Neural Payoff Machines: Predicting Fair and Stable Payoff Allocations Among Team Members

Anonymous Author(s)
Affiliation
Address
email

Abstract

In many multi-agent settings, participants can form teams to achieve collective outcomes that may far surpass their individual capabilities. Measuring the relative contributions of agents and allocating them shares of the reward that promote long-lasting cooperation are difficult tasks. Cooperative game theory offers solution concepts identifying distribution schemes, such as the Shapley value, that fairly reflect the contribution of individuals to the performance of the team or the Core, which reduces the incentive of agents to abandon their team. Applications of such methods include identifying influential features and sharing the costs of joint ventures or team formation. Unfortunately, using these solutions requires tackling a computational barrier as they are hard to compute, even in restricted settings. In this work, we show how cooperative game-theoretic solutions can be distilled into a learned model that generalizes to previously unobserved games by training neural networks to propose fair and stable payoff allocations. We show that our approach creates models that can generalize to games far from the training distribution and can predict solutions for more players than observed during training. Finally, we show how this approach can be used as an approximate precomputation for speeding up explainable AI methods.

1 Introduction

The ability of individuals to form teams and collaborate is crucial to their performance in many environments. Moreover, the success of humans as a species hinges on our capability to cooperate at scale [1]. Similarly, cooperation between learning agents is necessary to achieve high performance in many environments [2, 3, 4, 5] and is a fundamental problem in artificial intelligence [6, 7]. Often, individual agents are not incentivized by the joint reward achieved by a team but rather by their share of the spoils. Hence, teams are only likely to be formed when the overall gains obtained by the team are appropriately distributed between its members. However, understanding how collective outcomes arise from subsets of locally interacting parts, or fairly measuring the impact of individuals on the team's performance, remain open problems.

Direct applications exist in multiple domains. One example is identifying the most influential features that drive a model to make a certain prediction [8, 9, 10, 11]; one of the cornerstones of explainable AI [12]. Another example is sharing the costs of data acquisition or a joint venture in a fair way between participants [13, 14], or sharing gains between cooperating agents [15, 16]. In many legislative bodies individual participants have different weights, and passing a decision requires support from a set of participants holding the majority of the weight; different states in the US electoral college have different numbers of electors, and different countries in the EU Council of Ministers vary in their voting weight. In such situations we'd like to quantify the true political power held by each participant, or allocate a common budget between them [17, 18].

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

Cooperative game theory can provide strong theoretical foundations underpinning such applications and defines solution concepts that determine the relative impact of individuals on team performance, or the individual rewards agents are entitled to. *Power indices* such as the Shapley value [19] or Banzhaf index [20] attempt to divide the joint reward in a way that is *fair*, and have recently been used to compute feature importance [9]. In contrast, other solutions such as the Core [21] attempt to offer a *stable* allocation of payoffs, where individual agents are incentivised to continue working with their team, rather than attempting to break away from their team in favor of working with other agents. Despite their theoretical appeal, these solution concepts are difficult to apply in practice due to computational constraints. Calculating these solutions, even in restricted environments, is typically a hard problem [22, 23, 24].

Our contribution: We construct models that predict fair or stable payoff allocations among team members, combining solution concepts from cooperative game theory with the predictive power of neural networks. These neural "payoff machines" take in a representation of the performance or reward achievable by different subsets of agents, and output a suggested payoff vector allocating the total reward between the agents. By training the neural networks based on different cooperative solution concepts, the model can be tuned to aim for a fair distribution of the payoffs (the Shapley value or Banzhaf index) or to minimize the incentives of agents to abandon the team (the Least-Core [21, 25, 24]). Figure 1 depicts the two well-studied classes of games on which we evaluate our approach: Weighted Voting Games [17, 18, 23] and feature importance games in explainable AI [8, 9].

Weighted voting games (WVGs) are arguably the most well-studied class of cooperative games. Each agent is endowed with a weight and a team achieves its goal if the sum of the weights of the team members exceeds a certain threshold (quota). WVGs are a model of decision making bodies such as the US Electoral College or the EU Council of Ministers, and solving them allows estimating true political power [17, 18]. We train the neural networks by generating large sets of such games and computing their respective game theoretic solutions. Our empirical evaluation shows that the predictions for the various solutions (the Shapley value, Banzhaf index and Least-Core) accurately reflect the true game theoretic solutions on previously unobserved games. Furthermore, the resulting model can generalize even to games that are very far from the training distribution or with more players than the games in the training set.

Feature Importance Games are a model for quantifying the relative influence that various features have on driving a machine learning model to make a certain prediction [8, 9]. Solving these games for the Shapley value provides an intuitive understanding of the key factors that drove a model to reach a specific decision. This approach is model-agnostic, thus can be applied to make any "black-box" model more interpretable [12]. One drawback of this approach is the computational complexity of calculating the Shapley value, making such analysis slow even when using approximation algorithms. We show how a neural network can serve as way of pre-computing Shapley values, measuring the relative importance of features for a given model. After training, feature importance measures can be obtained instantly for each new input instance through a single forward pass on the trained neural network. Hence, our approximate pre-computation approach can significantly speed-up Explainable AI analysis.

2 Preliminaries

We provide a brief overview of cooperative game theory (examined in details in various books [26, 23]) and discuss how solution concepts in cooperative game theory have been applied in Explainable AI [8, 9, 27, 28, 29].

2.1 Cooperative Game Theory

A (transferable utility) **cooperative game** consists of a set $N = \{1, 2, \ldots, n\}$ of agents, or players, and a characteristic function $v : 2^n \to \mathbb{R}$ which maps each team of players, or *coalition* $C \subseteq N$, to a real number. This number indicates the joint reward the players obtained as a team. Games v where $v : 2^n \to \{0, 1\}$ (binary range) are *simple* games.

Weighted voting games (WVGs) are a restricted class of simple cooperative games [23], where each agent i has a weight w_i and a team of agents $C \subseteq N$ wins if the sum of the weights of its participants $\sum_{i=1}^{n} w_i$ exceeds a quota q. Formally, a WVG is defined as the triple $(\mathbf{w}, q, v)$ with weights $\mathbf{w} = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n_{\geq 0}$ and quota (threshold) $q \in \mathbb{R}_{\geq 0}$ where for any $C \subseteq N$ we

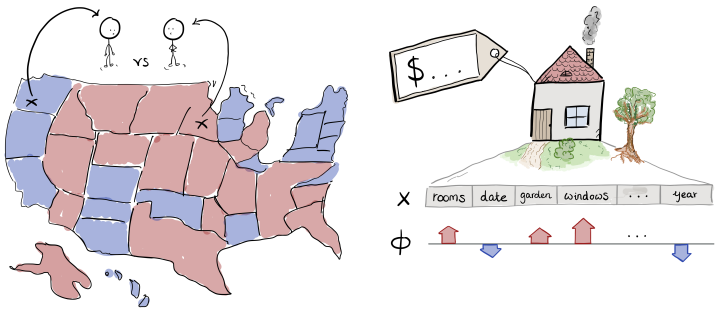


Figure 1: **Evaluation domains for our approach.** *Left*: Weighted Voting Games (WVGs) model decision making bodies such as the US Electoral College. Our approach predicts solution concepts which reflect the true political influence of players or stable payoff allocations (the Shapley value and the Least-Core). *Right*: Applying the Shapley value in Feature Importance Games enables quantifying the relative impact of features on the decisions of a model. In this example, a model predicts the price of a house based on several features. Our approach predicts the Shapley values ϕ of features, which are a commonly used metric for their impact. This speeds-up Explainable AI analysis, as each new instance can be analyzed quicky without a full computation of Shapley values.

have $v(C) = 1$ if $\sum_{i=1}^{n} w_i \geq q$ and otherwise $v(C) = 0$. If $v(C) = 0$ we say C is a losing coalition, and if $v(C) = 1$ we say it is a winning coalitions. WVGs have been thoroughly investigated as a model of voting bodies, such the US Electoral College or the EU Council of Ministers [17, 18].

The characteristic function defines the joint value of a coalition, but it does not specify how the value should be distributed among the agents. *Solution concepts* attempt to determine an allocation $\mathbf{p} = (p_1, \ldots, p_n)$ of the utility $v(N)$ achieved by the grand coalition of all the agents; an allocation $\mathbf{p}$ is called an *imputation* if for any player i we have $p_i \geq 0$ and $\sum_{i=1}^{n} p_i = v(N)$. [1] This allocation is meant to achieve some desiderata, such as fairly reflecting the contributions of individual agents, or achieving stability in the sense that no subset of agents is incentivized to abandon the team and form a new team. We describe three prominent solution concepts that we use in our analysis.

The Core. Rational players may abandon the grand coalition of all the agents if they can increase their individual utility by doing so. The Core is defined as the set of all payoff vectors where no subset of agents can generate more utility, as measured by the characteristic function, than the total payoff they are currently awarded by the payoff vector. As such, the Core is viewed as the set of *stable* payoff allocations. Formally, the core [21] is defined as the set of all imputations $\mathbf{p}$ such that $\sum_{i=1}^{n} p_i = v(N)$ and that $\sum_{i \in C} p_i \geq v(C)$ for any coalition $C \subseteq N$.

The ε-Core and Least-Core [25, 24]. Some games have empty cores, meaning that no payoff allocations achieves full stability (i.e. for any imputation $\mathbf{p}$ there exists at least one coalition C such that $v(C) > \sum_{i \in C} p_i$). In such cases, researchers have proposed minimizing the instability. A relaxation of the core is the **ε-core**, consisting of imputations $\mathbf{p}$ where for any value coalition C we have $p(C) \geq v(C) - \varepsilon$. Given an imputation $\mathbf{p}$ the difference $v(C) - \sum_{i \in C} p_i$ is called the *excess* of the coalition, and represents the total improvement in utility the members of C can achieve by abandoning the grand coalition and working on their own. For an imputation in the ε-core, no agent subset C can achieve an addition of ε in utility over the current total payoff offered to the team (i.e. no coalition has an excess of more than ε). The minimal ε for which the ε-core is non-empty is called the *Least-Core Value* (LCV). The Least-Core minimizes the incentive of any agent subset to abandon the grand coalition, and the LCV thus represents the degree of instability (excess) under the imputation that best minimizes this instability. We find the set of payoffs associated with the LVC through linear programming (full details in Appendix C).

We now discuss two *power indices*, payoff distributions reflecting the true influence a player has on the performance of the team, that fairly allocate the total gains of the teams among the agents in it.

The Shapley Value [19] measures the average marginal contribution of each player across all permutations of the players. The Shapley value is the unique solution concept that fulfills several

[1] In a WVG, the value of a coalition is bounded by 1, so if the grand coalition N indeed has a value of $v(N) = 1$ then a solution would be a payoff vector $\mathbf{p} = (p_1, \ldots, p_n)$ where $\sum_{i=1}^{n} p_i = 1$.

natural fairness axioms [30], and has thus found many applications from estimating feature importance [8, 9] to pruning neural networks [31, 32, 33]. Formally, we denote a permutation of the players by π , where π is a bijective mapping of N to itself, and the set of all such permutations by Π. By $C_\pi(i)$ we denote all players appearing before i in the permutation π. The Shapley value $\phi_i(v)$ of player i is defined as:

$$\phi_i(v) = \frac{1}{N!} \sum_{\pi \in \Pi} [v(C_\pi(i) \cup \{i\}) - v(C_\pi(i))] \tag{2.1}$$

Intuitively, one can consider starting with an empty coalition and adding the players' weights in the order of the permutation; the first player whose addition meets or exceeds the quota is considered the pivotal player in the permutation. The Shapley value then measures the proportion of permutations where a player is the pivotal player.

The Banzhaf index [20] is another method for distributing payoffs according to a players' ability to change the outcome of the game, but it reflects slightly different fairness axioms [34]. The Banzhaf index β_i of a player i is defined as the marginal contribution of a player across all *subsets* not containing that player:

$$\beta_i(G) = \frac{1}{2^{N-1}} \sum_{C \subseteq N \setminus \{i\}} [v(C \cup \{i\}) - v(C)] \tag{2.2}$$

In practice, we first compute the set of winning coalitions $C^{\text{win}} \subseteq C$ and count, for each player, the number of times it is critical or *pivotal*, that is, $v(\{C\} \setminus i) = 0$.

2.2 Explainable AI By Measuring Feature Importance Through Shapley Values

We briefly introduce the use of Shapley values in Explainable AI. For a more detailed discussion see [8, 9, 11, 12]. Shapley values allow quantifying the relative influence of features in driving a machine learning model to reaching its decision [12]. Through a cooperative game theory lens, one may view the features of a data instance as players, working together in a team to drive a model towards its current prediction. As such, we can measure the impact of a feature on the model's prediction by considering all the possible coalitions (combinations) of features, each time replacing the non-coalition features by their respective dataset mean (hence eliminating the unique information conveyed by the value that feature has in that data instance). Consider an instance with features $(\mathbf{x_1}, \ldots, \mathbf{x_k})$ resulting in a model prediction of $f_\theta(x_1, \ldots, x_k)$, and denote by μ_j the dataset mean of feature j. Given a subset C of features, denote by x_j^C the perturbed feature where $x_j^C = x_j$ if $j \in C$ and where $x_j^C = \mu_j$ if $j \notin C$. We can cast the "value" of a feature team C as the following characteristic function $v(C) = f_\theta(x_1^C, x_2^C, \ldots, x_j^C)$. Recently, convenient frameworks that approximate Shapley values such as SHAP [9] spurred the rapid adoption of the Shapley value within the explainable AI community [12].

3 Methods

As computing cooperative game-theoretic concept solutions is often intractable, we use deep learning approaches to efficiently approximate them. To evaluate our methods, we consider two domains in our experiments. The first domain is that of Weighted Voting Games (WVGs), where we use neural networks to measure the true influence of participants or to predict a stable payoff allocation. To train our models, we creates a dataset of randomly generated WVGs (with different distributions of their parameters). We also assess the quality of our models on the out-of-sample distributions and their generalization to a larger number of players.

The second domain is that of feature importance measures for explainable AI. We show how to apply our method to the Shapley-based feature importance setting discussed in Section 2.2. Thus, we can speed-up the computations of the relative impact of features on the predictions of a machine learning model. We do this by training our models to approximate Shapley values of features, and examining their performance on previously unobserved instances.

3.1 Weighted Voting Games

We consider two types of models: *fixed-size* models that always predict a fixed number of outputs and can handle games with a fixed number of players, and *variable-size* ones that predict a variable number of outcomes and can handle variable numbers of players. We now describe our data generation process, models, training procedure, and evaluation metrics used in weighted voting games.

3.1.1 Data and Models: Fixed-Size

For each n player game, we generate G independent and identically distributed games. The training dataset is given by $\mathcal{D}^n_{\text{fixed}} = \left\{\mathbf{X} \in \mathbb{R}^{G\times n}, \mathbf{P} \in \mathbb{R}^{G\times K}\right\}$ where K denotes the number of outputs of interest. The features are obtained in two steps. First, we sample a weight vector $\mathbf{w} \sim \text{Beta}(\alpha = 1, \beta = 1)$ on the interval from 1 to $2n$, and a quota $q \sim \mathcal{N}(\mu = \frac{1}{4}(2n+1)n, \sigma^2 = 2n)$. To create games where players are dependent on each other to achieve the task at hand, the quota distribution is parameterized such that the average drawn quota is half of the sum of the players' weights. We then normalize the weights by the quota to get the feature vector $\mathbf{x} = \frac{1}{q} \cdot \mathbf{w}$. In other words, we have n features which are the weights normalized as the respective proportion of the quota. Thus, a value $x_i > 1$ indicates that player i is a winning coalition by itself, and needs other players to meet the quota otherwise (Figure 2 on the left).

We train models to predict the three solutions: the Least-Core, the Shapley values, and the Banzhaf indices. For the Shapey values and the Banzhaf indices, the model predicts the payoff allocation $(p_1, \ldots, p_n)$ so $K \equiv n$. For the Least-Core solution, we train the model to not only predict the payoff allocation $p_1, \ldots, p_n$, but also to predict the Least Core Value ε_{min} (so in this case the model has $K = n + 1$ outputs). For each n player game, we produce a model $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^K$, where $\boldsymbol{\theta}$ are the model parameters. We use deep feedforward networks for $f_{\boldsymbol{\theta}}$. Appendix D.1 contains the full details about the experimental setup.

3.1.2 Data and Models: Variable-size

For the variable-size case, we consider a maximal number of possible players M, and for games with less than M players we pad the inputs with zeros. Hence, we generate a single dataset $\mathcal{D}_{\text{var}} = \left\{\mathbf{X} \in \mathbb{R}^{G\times M}, \mathbf{P} \in \mathbb{R}^{G\times K}\right\}$. Similarly to the fixed-size dataset, the feature matrix $\mathbf{X}$ contains the normalized weights and $\mathbf{P}$ the corresponding solutions with either $K = M+1$ or $K = M$, with the ground truth output vector again padded with zeros when there are fewer than M players. Hence, we allow for the prediction of up to M players, and we shuffle the data so that players are located at random positions. The Least Core Value ε_{min} is not shuffled and instead stored at the last element of each row (Figure 2 on the right).

We produce one model $g_{\boldsymbol{\eta}} : \mathbb{R}^n \to \mathbb{R}^K$ that can be used for different number of players n, where $\boldsymbol{\eta}$ are the model parameters. The model learns to allocate the joint payoff among at most M players. During prediction time we pad the input with zeros when there are fewer than M players, and we redistribute the payoffs allocated to non-player entries among the players according to their original share of the joint payoff (i.e. we ignore the outputs in the non-player positions, and renormalize the outputs in the player positions so they sum up to 1).

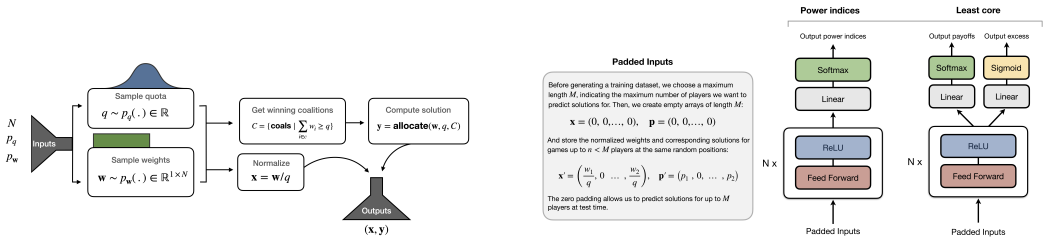


Figure 2: **Data and Models.** *Left*: Procedurally generating one weighted voting game. *Right*: Neural Architectures for Variable-size game prediction.

3.1.3 Training and Evaluation

During training we minimize the mean square error between the true and predicted solutions (over the payoffs of all players). For the variable-size models we also include the $M - n$ padded locations so that the model learns not to allocate value to non-player entries.

Evaluation metrics. Given a predicted payoff vector $\hat{\mathbf{p}} = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n)$, we consider multiple metrics for assessing the models' performance. First, we quantify the models' predictive performance via the **Mean Absolute Error**, defined for each game as $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |p_i - \hat{p}_i|$, where n is the number of players. For the Least-Core there is another natural game theoretic metric. The goal of the Least-Core is to minimize the incentives of any subset to abandon the current team and form its own sub-team. Given a suggested payoff vector $\hat{\mathbf{p}}$, the maximal excess $v(C) - \sum_{i\in C} \hat{p}_i$ over all

possible coalitions C measures the incentive to abandon the team, and serves as a good measure for the quality of the model.

Test data. The vector of weights and the quota together determine the set of winning coalitions in a WVG, which in turn establish the payoff allocation. To assess our models' ability to generalize to previously unobserved instances and see how well they generalize on games far outside of the training distribution, we sample the weights $\mathbf{w} \sim \text{Beta}(\alpha, \beta)$ with varying parameters for α and β and the distribution location (full details in Table 2). A visualization of the five standardized test distributions is found in Figure 9.

3.2 Explainable AI: Approximately Computing Feature Importance By Predicting Shapley Values

As discussed in Section 2.2, the Shapley value offers a principled method for identifying the relative influence that various features have on a machine learning model in making its prediction on a given instance. Despite its theoretical appeal, computing Shapley-based feature importance measures is a computationally demanding process; for each analyzed data instance, directly applying the Shapley formula requires considering all permutations and thus it is exponential in the number of features. Even for a moderate number of features and even when using approximations such as those implemented in recent packages such as SHAP [9], the process is slow.

We show that neural networks can provide a faster alternative for measuring feature importance at scale. A neural model for predicting Shapley values can be trained on a set of instances; following training this neural model can quickly predict Shapley values for many unseen instances, only requiring a single forward pass to obtain approximations of Shapley feature importance measures. Our procedure is to select a sampled sub-dataset $\mathbf{X}$ of M instances with K features each, and exactly compute the Shapley value for each feature, yielding a matrix $\mathbf{\Phi}$. Then, we train a neural network $h_{\boldsymbol{\psi}} : \mathbb{R}^K \to \mathbb{R}^K$ that predicts a Shapley value ϕ_{ij} for each feature x_{ij} (full details in Appendix D.2).

A data efficient model must exhibit good performance on the test set even with a relatively small number of training samples. To study the sample complexity, we perform a simple experiment. First, we create an array of $T = 999$ linearly spaced train-test splits $\mathbf{t} = (0.001, 0.002, \ldots, 0.998, 0.999)$. Each element t_i in $\mathbf{t}$ indicates the ratio of samples that are used for training the neural network, with the remaining $(1 - t_i)N$ samples used for evaluation. For each train-test partition in $\mathbf{t}$ then, we train a feed forward neural network on $t_i N$ samples by minimizing the MSE between the predicted and actual Shapley values (full details in Table 3) for 100 epochs, and evaluate on the $1 - t_i N$ samples.

4 Results

We present experimental results that allow us to assess how well neural models are able to learn a representation of the various solution concepts. We first describe the predictive performance of neural networks in the WVG setting, and consider properties of these solutions that make them hard to learn. We then consider the explainable AI domain, and study the performance and sample complexity of Shapley feature importance prediction.

4.1 Weighted Voting Games: Evaluation

For our WVG analysis we train a selection of fixed-size neural networks $f^N_{\boldsymbol{\theta}^*}$ for each number of players $n \in \{4, 5, \ldots, 19, 20\}$[2] on $G = 5,000$ games each. We also train a single variable-size model $g_{\boldsymbol{\eta}^*}$ that is trained on one dataset containing $G = 17,500$ games in total, consisting of $2,500$ games for each number of players $n \in \{4, 5, \ldots, 9, 10\}$ games. The data is padded with zeros to allow for payoff allocation up to $M = 20$ players (see details in Section 3.1).

4.1.1 Predictive Performance

Table 1 shows that the ability to handle games of variable numbers of players comes at a the cost of having a lower accuracy. However, even for variable-size models, and even under a significant distribution shift, the errors in predicting all solution concepts are low. We further note that the

[2] A direct computation of the Shapley value requires enumerating through a large list of permutations, which becomes computationally very costly when there are many players. Hence, for games with $n = 9$ players or more, we use Monte-Carlo approximations for the Shapley value to obtain the ground-truth solution. See full details in Appendix D.1)

error in predicting Least-Core based payoffs are generally larger than for the Shapley and Banzhaf power indices. One possible reason is that for these solutions, there are many cases where a small perturbation in the weights or quota results in a large pretubation of the Least-Core solution [35, 36].

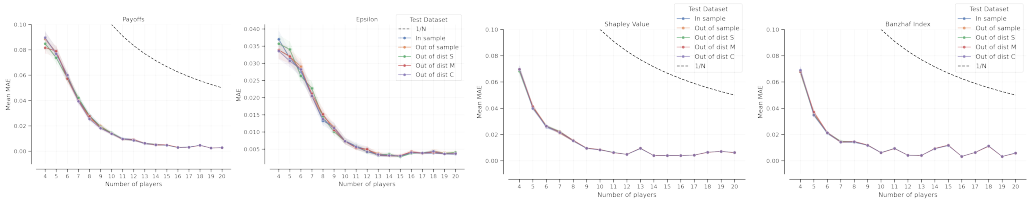
Table 1: Comparison of predictive performance across test datasets and solution concepts.

	Least core payoffs		**Least core excess**		**Shapley values**		**Banzhaf indices**	
Dataset	Mean MAE		MAE		Mean MAE		Mean MAE	
	Fixed	Variable	Fixed	Variable	Fixed	Variable	Fixed	Variable
In-sample	0.0222	0.0345	0.0110	0.0447	0.0161	0.0183	0.0141	0.0238
Out-of-sample	0.0222	0.0373	0.0111	0.0229	0.0163	0.0269	0.0141	0.0418
Slightly out-of-distribution	0.0218	0.0224	0.0112	0.0644	0.0159	0.0154	0.0141	0.0139
Moderately out-of-distribution	0.0216	0.0308	0.0110	0.0257	0.0157	0.0211	0.0142	0.0253
Significantly out-of-distribution	0.0219	0.0501	0.0108	0.0406	0.0163	0.0341	0.0144	0.0473

We now highlight several conclusions from our analysis of the results.

Fixed-size models display stable performance across solution concepts. Our first observation is that the fixed-size neural networks are adept at estimating solutions across the three considered concepts. The average error per player ranges from 0.005 to 0.087 (Least Core), 0.004 to 0.069 (Shapley values) and 0.012 to 0.067 (Banzhaf indices). Table 5 provides a complete summary of our models' in-sample performance.

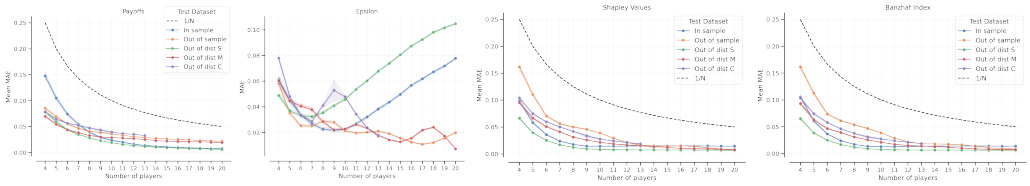
Fixed-size models are robust to shifts in the weight distribution. As shown in Figure 3, the predictive performance (Mean MAE) of the fixed-size models is consistent across test datasets. To account for the natural decrease in the MAE as N increases, we also display the average payoff per player ($1/N$). As expected, the error scales approximately with the average payoff per player.



(a) Least Core (b) Shapley value (c) Banzhaf index

Figure 3: **Performance fixed-size models.** We evaluate the performance on five test sets with 1000 samples per n-player game. Figures show the MAE and 95 % confidence intervals for each solution concept. The black dashed line indicates the average payoff per player as a function of n.

Variable-size models are robust to shifts in the weight distribution. Figure 4 shows that the variable-size models are also able to generalize outside the training distribution. Across test sets, we observe a stable performance that decays with the number of players, as is expected. For the Least Core, we see that there is an abrupt decrease in performance for the excess beyond 8 players.



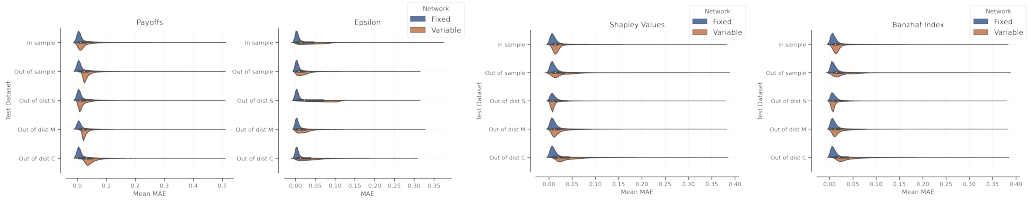
(a) Least Core (b) Shapley value (c) Banzhaf index

Figure 4: **Performance variable-size models.** We evaluate the performance on five test sets with 1000 samples per n-player game. Figures show the MAE and 95 % confidence intervals for each solution concept. The black dashed line indicates the average payoff per player as a function of n.

Variable-size models generalize to a larger number of players. Our main objective is to investigate how to leverage machine learning to perform scalable value estimation in large multi-agent settings. To test this, we train the variable-size models on games up $n = 10$ players, and evaluate on

$n+1, n+2, \ldots, n+10$ players (full details in section 3.1.2). Interestingly, Figure 4 demonstrates that there is no significant decrease in performance for games with more than $n = 10$ players across solution concepts: variable-size models are able to extrapolate beyond the number of players seen during training. This suggests that there is be valuable information in the small player games such that games of larger sizes can be inferred.

Fixed-size models outperform variable-size models. Table 1 contains the Mean MAEs across solution concepts and test sets. Corresponding error distributions are showed in Figure 5. From these, we conclude that in almost all settings the fixed-size networks outperform the variable-size networks. Variable-sized networks tend to have larger errors for all predicted variables and display more variance in their predictions.



(a) Least Core (b) Shapley value (c) Banzhaf index

Figure 5: **Comparing overall performance fixed-size vs. variable-size networks.**

4.1.2 Discontinuities in the Solution Space

One obstacle that makes it difficult to learn the function mapping from game parameters to their respective solutions is the fact that there are many discontinuities. For various parameters, such as the quota of the WVG, an infinitesimally small perturbation in the parameter results in a large shift in the solution. We now examine several examples of such discontinuities and see how the neural network models attempt to capture them.

Some solution concepts are step-wise constant functions which are difficult to capture. Consider taking a WVG and changing the weight of a player. This only changes the game theoretic solutions when it restructures the subset of winning coalitions. This means that the function outputs the same value until the weight reaches a certain threshold where the structures of winning coalitions change, at which point the solution can change drastically. Learning these kind of functions is difficult, as the error around the discontinuity point is often large. We inspect several specific WVGs to get see how the trained models capture these discontinuous jumps.

Our analysis examines an n player game with a fixed weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_n)$ and considers an array of quotas $\mathbf{q} = (q_1, q_2, \ldots, q_K) = (\min(\mathbf{w}), \min(\mathbf{w}) + \epsilon, \ldots, \sum_{i=1}^{n} w_i)$, where $\epsilon = 0.1$. We solve the game for each combination of the fixed weights and the changing quota in $\mathbf{q}$ to obtain a matrix $\mathbf{P}$. Figure 6 shows the fixed-size model predictions for two selected WVGs. The models capture the overall effect of the changes to the quota, but do poorly close to the discontinuity point (where the ground truth solution incurs a large change).

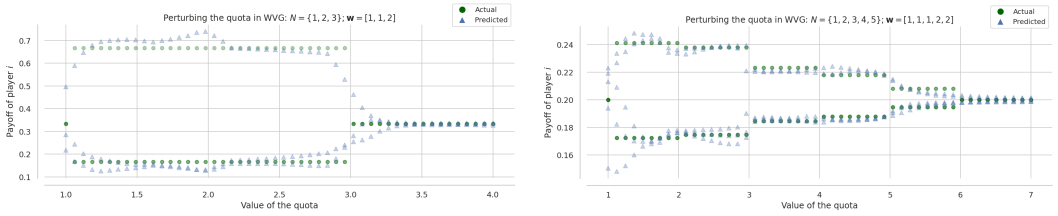


Figure 6: **Capturing step-wise jumps.** Individual actual payoffs (green dots) together with the model predictions (blue triangles) as the value of quota increases by $\epsilon = 0.1$ increments (full description in Appendix E).

4.2 Speeding Up Feature Importance Calculations By Predicting Shapley Values

Following the experimental procedure outlined in Section 3.2, we examine the sample efficiency and performance of neural networks trained to predict Shapley values on sets of features. We consider

two well-known datasets: the classical UCI Bank Marketing dataset [37] which has 15 different indicators with over 11,162 observations, and the Melbourne Housing dataset [38], which contains 20 features and 34,857 observations (full details in Appendix D.2). Overall, we find that neural networks achieve high performance on the test set with very few training samples.

Figure 7 displays the models' performance (in RMSE) as a function of samples available for training. We observe that the error decays approximately exponentially with the proportion of data used for training on both the Banking and Melbourne datasets. A model trained on just 1 percent (349 samples) of the Melbourne dataset has an RMSE of 19.72 on the resultant test set (34,508 samples). Increasing the number of training samples to 3 percent (1045 samples) results in a RMSE of less than 0.07 (on 33,812 samples), a 99.6 percent decrease in error. We observe a similar trend for the Banking dataset: the RMSE decreases with 93.8 percent (from 4.10 to 0.25) when the available training data increases from 0.5 to 4 percent. For both datasets, the error stabilizes after approximately 5 percent of the data is available for training. These results show that a very small portion of the instances is required for the neural model to generalize well to unseen instances. These results highlight the strength of this approach: the computational cost of training the Shapley prediction network is very small as compared to the speedup obtained on the vast majority of the data (and any subsequent instances analyzed later).

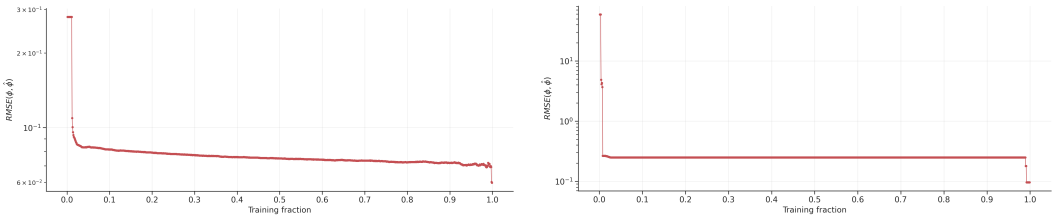


Figure 7: **Root Mean Squared Error as a function of training fraction.** The RMSE between the actual and predicted Shapley values scales approximately as a power law with the training fraction. *Left*: UCI Banking dataset *Right*: Melbourne Housing dataset.

5 Discussion and Conclusion

We considered "neural payoff machines", models that take in a representation of a cooperative game and solve it to produce a distribution of the joint gains of a team amongst its members. Our analysis focused on two concise representations of a characteristic function: Weighted Voting Games, and Feature Influence Games for explainable AI. Our analysis shows that neural models capture cooperative game solutions well, and can generalize well outside the training distribution, even extending to more players than in the training set.

Given the larger MAE for the Least-Core, we believe that this is a harder solution to learn than the Shapley value and Banzhaf index. Potentially, the fact that the Least core is complex and can only be computed by solving multiple Linear Programs makes a hard concept to grasp from examples.

The methods we discussed can drive better analysis of decision making bodies and for approximating feature importance, but such methods can also be applied to other classes of cooperative games, so long as one can generate data consisting of game samples and their solutions. For instance, the same approach can be used for other cooperative representations, such as graph-based games [24, 39, 40] or set-based representations [41]. The direct applications that we have considered in this paper are also intrinsically valuable on their own, as we allow speeding up explainable AI analysis and political influence estimation. Some questions remain open for future work. Can similar techniques be used for non-cooperative games or for other types of cooperative games? Can this approach be applied to other solutions such as the Kernel or Nucleolous [23]? Finally, are there better neural network designs, such as graph neural networks or transformers, that can exploit invariances or equivariances in the game's representation or be better equipped to deal with sharp discontinuities in the solutions?

References

[1] Joseph Henrich. The secret of our success. In *The Secret of Our Success*. Princeton University Press, 2015.

[2] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.

[3] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[4] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[5] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

[6] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[7] Allan Dafoe, Yoram Bachrach, Gillian Hadfield, Eric Horvitz, Kate Larson, and Thore Graepel. Cooperative ai: machines must learn to find common ground, 2021.

[8] Anupam Datta, Shayak Sen, and Yair Zick. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *2016 IEEE symposium on security and privacy (SP)*, pages 598–617. IEEE, 2016.

[9] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[10] Yoram Bachrach, Thore Graepel, Gjergji Kasneci, Michal Kosinski, and Jurgen Van Gael. Crowd iq: aggregating opinions to boost performance. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 535–542, 2012.

[11] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[12] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58:82–115, 2020.

[13] Eric Balkanski, Umar Syed, and Sergei Vassilvitskii. Statistical cost sharing. *Advances in Neural Information Processing Systems*, 30, 2017.

[14] Anish Agarwal, Munther Dahleh, and Tuhin Sarkar. A marketplace for data: An algorithmic solution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 701–726, 2019.

[15] Dermot Gately. Sharing the gains from regional cooperation: A game theoretic application to planning investment in electric power. *International Economic Review*, pages 195–208, 1974.

[16] Rachael Hwee Ling Sim, Yehong Zhang, Mun Choon Chan, and Bryan Kian Hsiang Low. Collaborative machine learning with incentive-aware model rewards. In *International Conference on Machine Learning*, pages 8927–8936. PMLR, 2020.

[17] Irwin Mann and Lloyd S Shapley. Values of large games. 6: Evaluating the electoral college exactly. Technical report, RAND CORP SANTA MONICA CA, 1962.

[18] Jesús Mario Bilbao, Julio R Fernandez, Nieves Jiménez, and Jorge J Lopez. Voting power in the european union enlargement. *European Journal of Operational Research*, 143(1):181–196, 2002.

[19] Lloyd S Shapley. A value for n-person games. aw kuhn, hw tucker, ed., contributions to the thoery of games ii, 1953.

[20] John F Banzhaf III. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

[21] Donald Bruce Gillies. *Some theorems on n-person games*. Princeton University, 1953.

[22] Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. Computational complexity of weighted threshold games. In *AAAI*, pages 718–723, 2007.

[23] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

[24] Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of operations research*, 19(2):257–266, 1994.

[25] Michael Maschler, Bezalel Peleg, and Lloyd S Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of operations research*, 4(4):303–338, 1979.

[26] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[27] Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

[28] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[29] Tom Yan and Ariel D Procaccia. If you like shapley then you'll love the core. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5751–5759, 2021.

[30] Pradeep Dubey. On the uniqueness of the shapley value. *International Journal of Game Theory*, 4(3):131–139, 1975.

[31] Shiqiang Wang. Efficient deep learning. *Nature Computational Science*, 1(3):181–182, 2021.

[32] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[33] Amirata Ghorbani and James Y Zou. Neuron shapley: Discovering the responsible neurons. *Advances in Neural Information Processing Systems*, 33:5922–5932, 2020.

[34] PD Straffin jr. "the shapley-shubik and banzhaf power indices as probabilities". *The Shapley Value: Essays in Honor of Lloyd S. Shapley*, pages 71–81, 1988.

[35] Edith Elkind and Dmitrii Pasechnik. Computing the nucleolus of weighted voting games. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 327–335. SIAM, 2009.

[36] Michael Zuckerman, Piotr Faliszewski, Yoram Bachrach, and Edith Elkind. Manipulating the quota in weighted voting games. *Artificial Intelligence*, 180:1–19, 2012.

[37] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[38] Tony Pino. Melbourne housing market data, 2018.

[39] Haris Aziz, Oded Lachish, Mike Paterson, and Rahul Savani. Power indices in spanning connectivity games. In *International Conference on Algorithmic Applications in Management*, pages 55–67. Springer, 2009.

[40] Yoram Bachrach, Jeffrey S Rosenschein, and Ely Porat. Power and stability in connectivity games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 999–1006, 2008.

[41] Naoki Ohta, Vincent Conitzer, Yasufumi Satoh, Atsushi Iwasaki, and Makoto Yokoo. Anonymity-proof shapley value: extending shapley value for coalitional games in open environments. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 927–934, 2008.

[42] John Deegan and Edward W Packel. A new index of power for simplen-person games. *International Journal of Game Theory*, 7(2):113–123, 1978.

[43] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[44] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

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] The abstract and introduction provide representative summaries of our work.
   (b) Did you describe the limitations of your work? [Yes] See section 4.1.2 and the Discussion.
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
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)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See methods and Appendix D.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (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] See section F.
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? [N/A]
   (b) Did you mention the license of the assets? [Yes] See Appendix G.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (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]

Appendix

A Notation

The following notation is used throughout this paper:

- n the number of players in the game
- G the number of games
- M the maximum sequence length in variable feedforward networks
- q the quota, threshold that determines when a coalition can complete the goal or task
- w_i the weight of a player i
- $x_i = w_i/q$, the normalized weight of a player i
- N the grand coalition
- C a coalition (subset of players)
- C^{min} the minimal set of winning coalitions: the coalitions where each player is pivotal
- $\mathbf{X}$ feature matrix
- $\mathbf{P}$ solution matrix
- p_i the allocated reward or payoff to a player i
- ε the least core value (LCV), i.e. joint payoff to sacrifice such that there exists a feasible payoff vector in the least core

B Motivating The Use Of The Shapley Value To Analyze Influence In Weighted Voting Games

We provide an intuitive example motivating the use of the Shapley value for measuring the true influence of participants in weighted voting games or political power in decision making bodies.

Consider a decision making body such as a parliament with 100 seats, and imagine that following the elections we have two large parties of 49 seats each, and one tiny party who has only managed to get 2 seats. However, note that having a majority means controlling a majority of the seats, i.e. more than 50 seats. This means that neither of the two big parties has a majority on its own (and of course the small party hasn't got a majority on its own).

Hence, to achieve a required majority and form a government or pass a bill, the parties have to work in teams. Clearly, a coalition consisting of the two big parties has a majority, as together they have 49+49=98 seats out of the total of 100. However, a coalition consisting of any of the two big parties and the one small party also has 49+2=51 seats, which is also a majority.

Hence, in the setting discussed above, *any* two parties form a winning coalition that has a majority and is able to form a government or pass a bill. This can be modelled as a weighted voting game, where the weights are $(w_1 = 49, w_2 = 49, w_3 = 2)$ and the quota is $q = 50$.

Intuitively, as any two parties or more are a winning team and as any single party is a losing team, we might say that the weights are irrelevant. Although the big parties each have almost 25 times the seats as the small party, they all are symmetric in their opportunities to form a winning coalition, and one could claim they actually have equal political power. Clearly, the small party is in a very strong negotiation position, and could demand for instance to control a significant proportion of the joint budget.

In the above weighted voting game, the Shapley value of all agents would be equal (each getting a value of $\frac{1}{3}$), as it considers the marginal contributions of the parties over all permutations of players. Due to the symmetry between the parties, they have identical marginal contributions.

The above specific example illustrates some of the principles behind the fairness axioms that the Shapley value exhibits, and illustrates its importance for measuring influence in decision making bodies.

C Algorithm For Generating Least Core Solutions

The least core solutions can be generated by considering all possible coalitions. A simple improvement, however, is to use the minimal set of winning coalitions [42] instead. In this section, we describe the idea behind this approach and the algorithm used in this paper.

Let $N \equiv \{1, 2, \ldots, n\}$ denote the set of all $N \in \mathbb{Z}_{\geq 0}$ players, let $q \in \mathbb{R}_{\geq 0}$ denote the quota and let $w_i \in \mathbb{R}_{\geq 0}$ denote the weights for player i, for all $i \in N$. The characteristic function is given by $v(C) \in [0, 1]$, for coalitions $C \subseteq N$.

The naive approach to find the least core solutions is by considering all winning coalitions C^{win} defined as

$$C^{\text{win}} \equiv \left\{ C \subseteq N \;\middle|\; C \neq \varnothing \text{ and } \sum_{i \in C} w_i \geq q \right\} \tag{C.1}$$

and to find a least core solution by solving LP-problem 8a. In LP-problem (8a), all winning coalitions are considered and there could be sub-coalitions in winning coalitions that are still winning and could deviate from the coalition by forming a smaller coalition. This yields a higher payoff for the players in the sub-coalition and is hence a Pareto improvement for the "decision-makers". Therefore, we can instead only consider the subset of *minimal winning* coalitions, given by

$$\begin{aligned} \min_{p_1, p_2, \ldots, p_n, \varepsilon} \quad & \varepsilon \\ \text{s.t.} \quad & \sum_{i \in C} p_i \geq 1 - \varepsilon \quad \forall\, C \in C^{\text{win}} \\ & \sum_{i \in N} p_i = 1 \\ & p_i \geq 0 \qquad \forall\, i \in N \end{aligned}$$

(a) Naive approach

$$\begin{aligned} \min_{p_1, p_2, \ldots, p_n, \varepsilon} \quad & \varepsilon \\ \text{s.t.} \quad & \sum_{i \in C} p_i \geq 1 - \varepsilon \quad \forall\, C \in C^{\min} \\ & \sum_{i \in N} p_i = 1 \\ & p_i \geq 0 \qquad \forall\, i \in N \end{aligned}$$

(b) Minimal approach

Figure 8: Algorithms for finding least core solutions.

$$C^{\min} \equiv \left\{ C \subseteq N \;\middle|\; C \neq \varnothing, \sum_{i \in S} w_i \geq q, \text{ and, for all } j \in C, \sum_{i \in C \setminus \{j\}} w_i < q \right\} \tag{C.2}$$

and solve the more efficient LP-problem 8b.

D Experimental details

In this section, we elaborate on the experimental procedures, model architectures and hyperparameters used in our experiments.

Code and notebook. Following the guidelines, we provide python code to reproduce our results, a pedagogical notebook (called CooperativeGameTheoryPrimer.ipynb) that introduces the most important game theory formalisms and the necessary data in the accompanied zip file.

D.1 Weighted Voting Games

Optimization procedure fixed-size feedforward networks. For each N-player dataset $\mathcal{D}^N_{\text{fixed}}$ we perform $R = 100$ independent runs with a maximum of $E = 6000$ epochs and an early-stopping criteria which breaks a run if, after a baseline of 500 epochs, the validation loss does not improve for 75 consecutive epochs. For each run, 70 percent is allocated for training and 30 percent for validation. After training, we select the model with the best validation loss. The selected models are stored and used for evaluation. In all experiments we use the Adam [43] optimizer, ReLU [44] activation functions for the hidden layers, Softmax functions for output layer of the payoffs and a Sigmoid for the output layer of epsilon.

Optimization procedure var-size feedforward network. We perform $R = 100$ independent runs for one variable dataset $\mathcal{D}_{\text{var}}$ with a maximum of $E = 15000$ number of epochs and an early-stopping criteria which breaks a run if, after a baseline of 500 epochs, the validation loss does not improve for 75 consecutive epochs. For each run, 70 percent is allocated for training and 30 percent for validation. After training, we select the model with the best validation loss, which is stored and used

for evaluation. In all experiments we use the Adam [43] optimizer, ReLU [44] activation functions for the hidden layers, Softmax functions for output layer of the payoffs and a Sigmoid for the output layer of epsilon.

Hyperparameters. All results are generated with multi-layer perceptrons (feedforward networks) of varying hidden-layer size and width. Further hyperparameters are the dropout rate, weight decay in the Adam optimizer, and the learning rate. Our standard model architectures are depicted in Figure 2 Right.

Approximated ground-truth Shapley values. To obtain the ground-truth solutions for Shapley values above nine players, we perform Monte-Carlo sampling to obtain a subset of all possible permutations. For each of the $R = 10$ resamples we randomly sample $P = 1000$ permutations and average across the resamples to obtain approximations of the ground-truth. To validate that the approximated labels are representative we compute the Mean MAE between the true and approximated Shapley values for $n = 5$ up to $n = 10$ players with 1000 games per n-player game. Across all n-player games the mean MAE between the true and approximated Shapley values was 0.0011 and never exceeded 0.0014. Figure 10 shows the distributions per player together with the true label (red dot) for 8 and 9 player games.

Parameterizations for test datasets. Figure 9 shows the five standardized test distributions that are generated with the parameters in Table 2.

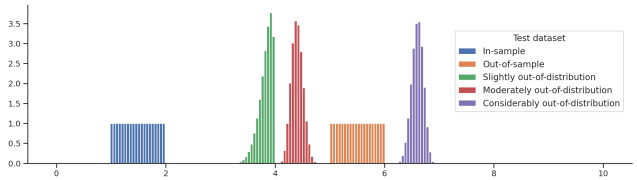


Figure 9: **Test data distributions in perspective.** To assess how robust models' are to shifts in the weight distribution, we re-parameterize the weight distribution in five different ways.

Table 2: Test distributions are generated by the following parameterizations of the beta distribution

Test set name	α	β	location
In-sample	1	1	1
Out-of-sample	1	1	2.5N
Slightly out-of-distribution	8	12	2
Moderately out-of-distribution	7	1.5	1.5N
Considerably out-of-distribution	12	8	3N

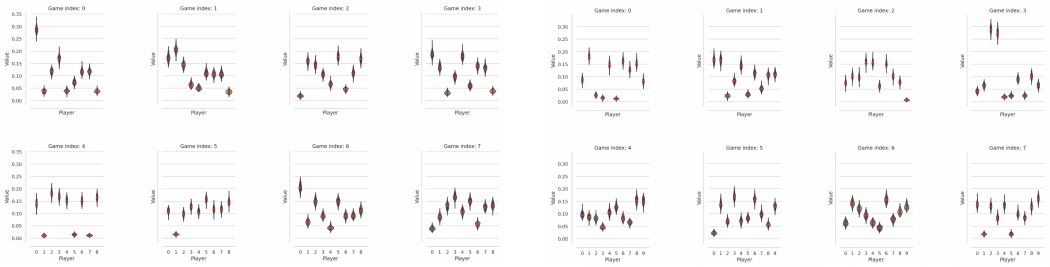


Figure 10: **Approximating ground-truth Shapley values.** Ten randomly sampled games with distribution for each player in the game, the red dot indicates the true Shapley value (i.e. when computing the marginal value for each permutation).

D.2 Explainable AI

To train a neural network to predict the relative importance of each feature in a dataset, our procedure is as follows:

1. **Select a dataset.** The first step is to select a dataset of choice. This can be a set of images or a set of features. In our example, we end up with a feature matrix $\mathbf{X}^{N\times F}$ where each row is an observation and each column a feature, and a target vector $\mathbf{y} \in \mathbb{R}^N or [0,1]$ depending on the task.

2. **Feature to target.** Following, we estimate a model $f_{\boldsymbol{\theta}} : \mathbb{R}^F \to \mathbb{R}$ that predicts one output for each set of features.

3. **Compute Shapley values.** We use the trained model from step 2 to obtain Shapley values with the SHAP package [9], yielding a matrix $\mathbf{\Phi} \in \mathbb{R}^{N\times F}$.

4. **Feature to Shapley.** Finally, we train a model $h_{\boldsymbol{\psi}} : \mathbb{R}^F \to \mathbb{R}^F$ that estimates a Shapley value ϕ_{nf} for each feature x_{nf}.

Datasets and preprocessing. Table 4 summarizes the considered datasets. The UCI Banking dataset contains 15 features and a binary target (whether the client has subscribed a term deposit), whereas the Melbourne housing dataset contains 21 features and a regression target (the price of a house). We exclude the "duration" feature in the Banking dataset to prevent data leakage, as is recommended in the documentation. Moreover, we exclude the "Rooms" feature as there is a lot of overlap in this and the "Bedroom2" feature. The numerical values in both datasets are z-scored, and categorical features are encoded as numbers. Missing values are encoded as zeros, so that the complete dataset can be used for our experiment.

Feature to target. We partition the dataset into train and test and train a Random Forest Classification model on the UCI Banking set, obtaining a final test accuracy of 72 %. Moreover, we train a decision tree regression model to predict the price given the set of house features and obtain a final MSE of 1.21 on the test dataset.

Compute Shapley values. We use the trained models to obtain Shapley values for each instance in the feature dataset $\mathbf{X}$ using the SHAP kernel explainer object [9].

Feature to Shapley: procedure. Finally, we can train a neural network using the datasets $\mathbf{X}, \mathbf{\Phi}$. To investigate the sample complexity required to learn a representative mapping from features to Shapley values, we conduct the following experiment. First, we create an array of $T = 999$ linearly spaced training splits $\mathbf{t} = (0.001, 0.002, \ldots, 0.998, 0.999)$. Each element in t_i in $\mathbf{t}$ indicates the ratio of samples that are used for training the neural network, leaving the remaining $(1 - t_i)N$ samples for evaluation. For each train/test partition in $\mathbf{t}$ then, train a feed forward neural network on $t_i N$ samples through minimizing the MSE between the predicted and actual Shapley values (architecture and hyperparameters specified in Table 3) for 100 epochs, and evaluate on the $1 - t_i N$ samples. Figure 7 depicts the resulting RMSE for both datasets as a function of the training fraction.

Table 3: Hyperparameters used for XAI experiments.

Hyperparameter	
Number of layers	3
Hidden size	128
Adam learning rate	1e-4
Adam ϵ	1e-5
Dropout rate	0.1

Table 4: Number of features pre and post processing for each considered dataset.

Dataset	Number of features		Number of observations	
	Pre	Post	Pre	Post
UCI Banking [37]	15	14	11162	11162
Melbourne Housing [38]	20	20	34857	34857

E Additional Results

In-sample predictive performance. For completeness, we provide the MAEs for each solution concepts on the in sample dataset up to $n = 15$ players in Table 5.

Table 5: Predictive performance across solution concepts on in-sample test dataset.

	Least core payoffs		**Least core excess**		**Shapley values**		**Banzhaf indices**	
Metric	Mean MAE		MAE		Mean MAE		Mean MAE	
N	Fixed	Variable	Fixed	Variable	Fixed	Variable	Fixed	Variable
4	0.087	0.140	0.034	0.058	0.069	0.072	0.067	0.097
5	0.077	0.096	0.031	0.045	0.040	0.047	0.035	0.060
6	0.058	0.069	0.028	0.033	0.026	0.031	0.021	0.037
7	0.040	0.051	0.021	0.027	0.022	0.023	0.014	0.024
8	0.027	0.036	0.015	0.021	0.016	0.018	0.015	0.017
9	0.019	0.031	0.012	0.019	0.010	0.015	0.012	0.014
10	0.014	0.023	0.007	0.017	0.009	0.013	0.006	0.013
11	0.010	0.021	0.006	0.020	0.006	0.012	0.009	0.013
12	0.009	0.018	0.005	0.023	0.005	0.011	0.004	0.013
13	0.006	0.016	0.003	0.027	0.009	0.010	0.004	0.014
14	0.005	0.014	0.003	0.032	0.004	0.009	0.009	0.014
15	0.005	0.013	0.003	0.036	0.004	0.009	0.012	0.014
Avg	0.019	0.015	0.029	0.051	0.018	0.022	0.017	0.027

Capturing step-wise jumps: full description. We provide a more detailed description to Figure 6. In the Figure on the **left**, each player is a winning coalition by itself when $q = 1$. Up to $q = 3$, player 3 is more critical than players 1 and 2 and receives a larger share of the joint payoffs. All players obtain equal payoffs when the only winning coalition is the grand coalition. On the **right** side: each player is a winning coalition by itself when $q = 1$, which results in an equal payoff of 0.2 for each player. Increasing the quota continuously alters the set of winning coalitions, thereby the relative importance of each player. At $q = 3$ players 1, 2 and 3 have become less critical and obtain a payoff of 0.18 each compared to 0.22 for player 4 and 5. Finally, when $q = \sum_{i=1}^{N} w_i = 7$, the only winning coalition that can be formed is the grand coalition: each player is required and the joint payoff is divided equally.

F Hardware details

We trained our neural networks on an internal compute cluster with 10 Quadro RTX 6000 GPUs and two Intel(R) Xeon(R) Gold 5218 CPUs, each of which has 16 physical cores (32 logical processors).

G Asset licensing

Two main assets were used in this work:

- The UCI Banking dataset
- The Melbourne Housing dataset

The Melbourne Housing dataset was released under the license CC BY-NC-SA 4.0. The UCI Banking dataset is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) license.