Final Project Report: Adversarial Q-Learning to play Tag in Random Environments

Ian Sinclair
COMP 3501-1

November 22, 2022
Dr. Stephen Hutt

Abstract

Reinforcement learning is a common technique in the development of intelligent agents. In particular, many agents can take information about their surroundings to learn complex tasks such as playing chess or navigating distressed environments. However, as these tasks become more complicated it may become relevant to preferentially select informational attributes of the environment that most help in the learning process. For example, if an agent is given to much information is may be unable to discern what is important to completing the task, while if given to little information, the agent may miss something critical. Within the scope of this report, we examine state space reduction techniques that limit the amount of information an agent receives from its environment to improve learning speed. Fundamentally this utilizes a discrete Q-learning algorithm to manage policy generation over time. Our experiment is to design adversarial agents to compete in a game of tag over a randomly generated environment. The randomness of the game environment makes it more difficult for the agents to learn adaptive search and/or evade policies and highlights the effectiveness of our state space reduction technique.

1 Outline

Two agents are designed to compete in a game of tag. The game is played on a rectangular grid where each game object can occupy a single space. Aside from the agents, there are walls that block movement. In general agent 1 (seeker) is tasked with seeking out agent 2 (runner) and occupying that same space. This results in agent 1 winning the game and getting a reward and agent 2 losing reward. Additionally, if agent 2 survives 200 game steps the game ends. This will evolve two learning policies for the different agents that are optimized in competition with each other. After each game, a new random environment is created and the agents are allowed to play again. This continues for $100,000$ epocs. The randomness contributes to adaptive policies and allows for the agents to be placed in any environment and still preform fairly well in terms of meeting their distinct objectives. And as a consequence of this, after training the agents are placed in very complex more principally generated terrains to test the strength of their policies.

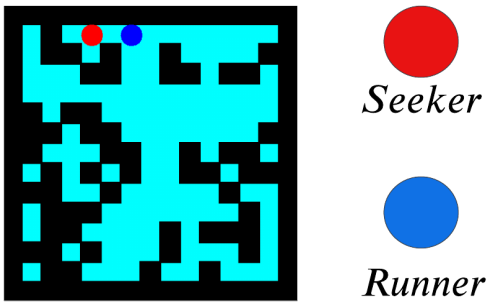

Figure 1: Concept art: example of randomly generated game board. Along with labeled seeker and runner agents.

1.1 Purpose

Within the scope of this report, techniques to encode environment information is considered. In particular, we seek to find a sufficiently small encoding of each game state to increase training speed and lower model complexity while still offering a 'good enough' agent policy.

1.2 Evaluation and Success

The game ends after agent 1 tags agent 2 or after a certain preset max game length. And so we can use the game length as a measure of success or failure for both agents. In general averaging over a large number of trials, if the game lengths show statistical deviation from random motion of both agents it is a good indicator of learning. Note, that if this deviation is strongly negative, it would be clear that the seeker is learning. However, if the deviation is strongly positive then either the runner could be learning or both agents could be thrashing. And so a second measure is added so examine cases of thrashing. In particular, we take the normalized average distance between the two agents in each game averaged across a 1000 game window.

$$H_i = \frac{max(distance(seeker \rightarrow runner) - min(distance(seeker \rightarrow runner)}{max(distance(seeker \rightarrow runner)}$$

For a particular game instance i. Generally if agents are thrashing this value will be small because the minimum distance between the agents will resemble the maximum distance. While, if the agents are attempting to move closer together, this value will tend towards 1. And so the larger the value H_i the better.

2 Formalism

2.1 Markov Decision Process

Reinforcement learning seeks to find an optimal control policy to navigate an environment or complete some task. This is typically done by learning connections between game states through repeatedly preforming actions and getting different amounts of reward. In general this is based on the Markov Decision process. Here let $\mathcal{M} = (S, A, \rho, R)$; where,

- S : is the set of all states the agent is capable of encoding in the environment.
  - $s \in S$ a particular state s in the state space is essentially a snap shot of the agents environment at a particular time step. As the agent moves it transitions between states.
- A : the set of actions available to the agent at a particular state. In our case, the set of actions at any state is 8 coordinate transitions, North, North-East, East,... etc.
- ρ : Is a transition model depicting the probability that the agent will move from state s to new state s' using action a. Or rather specifies the map, $\rho : S \times A \times S \rightarrow [0, 1]$.[Parr and Russell(1997)] Note this is usually not known and so is learned or approximated over time.
- R : This is a reward function that maps, $S \times A \times S \rightarrow \mathbb{R}$, or in general given a state action and new state triple determines the amount of reinforcement the agent should receive for that decision. This defines the optimization portion of the algorithm and determines the type of strategy each agent learn. And is generally controlled by the function, $R_a(s, s') \in \mathbb{R}$ for some $a \in A$, $s, s' \in S$. [Xia et al.(2019)Xia, Di, Guo, and Li]

Generally speaking a reinforcement learning algorithm sets an agent to navigate its environment and attempt to find a set of actions or policy, π, of behavior that maximizes the agents total reward. And so for a particular state s, is interested in the perceived value of traveling to state s' with a particular action α.

2.2 Policy Generation

Let π be a behavioral policy for an agent in an environment. Then the value associated with π at state s can be found by,

$$V_0^\pi = 0$$
$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} \rho(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$

Where γ is a discounting rate for previous experiences. Where,

$$\pi^* = \arg\max_{\pi} V^{\pi}(S)$$

And, for some state $s \in S$ an optimal action is selected by,

$$\alpha^* = \pi^*(s) = \arg\max_{a} V^{\pi^*}(s)$$

Importantly, notice that for this method ρ and R must be known and well-defined. Which is not always the case. And for our tag game problem, because both agents are moving calculating state transitions is likely untenable, and so a different version of policy generation is used using online learning (testing trial and error approach).

2.3 Q-Learning

Q-learning is a type of reinforcement learning that relaxes the need to maintain the exact reward value of each state action pair, and rather encodes a quantity conducive of finding an optimal policy but not representative of the actual rewards. In general this is done by maintaining a look up Q-table of every state action pair, (s, α), pointing to s Q-value.

$$\pi^*(s) = \arg\max_{a \in A} Q(s, a)$$

The value of $Q(s, a)$ is populated by updates from online learning or rather by trying an action an incorporating the receive reward/punishment into the associated Q value. This is done by sampling with the Bellman equation.

$$sample \leftarrow R(s, a, s') + \gamma \max_{a'} Q(s', a')$$

Where $R(s, a, s')$ is the reward from moving from state s to s' using action a. And $\max_{a'} Q(s'a')$ represents the best estimated reward leaving state s'. And so build a transition between state s,s' and the next state after s'. From here, the Q value, $Q(s, a)$ is updated with the sample.

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha[sample]$$

Where $\alpha \in [0, 1]$ is a learning rate that prioritizes newer samples. And so for higher α the algorithm will weight rewards it has experiences more recently higher than older experiences. Which can be useful if the reward function, R changes with time. Because the reward structure for out game of tag is sparse, (agents only receiving reward when they win or lose the game) we keep the learning rate low, to prevent 0 reward states that contribute to optimal behavior from deteriorating.

2.4 $\epsilon-$greedy Strategy and Exploration vs. Exploitation

An important debate in any online reinforcement learning algorithm is whether to move randomly to better populate the far reaches of the Q table, or to exploit the best found policy and learn better efficiencies. Generally, this is solved using an ϵ greedy strategy that forces the agents to move with a high degree of randomness early on in training then slowly begin add more principled movements later. This strikes a balance between searching for new strategies and improving the ones that have already been found. Our implementation begins with 50% random movement that linearly decreases to 0% at the last training game.

3 Adversarial Reinforcement Learning

From here, adversarial Multi-Agent Reinforcement Learning (MARL) is a technique that allows multiple agents to interact with each other and the environment at the same time during training. And so ideally agents will learn policies relative to each other and improve more quickly. This can make the learning task more complicated and is still somewhat unexplored. Fortunately much of the construction from single agent reinforcement learning can be transferred to multi-agent. This section is is a brief review of current and concerns techniques of MARL and how they are adapted to our game environment.

First, define a stochastic multi-agent game environment by, the vector, $< N, S, \{A^i\}_{i=1,\cdots,N}, P, R, \gamma >$ where N is the number of agents, S is the state space shared by all agents, A^i is the set of actions available to agent i. And P, R, γ are the transition and reword functions along with the discounting rate respectively. [Yang and Wang(2020)] This construction allows the MDP and Q table to solved similarly to a single agent system.

From here we consider another attempt towards MARL in a game of tag from [Raj and Salim(2022)] which used deep reinforcement learning to find an optimal policy between the seeker and runner agents. In the environment of established by [Raj and Salim(2022)] agents where highly successful; however, their altering their environment constituted retraining and so were less focused on adaptive. In addition, their deep learning model seemed significant and would likely take a long time to train. Here we examine the minimum sized model in terms of state space size that can find a 'good enough' policy, or the smallest state space size that is still able to support learning from the agents.

4 State Space Reduction

Basic Q-learning is a reinforcement learning strategy that maintains a memory of every state an agent has visited in a Q-table. Along with a metric for the strength of each action from that state. This means that without using some approximation function every state relevant to the agents policy must be experienced during training. This means that for complex environments the necessary amount of training to accommodate sufficient learning for each state may be untenable. And so generally, smaller state spaces are preferred as long as they still allow for convergence to an optimal policy. Correspondingly, it may be possible to create a smaller abstraction of a larger state space that retains much of the same utility in a much more compact form. This would involve representing a large subset of states, $C \subseteq S$, as a single state, $C \simeq \phi$, where a single state ϕ is sufficiently representative of all the elements of C to ensure the same optimal policy is found. From here, if such an abstraction exists for all distinct partitions of S, then $\Phi = \{\phi_1, \phi_2, \cdots \phi_n\} \simeq S$. In general such an abstraction requires,

- 'The difference between the transition function and reward function in both models has to be a small value.'
- 'For each policy on the original state space there must exist a policy in the abstract model. And if a state s is not reachable from state s' in the abstract model, then there should not exist a policy that leads from s to s' in the original state space.'[Asadi and Huber(2004)]

From here we consider a state space construction from our game of tag.

4.1 State Space Cardinally

Critically, our implementation of tag as a grid is a perfect information game where both agents can have access to all the information in the entire environment. And so let S be a maximum state space where $\forall s \in S$, s maintains as much information about the game as possible. Assuming an admissible Markov Decision Process, S ensures that an optimal policy, π^*_S can be found. Here because the environment of our tag game is random, each state s would be a snapshot of the entire tag game board including the position of every wall and enemy agents. Unfortunately, for large games, the amount of states needed to population the look-up table would be exponentially large. For example, consider a 10×10 grid, where each grid tile can assume one value of $\{wall, empty, agent\}$ with independent uniform probability. Then the number of possible grid configurations is $|S| = 3^{10\times 10} = 5.1537752 \times 10^{47}$, which is untenable. And so encoding a new state space, Φ that is representative of S can be beneficial.

4.2 Partitions

Again let S be an original state space in the Markov decision process $\mathcal{M} = (S, A, \rho, R)$, then consider a partition, P, of S such that, $P = \{B_1, B_2, \cdots, B_m\}$ where B_i is a cluster of states in S, and P saturates S. Then by, [Dean and Givan(1997)] consider the following,

Definition 1 For any initial partition P there exists a unique coarsest homogeneous refinement of P

That is to say that a refinement P^* of P, distributes the elements of S into distinct blocks B_i such that each B_i is optimally representable by a reduced (or constant) number of states under some transformation function h. Or rather, $B_i \simeq c_i$ and $\bigcup_i B_i = S$.

From here, a central idea behind our state space representation is the notion of stable partitions.

> For some block C in a partition P, C is stable with respect to some other block $B \in P$, $B \neq C$, and action $\alpha \in A$, iff every state in C has the same probability of moving into block B with action α.[Dean and Givan(1997)]

$$\exists c \in [0,1], \forall p \in C, \mathbf{Pr}(X_{t+1} \in B | X_t = p, U_t = \alpha) = c$$

there is some $c \in [0, 1]$ where if the current state $X_t = p \in C$, then, the probability of moving into block B is the same across all $p \in C$.
Then C is stable if $\forall B_i \in P$, C is stable with respect to B. [Dean and Givan(1997)]

Which seems to have an obvious extension, that the information of block C can be packaged into a single element. Let $\phi \in C$ be an element of C where C is stable. Then,

$$\forall c' \in C, s, \alpha \in S \times A, \quad \mathbf{Pr}(X_{t+1} = s | X_t = c', U_t = \alpha) = \mathbf{Pr}(X_{t+1} = s | X_t = \phi, U_t = \alpha)$$

And so we can encode all the information of C into a single element ϕ. Next define the transformation function, h, by $h(C) = \phi$ or rather take ϕ as a generator such that, $C \simeq \phi$.
Now, if a refined partition P is stable, or for all $B_i \in P$, B_i is stable, then of course,

$$P = \{B_1, B_2, \cdots, B_m\} \simeq \{\phi_1, \phi_2, \cdots, \phi_m\} = \Phi$$

And because P is a partition of S, such that, $\bigcup_i B_i = S$ it follows that,

$$h(S) \subseteq \Phi$$
$$S \simeq \{\phi_1, \phi_2, \cdots, \phi_m\}$$

Which encodes the information of S into m generating states.

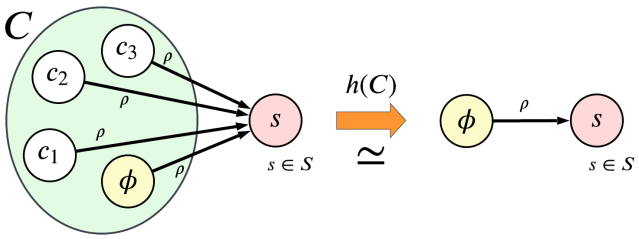

Figure 2: Concept Art: Minimal packing of stable block C with representative ϕ and equivalent transformed expression ϕ to $s \in S$.

The article [Dean and Givan(1997)] continues to establish methods to find refined and stable partitions for Bayesian networks and reinforcement learning algorithms; however, for the scope of this report we are only interesting in establishing the foundation for state space reduction and demonstrate while smaller abstractions may still be representative of the larger state space.

4.3 State Space of Tag

To that respect we are less interested in a theoretically admissible state space encoding that corresponds to the purely optimal policy. And alternatively, are focusing on a minimal state space representation that is still capable of demonstrating principled strategies. However, we use principals in block transition representation to guide our construction of state spaces. Consequently, we consider three levels of abstraction that can be combined to make an admissible reduced state space. Each level is then tested in online training to examine the quality of policy generation. And so note the three independent abstractions.

- Grid

  This encodes the immediate surroundings of each agent by centering a 3×3 grid at their location. Where each grid tile provides the agent with the game object incident with that location. Notice that this means agents have no information on the location of their target until they are right next to each other.

- Quadrant Mapping

  Quadrant Mapping allows each agent to know the general location of their target. If they are either North-East, North-West, South-East or South-West of the agents current location.

- Time Step Decoupling

Because of the sparseness of the reward system and the limitations on the diminsionality of each state it is likely that many states will be repeated along a pack to an agents target. Then, because these states are reached at different times they may favor different actions. However, because Q learning treats every distinct state s the same, these states would have the same policy driven action. And so time stamps are added to each state referencing the amount of times that state has been visited in a single game.

$$s_0 = 0$$
$$s_{t+1} = s_t + 1 \mod 200$$

Where $s \in S$ is some state, and t is the number of times that state has been visited in a particular game. This allows for the Q table to treat states reached at different times independently and so an optimal policy can distinguish 'good' actions between these positions. Note, in the event of an infinite game time, with no natural stopping condition, this time stamping could diverge. This is why we added mod 200, to ensure that our state space remains discrete. (200 is the current max game length.)

These three abstractions make up the experimental portion of the project. Or rather we test each principled combination of abstractions against random movement to determine how small our state space can be to still facilitate learning. Critically, every experiment level contains the first layer abstraction (grid). And so groups are as follows.

- Grid
- Grid + quadrant mapping
- Grid + time stamp decoupling
- grid + quadrant mapping + time stamp decoupling

The results of which are analyzed in terms of the success parameters mentioned in the outline section.

5 Results

Here we consider the response of each state space representation when trained in the same random environment for $100,000$ epocs. Then, evaluation results are accumulated for an additional $100,000$ games after training is complete. Primarily we focus on average game length over 1000 game intervals and so take 100 data points from the evaluation epocs. Similarly, the average displacement of the agents is considered. Where for a particular game, i, let t be any time step in that game. Then, d_t be the distance between the agents at time step t. Then the average distance over 1000 game windows is,

$$mean(h_i)_{i=1,2,3,\cdots,1000}, \qquad h_i = \frac{\max d_t - \min d_t}{\max d_t} \in [0,1]$$

And so notice that in the event of thrashing this value will be close to 0. And for optimal play, the seeker should always be able to close the distance on the runner and so this value should be closer to 1.

5.1 Abstraction Level 1

Here the agents are only allowed to use a 3×3 grid to encode each state. Correspondingly, consider the average game length chart, Notice here the agents are moving similarly to random chance and are able to find each other (and end the

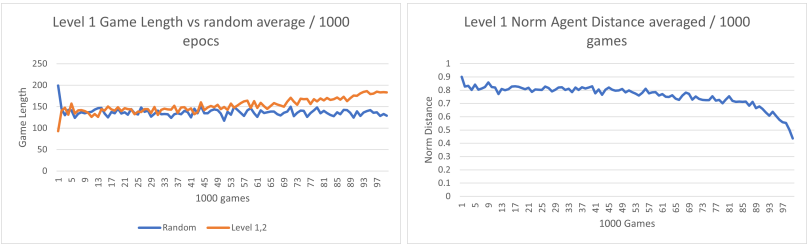

Figure 3: (1)Displays game length vs random agents averaged per 1000 games. (2) Image displaying the normalized change in distance between the agents averaged on 1000 game windows.

game) about as often as the random agents. This is consistent with expectations because the agents have no ability to know which part of the environment their target is in.

5.2 Abstraction Level $1, 2$

Here the agents are only permitted to use the grid and the quadrant map detailing vaguely where their target is.

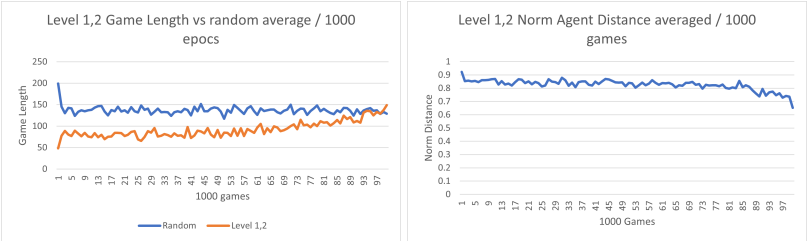

Figure 4: Image displaying the normalized change in distance between the agents averaged on 1000 game windows.

Here notice that the seeker is initially able to track down the runner much faster than random chance. However, trends upwards. In addition the average distance graph demonstrates that the seeker is able to consistently close ground on the runner, as would be expected if the games are consistently able to be ended early.

5.3 Abstraction Level $1, 3$

In this level the agents are given a 3×3 grid along with time stamps decoding allows duplicated states in a single game to be treated as different states. Consider the game length diagram.

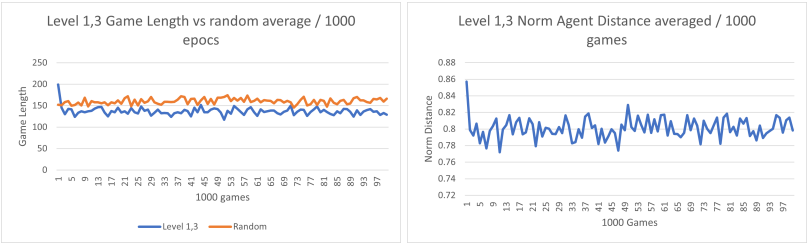

Figure 5: (1)Displays game length vs random agents averaged per 1000 games. (2) Image displaying the normalized change in distance between the agents averaged on 1000 game windows.

Notice here that the average game time is slightly below random collision implying that the seeker's strategy is potentially giving it an advantage. Also consider the amount of distance the seeker is able to close per game. Additionally notice here that the normalized distance closed by the seeker in averaged over many games is close to 1 which implies that this agent is still able to move closer to the runner. However, this may also be due to the seeker favoring random motion over thrashing which would statistically bring the agent closer to its target.

5.4 Abstraction Level $1, 2, 3$

Here we allow states to be encoded with a grid, a quadrant map, and a time sequence. First consider the plot,

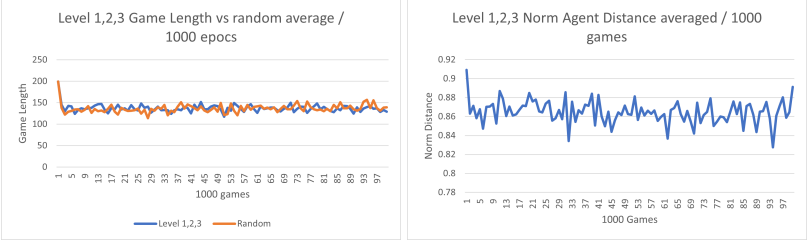

Figure 6: (1) Displays game length vs random agents averaged per 1000 games. (2) Image displaying the normalized change in distance between the agents averaged on 1000 game windows.

Note the average game length for the policy holding agents approaches that of random agents implying that the runner is preforming well in correlation to the seeker. Now consider the plot of average change over time. Here note that the normalize distance tends closer to 1 demonstrating that the seeker is correctly tracking the runner. This is a good indicator that the agents are not thrashing.

6 Conclusions

The scope of the experiment focused on multi-agent reinforcement learning for a game of tag. In which two agents learn policies in proxy of each other, ideally resulting is learning mechanisms that are not easily expressed by the single agent reward function. In addition to this we allow our game environments to vary is both size and density contributing to the agents motivation to learn adaptable strategies. Critically our evaluation metric used average game length across many trials to determine the quality of the runner/seeker pair; then, used normalized change in distance averaged between games to look for thresholding. Unfortunately, these metrics proved ineffective, because the conclusions made by the data is not consistent with observing the agents policies. For example, most levels of state abstraction that were tested where able to at least meet game endings by random chance, which would either imply that the agents are moving randomly, or moving on a slight improvement to random. However, it can be noted that abstraction level $1, 3$ thrashes consistently, and abstraction level $1, 2, 3$ finds a policy that is visually much better to random however, the games are still prolong because the relative skill of both agents is similar. And so a better casting of the game length metric may be to determine if two agents have equivalent skill. If they meet at the level of random collision it is likely they are both capable of performing at least semi-correct actions at each step. If one agent had an 'edge' over the other this metric would likely deviate from random collision. It would be an interesting follow up experiment to examine the policy strength of agents competing against different levels of state abstraction. The second metric also needs to be improved to re-factored. Critically, this metric can only be used to detect thrashing, and not agent policy strength. This is because the randomly moving agents also maintained a very high score (close to 1) on this metric. Indicating that random movement does not fall into the realm of thrashing detectable by this metric. However, also does represent the lowest control policy for the agents, or rather a set of behaviors the agents must be able to beat to prove they are learning better policies.

References

[Asadi and Huber(2004)] M. Asadi and M. Huber. State space reduction for hierarchical reinforcement learning. 01 2004.

[Dean and Givan(1997)] T. L. Dean and R. Givan. Model minimization in markov decision processes. In *AAAI/IAAI*, 1997.

[Parr and Russell(1997)] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10. MIT Press, 1997. URL https://proceedings.neurips.cc/paper/1997/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[Raj and Salim(2022)] R. Raj and A. Salim. Deep multi-agent reinforcement learning for tag game. In G. Sanyal, C. M. Travieso-González, S. Awasthi, C. M. Pinto, and B. R. Purushothama, editors, *International Conference on Artificial Intelligence and Sustainable Engineering*, pages 7–18, Singapore, 2022. Springer Singapore.

[Xia et al.(2019)Xia, Di, Guo, and Li] W. Xia, C. Di, H. Guo, and S. Li. Reinforcement learning based stochastic shortest path finding in wireless sensor networks. *IEEE Access*, 7:157807–157817, 2019. doi: 10.1109/ACCESS.2019.2950055.

[Yang and Wang(2020)] Y. Yang and J. Wang. An overview of multi-agent reinforcement learning from game theoretical perspective, 2020. URL https://arxiv.org/abs/2011.00583.