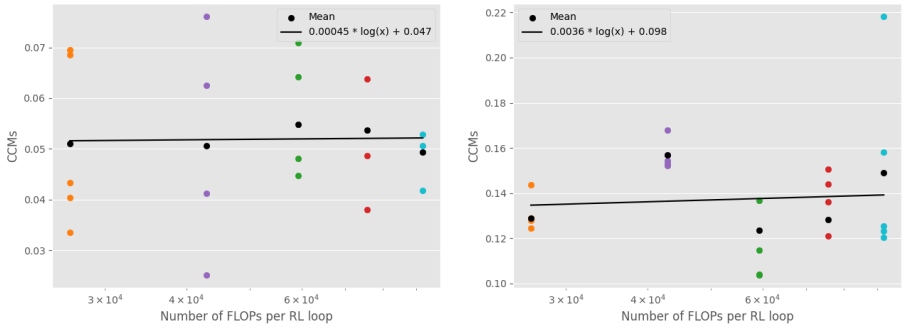


(a) $E = 2$ (b) $E = 3$

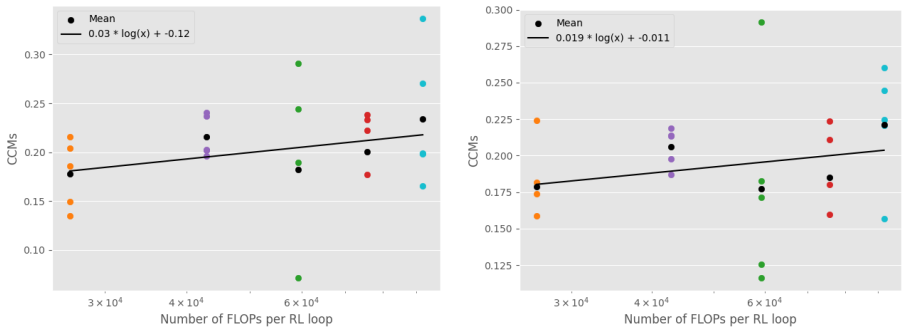


(c) $E = 4$ (d) $E = 5$

Figure 14: CCMs w.r.t. FLOPs for varying values of E in fixed-width experiments on Simple Tag

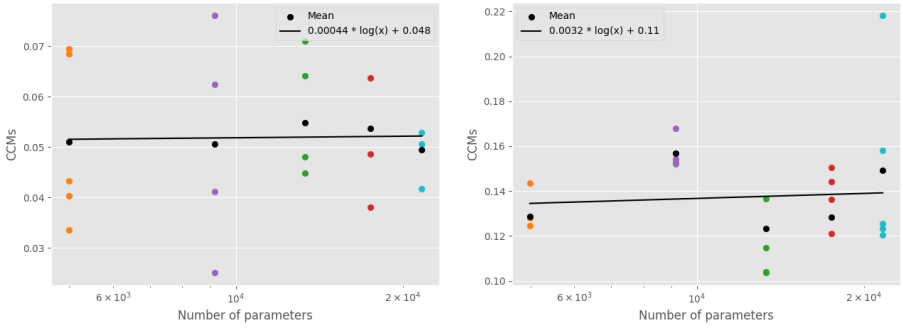


(a) $E = 2$ (b) $E = 3$

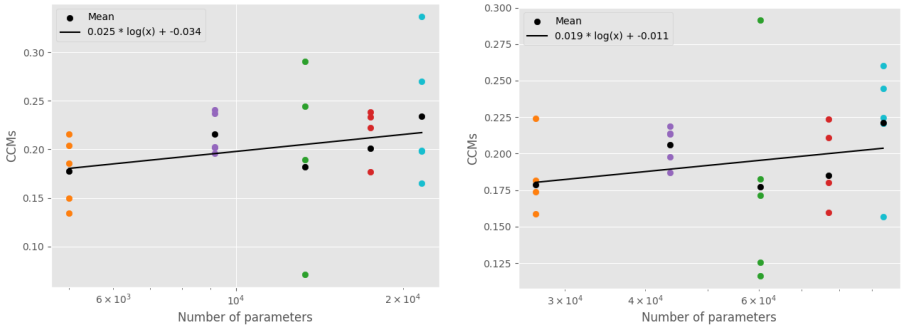


(c) $E = 4$ (d) $E = 5$

Figure 15: Regression fit for best CCMs across checkpoints w.r.t. FLOPs for varying values of E in fixed-width experiments on Simple Tag

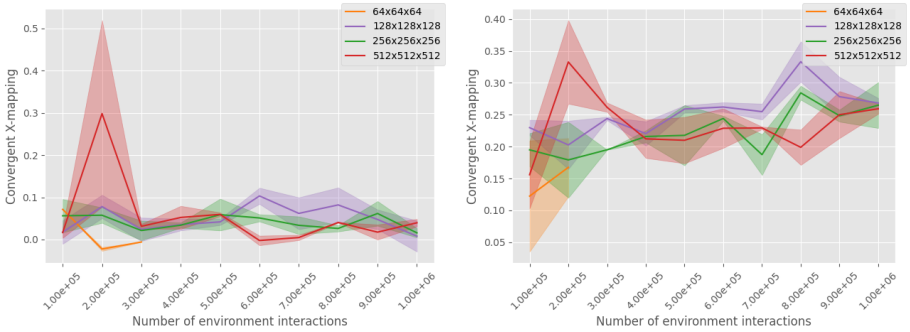


(a) $E = 2$ (b) $E = 3$



(c) $E = 4$ (d) $E = 5$

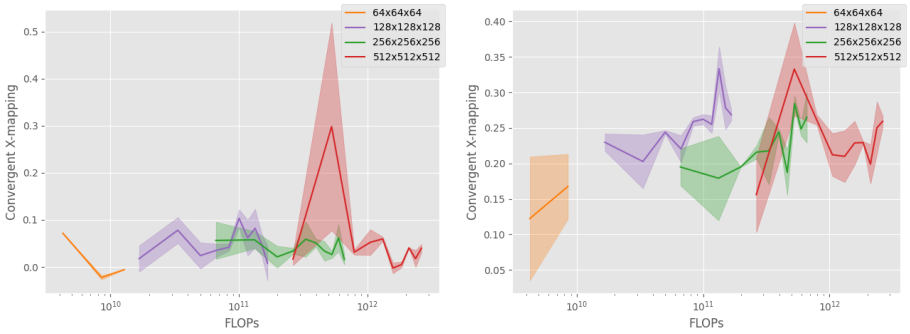
Figure 16: Regression fit for best CCMs across checkpoints w.r.t. number of parameters for varying values of E in fixed-width experiments on Simple Tag



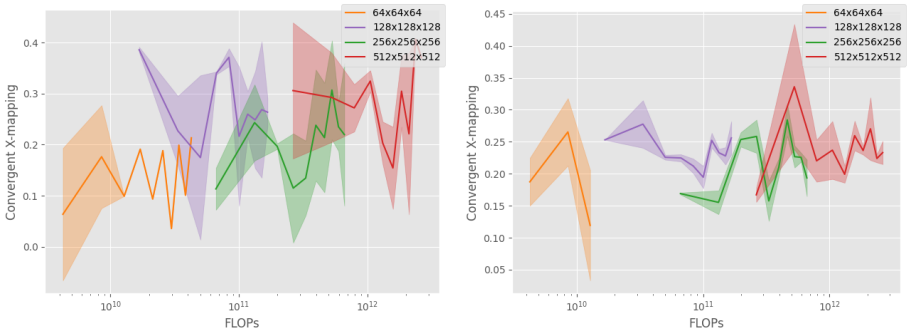
(a) $E = 2$ (b) $E = 3$

(c) $E = 4$ (d) $E = 5$

Figure 17: CCMs for varying values of E in fixed-depth experiments on Simple Adversary

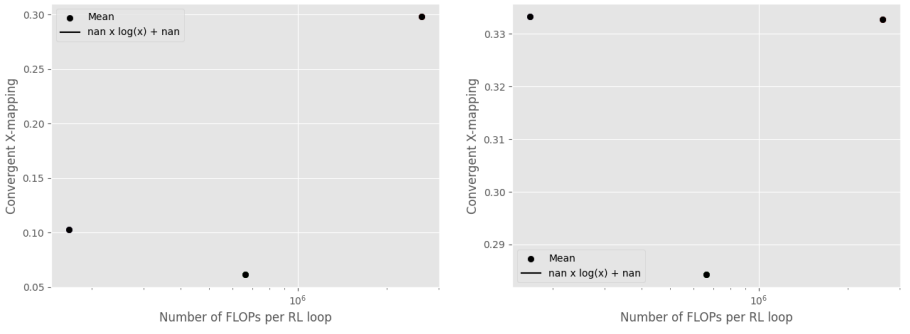


(a) $E = 2$ (b) $E = 3$

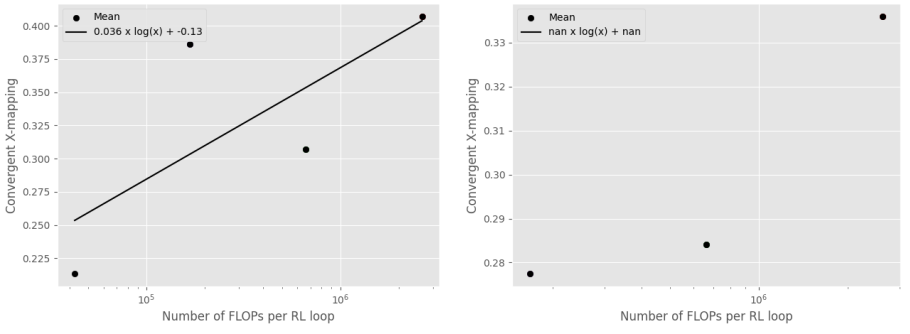


(c) $E = 4$ (d) $E = 5$

Figure 18: CCMs w.r.t. FLOPs for varying values of E in fixed-depth experiments on Simple Adversary

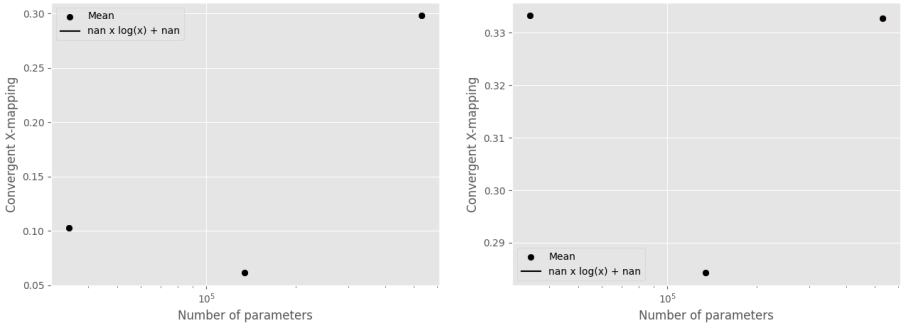


(a) $E = 2$ (b) $E = 3$



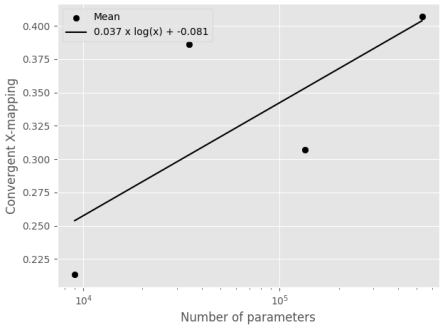
(c) $E = 4$ (d) $E = 5$

Figure 19: Regression fit for best CCMs across checkpoints w.r.t. FLOPs for varying values of E in fixed-depth experiments on Simple Adversary

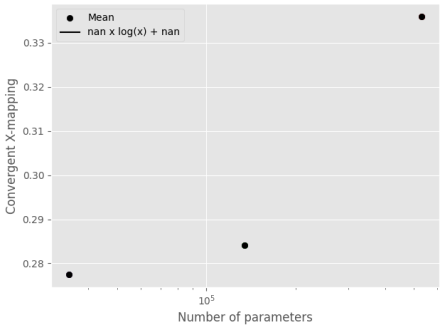


(a) $E = 2$

(b) $E = 3$

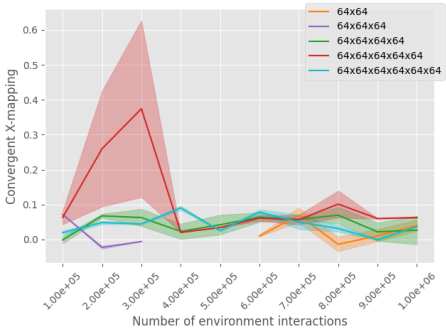


(c) $E = 4$

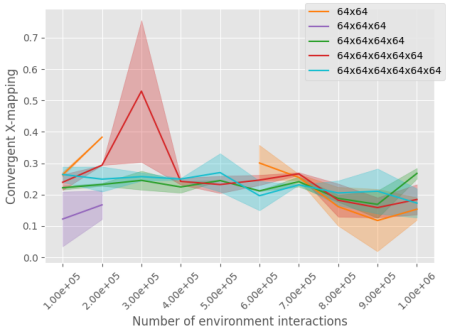


(d) $E = 5$

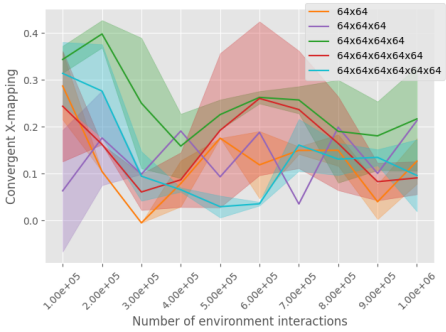
Figure 20: Regression fit for best CCMs across checkpoints w.r.t. number of parameters for varying values of E in fixed-depth experiments on Simple Adversary



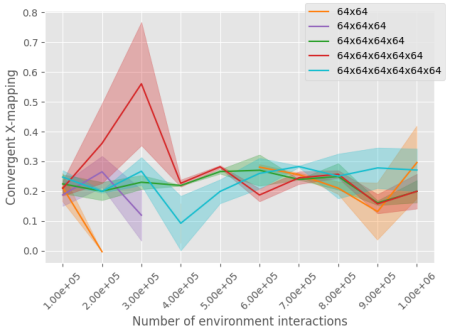
(a) $E = 2$



(b) $E = 3$

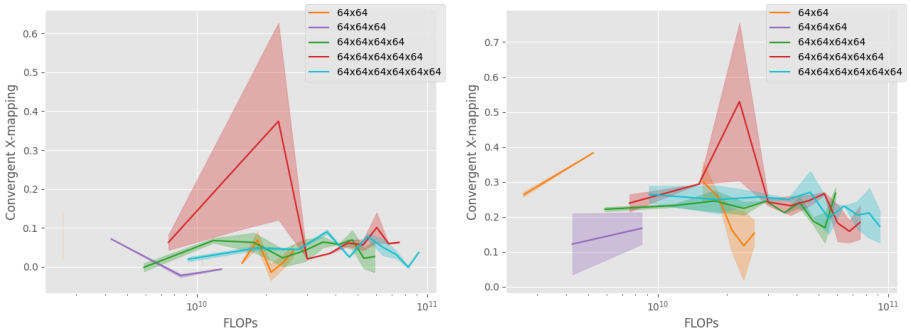


(c) $E = 4$

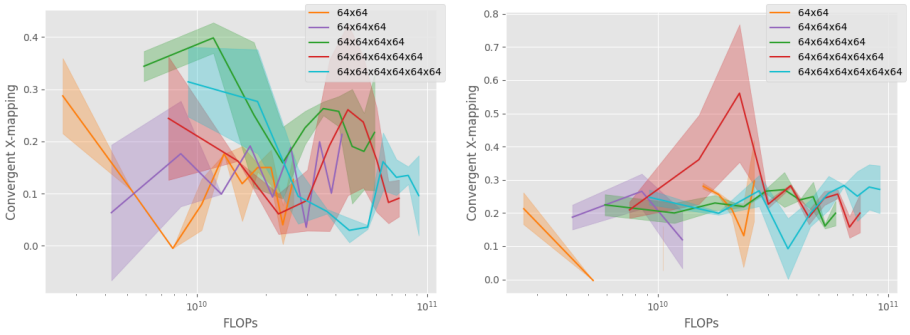


(d) $E = 5$

Figure 21: CCMs for varying values of E in fixed-width experiments on Simple Adversary

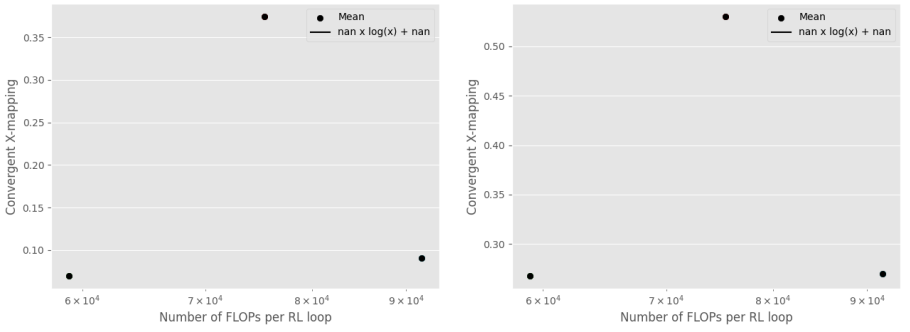


(a) $E = 2$ (b) $E = 3$

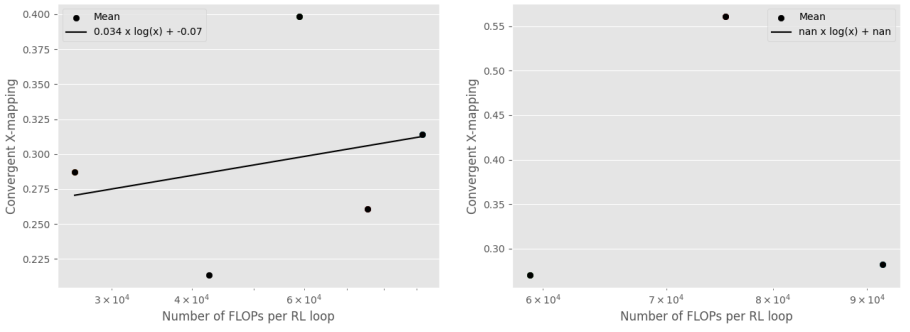


(c) $E = 4$ (d) $E = 5$

Figure 22: CCMs w.r.t. FLOPs for varying values of E in fixed-width experiments on Simple Adversary

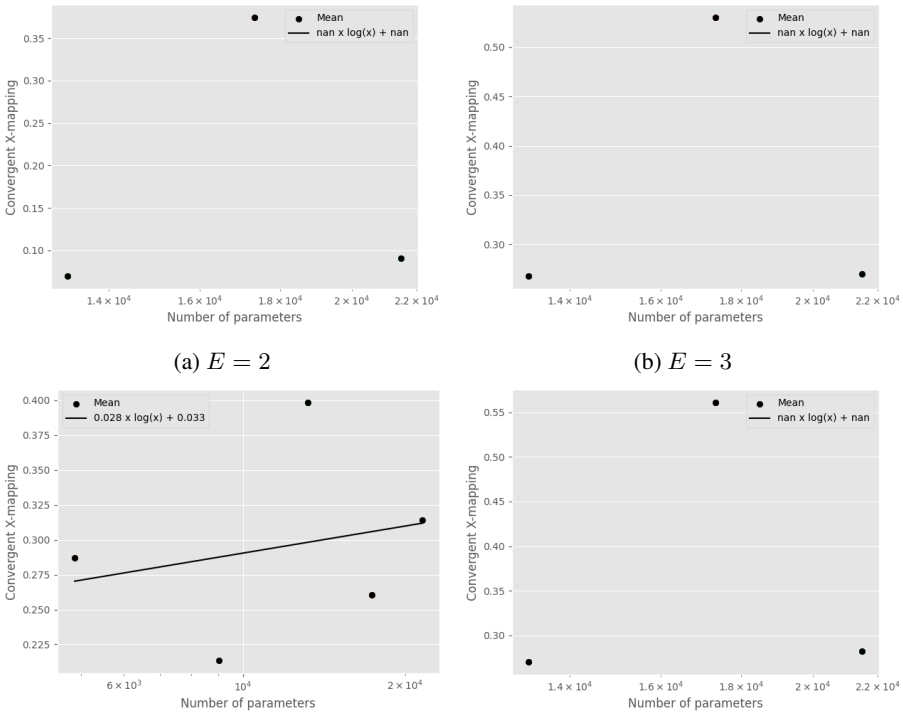


(a) $E = 2$ (b) $E = 3$



(c) $E = 4$ (d) $E = 5$

Figure 23: Regression fit for best CCMs across checkpoints w.r.t. FLOPs for varying values of E in fixed-width experiments on Simple Adversary



(a) $E = 2$ (b) $E = 3$

(c) $E = 4$ (d) $E = 5$

Figure 24: Regression fit for best CCMs across checkpoints w.r.t. number of parameters for varying values of E in fixed-width experiments on Simple Tag