Turbocharging Solution Concepts: Solving NEs, CEs and CCEs with Neural Equilibrium Solvers

Anonymous Author(s)
Affiliation
Address
email

Abstract

Solution concepts such as Nash Equilibria, Correlated Equilibria, and Coarse Correlated Equilibria are useful components for many multiagent machine learning algorithms. Unfortunately, solving a normal-form game could take prohibitive or non-deterministic time to converge, and could fail. We introduce the Neural Equilibrium Solver which utilizes a special equivariant neural network architecture to approximately solve all games of fixed shape, buying speed and determinism. We define a flexible equilibrium selection framework, that is capable of uniquely selecting an equilibrium that minimizes relative entropy, or maximizes welfare. The network is trained without needing to generate any supervised training data. We argue that such a network is a powerful module for many possible multiagent algorithms.

1 Introduction

Solution concepts such as Nash Equilibrium (NE) [28], Correlated Equilibrium (CE) [3], and Coarse Correlated Equilibrium (CCE) [27] are useful components for many machine learning algorithms [13]. They are used in Double Oracle [26], PSRO [21], and JPSRO [23] to determine what distributions of co-players agents should train with. Furthermore equilibria have been critical in driving agents to superhuman performance in Go [36], Chess [37], and StarCraft [41]. Unfortunately, solving for an equilibrium of a game can be computationally complex [7, 6], and often requires deploying iterative solvers, which can scale but may (i) take a non-deterministic amount of time to converge, (ii) take a prohibitive amount of time to do so, and (iii) may fail unpredictably on ill-conditioned problems. Furthermore, classical methods [12, 22, 25] (i) do not scale, and (ii) are non-differentiable. This limits the applicability of equilibrium solution concepts in multiagent machine learning algorithms.

We propose the Neural Equilibirum Solver (NES), a neural network that approximately solves for such solutions, allowing equilibria to be found quickly and in a deterministic amount of time. This network is trained to optimize a composite objective function that weights accuracy of the returned equilibrium against auxiliary objectives that a user may desire such as maximum entropy, maximum welfare, or minimum distance to some target distribution. We introduce several innovations into the design and training of NES so that it is efficient and accurate. Unlike most supervised deep learning models, NES avoids the need for explicitly constructing a labeled dataset of (game, equilibrium) pairs. Instead we derive a loss function that can be minimized in an unsupervised fashion from only game inputs. We also exploit the duality of the equilibrium problem. Instead of solving for equilibria in the primal space, we train NES to solve for them in the dual space, which has a much lower dimensional representation. We utilize a training distribution that efficiently represents the space of all normal-form games of a desired shape and use an invariant preprocessing step to map games at test time to this space. In terms of the network architecture, we design a series of layers that are equivariant to symmetries in games such as permutations of players and strategies, which reduces

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the number of training steps and improves generalization performance. This network can either be pretrained before being deployed, trained online alongside another machine learning algorithm, or a mixture of both.

2 Preliminaries

Game Theory Game theory is the study of the interactive behaviour of rational payoff maximizing agents in the presence of other agents. The environment that the agents operate in is called a game. We focus on a particular type of single-shot, simultaneous move game called a *normal-form* game. A normal-form game consists of N players, a set of strategies available to each player, $a_p \in \mathcal{A}_p$, and a payoff for each player under a particular joint action, $G_p(a)$, where $a = (a_1, ..., a_N) = (a_p, a_{-p}) \in \mathcal{A} = \otimes_p \mathcal{A}_p$. The subscript notation $-p$ is used to mean "all players apart from player p". The distribution of play is described by a joint $\sigma(a)$. The goal of each player is to maximize their expected payoff, $\sum_{a \in \mathcal{A}} \sigma(a) G_p(a)$. Players could play independently by selecting a strategy according to their marginal $\sigma(a_p)$ over joint strategies, such that $\sigma(a) = \otimes_p \sigma(a_p)$. However this is limiting because it does not allow players to coordinate. A mediator called a *correlation device* could be employed to allow players to execute arbitrary joint strategies $\sigma(a)$ that do not necessarily factorize into their marginals. Such a mediator would sample from a publicly known joint $\sigma(a)$ and secretly communicate to each player their recommended strategy. Game theory is most developed in a subset of games: those with two players and a restriction on the payoffs, $G_1(a_1, a_2) = -G_2(a_1, a_2)$, known as zero-sum. Particularly in N-player, general-sum games, it is difficult to define a single criterion to find solutions to games. One approach is instead to consider joints that are in equilibrium: distributions such that no player has incentive to unilaterally deviate from a recommendation.

Equilibrium Solution Concepts Correlated Equilibria (CEs) [3] can be defined in terms of linear inequality constraints. The deviation gain of a player is the change in payoff the player achieves when deviating to action a'_p from a recommended action a''_p, when the other players play a_{-p}.

$$A_p^{\text{CE}}(a'_p, a''_p, a) = A_p^{\text{CE}}(a'_p, a''_p, a_p, a_{-p}) = \begin{cases} G_p(a'_p, a_{-p}) - G_p(a''_p, a_{-p}) & a_p = a''_p \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

A distribution, $\sigma(a)$, is in ϵ-CE if the deviation gain is no more than some constant $\epsilon_p \leq \epsilon$ for every pair of recommendation, a''_p, and deviation strategies, a'_p, for every player, p. These linear constraints geometrically form a convex polytope of feasible solutions.

$$\epsilon\text{-CE:} \qquad \sum_{a \in \mathcal{A}} \sigma(a) A_p^{\text{CE}}(a'_p, a''_p, a) \leq \epsilon_p \qquad \forall p \in [1, N], a''_p \neq a'_p \in \mathcal{A}_p \tag{2}$$

Coarse Correlated Equilibria (CCEs) [27] are similar to CEs but a player may only consider deviating before receiving a recommended strategy. Therefore the deviation gain for CCEs can be derived from the CE definition by summing over all possible recommended strategies a''_p.

$$A_p^{\text{CCE}}(a'_p, a) = \sum_{a''_p \in \mathcal{A}_p} A_p^{\text{CE}}(a'_p, a''_p, a) = G_p(a'_p, a_{-p}) - G_p(a) \tag{3}$$

A distribution, $\sigma(a)$, is in ϵ-CCE if the deviation gain is no more than some constant $\epsilon_p \leq \epsilon$ for every deviation strategy, a'_p, and for every player, p.

$$\epsilon\text{-CCE:} \qquad \sum_{a \in \mathcal{A}} \sigma(a) A_p^{\text{CCE}}(a'_p, a) \leq \epsilon_p \qquad \forall p \in [1, N], a'_p \in \mathcal{A}_p \tag{4}$$

NEs [28] have similar definitions to CCEs but have an extra constraint: the joint distribution factorizes $\otimes_p \sigma(a_p) = \sigma(a)$, resulting in nonlinear constraints[1].

$$\epsilon\text{-NE:} \qquad \sum_{a \in \mathcal{A}} \otimes_q \sigma(a_q) A_p^{\text{CCE}}(a'_p, a) \leq \epsilon_p \qquad \forall p \in [1, N], a'_p \in \mathcal{A}_p \tag{5}$$

[1] This is why NEs are harder to compute than (C)CEs.

Note that the definition of the NE uses the same deviation gain as the CCE definition. Another remarkable fact is that the marginals of any joint CCE in two-player constant-sum games, $\sigma(a_p) = \sum_{a_{-p}} \sigma(a)$, is also an NE, when $\epsilon_p = 0$. Therefore we can use CCE machinery to solve for NEs in such classes of games.

When a distribution is in equilibrium, no player has incentive to *unilaterally* deviate from it to achieve a better payoff. There can however be many equilibria in a game, choosing amongst these is known as the *equilibrium selection problem* [14]. For (C)CEs, the valid solution space is convex (Figure 1), so any strictly convex function will suffice (in particular, Maximum Entropy (ME) [30] and Maximum Gini (MG) [23] have been proposed). For NEs, the solution space is convex for only certain classes of games such as two-player constant-sum games. Indeed, NEs are considered fundamental in this class where they have powerful properties, such as being unexploitable, interchangeable, and tractable to compute. However, for N-player general-sum games (C)CEs may be more suitable as they remain tractable and permit coordination between players which results in higher-welfare equilibria. If $\epsilon_p \geq 0$, there must exist at least one NE, CE, and CCE for any finite game, because a NE always exists and NE $\subseteq$ CE $\subseteq$ CCE.

Neural Network Solvers Approximating NEs using neural networks is known to be PAC learnable [10]. Learning NEs [26] and CCEs [15] on a single game is well studied. There is also work learning (C)CEs [4, 19] and training neural networks to approximate NEs [10, 16] on subclasses of games. But, to the best of our knowledge, no work exists training a general approximate solver for the full space of games for (C)CEs with flexible selection criteria.

3 Maximum Welfare Minimum Relative Entropy (C)CEs

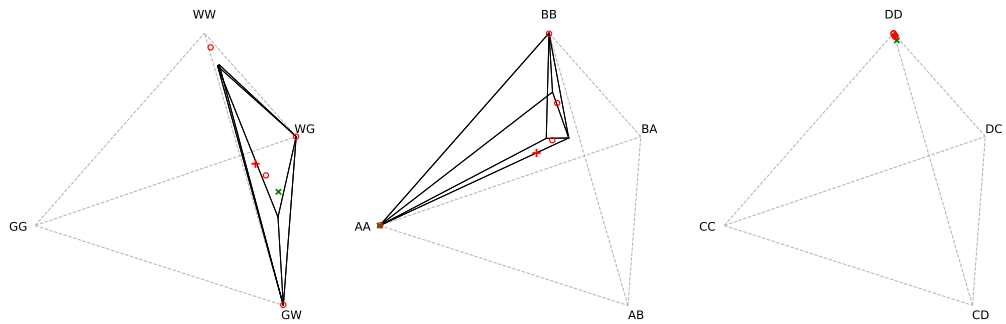
Previous work has argued that having a unique objective to solve for equilibrium selection is important. However justifying why a particular solution is better than another is thorny. Appeals to Occam's razor and the principle of maximum entropy have been used to justify entropy maximizing equilibria. In maximum entropy selection, payoffs are ignored and selection is based on minimizing the distance to the uniform distribution. This has two interesting properties: (i) it makes defining unique solutions easy (ii) the solution is invariant to offset and positive scaling of the payoff tensor. While these solutions are unique, they both result in weak and low payoff equilibria because they find solutions on the boundary of the polytope. Meanwhile, the literature tends to favour Maximum Welfare (MW) because it results in large payoffs for the agents and is a linear objective, however it is not unique in general. We consider a composite objective function composed of (i) Minimum Relative Entropy (MRE) between a target joint, $\hat{\sigma}(a)$, and the equilibrium joint, $\sigma(a)$, (ii) distance between a target approximation, $\hat{\epsilon}_p$, and the equilibrium approximation, ϵ_p, (iii) maximum of a linear objective, $W(a)$. The objective is constrained by the (i) distribution constraints ($\sum_a \sigma(a) = 1$ and $\sigma(a) \geq 0$) and, (ii) either CCE constraints (Equation (4)) or CE constraints (Equation (2)).

$$\arg\max_{\sigma,\epsilon_p} \mu \sum_{a\in\mathcal{A}} \sigma(a)W(a) - \sum_{a\in\mathcal{A}} \sigma(a) \ln\left(\frac{\sigma(a)}{\hat{\sigma}(a)}\right) - \rho \sum_p \left(\epsilon_p^+ - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon_p^+ - \epsilon_p}{(\epsilon_p^+ - \hat{\epsilon}_p)}\right) \tag{6}$$

The approximation weight, ρ, and welfare weight, μ, are hyperparemeters that control the balance of the optimization. The maximum approximation parameter, ϵ_p^+, is another constant that is usually chosen to be equal to the payoff scale (Section 4.1). The approximation term is designed to have a similar form to the relative entropy and is maximum when $\hat{\epsilon}_p = \epsilon_p$. We refer to this equilibrium selection framework as Target Approximate Maximum Welfare Minimum Relative Entropy ($\hat{\epsilon}$-MWMRE).

3.1 Dual of ϵ-MWMRE (C)CEs

Rather than performing a constrained optimization, it is easier to optimize the dual Lagrangian, $\arg\min_{\alpha_p} L^{\text{(C)CE}}$ (derived in Section A), where $\alpha_p^{\text{CE}}(a'_p, a''_p) \geq 0$ are the dual deviation gains corresponding to the CE constraints, and $\alpha_p^{\text{CCE}}(a'_p) \geq 0$ are the dual deviation gains corresponding to the CCE constraints. Note that we do not need to optimize over the primal joint, $\sigma(a)$. Choosing one



(a) Traffic Lights (b) Pure Coordination (c) Prisoner's Dilemma

Figure 1: Diagrams for three 2×2 normal-form games, showing their (C)CE solution polytope on the joint simplex. An MWME NES, trained by sampling over the space of payoffs and welfare targets, is used to approximate the MW(C)CE solution ($\times$). An MRE NES, trained by sampling over the space of payoffs and joint targets, is used to approximate the ME(C)CE ($+$), and all pure joint target MRE(C)CEs ($\circ$). The networks have never trained on these games.

of the elements in the curly brackets, the Lagrangian is defined:

$$\overset{\text{(C)CE}}{L} = \ln\left(\sum_{a\in\mathcal{A}} \hat{\sigma}(a) \exp\left(\overset{\text{(C)CE}}{l}(a)\right)\right) + \sum_p \epsilon_p^+ \left\{\sum_{a'_p, a''_p} \overset{\text{CE}}{\alpha}_p(a'_p, a''_p), \sum_{a'_p} \overset{\text{CCE}}{\alpha}_p(a'_p)\right\} - \rho \sum_p \overset{\text{(C)CE}}{\epsilon}_p \tag{7}$$

The logits $l^{\text{(C)CE}}(a)$ are defined:

$$\overset{\text{(C)CE}}{l}(a) = \mu \sum_a W(a) - \sum_p \left\{\sum_{a'_p, a_p} \overset{\text{CE}}{\alpha}_p(a'_p, a_p) \overset{\text{CE}}{A}_p(a'_p, a''_p, a), \sum_{a'_p} \overset{\text{CCE}}{\alpha}_p(a'_p) \overset{\text{CCE}}{A}_p(a'_p, a)\right\} \tag{8}$$

The primal joint and primal approximation parameters are defined:

$$\overset{\text{(C)CE}}{\sigma}(a) = \frac{\hat{\sigma}(a) \exp\left(\overset{\text{(C)CE}}{l}(a)\right)}{\sum_{a\in\mathcal{A}} \hat{\sigma}(a) \exp\left(\overset{\text{(C)CE}}{l}(a)\right)} \tag{9}$$

$$\overset{\text{(C)CE}}{\epsilon}_p = (\hat{\epsilon}_p - \epsilon_p^+) \exp\left(-\frac{1}{\rho}\left\{\sum_{a'_p, a''_p} \overset{\text{CE}}{\alpha}_p(a'_p, a''_p), \sum_{a'_p} \overset{\text{CCE}}{\alpha}_p(a'_p)\right\}\right) + \epsilon_p^+ \tag{10}$$

4 Neural Network Training

The network maps the payoffs of a game, $G_p(a)$, and the targets ($\hat{\sigma}(a)$, $\hat{\epsilon}_p$, $W(a)$) to the dual deviation gains, $\alpha_p^{\text{(C)CE}}$, that define the equilibrium. The duals are a much more space efficient objective target ($\sum_p |\mathcal{A}_p|^2$ for CEs and $\sum_p |\mathcal{A}_p|$ for CCEs) than the full joint ($\prod_p |\mathcal{A}_p|$), particularly when scaling the number of strategies and players. The joint, $\sigma(a)$, and approximation, ϵ_p, can be computed analytically from the dual deviation gains and the inputs using Equations (9) and (10). We call the resulting architecture a Neural Equilibrium Solver (NES).

4.1 Input and Preprocessing

The MRE objective, and the (C)CEs constraints are invariant to payoff offset and scaling. Therefore we can assume that the payoffs have been standardized without loss of generality. Each player's payoff should have zero-mean. Furthermore, it should be scaled such that the $L_m = ||\cdot||_m$ norm of the payoff tensor equals Z_m. We will ensure both these properties by including a preprocessing layer in NES. The inputs ($G_p^{L_m}(a)$, $\hat{\sigma}^{L_1}(a)$, $\hat{\epsilon}_p^{L_m}$, $W^{L_m}(a)$) are then broadcast and concatenated together so that they result in an input of shape $[C, N, |\mathcal{A}_1|, ..., |\mathcal{A}_N|]$, where the channel dimension $C = 4$, if all inputs are required.

4.2 Training Distribution

Zero-mean and constant norm scale geometrically results in the edge of a disc. It makes sense to uniformly sample over this space so as to not bias the training data to a particular offset or scaling. The literature tends to favour the unnormalized uniform distribution to generate random games, which would unnecessarily require a network to learn scale invariance. When normalized to a constant scale and zero-mean the uniform results in the L_∞ disc. We propose using disc distributions for training, and use the L_2 disc in the experiments in this paper. We choose Z_m such that the elements of the payoffs maintain unit variance (for example, $Z_2 = \sqrt{|\mathcal{A}|}$).

4.3 Gradient Calculation

The gradient update, is found by taking the derivative with respect to the dual deviation gains (Equation (7)). Note that computing a gradient does not require knowing the optimal joint $\sigma^*(a)$, so the network can be trained in an unsupervised fashion, from randomly generated inputs, $G_p(a)$, $\hat{\sigma}(a)$, $\hat{\epsilon}_p$, and $W(a)$. Furthermore, many powerful neural network optimizers [39, 20, 11] and ML frameworks [1, 5, 31] can be leveraged to update network parameters.

$$\frac{\partial L^{\text{(C)CE}}}{\partial \left\{\alpha_p^{\text{CE}}(a'_p, a''_p), \alpha_p^{\text{CCE}}(a'_p)\right\}} = \overset{\text{(C)CE}}{\epsilon_p} - \sum_a \left\{ \overset{\text{CE}}{A_p}(a'_p, a''_p, a), \overset{\text{CCE}}{A_p}(a'_p, a) \right\} \overset{\text{(C)CE}}{\sigma}(a) \tag{11}$$

4.4 Equivariant Architectures

The ordering of strategies and players in a normal-form game is unimportant, therefore the output of the network should be equivariant under two types of permutation; (i) strategy permutation, and (ii) player permutation. Specifically, for some strategy permutation $\tau_p(1), ..., \tau_p(|\mathcal{A}_p|)$ applied to each element of a player's inputs (payoffs, target joint, and welfare), the outputs must also have permuted dimensions: $(\alpha_p(a_p) = \alpha_p(\tau_p(a_p))$ and $\sigma^\tau(a_1, ..., a_N) = \sigma(\tau_1(a_1), ..., \tau_N(a_N)))$. Likewise, for some player permutation $\tau(1), ..., \tau(N)$, the outputs must be transposed: $(\alpha_p^\tau(a_p) = \alpha_{\tau(p)}(a_{\tau(p)})$ and $\sigma^\tau(a_1, ..., a_N) = \sigma(a_{\tau(1)}, ..., a_{\tau(N)}))$. The latter equivariance can only be exploited by a network if all players have the same number of strategies ("cubic games"). There are $|\mathcal{A}_p|!$ possible strategy permutations for each player and $N!$ player permutations, resulting in $N!\left(|\mathcal{A}_p|!\right)^N$ possible equivariant permutations of each sampled payoff. Note that this is much greater than the number of joint strategies in a game, $N!\left(|\mathcal{A}_p|!\right)^N \gg |\mathcal{A}_p|^N$, which is an encouraging observation when considering how this approach will scale to large games. Utilizing an equivariant [34, 42] architecture is therefore crucial to scale to large games because each sample represents many possible inputs. Equivariant architectures have been used before for two-player games [16].

Payoffs Transformations The main layers of the architecture consist of activations with shape $[C, N, |\mathcal{A}_1|, ..., |\mathcal{A}_N|]$, which is the same shape as a payoff tensor (with a channel dimension). We refer to layers with this shape as "payoff" layers. We consider transformations of the form:

$$g_{l+1}(c_{l+1}, p, a_1, ..., a_N) = f\left(\sum_{c_l^i}^{IC_l} w(c_{l+1}, c_l^i)\text{Stack}_i^I\left[h_i\left(g_l(c_l, p, a_1, ..., a_N)\right)\right] + b(c_{l+1})\right) \tag{12}$$

where f is any equivariant nonlinearity[2], and h_i is one of many possible equivariant pooling functions (Section C). For example, consider one such function, $h_i = \sum_{a_1} g(p, a_1, ..., a_N)$, which is invariant across any permutation of a_1 (similar to sum-pooling in CNNs), and equivariant over permutations of $p, a_2, ..., a_N$. In general we can use $\phi_{\subseteq\{p, a_1, ...a_N\}} g(p, a_1, ..., a_N)$, where ϕ is any equivariant continuous pooling function (mean-pooling and max-pooling are good choices). If all players have an equal number of strategies, for some functions, weights can be shared over all $p \in [1, N]$ because of symmetry [35]. Note that number of trainable parameters scales with the number of input and output channels, and not with the size of the game. The basic layer, g_{l+1}, therefore comprises of a linear transform of a stacked, broadcasted set of pooling functions.

[2] Common nonlinearities such as ReLU, tanh, sigmoid, and SoftMax are all equivariant.

Payoffs to CCE Duals Transformations Payoffs can be transformed to CCE duals, $\alpha_p^{\text{CCE}}(c_{l+1}, a'_p)$, by using a combination of a subset of the equivariant functions h_i discussed above that sum over at least $-p$. If the number of strategies are equal for each player, the transformation weights can be shared and the duals can be stacked into a single object for more efficient computation in later layers: $\alpha^{\text{CCE}}(c_{l+1}, p, a'_p) = \underset{p}{\text{Stack}}\left(\alpha_p^{\text{CCE}}(c_{l+1}, a'_p)\right)$.

Payoffs to CE Duals Transformations The transformation to produce the CE duals is more complex. CE duals, $\alpha_p^{\text{CCE}}(c_{l+1}, a''_p, a'_p)$, need to be *symmetrically equivariant*. This property can be obtained by (i) independently generating two CCE duals and, (ii) taking outer operations, $\boxdot$ (for example, sum or product), over them.

$$\alpha_p^{\text{CE}}(c_{l+1}, a''_p, a'_p) = \hat{f}\left(\alpha_p^{\text{CCE}}(c_{l+1}, a''_p) \boxdot \alpha_p^{\text{CCE}}(c_{l+1}, a'_p)\right) \tag{13}$$

Where $\hat{f}$ is any equivariant nonlinearity *with zero diagonal*[3]. We know that the diagonal is zero because it represents the dual of the deviation gain when deviating from a strategy to itself, which is zero, and therefore cannot be violated. This is a useful property which will be exploited in later dual layers. These can also be stacked if players have equal number of strategies: $\alpha^{\text{CCE}}(c_{l+1}, p, a'_p, a''_p) = \underset{p}{\text{Stack}}\left(\alpha_p^{\text{CCE}}(c_{l+1}, a'_p, a''_p)\right)$.

CCE Duals Transformations Because the payoff activations are high-dimensional, it is worthwhile to operate on them in dual space. When transforming CCE duals we consider a mapping:

$$\alpha_{l+1}^{\text{CCE}}(c_{l+1}, p, a'_p) = f\left(\sum_{c_l^i}^{IC_l} w(c_{l+1}, c_l^i)\text{Stack}_i^I\left[h_i\left(\alpha_l^{\text{CCE}}(c_l, p, a'_p)\right)\right] + b(c_{l+1})\right) \tag{14}$$

where f is any equivairant nonlinearity, and h_i is from a set of only two possible equivariant transformation functions (and two more if the game is cubic). For the final layer, we use a SoftPlus nonlinearity to ensure the output is nonnegative and has gradient everywhere.

CE Duals Transformations When transforming CE duals we consider functions of the form:

$$\alpha_{l+1}^{\text{CE}}(c_{l+1}, p, a'_p, a''_p) = \hat{f}\left(\sum_{c_l}^{IC_l} w(c_{l+1}, c_l^i)\text{Stack}_i^I\left[h_i\left(\alpha_l^{\text{CE}}(c_l, p, a'_p, a''_p)\right] + b(c_{l+l})\right)\right) \tag{15}$$

For the CE case, the equivariant linear transformations are more complex: it is symmetric over the recommended and deviation strategies. Fortunately this is a well studied equivariance class [38], which can be fully covered by combining seven transforms (Section C) which comprise of different sums and transpositions of the input.

Activation Variance Because the equivariant network possibly involves summing over dimensions of the inputs, activations are no longer independent of one another, so extra care needs to be taken when initializing the network to avoid variance explosion. To combat this we use three techniques: (i) inputs are scaled to unit variance as described previously (Section 4.1), (ii) the network is randomly initialized with variance scaling to ensure the variance at every layer is one, and (iii) we use BatchNorm [18] between every layer. We also used weight decay to regularize the network.

Advanced Architectures More advanced architectures such as ResNet [17] or Transformers [40] are possible. An example of the final architecture is summarized in Figure 2.

4.5 Parameterizations

The composite objective framework allows us to define a number of combinations of auxiliary objectives. We highlight several interesting specifications (Table 1). The most basic is Maximum Entropy (ME) which simply finds the unique equilibrium closest to the uniform distribution according to the relative entropy distance. This distribution need not be uniform, it could be any target

[3]Masking is sufficient: e.g. $\hat{f}(a'_p, a''_p) = (1 - I(a'_p, a''_p))f(a'_p, a''_p)$, where I is the identity matrix.

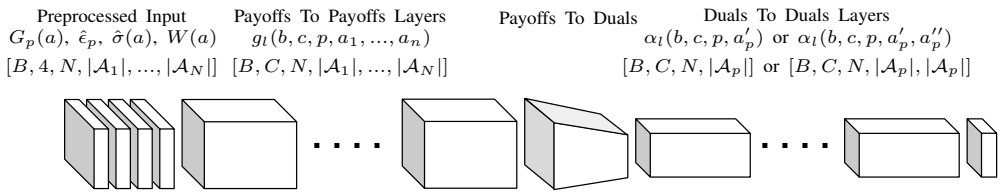


Figure 2: Network architecture showing the name, indices and shape (**B**atch, **C**hannels, **N**umber of players, **A**ctions per player) of each layer layer. Other architectures are possible, for example some of the inputs (target approximation, target joint, or welfare) could be passed in at a later layer.

distribution. We could instead favour a welfare objective parameterized on the payoffs to find a Maximum Utilitarian (MU) solution. The two previous solutions can be generalized to solve for any welfare (Maximum Welfare Maximum Entropy (MWME)) or any target (Minimum Relative Entropy (MRE)). Furthermore we need not limit ourselves to approximation parameters equal to zero, for example by finding the minimum possible approximation parameter we have the Maximum Strength (MS) solution. Finally, we can find solutions for any approximation parameter for the solutions discussed so far ($\hat{\epsilon}$-MWME and $\hat{\epsilon}$-MRE).

5 Performance Experiments

Traditionally performance of NE, and (C)CE solvers has focused on evaluating time to converge to a solution within some tolerance. Feedforward neural networks can produce *batches* of solutions quickly[4] and deterministically. For non-trivial games this is much faster than what an iterative solver could hope to achieve. We therefore focus our evaluation on the trade-offs of the neural network solver, namely (i) how long it takes to train, and (ii) how accurate the solutions are. For the latter we use two metrics:

$$\text{Solver Gap: } \frac{1}{2}\sum_{a} |\sigma^*(a) - \sigma(a)| \qquad \text{(C)CE Gap: } \sum_{p} \max_{.} \sum_{a} \left(A_p(., a) - \epsilon_p\right) \sigma(a)$$

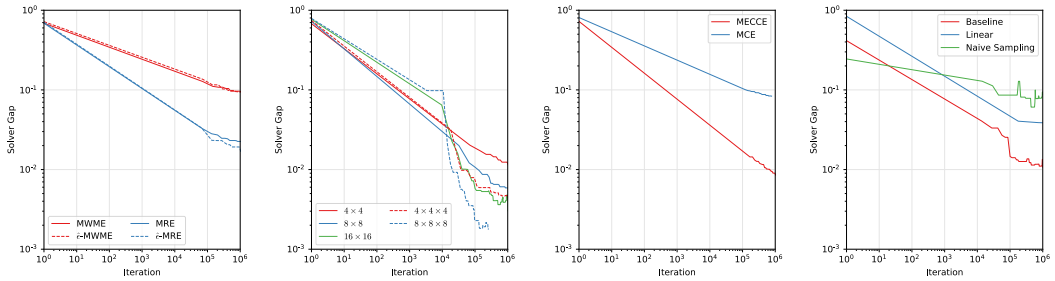
The first (Solver Gap) measures the distance to the exact solution found by an iterative solver[5], $\sigma^*(a)$, and is bounded between 0 and 1. The second ((C)CE Gap) measures the distance to the equilibrium solution polytope, and is zero if it is within the polytope.

Parameterization Sweeps We show performance across a number of parameterizations, including (i) different equilibrium selection criteria (Figure 3a), (ii) different shapes of games (Figure 3b), and (iii) different solution concepts (Figure 3c).

Classes of Games It is known that some distributions of game payoffs are harder for some methods to solve than others [33]. We compare performance across a number of classes of games (Table 2) for a single MECCE and a single MECE Neural Equilibrium Solver trained on 8×8 games. Figure 4 shows the worst, mean, and best performance over 512 samples from each class in terms of (i) distance to any equilibrium, and (ii) distance to the target equilibrium found by an iterative solver. We also plot the performance of a uniform joint as a naive baseline, as the gap can be artificially reduced by scaling the payoffs. In regards to equilibrium violation, ME is tricky because it lies on the boundary of the polytope, so some violation is expected in an approximate setting. The plots showing the failure rate and run time of the iterative solver are to intuit difficult classes. We see that for most classes the NES is very accurate with a solver gap of around 10^{-2}. Some classes of games are indeed more difficult and these align with games that iterative equilibrium solvers struggle with. This hints that difficult games are ill-conditioned.

[4] We found inference, $\frac{\text{step time}}{\text{batch size}}$, to be around $1\mu\frac{\text{s}}{\text{example}}$ on our hardware.

[5] We use an implementation in CVXPY [8, 2] which leverages the ECOS [9] solver.



(a) Equilibrium Selection (b) Shapes (c) Solution Concepts (d) Ablation

Figure 3: Sweeps and ablation studies where (a) shows 4×4 games over different equilibrium selection, (b) shows MECCE over games with different numbers of players and strategies, (c) shows CE and CCE concepts on 8×8 games, and (d) shows ablation experiments on MECCE $4 \times 4 \times 4$ games.

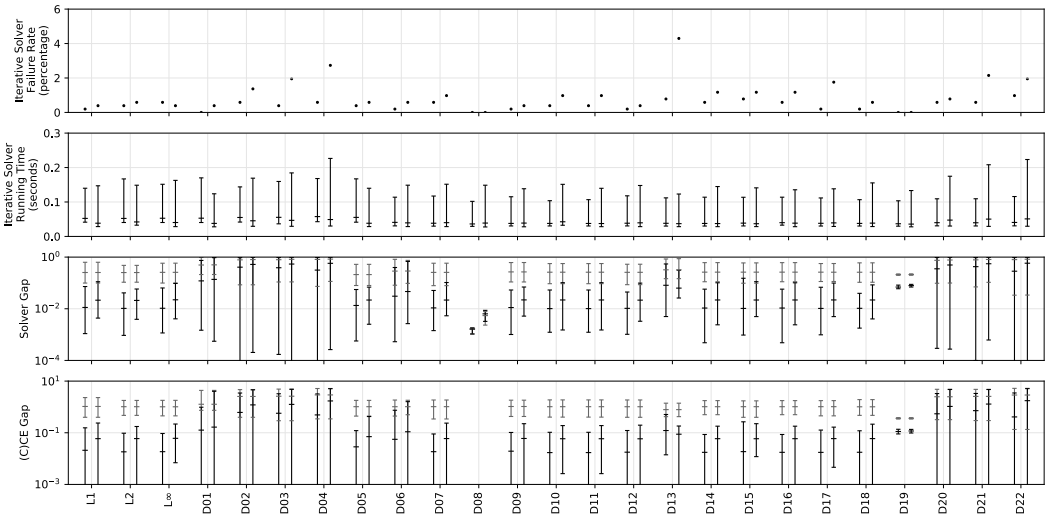


Figure 4: Performance of MECCE (left) and MECE (right) over 512 samples on the three classes introduced in this paper (Section 4.1), and on a subset [32] of GAMUT [29] games (Table 2).

Ablations We show the performance (Figure 3d) of the proposed method compared with (i) a linear network, and (ii) no invariant pre-processing with naive payoff sampling. Both result in significant reduction in performance.

6 Applications

The key benefit of this work is that it allows fast and general solving of games at the expense of guaranteed correctness. This allows the application of solution concepts into areas that would be otherwise be too time-expensive and are not as sensitive to approximations. There are several possible applications for this approach, most of which we leave to future work, but mention here.

Warm-Start Iterative Solvers Many iterative solvers start with a guess of the parameters and refine them over time to find an accurate solution. It is possible to use NES to warm-start iterative solver algorithms, potentially significantly improving convergence.

Polytope Approximation The framework can be used to approximate the full space of solutions by finding extreme points of the convex polytope. Because of convexity, any convex mixture of these extreme points is also a valid solution. Two approaches could be used to find extreme points (i) using different welfare objectives (ii) or using different target joint objectives. For example, using pure

joint targets:

$$W(a) = \begin{cases} 1, & \text{if } a = \hat{a} \\ 0, & \text{otherwise} \end{cases} \qquad \hat{\sigma}(a) = \begin{cases} 1^-, & \text{if } a = \hat{a} \\ 0^+, & \text{otherwise} \end{cases}$$

These could be computed in a single batch, and would cover a reasonably large subset of full polytope of solutions (the latter approach is demonstrated in Figure 1). It would be easy to develop an algorithm that refines the targets at each step to gradually find all vertices of the polytope, if desired.

Differentiable Model and Mechanism Design Mechanism design (MD) is a sub-field of economics often described as "inverse game theory", where instead of studying the behavior of rational payoff maximizing agents on a given game, we are tasked with designing a game so that rational payoff maximizing participants will exhibit behaviours *at equilibrium* that we deem desirable. The field has a long history to which it is near impossible to do justice; see [24] for a review. The work presented here could impact MD in two ways. First, by making it easy to compute equilibrium strategies, NES could widen the class of acceptable output games, relaxing the restrictive requirements (e.g. strategic dominance) often imposed of the output games out of concern more permissive solution concepts could be hard for participants to compute. Second, NES maps payoffs to joint strategies and it is differentiable, one could imagine turning a mechanism design task to a optimization problem that could be solved using standard gradient descent (e.g. design a general-sum game where strategies at equilibrium maximize some non-linear function of welfare and entropy, with payoff lying in a useful convex and closed subset).

7 Discussion

With Neural Equilibrium Solvers (NES) it is possible to quickly find approximate equilibrium solutions using a variety of selection criteria. There are many exciting applications of such a network to MARL algorithms and mechanism design.

The main limitation of the approach is that, while the network is extremely efficient in terms of parameters, the activation space of the network is large, particularly with a large number of players and strategies which limits the size of games that can be tackled.

Solving for equilibria has the potential to promote increased cooperation in general-sum games, which could increase the welfare of all players. However, if a powerful and unethical actor had influence on the game being played, welfare gains of some equilibria could unfairly come at the expense of other players.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[3] R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

[4] Y. Bai, C. Jin, and T. Yu. Near-optimal reinforcement learning with self-play. *CoRR*, abs/2006.12007, 2020. URL `https://arxiv.org/abs/2006.12007`.

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[6] X. Chen, X. Deng, and S.-H. Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57, 2009.

[7] C. Daskalakis, P. Goldberg, and C. Papadimitriou. The complexity of computing a Nash equilibrium. *SIAM J. Comput.*, 39:195–259, 02 2009.

[8] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[9] A. Domahidi, E. Chu, and S. Boyd. ECOS: An SOCP solver for embedded systems. In *European Control Conference (ECC)*, pages 3071–3076, 2013.

[10] Z. Duan, Y. Du, J. Wang, and X. Deng. Learning to compute approximate nash equilibrium for normal-form games. *CoRR*, abs/2108.07472, 2021. URL https://arxiv.org/abs/2108.07472.

[11] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[12] H. Fargier, P. Jourdan, and R. Sabbadin. A path-following polynomial equations systems approach for computing nash equilibria. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 418–426, 2022.

[13] A. Greenwald and K. Hall. Correlated-q learning. pages 242–249, 2003.

[14] J. Harsanyi and R. Selten. *A General Theory of Equilibrium Selection in Games*, volume 1. The MIT Press, 1 edition, 1988.

[15] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[16] J. S. Hartford, J. R. Wright, and K. Leyton-Brown. Deep learning for predicting human strategic behavior. 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. URL https://proceedings.neurips.cc/paper/2016/file/7eb3c8be3d411e8ebfab08eba5f49632-Paper.pdf.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015. URL http://arxiv.org/abs/1502.03167.

[19] C. Jin, Q. Liu, Y. Wang, and T. Yu. V-learning - A simple, efficient, decentralized algorithm for multiagent RL. *CoRR*, abs/2110.14555, 2021. URL https://arxiv.org/abs/2110.14555.

[20] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14.

[21] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *NIPS*. 2017.

[22] C. Lemke and J. Howson, Jr. Equilibrium points of bimatrix games. *Journal of the Society for industrial and Applied Mathematics*, 12(2):413–423, 1964.

[23] L. Marris, P. Muller, M. Lanctot, K. Tuyls, and T. Graepel. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. *CoRR*, abs/2106.09435, 2021. URL https://arxiv.org/abs/2106.09435.

[24] E. S. Maskin. Mechanism design: How to implement social goals. *American Economic Review*, 98(3):567–76, 2008.

[25] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory, version 16.0.1, 2016.

[26] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

[27] H. Moulin and J.-P. Vial. Strategically zero-sum games: the class of games whose completely mixed equilibria cannot be improved upon. *International Journal of Game Theory*, 7(3-4): 201–221, 1978.

[28] J. Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[29] E. Nudelman, J. Wortman, Y. Shoham, and K. Leyton-Brown. Run the gamut: A comprehensive approach to evaluating game-theoretic algorithms. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, page 880–887, USA, 2004. IEEE Computer Society. ISBN 1581138644.

[30] L. E. Ortiz, R. E. Schapire, and S. M. Kakade. Maximum entropy correlated equilibria. In M. Meila and X. Shen, editors, *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, volume 2 of *Proceedings of Machine Learning Research*, pages 347–354, San Juan, Puerto Rico, 21–24 Mar 2007. PMLR.

[31] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[32] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a nash equilibrium. *Games and Economic Behavior*, 63(2):642–662, 2008. ISSN 0899-8256. doi: https://doi.org/10.1016/j.geb.2006.03.015. URL https://www.sciencedirect.com/science/article/pii/S0899825606000935. Second World Congress of the Game Theory Society.

[33] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a nash equilibrium. *Games and Economic Behavior*, 63(2):642–662, 2008.

[34] J. Shawe-Taylor. Symmetries and discriminability in feedforward network architectures. *IEEE transactions on neural networks*, 4 5:816–26, 1993.

[35] J. Shawe-Taylor. Introducing invariance: a principled approach to weight sharing, 1994.

[36] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[37] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419): 1140–1144, 2018. doi: 10.1126/science.aar6404. URL https://www.science.org/doi/abs/10.1126/science.aar6404.

[38] E. H. Thiede, T. Hy, and R. Kondor. The general theory of permutation equivariant neural networks and higher order graph variational encoders. *CoRR*, abs/2004.03990, 2020. URL https://arxiv.org/abs/2004.03990.

[39] T. Tieleman, G. Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[41] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019.

[42] J. Wood and J. Shawe-Taylor. Representation theory and invariant neural networks. *Discrete Applied Mathematics*, 69(1):33–60, 1996. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(95)00075-3. URL https://www.sciencedirect.com/science/article/pii/0166218X95000753.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Main limitation is that the activation space of the network is very large.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We briefly discuss the abuse of equilibria could cause harm if used by a malicious actor, as well as the obvious benefits. We think most of game theory can be abused this way. Users must make sure they apply any work in an ethical way.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] We have no ethics concerns.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Proofs are in appendices.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We have not included code, although we are working on open-sourcing a library. We hope this will be available by rebuttal time. We have described the approach in enough detail that a competent practitioner should be able to reproduce our results in less than a week.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We did not run with multiple random seeds due to computation constraints. We do, however, report the worst, mean, and best case performance over many examples.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use third party code which we cite (e.g. GAMUT).
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]