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

Luke Marris
DeepMind (marris@deepmind.com), UCL

Ian Gemp DeepMind | **Thomas Anthony** DeepMind | **Andrea Tacchetti** DeepMind | **Siqi Liu** DeepMind, UCL | **Karl Tuyls** DeepMind

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 the space of 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 show remarkable zero-shot generalization to larger games. We argue that such a network is a powerful component for many possible multiagent algorithms.

1 Introduction

Normal-form solution concepts such as Nash Equilibrium (NE) [28, 46], Correlated Equilibrium (CE) [3], and Coarse Correlated Equilibrium (CCE) [45] are useful components and subroutines for many multiagent machine learning algorithms. For example, value-based reinforcement learning algorithms for solving Markov games, such as Nash Q-learning [27] and Correlated Q-Learning [19] maintain state action values for every player in the game. These action values are equivalent to per-state normal-form games, and policies are equilibrium solutions of these games. Critically, this policy will need to be recomputed each time the action-value is updated during training, and for large or continuous state-space Markov games, every time the agents need to take an action. Another class of multiagent algorithms are those in the space of Empirical Game Theoretic Analysis (EGTA) [60, 61] including PSRO [35, 44], JPSRO [40], and NeuPL [39, 38]. These algorithms are capable of training policies in extensive-form games, and require finding equilibria of empirically estimated normal-form games as a subroutine (the "meta-solver" step). In particular, these algorithms have been critical in driving agents to superhuman performance in Go [54], Chess [55], and StarCraft [59].

Unfortunately, solving for an equilibrium can be computationally complex. NEs are known to be PPAD [9, 8]. (C)CEs are defined by linear constraints, and if a linear objective is used to select an equilibrium, can be solved by linear programs (LPs) in polynomial time. However, in general, the solutions to LPs are non-unique (e.g. zero-sum games), and therefore are unsuitable equilibrium selection methods for many algorithms, and unsuitable for training neural networks which benefit from unambiguous targets. Objectives such as Maximum Gini [41] (a quadratic program (QP)), and Maximum Entropy [48] (a nonlinear program), are unique but are more complex to solve.

As a result, solving for equilibria often requires deploying iterative solvers, which theoretically can scale to large normal-form games but may (i) take an unpredictable amount of time to converge, (ii)

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

take a prohibitive amount of time to do so, and (iii) may fail unpredictably on ill-conditioned problems. Furthermore, classical methods [16, 36, 43] (i) do not scale, and (ii) are non-differentiable. This limits the applicability of equilibrium solution concepts in multiagent machine learning algorithms.

Therefore, there exists an important niche for approximately solving equilibria in medium sized normal-form games, quickly, in batches, reliably, and in a deterministic amount of time. With appropriate care, this goal can be accomplished with a Neural Network which amortizes up-front training cost to map normal-form payoffs to equilibrium solution concepts quickly at test time. We propose the Neural Equilibirum Solver (NES). 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 to explicitly construct 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 smaller 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 the number of training steps and improves generalization performance. The network architecture is independent of the number of strategies in the game and we show interesting zero-shot generalization to larger games. 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". Games are sometimes referred to by their shape, for example: $|A_1| \times ... \times |A_N|$. 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) [45] 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 [46] 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].

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

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* [20]. For (C)CEs, the valid solution space is convex (Figure 1), so any strictly convex function will suffice (in particular, Maximum Entropy (ME) [48] and Maximum Gini (MG) [40] 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. Learning NEs [44] and CCEs [21] on a single game is well studied.

Neural Network Solvers Approximating NEs using neural networks is known to be PAC learnable [14]. There is also work learning (C)CEs [4, 32] and training neural networks to approximate NEs [14, 22] on subclasses of games. Learned NE meta-solvers have been deployed in PSRO [17]. Differentiable neural networks have been developed to learn QREs [37]. NEs for contextual games have been learned using fixed point (deep equilibrium) networks [24]. A related field, L2O [7], aims to learn an iterative optimizer more suited to a particular distribution of inputs, while this work focuses on learning a direct mapping. To the best of our knowledge, no work exists training a general approximate mapping from the full space of games to (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. The principle of maximum entropy [30] has been used to find unique equilibria [48]. 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 transformations (such as 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 high value for the agents and is a linear objective, however in general it is not unique. We consider a composite objective function composed of (i) Minimum Relative Entropy (MRE, also known as Kullback-Leibler divergence) between a target joint, $\hat{\sigma}(a)$,

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

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 hyperparameters 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 solve the dual problem, $\arg\min_{\alpha_p} L^{\text{(C)CE}}$ (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 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:

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

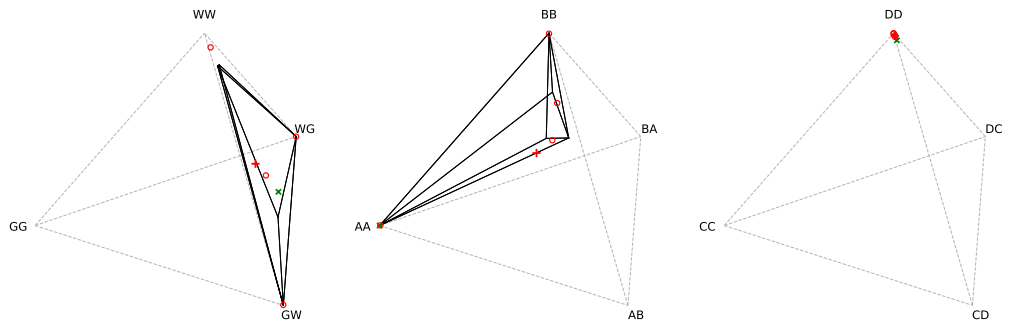
$$\overset{\text{(C)CE}}{\epsilon}_p = (\hat{\epsilon}_p - \epsilon_p^+) \exp\left(-\frac{1}{\rho}\left\{\sum_{a'_p, a''_p} \overset{\text{CE}}{\alpha}_p(a'_p, a''_p), \sum_{a'_p} \overset{\text{CCE}}{\alpha}_p(a'_p)\right\}\right) + \epsilon_p^+ \tag{10}$$

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 significantly 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). The network is trained by minimizing the loss, $L^{\text{(C)CE}}$ (Equation (7)). 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, where Z_m is a scale hyperparameter chosen such that the elements of



(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 (in two-strategy games CEs and CCEs are equivalent). 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.

the inputs have unit variance (a property that ensures neural networks train quickly with standard parameter initializations [18]). We will ensure both these properties by including a preprocessing layer in NES. The preprocessed inputs ($G_p(a)$, $\hat{\sigma}(a)$, $\hat{\epsilon}_p$, $W(a)$) 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

The literature favours sampling games from the uniform or normal distribution. This introduces two problems: (i) it biases the distribution of games solvable by the network, and (ii) unnecessarily requires the network to learn offset and scale invariance in the payoffs. Recall that the space of equilibria are invariant to offset and positive scale transformations to the payoff. Zero-mean and L_2 norm scaling geometrically results in the surface of an $(|\mathcal{A}| - 1)$-ball centered on the origin (Section D.4). We propose using this invariant subspace for training. We choose the norm scaling constant, $Z_2 = \sqrt{|\mathcal{A}|}$, such that the elements of the payoffs maintain unit variance.

4.3 Gradient Calculation

The gradient update is found by taking the derivative of the loss (Equation (7)) with respect to the dual variables, α. 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 $W(a)$.

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

The dual variables, $\{\alpha_p^{\text{CE}}(a'_p, a''_p), \alpha_p^{\text{CCE}}(a'_p)\}$, are outputs of the neural network, with learned parameters θ. Gradients for these parameters can be derived using the chain rule:

$$\frac{\partial L^{\text{(C)CE}}}{\partial \theta} = \frac{\partial L^{\text{(C)CE}}}{\partial \left\{\alpha_p^{\text{CE}}(a'_p, a''_p), \alpha_p^{\text{CCE}}(a'_p)\right\}} \frac{\partial \left\{\alpha_p^{\text{CE}}(a'_p, a''_p), \alpha_p^{\text{CCE}}(a'_p)\right\}}{\partial \theta}$$

Backprop efficiently calculates these gradients, and many powerful neural network optimizers [57, 34, 15] and ML frameworks [1, 5, 49] can be leveraged to update the network parameters.

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 [52, 62] 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 [22].

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:

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

where f is any equivariant nonlinearity[2], w are learned network weights, b are learned network biases, and ϕ_i is one of many possible equivariant pooling functions (Section C) and Con is the concatenate function along the channel dimension. For example, consider one such function, $\phi_i = \sum_{a_1}$, 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 $\phi_{\subseteq\{p, a_1, ...a_N\}}g(p, a_1, ..., a_N)$ (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 [53]. Note that the number of trainable parameters scales with the number of input and output channels, and not with the size of the game (Figure 2), therefore it is possible for the network to generalize to games with different numbers of strategies. The basic layer, g_{l+1}, therefore comprises of a linear transform of a concatenated, broadcasted set of pooling functions.

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 ϕ_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: $\alpha^{\text{CCE}}(c_{l+1}, p, a'_p) = \text{Stack}_p\left(\alpha_p^{\text{CCE}}(c_{l+1}, a'_p)\right)$.

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.

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

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: $\alpha^{\text{CCE}}(c_{l+1}, p, a'_p, a''_p) = \text{Stack}_p\left(\alpha_p^{\text{CCE}}(c_{l+1}, a'_p, a''_p)\right)$.

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:

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

[2]Common nonlinearities such as element-wise (ReLU, tanh, sigmoid), and SoftMax are all equivariant.

[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.

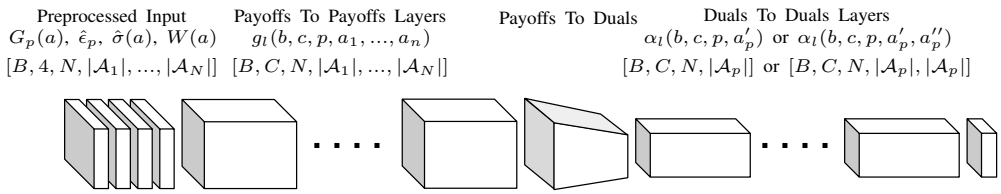


Figure 2: Network architecture showing the name, indices and shape (**B**atch, **C**hannels, **N**umber of players, **A**ctions per player) of each layer layer. Other architectures are possible, for example some of the inputs (target approximation, target joint, or welfare) could be passed in at a later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

where f is any equivariant nonlinearity, and ϕ_i is from a set (Section C.2) 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:

$$\alpha^{\text{CE}}_{l+1}(c_{l+1}, p, a'_p, a''_p) = \hat{f}\left(\sum_{c_l}^{IC_l} w(c_{l+1}, c^i_l)\text{Con}^I_i\left[\phi_i\left(\alpha^{\text{CE}}_l(c_l, p, a'_p, a''_p)\right] + b(c_{l+l})\right)\right) \tag{15}$$

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 [56], 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 [29] between every layer. We also used weight decay to regularize the network.

Advanced Architectures More advanced architectures such as ResNet [23] or Transformers [58] are possible. An example of the final architecture is summarized in Figure 2.

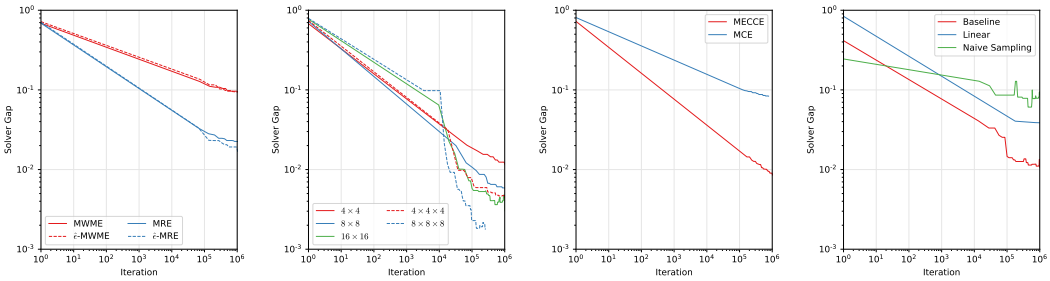
4.5 Parameterizations

The composite objective framework allows us to define a number of combinations of auxiliary objectives. We highlight several interesting specifications (Appendix Table 3). 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 distribution. We could instead favour a welfare objective parameterized on the payoffs to find a Maximum Welfare (MW) 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

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



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

Figure 3: Sweeps and ablation studies showing the average solver gap of three experiment seeds evaluated over 512 sampled games against the number of train steps. Subfigure (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.

solver, namely (i) how long it takes to train, and (ii) how accurate the solutions are. For the latter we use two metrics:

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

The first (Solver Gap) measures the distance to the exact unique solution found by an iterative solver[5], $\sigma^*(a)$, and is bounded between 0 and 1, and is zero for perfect prediction. 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 [51]. We compare performance across a number of classes of transfer games (Appendix Table 4) for a single MECCE and a single MECE [48] 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. The baseline iterative solver take about 0.05s to solve a single game, the network can solve a batch of 4096 games in 0.0025s. 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.

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 (each element sampled using a uniform distribution). Both result in significant reduction in performance.

Scaling Due to the size of the representation of the payoffs, $G_p(a)$, the inputs and therefore the activations of the network grow significantly with the number of joint strategies in the game. Therefore without further work on sparser payoff representation, NES is limited by size of payoff inputs. For further discussion see Section 7. Nevertheless, Table 1 shows good performance when scaling to moderately sized games. Note that the "solver gap" metric is incomplete on larger games because the ECOS evaluation solver fails to converge.

[5]We use an implementation in CVXPY [12, 2] which leverages the ECOS [13] solver.

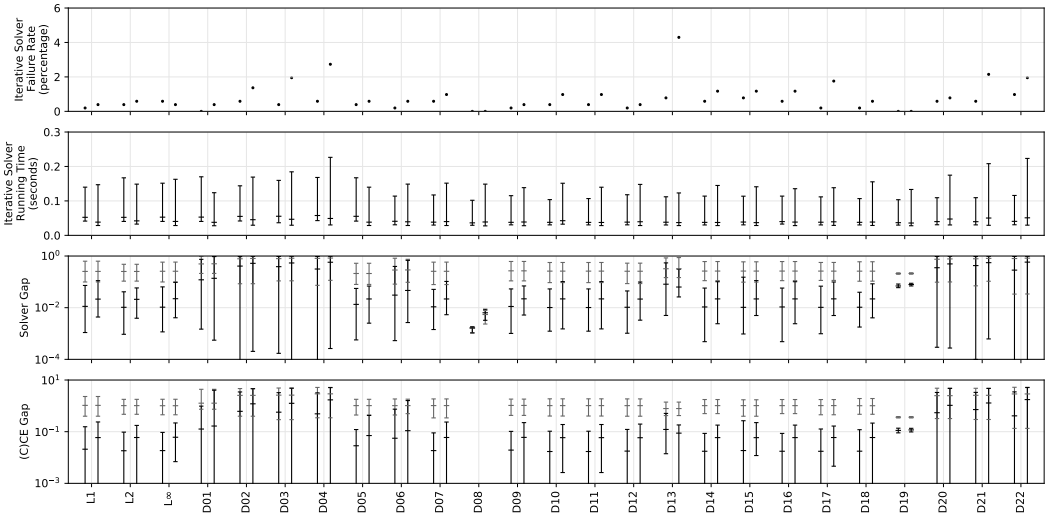


Figure 4: Worst, mean, and best performance of MECCE (left in pair) and MECE (right in pair) over 512 samples on the three classes introduced in this paper (Section 4.1), and on a subset [50] of transfer GAMUT [47] games (Appendix Table 4). The network was only trained on the "L_2 invariant subspace" distribution of games. The gray range indicates the performance under a uniform distribution baseline.

Table 1: Scaling experiments showing the gaps of five NES models for larger games, with a uniform baseline, over 128 samples. The ECOS solver used to evaluate "solver gap" fails on large games.

Game	CCE Gap under uniform	CCE Gap under NES	Solver Gap under uniform	Solver Gap under NES	Success Fraction
4×4	1.1006	0.0274	0.3552	0.0120	100%
8×8	1.0043	0.0163	0.2513	0.0054	99%
16×16	0.8861	0.0173	0.2014	0.0034	98%
32×32	0.7376	0.0215	–	–	0%
64×64	0.5864	0.0288	–	–	0%

Generalization An interesting property of the NES architecture is that its parameters do not depend on the number of strategies in the game. Therefore we can test the generalization ability of the network zero-shot on games with different numbers of strategies (Table 2). There are two observations: (i) NES only weakly generalizes to other game sizes under the solver gap metric, and (ii) NES strongly generalizes to larger games under the CCE gap, remarkably achieving zero violation. Therefore the network retains the ability to reliably find CCEs in larger games, but does struggle to accurately select the target MWMRE equilibrium. This could be mitigated by training the network on a mixture of game sizes, which we leave to future work.

6 Applications

With Neural Equilibrium Solvers (NES) it is possible to quickly find approximate equilibrium solutions using a variety of selection criteria. This allows the application of solution concepts into areas that would otherwise be too time-expensive and are not as sensitive to approximations.

Inner Loop of MARL Algorithms For algorithms [28, 27, 19, 35, 41, 39] where speed is critical, and the size of games is modest, but numerous, and approximations can be tolerated.

Warm-Start Iterative Solvers Many iterative solvers start with a guess of the parameters and refine them over time to find an accurate solution [14]. It is possible to use NES to warm-start iterative solver algorithms, potentially significantly improving convergence.

Table 2: Generalization experiments showing how an 8×8 network generalizes to games with a different number of strategies, over 128 samples.

Game	CCE Gap under uniform	CCE Gap under NES	Solver Gap under uniform	Solver Gap under NES	Success Fraction
4×4	1.1006	4.4445	0.3552	0.1500	100%
$\mathbf{8 \times 8}$	1.0043	0.0163	0.2513	0.0054	99%
16×16	0.8861	0.0000	0.2014	0.1089	98%
32×32	0.7376	0.0000	–	–	0%
64×64	0.5864	0.0000	–	–	0%

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 [42] 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). A related idea is to find a game that produces a certain equilibrium [37]. Given an equilibrium, a payoff could be trained through the differentiable model that results in the desired specific behaviour.

7 Discussion

The main limitation of the approach is that 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. Future work could look at restricted classes of games, such as polymatrix games [10, 11], or graphical games [31], which consider only local payoff structure and have much smaller payoff representations. This is a promising direction because NES otherwise has good scaling properties: (i) the dual variables are space efficient, (ii) there are relatively few parameters, (iii) the number of parameters is independent of the number of strategies in the game, (iv) equivariance means each training sample is equivalent to training under all payoff permutations, and (v) there are promising zero-shot generalization results to larger games.

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] A. Brock, S. De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. *CoRR*, abs/2102.06171, 2021. URL `https://arxiv.org/abs/2102.06171`.

[7] T. Chen, X. Chen, W. Chen, H. Heaton, J. Liu, Z. Wang, and W. Yin. Learning to optimize: A primer and a benchmark, 2021. URL `https://arxiv.org/abs/2103.12828`.

[8] 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.

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

[10] A. Deligkas, J. Fearnley, T. P. Igwe, and R. Savani. An empirical study on computing equilibria in polymatrix games. *arXiv preprint arXiv:1602.06865*, 2016.

[11] A. Deligkas, J. Fearnley, R. Savani, and P. Spirakis. Computing approximate nash equilibria in polymatrix games. *Algorithmica*, 77(2):487–514, 2017.

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

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

[14] Z. Duan, Y. Du, J. Wang, and X. Deng. Towards the pac learnability of nash equilibrium. *CoRR*, abs/2108.07472, 2021. URL `https://arxiv.org/abs/2108.07472`.

[15] 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.

[16] 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.

[17] X. Feng, O. Slumbers, Z. Wan, B. Liu, S. McAleer, Y. Wen, J. Wang, and Y. Yang. Neural auto-curricula in two-player zero-sum games. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3504–3517. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper/2021/file/1cd73be1e256a7405516501e94e892ac-Paper.pdf`.

[18] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In Y. W. Teh and M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `https://proceedings.mlr.press/v9/glorot10a.html`.

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

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

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

[22] 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`.

[23] 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`.

[24] H. Heaton, D. McKenzie, Q. Li, S. W. Fung, S. J. Osher, and W. Yin. Learn to predict equilibria via fixed point networks. *CoRR*, abs/2106.00906, 2021. URL `https://arxiv.org/abs/2106.00906`.

[25] T. Hennigan, T. Cai, T. Norman, and I. Babuschkin. Haiku: Sonnet for JAX, 2020. URL `http://github.com/deepmind/dm-haiku`.

[26] M. Hessel, D. Budden, F. Viola, M. Rosca, E. Sezener, and T. Hennigan. Optax: composable gradient transformation and optimisation, in jax!, 2020. URL `http://github.com/deepmind/optax`.

[27] J. Hu and M. Wellman. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 01 2003. doi: 10.1162/1532443041827880.

[28] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 242–250, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

[29] 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`.

[30] E. T. Jaynes. Information theory and statistical mechanics. *Phys. Rev.*, 106:620–630, May 1957.

[31] A. X. Jiang, K. Leyton-Brown, and N. A. Bhat. Action-graph games. *Games and Economic Behavior*, 71(1):141–173, 2011. ISSN 0899-8256. doi: https://doi.org/10.1016/j.geb.2010.10.012. Special Issue In Honor of John Nash.

[32] 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`.

[33] N. Jouppi, D. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson. A domain-specific supercomputer for training deep neural networks. *Communications of the ACM*, 63:67–78, 06 2020. doi: 10.1145/3360307.

[34] 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`.

[35] 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.

[36] 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.

[37] C. K. Ling, F. Fang, and J. Z. Kolter. What game are we playing? end-to-end learning in normal and extensive form games. *CoRR*, abs/1805.02777, 2018. URL `http://arxiv.org/abs/1805.02777`.

[38] S. Liu, M. Lanctot, L. Marris, and N. Heess. Simplex neural population learning: Any-mixture Bayes-optimality in symmetric zero-sum games. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 13793–13806. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/liu22h.html`.

[39] S. Liu, L. Marris, D. Hennes, J. Merel, N. Heess, and T. Graepel. NeuPL: Neural population learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=MIX3fJkl_1`.

[40] 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`.

[41] L. Marris, P. Muller, M. Lanctot, K. Tuyls, and T. Graepel. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 7480–7491. PMLR, 18–24 Jul 2021. URL `http://proceedings.mlr.press/v139/marris21a.html`.

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

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

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

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

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

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] W. Walsh, R. Das, G. Tesauro, and J. Kephart. Analyzing complex strategic interactions in multi-agent systems. 01 2002.

[61] M. P. Wellman. Methods for empirical game-theoretic analysis. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 1552–1556. AAAI Press, 2006. URL http://www.aaai.org/Library/AAAI/2006/aaai06-248.php.

[62] 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. 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]

A Approximate Target Maximum Welfare Minimum Relative Entropy Equilbiria

We use a Minimum Relative Entropy (RME) (also known as minimum KL divergence) $\sum_a \sigma(a) \ln\left(\frac{\sigma(a)}{\hat{\sigma}(a)}\right)$, where $\hat{\sigma}(a) > 0$ is a full-support joint such that, $\sum_a \hat{\sigma}(a) = 1$. This objective is similar to Maximum Entropy Correlated Equilibrium (MECE) [48], and the proofs here are similar to the framework set out there. A drawback of MECE is that it is not easy to determine the minimum ϵ_p permissible. If we choose ϵ_p that does not permit a valid solution, then the parameters will diverge. We can circumvent this problem by optimizing the distance to a target $\hat{\epsilon}_p$. We engineer this target, $\min_{\epsilon_p} \rho \sum_p \left(\epsilon_p^+ - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon_p^+ - \epsilon_p}{(\epsilon_p^+ - \hat{\epsilon}_p)}\right)$, to have a global minimum at $\epsilon_p = \hat{\epsilon_p}$, where $0 < \rho < \infty$ is a hyper-parameter used to control the balance between the distance to the target distribution and the distance to the target approximation parameter. And μ is for balancing the linear objective.

A.1 CEs

Theorem 1 (ϵ-MWMRE CE)**.** *The $\hat{\epsilon}$-MWMRE CE solution is equivalent to minimizing the loss:*

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

With logits defined as:

$$\overset{CE}{l}(a) = \mu \sum_a W(a) - \sum_{p, a'_p, a''_p} \overset{CE}{\alpha}_p(a'_p, a''_p) \overset{CE}{A}_p(a'_p, a''_p, a)$$

And primal variables defined:

$$\overset{CE}{\sigma}(a) = \frac{\hat{\sigma}(a) \exp\left(\overset{CE}{l}(a)\right)}{\sum_{a\in\mathcal{A}} \hat{\sigma}(a) \exp\left(\overset{CE}{l}(a)\right)} \qquad \overset{CE}{\epsilon}_p = (\hat{\epsilon}_p - \epsilon_p^+) \exp\left(-\frac{1}{\rho} \sum_{a'_p, a''_p} \overset{CE}{\alpha}_p(a'_p, a''_p)\right) + \epsilon_p^+$$

Proof. Construct a Lagrangian, $\max_{\sigma,\epsilon} \min_{\alpha,\beta,\lambda} L^{\sigma,\epsilon_p}_{\alpha_p,\beta,\lambda}$, where the primal variables are being maximized and the dual variables are being minimized.

$$\begin{aligned}
L^{\sigma,\epsilon_p}_{\alpha_p,\beta,\lambda} &= -\sum_a \sigma(a) \ln\left(\frac{\sigma(a)}{\hat{\sigma}(a)}\right) + \mu \sum_{a\in\mathcal{A}} W(a)\sigma(a) - \rho \sum_p \left(\epsilon_p^+ - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon_p^+ - \epsilon_p}{(\epsilon_p^+ - \hat{\epsilon}_p)}\right) \\
&\quad + \sum_a \beta(a)\sigma(a) - \lambda\left(\sum_a \sigma(a) - 1\right) - \sum_{p, a'_p, a''_p} \alpha_p(a'_p, a_p)\left(\sum_a \sigma(a) A_p(a'_p, a''_p, a) - \epsilon_p\right) \\
&= \sum_a \sigma(a)\left(-\ln\left(\frac{\sigma(a)}{\hat{\sigma}(a)}\right) + \mu W(a) + \beta(a) - \lambda - \sum_{p, a'_p, a''_p} \alpha_p(a'_p, a''_p) A_p(a'_p, a''_p, a)\right) \\
&\quad + \sum_{p, a'_p, a''_p} \alpha_p(a'_p, a''_p)\epsilon_p + \lambda - \rho \sum_p \left(\epsilon_p^+ - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon_p^+ - \epsilon_p}{(\epsilon_p^+ - \hat{\epsilon}_p)}\right)
\end{aligned}$$

Taking the derivatives with respect to the joint distribution $\sigma(a)$, and setting to zero.

$$\frac{\partial L^{\sigma,\epsilon_p}_{\alpha_p,\beta,\lambda}}{\partial \sigma(a)} = -\ln\left(\frac{\sigma(a)}{\hat{\sigma}(a)}\right) - 1 + \mu W(a) + \beta(a) - \lambda - \sum_{p, a'_p, a_p} \alpha_p(a'_p, a_p) A_p(a'_p, a_p, a) = 0$$

$$\sigma^*(a) = \hat{\sigma}(a) \exp\left(-\lambda - 1 + \mu W(a) + \beta(a) - \sum_{p, a'_p, a_p} \alpha_p(a'_p, a_p) A_p(a'_p, a_p, a)\right)$$

Substituting back in:

$$L^{\epsilon_p}_{\alpha_p,\beta,\lambda} = \sum_a \sigma^*(a) + \sum_{p,a'_p,a_p} \alpha_p(a'_p, a_p)\epsilon_p + \lambda - \rho \sum_p \left(\epsilon^+_p - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon^+_p - \epsilon_p}{(\epsilon^+_p - \hat{\epsilon}_p)}\right)$$

Taking the derivative with respect to λ and setting to zero.

$$\frac{\partial L^{\epsilon_p}_{\alpha_p,\beta,\lambda}}{\partial \lambda} = -\sum_a \sigma^*(a) + 1 = 0$$

$$\exp\left(\lambda^* + 1\right) = \sum_a \hat{\sigma}(a) \exp\left(\mu W(a) + \beta(a) - \sum_{p,a'_p,a''_p} \alpha_p(a'_p, a''_p) A_p(a'_p, a''_p, a)\right)$$

Substituting back in:

$$L^{\epsilon_p}_{\alpha_p,\beta} = \ln\left(\sum_a \hat{\sigma}(a) \exp\left(\mu W(a) + \beta(a) - \sum_{p,a'_p,a''_p} \alpha_p(a'_p, a''_p) A_p(a'_p, a''_p, a)\right)\right)$$
$$+ \sum_{p,a'_p,a''_p} \alpha_p(a'_p, a''_p)\epsilon_p - \rho \sum_p \left(\epsilon^+_p - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon^+_p - \epsilon_p}{(\epsilon^+_p - \hat{\epsilon}_p)}\right)$$

Noting that the term is minimized when $\beta(a) = 0$, and that we can lift the $\hat{\sigma}(a)$ term into the exponential, we have:

$$L^{\epsilon_p}_{\alpha_p} = \ln\left(\sum_a \hat{\sigma}(a) \exp\left(\mu W(a) - \sum_{p,a'_p,a''_p} \alpha_p(a'_p, a''_p) A_p(a'_p, a''_p, a)\right)\right)$$
$$+ \sum_{p,a'_p,a''_p} \alpha_p(a'_p, a''_p)\epsilon_p - \rho \sum_p \left(\epsilon^+_p - \epsilon_p\right) \ln\left(\frac{1}{\exp(1)} \frac{\epsilon^+_p - \epsilon_p}{(\epsilon^+_p - \hat{\epsilon}_p)}\right)$$

Taking the derivatives with respect to the approximation parameter ϵ_p, and setting to zero.

$$\frac{\partial L^{\epsilon_p}_{\alpha_p}}{\partial \epsilon_p} = \rho \ln\left(\frac{\epsilon^+_p - \epsilon^*_p}{\epsilon^+_p - \hat{\epsilon}_p}\right) + \sum_{a'_p,a_p} \alpha_p(a'_p, a_p) = 0 \implies \epsilon^*_p \quad = (\hat{\epsilon}_p - \epsilon^+_p) \exp\left(-\frac{1}{\rho} \sum_{a'_p,a_p} \alpha_p(a'_p, a_p)\right) + \epsilon^+_p$$

Therefore:

$$\ln\left(\frac{1}{\exp(1)} \frac{\epsilon^+_p - \epsilon^*_p}{(\epsilon^+_p - \hat{\epsilon}_p)}\right) = \ln\left(\frac{1}{\exp(1)} \exp\left(-\frac{1}{\rho} \sum_{a'_p,a_p} \alpha_p(a'_p, a''_p)\right)\right) = -\frac{1}{\rho} \sum_{a'_p,a''_p} \alpha_p(a'_p, a''_p) - 1$$

Substituting back in:

$$L_{\alpha_p} = \ln\left(\sum_a \hat{\sigma}(a) \exp\left(\mu W(a) - \sum_{p,a'_p,a''_p} \alpha_p(a'_p, a''_p) A_p(a'_p, a''_p, a)\right)\right)$$
$$+ \sum_{p,a'_p,a''_p} \epsilon^+_p \alpha_p(a'_p, a''_p) - \rho \sum_p (\hat{\epsilon}_p - \epsilon^+_p) \exp\left(-\frac{1}{\rho} \sum_{a'_p,a''_p} \alpha_p(a'_p, a''_p)\right)$$

□

A.2 CCEs

Theorem 2 (CCE)**.** *The $\hat{\epsilon}$-MWMRE CCE solution is equivalent to minimizing the loss:*

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

With logits defined as:

$$\overset{CCE}{l}(a) = \mu \sum_a W(a) - \sum_{p,a'_p} \overset{CCE}{\alpha}_p(a'_p)\overset{CCE}{A}_p(a'_p, a)$$

And primal variables defined:

$$\overset{CCE}{\sigma}(a) = \frac{\hat{\sigma}(a)\exp\left(\overset{CCE}{l}(a)\right)}{\sum_{a\in\mathcal{A}} \hat{\sigma}(a)\exp\left(\overset{CCE}{l}(a)\right)} \qquad \overset{CCE}{\epsilon}_p = (\hat{\epsilon}_p - \epsilon_p^+)\exp\left(-\frac{1}{\rho}\sum_{a'_p} \overset{CCE}{\alpha}_p(a'_p)\right) + \epsilon_p^+$$

Proof. Similar proof to Theorem 1. □

B Unit Variance Scaling

The inputs are preprocessed so that the network does not need to learn to be invariant to the offset or scale. This is achieved by using a zero-mean offset, and normalizing by an m-norm, scaled with a constant Z_m. This constant is chosen such that the variance of the elements is one.

$$G_p^{L_m}(a) = Z_m \frac{G_p(a) - \frac{1}{|\mathcal{A}|}\sum_a G_p(a)}{\left\|G_p(a) - \frac{1}{|\mathcal{A}|}\sum_a G_p(a)\right\|_m} \tag{16a}$$

$$\hat{\epsilon}_p^{L_m} = \text{clip}\left(\frac{\hat{\epsilon}_p}{\left\|G_p(a) - \frac{1}{|\mathcal{A}|}\sum_a G_p(a)\right\|_m}, -\hat{\epsilon}^+ = -Z_m, +\hat{\epsilon}^+ = +Z_m\right) \tag{16b}$$

$$W^{L_m}(a) = Z_m \frac{W(a) - \frac{1}{|\mathcal{A}|}\sum_a W(a)}{\left\|W(a) - \frac{1}{|\mathcal{A}|}\sum_a W(a)\right\|_m} \tag{16c}$$

$$\hat{\sigma}^{L_1}(a) = Z_\sigma\left(\hat{\sigma}(a) - \frac{1}{|\mathcal{A}|}\right) \tag{16d}$$

Some scale factors are:

$$Z_\sigma = |\mathcal{A}|\sqrt{\frac{|\mathcal{A}|+1}{|\mathcal{A}|-1}} \qquad Z_2 = \sqrt{|\mathcal{A}|} \tag{17}$$

The constant, Z_σ, of the joint is derived from the variance of elements of a flat Dirichlet distribution. The constant, Z_2, of the the L_2 ring is derived from the mean of the Chi-Squared distribution.

C Equivariant Pooling Functions

Functions which maintain equivariance over player and strategy permutations are useful building blocks for neural network architectures. These comprise of two components; (i) an equivariant pooling function ϕ, such as mean sum, min, or max, and (ii) the reduction dimensions.

An *equivairant pooling function* has three properties: (i) it collapses one or more of the dimensions of a tensor, (ii) the operation is invariant to the order of the elements in the collapsed dimensions, and

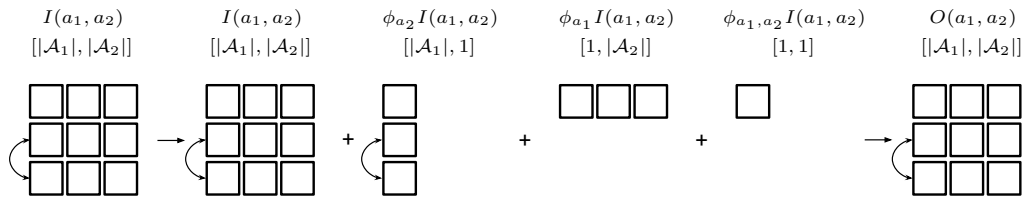


Figure 5: Equivariant Pooling Functions, mapping an input $I(a_1, a_2)$ to an output $O(a_1, a_2)$. Swapping the second and third row (for example) of the input results in the same swap in the outputs.

(iii) the operation is equivariant to the order of the elements in the non-collapsed dimensions. For example, the reduction $\sum_{a_2} G_1(a_1, a_2) = R_1(a_1)$ (i) reduces the dimensionality from $|\mathcal{A}_1| \times |\mathcal{A}_2|$ to $|\mathcal{A}_1|$, (ii) reordering the columns of $G_1(a_1, a_2)$ does not change the calculation, but (iii) reordering the rows of $G_1(a_1, a_2)$ results in an equivariant output, where $R_1(a_1)$ is reordered in the same way.

Such combinations of pooling functions and reduction dimensions can be combined to construct a network that is equivariant to strategy and player permutation. Consider several such pooling functions composed together (Figure 5).

C.1 Payoffs to Payoffs

Some equivariant functions which map payoff structures to payoff structures are:

$$g(p, a_1, ..., a_N) \quad (18a) \qquad \underset{p, a_q}{\phi}\, g(p, a_1, ..., a_N) \quad (18e) \qquad \underset{a_p}{\phi}\, g(q, a_1, ..., a_N) \quad (18i)$$

$$\underset{a_1, ..., a_N}{\phi}\, g(p, a_1, ..., a_N) \quad (18b) \qquad \underset{p, a_{-q}}{\phi}\, g(p, a_1, ..., a_N) \quad (18f) \qquad \underset{a_{-p}}{\phi}\, g(q, a_1, ..., a_N) \quad (18j)$$

$$\underset{p, a_1, ..., a_N}{\phi}\, g(p, a_1, ..., a_N) \quad (18c) \qquad \underset{a_q}{\phi}\, g(p, a_1, ..., a_N) \quad (18g) \qquad \underset{a_q}{\phi}\, g(q, a_1, ..., a_N) \quad (18k)$$

$$\underset{p}{\phi}\, g(p, a_1, ..., a_N) \quad (18d) \qquad \underset{a_{-q}}{\phi}\, g(p, a_1, ..., a_N) \quad (18h) \qquad \underset{a_{-q}}{\phi}\, g(q, a_1, ..., a_N) \quad (18l)$$

$$\forall\, q \in [1, N] \qquad\qquad \forall\, q \in [1, N]$$

If all players have an equal number of strategies, for some primitives (Equations (18e)-(18l)), weights can be shared over all $q \in [1, N]$ because of symmetry.

C.2 CCE Duals to CCE Duals

All equivariant CCE dual pooling functions are given below. Equations (19c) and (19d) may only be used for cube shaped games.

$$\alpha_p(a'_p) \quad (19a) \qquad \underset{a'_p}{\phi}\, \alpha_p(a'_p) \quad (19b) \qquad \underset{p}{\phi}\, \alpha_p(a'_p) \quad (19c) \qquad \underset{p, a'_p}{\phi}\, \alpha_p(a'_p) \quad (19d)$$

C.3 CE Duals to CE Duals

All zero-diagonal equivariant CE dual pooling functions are:

$$\alpha_p(a'_p, a''_p) \quad (20a) \qquad \underset{a'_p}{\phi}\, \alpha_p(a'_p, a''_p) \quad (20c) \qquad \underset{a''_p}{\phi}\, \alpha_p(a'_p, a''_p) \quad (20e) \qquad \underset{a'_p, a''_p}{\phi}\, \alpha_p(a'_p, a''_p) \quad (20g)$$

$$\alpha_p(a''_p, a'_p) \quad (20b) \qquad \underset{a'_p}{\phi}\, \alpha_p(a''_p, a'_p) \quad (20d) \qquad \underset{a''_p}{\phi}\, \alpha_p(a''_p, a'_p) \quad (20f) \qquad \underset{p, a'_p, a''_p}{\phi}\, \alpha_p(a'_p, a''_p) \quad (20h)$$

Table 3: Possible Neural Equilibrium Solver solution parameterizations.

	$G_p(a)$	$\hat{\sigma}(a)$	$\hat{\epsilon}_p$	$\hat{\epsilon}^+$	$W(a)$	ρ	μ
ME	$\sim L_m$	$\frac{1}{\lvert\mathcal{A}\rvert}$	0	Z_m	0	$\gg 1$	0
MT	$\sim L_m$	$\hat{\sigma}(a)$	0	Z_m	0	$\gg 1$	0
MU	$\sim L_m$	$\frac{1}{\lvert\mathcal{A}\rvert}$	0	Z_m	$\sum_p G_p(a)$	$\gg 1$	$\gg 1$
MWME	$\sim L_m$	$\frac{1}{\lvert\mathcal{A}\rvert}$	0	Z_m	$\sim L_m$	$\gg 1$	$\gg 1$
MRE	$\sim L_m$	$\sim \text{Dir}(1)$	0	Z_m	0	$\gg 1$	0
MS	$\sim L_m$	$\frac{1}{\lvert\mathcal{A}\rvert}$	$-Z_m$	Z_m	0	$\gg 1$	0
$\hat{\epsilon}_p$-ME	$\sim L_m$	$\frac{1}{\lvert\mathcal{A}\rvert}$	$\sim \text{U}(-Z_m, Z_m)$	Z_m	0	$\gg 1$	0
$\hat{\epsilon}_p$-MWME	$\sim L_m$	$\frac{1}{\lvert\mathcal{A}\rvert}$	$\sim \text{U}(-Z_m, Z_m)$	Z_m	$\sim L_m$	$\gg 1$	$\gg 1$
$\hat{\epsilon}_p$-MRE	$\sim L_m$	$\sim \text{Dir}(1)$	$\sim \text{U}(-Z_m, Z_m)$	Z_m	0	$\gg 1$	0

D Experiment Architecture and Hyper-Parameters

The architecture and hyper-parameters were chosen from a coarse sweep. The performance of architecture was not very sensitive to parameterization: similar settings will work well, or even better. Nevertheless we provide the details of the exact architecture used in the experiments.

D.1 Architecture

All experiments use the same network architecture, with either CCE or CE dual parameterization, implemented in JAX [5] and Haiku [25]. We used pooling functions (Equations (18a)-(18d) and (18k)-(18l)) for the payoffs to payoffs layers, and used all the pooling functions for dual layers. For ϕ we used both mean and max pooling together. The we used 5 payoffs to payoffs layers, each with 32 channels, a payoffs to duals layer with 64 channels and 2 duals to duals layers with 32 channels, which we denote $[(32, 32, 32, 32, 32), 64, (32, 32)]$. The network has 79,905 parameters. All nonlinearities are ReLUs apart from the final layer where we use a Softplus. Between every layer we use BatchNorm [29] with learned scale and variance correction. The network was initialized such that the variance of activations at every layer is unity. This was done empirically by passing a dummy batch of data through the network and calculating the variance.

D.2 Hyper-Parameters

We used a training batch size of 4096, the Optax [26] implementation of Adam [34] (learning rate 4×10^{-4}) optimizer with adaptive gradient clipping [6] (clipping 10^{-3}). We used a learning rate schedule with (iteration, factor) pairs of $[(1 \times 10^5, 1.0), (1 \times 10^6, 0.6), (4 \times 10^6, 0.3), (7 \times 10^6, 0.1), (1 \times 10^7, 0.06), (1 \times 10^8, 0.03)]$. We included a weight decay loss (learning rate 1×10^{-7}).

D.3 Network Parameterizations

Different possible network parameterizations can be found in Table 3.

D.4 Game Distributions

A list of different game distributions can be found in Table 4. The geometric interpretation of the invariant subspace of games is shown in Figure 6.

D.5 Hardware

We trained our network on a 32 core TPU v3 [33], and evaluated on an 8 core TPU v2 [33]. For intuition, the 8×8 network trains at around 400 batches per second (1,638,400 examples per second). Evaluation is even faster. Bigger games take longer, and scales approximately linearly with the number of joint actions in the game.

Table 4: Classes of random games. We consider the three scale and offset invariant classes introduced in this paper (Section 4.1), and on a subset [50] of GAMUT [47] games using the functions in parenthesis and parameterized with the `-random_params` flag.

Name	Game Description
L_1	L_1 Invariant
L_2	L_2 Invariant
L_∞	L_∞ Invariant
D1	Bertrand Oligopoly (`BertrandOligopoly`)
D2	Bidirectional LEG, Complete Graph (`BidirectionalLEG-CG`)
D3	Bidirectional LEG, Random Graph (`BidirectionalLEG-RG`)
D4	Bidirectional LEG, Star Graph (`BidirectionalLEG-SG`)
D5	Covariance Game, $\rho = 0.9$ (`CovariantGame-Pos`)
D6	Covariance Game, $\rho \in [-1/(N-1), 1]$ (`CovariantGame`)
D7	Covariance Game, $\rho = 0$ (`CovariantGame-Zero`)
D8	Dispersion Game (`DispersionGame`)
D9	Graphical Game, Random Graph (`GraphicalGame-RG`)
D10	Graphical Game, Road Graph (`GraphicalGame-Road`)
D11	Graphical Game, Star Graph (`GraphicalGame-SG`)
D12	Graphical Game, Small-World (`GraphicalGame-SW`)
D13	Minimum Effort Game (`MinimumEffortGame`)
D14	Polymatrix Game, Complete Graph (`PolymatrixGame-CG`)
D15	Polymatrix Game, Random Graph (`PolymatrixGame-RG`)
D16	Polymatrix Game, Road Graph (`PolymatrixGame-Road`)
D17	Polymatrix Game, Small-World (`PolymatrixGame-SW`)
D18	Uniformly Random Game (`RandomGame`)
D19	Travelers Dilemma (`TravelersDilemma`)
D20	Uniform LEG, Complete Graph (`UniformLEG-CG`)
D21	Uniform LEG, Random Graph (`UniformLEG-RG`)
D22	Uniform LEG, Star Graph (`UniformLEG-SG`)

Table 5: Payoffs for two games that show that maximum welfare cannot be discovered via a MW objective to the distribution given by a softmax of welfare. Both games are symmetric, payoffs are given for the row player

(a) CE MW Counterexample

	1	2	3	4
1	-4	2, -2	-999	-999
2	-2, 2	1	-999	-999
3	-999	-999	-3	2, -2
4	-999	-999	-2, 2	1.1

(b) CCE MW Counterexample

	1	2	3	4
1	2	0	0	0, 2
2	0	3	0, 3	-10, 7
3	0	3, 0	-10	-10, -6
4	2, 0	7, -10	-6, -10	0

E Relative Entropy and Welfare Objectives

Any solution can be realised by some relative entropy objective, since if a joint distribution $\sigma(a)$ is a (C)CE, then the solution with a relative entropy objective to $\hat{\sigma}(a) = \sigma(a)$ itself is optimised by $\sigma(a)$. One might imagine therefore that a relative entropy objective could be chosen to induce a maximum welfare solution, based on the payoffs. If possible, this would allow the MWMRE to be simplified. However, it is not straightforward to determine a priori which relative entropy objective(s) will lead to the maximum welfare. This means that relative entropy objectives are insufficient for finding Maximum Welfare solutions.

For example, consider a welfare $W(a) = \sum_p G_p(a)$. We might try to induce a welfare maximising solution by choosing a target joint $\hat{\sigma}(a) = \lim_{T\to\infty} \text{SoftMax}(TW(a))$, where T is the temperature parameter. Finding a CE that minimizes relative entropy to $\hat{\sigma}$ would place high mass on the highest welfare joint action, but is not equivalent to maximizing the linear objective $\sum_a \sigma(a)W(a)$, for either CCEs or CEs.

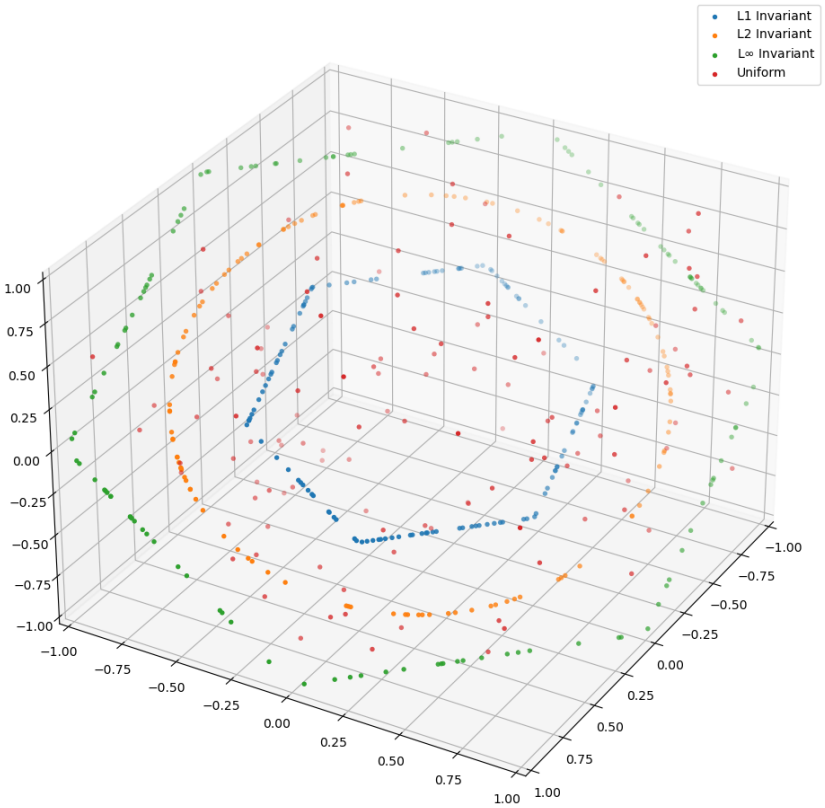


Figure 6: Shows 128 samples of payoffs for player 1 in 3×1 shaped games, under four different distributions. A 3×1 game is not theoretically interesting, but is used here because visualizing above 3 dimensions is difficult. Normalizing by an offset to result in zero-mean payoffs, and by a positive scale to result in unit norm payoffs, is geometrically the surface of an $(|\mathcal{A}| - 1)$-ball. It is straightforward to uniformly sample over such a space. Furthermore, no interesting game structure is lost by only considering this subspace, because offset and positive scale transformations are invariant transformations. It is easy to map any payoff to this invariant subspace, so the neural network can handle any game of appropriate shape at test time. Meanwhile, a naive sample method, such as uniformly sampling payoffs is an unwieldy input for a neural network to decode.

Consider game (a) in Table 5. This game consists of two games of chicken side-by-side, which are mutually incompatible (i.e. the players must co-ordinate to play the same game of chicken to avoid a very large negative payoff for both). The softmax relative entropy objective will prefer the action pair (4, 4) over all others, as it gives slightly higher payoffs than (2, 2). Notice that a CE that recommends (4, 4) must also recommend the action pair (4, 3) some of the time in order to disincentivise the row player from deviating from action 4 to action 3. Similarly, a CE that recommends (2, 2) must also recommend (2, 1) some of the time to disincentivise the row player from deviating from action 2 to action 1.

Crucially, because (1, 1) has a lower payoff for the row player than (3, 3), in CEs consisting (2, 2) the mediator doesn't have to recommend (2, 1) as often as it has to recommend (4, 3) to form an effective disincentive. The result is that a CE that plays exclusively the joint actions $(1, 2)$, $(2, 1)$ and $(2, 2)$ can achieve higher welfare than any that plays $(4, 4)$, despite never playing the welfare maximising joint action.

Game (b) in Table 5 provides a counterexample for the CCE case. It works in a similar way to the CE counterexample: there are two high welfare joint strategies, (1, 1) and (2, 2). The latter has higher welfare, but if played too much a deviation to strategy 4 is incentivised. In the limit of T, the relative entropy objective selects whichever equilibrium has the highest probability of the max-welfare joint.

To disincentivse the row player's deviation to strategy 4, the column player must either: play only strategies 1 and 4, because the strategies 1 and 4 have the same payoff for the row player, or play strategy 3 sufficiently frequently that the benefit of deviating from strategy 2 to strategy 4 is nullified.

The first option gives rise to the max-welfare CCE, which plays (1, 1) with probability 1. The second gives rise to the CCE that plays (2, 2) with the highest possible probability: 0.2. It plays (2, 3) and (3, 2) with probability 0.4 each. This is chosen by the relative entropy objective, but gives each player an average payoff of 1.8, which is equal to the payoff for deviating to action 4 in this equilibrium, but lower than the payoff of (1, 1).