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 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. An important application of our framework is Explainable AI: our approach can be used to speed-up Shapley value computations on many instances.

1 Introduction

The ability of individuals to form teams and collaborate is crucial to their performance in many environments. The success of humans as a species hinges on our capability to cooperate at scale [27]. Similarly, cooperation between learning agents is necessary to achieve high performance in many environments [32, 7, 51, 28] and is a fundamental problem in artificial intelligence [47, 13]. Individual agents are often 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 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 [15, 33, 6, 49]; one of the cornerstones of explainable AI [2]. Another example is sharing the costs of data acquisition or a joint venture in a fair way between participants [8, 1], or sharing gains between cooperating agents [24, 46]. 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. Here, would like to quantify the true political power held by each participant, or allocate a common budget between them [36, 10].

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. The field provides solution concepts that measure the relative impact of individuals on team performance, or the individual rewards agents are entitled to. *Power indices* such as the Shapley value [45] or Banzhaf index [9] attempt to divide the joint reward in a way that is *fair*, and have recently been used to compute feature importance [33]. In contrast, other solutions such as the Core [26] 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. Computing them is typically a hard problem, even in restricted environments [21, 12, 17].

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 [26, 37, 17]). Figure 1 depicts the two well-studied classes of games on which we evaluate our approach: weighted voting games [36, 10, 12] and feature importance games in explainable AI [15, 33].

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). 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 of features on the outcome of a machine learning model [15, 33]. Solving these games for the Shapley value (or other game theoretic measure) provides a way to reverse-engineer 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 [2]. One drawback of this approach is the computational complexity of calculating the Shapley value, making such analysis slow even when using approximation algorithms. Our approach provides a way to significantly speedup Explainable AI analyses, particularly for datasets with a large number of instances.

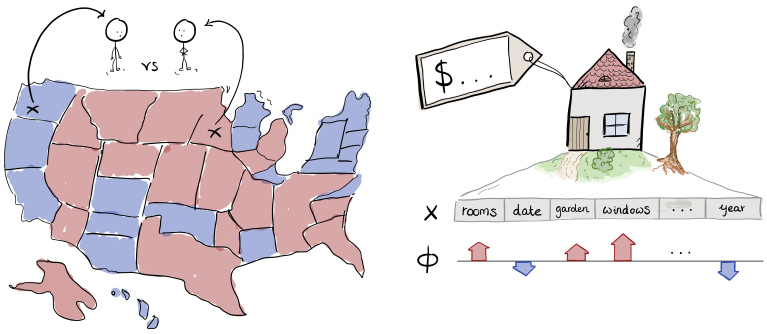


Figure 1: **Evaluation domains for our approach.** *Left*: Weighted voting games (WVGs) model decision making bodies such as the US Electoral College [36, 10]. Our approach predicts solution concepts which reflect the true political influence of players or stable payoff allocations. *Right*: Applying 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 quickly without a full computation of Shapley values.

2 Preliminaries

We provide a brief overview of cooperative game theory (examined in details in various books [42, 12]) and discuss how solution concepts in cooperative game theory have been applied in Explainable AI [15, 33, 34, 14, 53].

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 [12], 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 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 [36, 10].

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 [26] 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 [37, 17]. 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 [45] 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 natural fairness axioms [19], and has thus found many applications from estimating feature importance [15, 33] to pruning neural networks [52, 23, 25]. 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

[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 $\sum_{i=1}^{n} p_i = 1$.

$\phi_i(v)$ 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 [9] 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 [48]. 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 Speeding Up Explainable AI In Large Datasets

An important goal of our work is to speedup Shapley/Core computations of many data instances. In machine learning, we train a model $f_{\boldsymbol{\theta}}$ to learn a mapping from some set of input features to an outcome. This could we can train a model to predict the price of a house based on a number of features, such as the number of rooms, the year it was built, and so on (Figure 1). In such settings, it is desirable, but challenging, to explain the model outputs in terms of the input features. Explainable AI addresses this issue, and recent years showed several applications of game theoretic metrics for measuring feature importance in the machine learning community [49, 2].

The fastest method to approximate Shapley values (also used in the SHAP package) is a Monte-Carlo approach [33]. A number of other methods exist whose runtime and accuracy depend on the number of samples used, usually on the order of several thousands [30, 15, 35, 5]. In a model setting, the characteristic function takes the value of the trained model output for a given instance: $f_{\boldsymbol{\theta}^*}(\mathbf{x})$. The Shapley value of a feature i in a data instance x, $\phi_{x,i}$ is the effect that feature has on the model outcome. Sampling based-methods compute the contributions with respect to a base value, which is the average model output across all instances: $f_{\boldsymbol{\theta}^*}(x) = \mathbb{E}[f_{\boldsymbol{\theta}^*}(x)] + \sum i = 1^N \phi_{x,i}$. Sampling based methods are not ideal for large datasets because they require a large number of re-evaluation samples per computation. We show how our approach can be employed to speedup Shapley or Core computations of many instances by training models to learn representations of feature attribution schemes.

3 Methods

Our approach uses machine learning to create game-theoretic estimators. We generate synthetic datasets of games to train our models, spending compute up-front to allow for instant solutions afterwards. Our first domain concerns weighted voting games (WVGs) as they provide a generic framework for studying cooperation in multi-agent settings.

Afterwards, we apply our approach to the Shapley-based feature importance setting. The main idea is that 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 models to approximate Shapley values of features, and examine 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 for 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 E.2).

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 pad the inputs with zeros for games with less than M players. 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 (LCV) is not shuffled but stored at the last element of each row (Figure E.1).

We produce a single 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.

3.1.3 Training and Evaluation

During training we minimize the Mean Square Error (MSE) between the true and predicted solutions. 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 (MAE), 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. We sample weights $\mathbf{w} \sim \text{Beta}(\alpha, \beta)$ with varying parameters for α and β to assess our models' ability to generalize to previously unobserved instances their ability generalize to games far outside of the training distribution (full details in Table 2, Figure D.1).

3.2 Feature Importance Games

We perform an experiment to show that neural networks can provide a faster alternative for measuring feature importance at scale. We select a dataset (details in Appendix D), train a model, and use those to construct a dataset from features to Shapley values using the SHAP KernelExplainer [33] Following, we partition our dataset into a train and test, and incrementally change the proportions between the two. For each increment, we train a model for 100 epochs and test it on the remainder of the unseen instances.

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 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 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 perturbation of the Least-Core solution [20, 55].

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	0.030	0.043	0.015	0.034	0.019	0.022	0.018	0.028
Out-of-sample	0.030	0.044	0.015	0.027	0.018	0.036	0.018	0.056
Slightly out-of-distribution	0.029	0.028	0.015	0.050	0.018	0.019	0.017	0.018
Moderately out-of-distribution	0.030	0.035	0.014	0.029	0.018	0.026	0.018	0.032
Significantly out-of-distribution	0.031	0.045	0.014	0.036	0.018	0.029	0.018	0.039

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

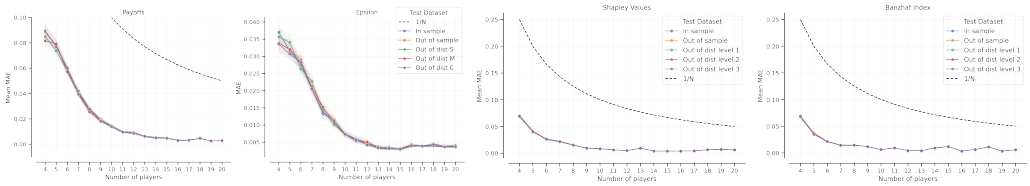
Fixed-size models display stable performance across solution concepts. Our first observation is that the fixed-size neural networks are adept at estimating solutions across the three considered concepts. The average error per player ranges from 0.005 to 0.087 (Least Core), 0.004 to 0.069 (Shapley values) 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 2, 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.

Variable-size models are robust to shifts in the weight distribution. Figure 3 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.

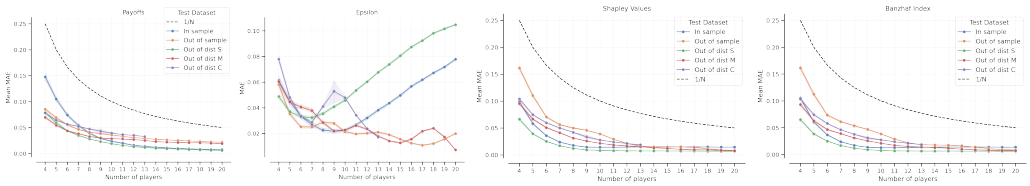
Variable-size models generalize to a larger number of players. Our main objective is to investigate how to leverage machine learning to perform scalable value estimation in large multi-agent settings. To test this, we train the variable-size models on games up $N = 10$ players, and evaluate on $N + 1, N + 2, \ldots, N + 10$ players (full details in section 3.1.2). Figure 3 demonstrates that

[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)



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

Figure 2: **Performance fixed-size models.** We evaluate the performance on five test sets with 1000 samples per N player each. 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.

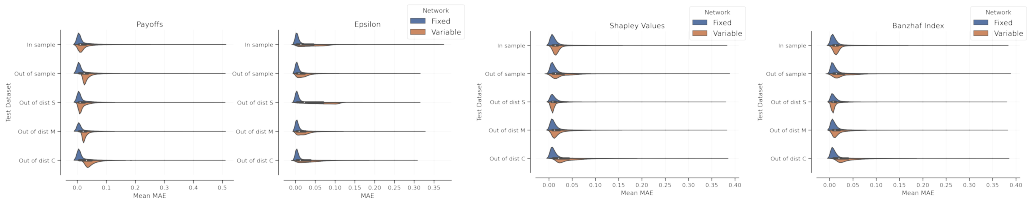


(a) Least Core (b) Shapley value (c) Banzhaf index

Figure 3: **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.

there is no significant decrease in performance for games with more than ten 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 4. 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 4: **Comparing overall performance fixed-size vs. variable-size networks.**

4.1.2 Discontinuities in the Solution Space

The function mapping from game parameters to solutions contains discontinuities. Discontinuous jumps emerge from the players' interdependence and the effect of the quota and are difficult for a model to learn. We analyze two examples that demonstrate how our models respond to such transitions.

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. Thus, 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.

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 5 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).

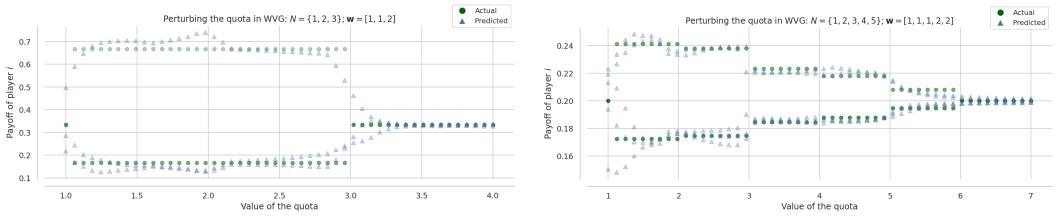


Figure 5: **Capturing step-wise jumps.** Individual actual payoffs (green dots) together with the model predictions (blue triangles) as the value of quota increases by $\varepsilon = 0.1$ increments (full description in Appendix F).

4.2 Speeding Up Feature Importance Calculations By Predicting Shapley Values

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 [39] (17 features, 11,162 observations) and the Melbourne Housing dataset [44] (21 feature, 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 F.1 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 of the Melbourne dataset has an RMSE of 19.72 on the resultant test set. Increasing the number of training samples to 3 percent results in a RMSE of less than 0.07, 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. 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).

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 importance games for explainable AI. Our analysis shows that neural models capture cooperative game solutions well, and can generalize well outside the training distribution, even extrapolating to more players than in the training set.

We saw that the Least core is a harder solution concept 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 [17, 3, 4] 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 [12]? 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] A. Agarwal, M. Dahleh, and T. Sarkar. A marketplace for data: An algorithmic solution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 701–726, 2019.

[2] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information fusion*, 58: 82–115, 2020.

[3] H. Aziz, O. Lachish, M. Paterson, and R. Savani. Power indices in spanning connectivity games. In *International Conference on Algorithmic Applications in Management*, pages 55–67. Springer, 2009.

[4] Y. Bachrach, J. S. Rosenschein, and E. 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.

[5] Y. Bachrach, E. Markakis, E. Resnick, A. D. Procaccia, J. S. Rosenschein, and A. Saberi. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems*, 20(2):105–122, 2010.

[6] Y. Bachrach, T. Graepel, G. Kasneci, M. Kosinski, and J. 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.

[7] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[8] E. Balkanski, U. Syed, and S. Vassilvitskii. Statistical cost sharing. *Advances in Neural Information Processing Systems*, 30, 2017.

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

[10] J. M. Bilbao, J. R. Fernandez, N. Jiménez, and J. J. Lopez. Voting power in the european union enlargement. *European Journal of Operational Research*, 143(1):181–196, 2002.

[11] F. Bistaffa, A. Farinelli, G. Chalkiadakis, and S. D. Ramchurn. A cooperative game-theoretic approach to the social ridesharing problem. *Artificial Intelligence*, 246:86–117, 2017.

[12] G. Chalkiadakis, E. Elkind, and M. Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

[13] A. Dafoe, Y. Bachrach, G. Hadfield, E. Horvitz, K. Larson, and T. Graepel. Cooperative ai: machines must learn to find common ground, 2021.

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

[15] A. Datta, S. Sen, and Y. 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.

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

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

[18] A. Dinar, A. Ratner, and D. Yaron. Evaluating cooperative game theory in water resources. *Theory and decision*, 32(1):1–20, 1992.

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

[20] E. Elkind and D. 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.

[21] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *AAAI*, pages 718–723, 2007.

[22] M. G. Fiestras-Janeiro, I. García-Jurado, A. Meca, and M. A. Mosquera. Cooperative game theory and inventory management. *European Journal of Operational Research*, 210(3):459–466, 2011.

[23] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

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

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

[26] D. B. Gillies. *Some theorems on n-person games*. Princeton University, 1953.

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

[28] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364(6443):859–865, 2019.

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

[30] D. Leech. Computation of power indices. 2002.

[31] D. Leech. Designing the voting system for the council of the european union. *Public Choice*, 113(3):437–464, 2002.

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

[33] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[34] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal, and S.-I. Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67.

[35] S. Maleki, L. Tran-Thanh, G. Hines, T. Rahwan, and A. Rogers. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

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

[37] M. Maschler, B. Peleg, and L. S. Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of operations research*, 4(4):303–338, 1979.

[38] F. Mirzaei-Nodoushan, O. Bozorg-Haddad, and H. A. Loáiciga. Evaluation of cooperative and non-cooperative game theoretic approaches for water allocation of transboundary rivers. *Scientific Reports*, 12(1):1–11, 2022.

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

[40] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[41] N. Ohta, V. Conitzer, Y. Satoh, A. Iwasaki, and M. 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] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1994.

[43] R. Patel, M. Garnelo, I. Gemp, C. Dyer, and Y. Bachrach. Game-theoretic vocabulary selection via the shapley value and banzhaf index. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2789–2798, 2021.

[44] T. Pino. Melbourne housing market data, 2018.

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

[46] R. H. L. Sim, Y. Zhang, M. C. Chan, and B. K. H. Low. Collaborative machine learning with incentive-aware model rewards. In *International Conference on Machine Learning*, pages 8927–8936. PMLR, 2020.

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

[48] P. 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.

[49] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[50] M. Vázquez-Brage, A. van den Nouweland, and I. Garcıa-Jurado. Owen's coalitional value and aircraft landing fees. *Mathematical Social Sciences*, 34(3):273–286, 1997.

[51] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

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

[53] T. Yan and A. 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.

[54] H. P. Young. Cost allocation. *Handbook of game theory with economic applications*, 2:1193–1235, 1994.

[55] M. Zuckerman, P. Faliszewski, Y. Bachrach, and E. Elkind. Manipulating the quota in weighted voting games. *Artificial Intelligence*, 180:1–19, 2012.

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 G.
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 H.
   (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 Shapley values for measuring the true influence of participants, in weighted voting games or political power in decision making bodies.

Consider 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 [16] 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 C.1a. In the LP-problem (C.1a), all winning

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

(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 C.1: Algorithms for finding least core solutions.

coalitions are considered, Inside those winning coalitions there exists a subset, that could branch out and form 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

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

and solve the more efficient LP-problem C.1b.

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 [29] optimizer, ReLU [40] 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 [29] optimizer, ReLU [40] 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 E.1 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 D.2 shows the distributions per player together with the true label (red dot) for 8 and 9 player games.

Parameterizations for test datasets. Figure D.1 shows the five standardized test distributions that are generated with the parameters in Table 2.

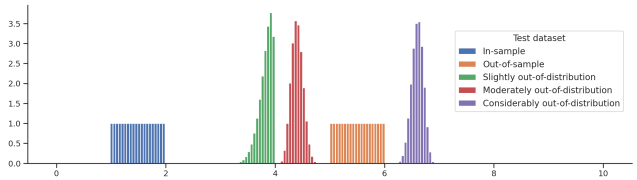


Figure D.1: **Test data distributions in perspective.** To assess how robust models' are to shifts in the weight distribution, we re-parameterize the weight distribution in five different ways.

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

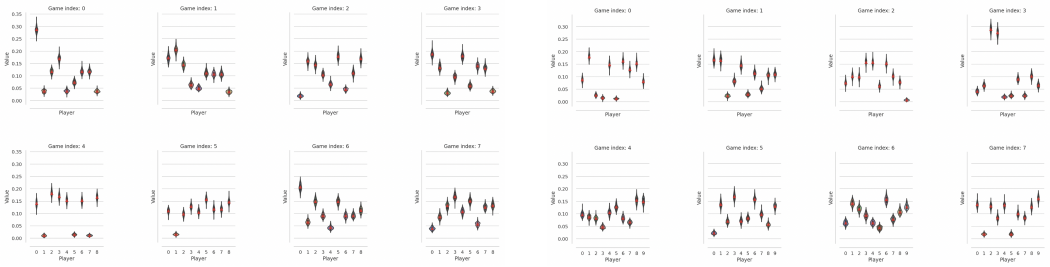


Figure D.2: **Approximating ground-truth Shapley values.** Ten randomly sampled games with distribution for each player in the game, the red dot indicates the true Shapley value (i.e. when computing the marginal value for each permutation).

D.2 Explainable AI

We provide a step-wise explanation of how we performed the experiment in our explainable AI section, that is, train a neural network to predict the Shapley values for a dataset of features:

1. **Select a dataset.** The first step is to select a dataset of choice. 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 $y \in [0, 1]$ 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 through sampling.** We use the trained model from step 2 to obtain Shapley values with the SHAP package KernelExplainer[33], which is based on LIME [33]. This yiels a matrix of Shapley values, one for each feature in every data instance: $\mathbf{\Phi} \in \mathbb{R}^{N \times F}$.

4. **Feature to Shapley.** Finally, 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 $1 - t_i N$ remainder (unseen) samples.

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 [33].

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 F.1 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 [39]	15	14	11162	11162
Melbourne Housing [44]	20	20	34857	34857

E Additional Methods

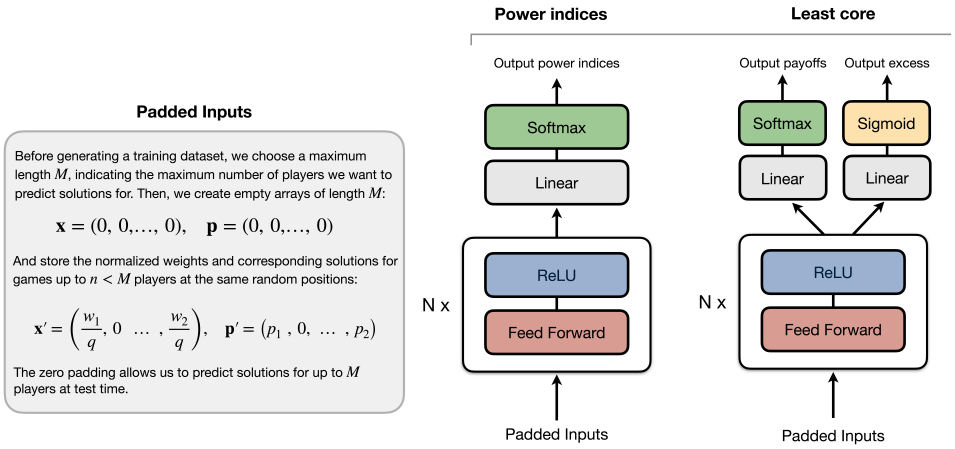


Figure E.1: **Neural architectures for the variable-size game predictions.** We add zero-padding to allow one model to predict solutions of games of variable sizes. We choose a maximum length $M = 20$, and add zero-padding to games with less than M players. Full details on our data in Section 3.1.2

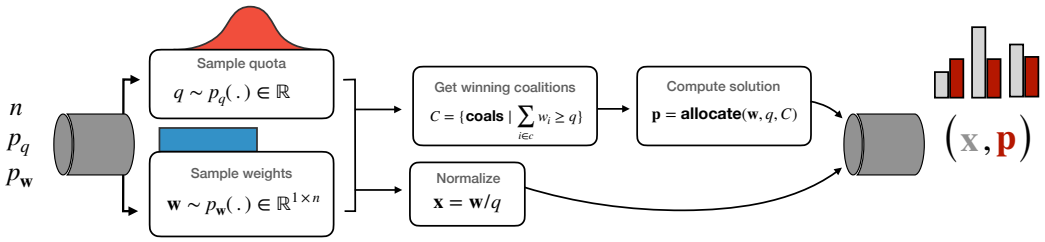


Figure E.2: **Procedurally generating and solving one weighted voting game.** We sample the weights and quota to create a weighted voting game. The model inputs are divided by the quota and the payoffs are computed by solving the game, provided a solution concept. One game instance consists of a set of normalized weights $\mathbf{x}$ and the allocated payoffs $\mathbf{p}$.

F 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 5. 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.

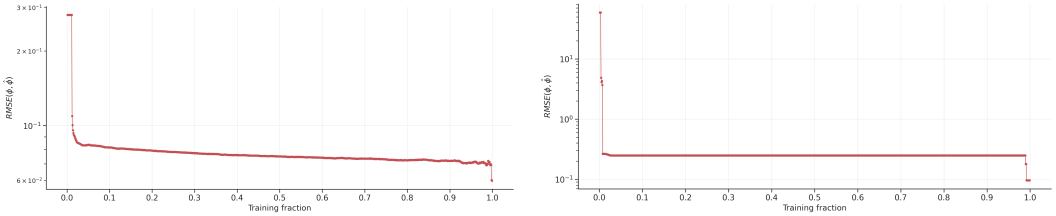


Figure F.1: **Root Mean Squared Error as a function of training fraction.** The RMSE between the actual and predicted Shapley values scales approximately as a power law with the training fraction. *Left*: UCI Banking dataset *Right*: Melbourne Housing dataset.

G 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).

H 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.

I Evaluating the Speedup Achieved by Our Approach, Depending on the Size of the Dataset

Many domains involve datasets with a large number of features, sometimes with over thousands or millions of data instances. A prominent example is language modelling, where large bodies of text are generated, translated or processed in different manners [43]. In such cases, computing the Shapley value for each data instance using sampling-based methods takes an extremely long time.

Our approach is targeted at speeding up feature importance computations of many instances. We perform a simple experiment to demonstrate the effectiveness of a model-based approach for explaining model decisions. We take the Melbourne Housing dataset and obtain 9000 instances by 13 features after preprocessing steps (encoding the categorical features and standardization). First, we generate a ground-truth dataset by setting the number of samples in the SHAP package (KernelExplainer) to 5000 samples. The number of samples in the KernelExplainer determines the number of times a model is re-evaluated for every prediction. A higher number of samples will result in more accurate (and lower variance in the) Shapley values, but require a longer compute time. Computing the Shapley values on our 9000 samples takes 4.13 hours. Figure I.1 (left) shows that the inference time scales approximately linearly with the number of re-evaluation samples.

To contrast the difference in computation time, we compute Shapley values on 10 % of the data (computation time is 29 minutes) train the neural network in 30 seconds, and predict Shapley values on the remaining 90% of the data (less than a second). All together, this procedure takes roughly 12

% of the time of it took to compute Shapley values for the entire dataset. This speeds-up the whole procedure by 8x while keeping a reasonable prediction quality on the remaining unseen data (Figure I.1 right).

Next, we use SHAP again to compute Shapley values for 90% of the data, 8100 instances, this time with the default number of model re-evaluations samples, yielding slightly accurate Shapley values (the default is $2 \times$ number of features $+ 2048$). We quantify the trade-off in prediction quality as the by comparing the model predicted Shapley values to the Shapley values obtained with the default number of samples.

Concretely, we measure the Mean Mean Absolute Error (MMSE) of our model predictions and the SHAP package to the ground-truth dataset. We note that speedup does come at a slightly reduced empirical performance: the model MSE on the test set is, MSE $= 1.15 \times 10^{-3}$, compared to the MSE with SHAP: MSE $= 7.4 \times 10^{-5}$. However, we expect that hyperparameter tuning and better preprocessing will improve the quality of the model predictions.

All together, it is easy to see how this win in computation time translates to larger datasets. Consider a similar dataset with 100,000 or a million entries. Computing Shapley values on the entire dataset with SHAP will take weeks (40 or 400 hours), while the time for our procedure remains relatively the same: building the train set and training the neural network only takes a fraction of the time required by the sampling-based approach: the analysis using our method would take a few hours.

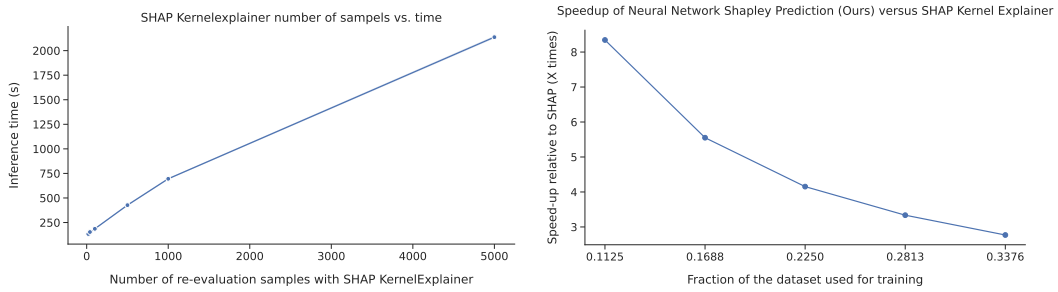


Figure I.1: **Speeding up Shapley value computations.** *Left*: The inference time, here the time it takes to compute the shapley values of 9000 instances, scales approximately linearly with the number of samples. *Right*: The x times speedup provided by our model-based approach contrasted by the fraction of data used for training.

J Heuristics and Simpler Models

We now consider a common heuristic for payoff allocation in weighted voting games, the *weight proportional* payoff allocation. We use this as a benchmark to evaluate our models' performance against. We find that our fixed models outperform this benchmark heuristic across solution concepts and test sets.

WeightProportional baseline. The WeightProportional (WP) heuristic divides the payoffs in a way that is directly proportional to the players' weights, that is, the payoff of a player j is:

$$\mathbf{p}_j = \frac{w_j}{\sum_{i=1}^{n} w_i} \tag{J.1}$$

for any $n-$player game. WP provides meaningful benchmark as it is the most naive method for allocating payoffs. The resulting payoffs (and thus errors) do not take into account the quota, which may have a major impact on the correct solutions by means of the emergent set of winning coalitions. As such, any improvement on this benchmark thus suggests a better understanding of the collaborative structures in the game at hand.

We also compare the predictive performance of our models against the predictive performance of multinomial regression models. These models have just one layer with n inputs and n outputs for each $n-$player game and a softmax nonlinearity at the end. Across all game sizes, our models outperform the linear models by 45 % to 95 % percent. Results are found in Table 6.

Table 6: Comparing neural network performance to multinomial logistic regression models.

	Linear Core	Neural net Core	**Linear Shapley**	Neural net Shapley	**Linear Banzhaf**	Neural net Banzhaf
N						
4	0.212	0.089	0.137	0.07	0.116	0.069
5	0.147	0.078	0.098	0.041	0.083	0.035
6	0.124	0.059	0.080	0.026	0.071	0.021
7	0.108	0.041	0.070	0.022	0.063	0.014
8	0.102	0.028	0.065	0.015	0.059	0.014
9	0.089	0.020	0.056	0.009	0.051	0.012
10	0.078	0.014	0.050	0.008	0.045	0.006
11	0.072	0.010	0.045	0.006	0.042	0.009
12	0.065	0.009	0.041	0.005	0.041	0.004
13	0.062	0.006	0.036	0.009	0.035	0.004
14	0.056	0.005	0.036	0.004	0.034	0.009
15	0.057	0.005	0.032	0.004	0.032	0.0012
16	0.051	0.003	0.029	0.004	0.030	0.003
17	0.051	0.003	0.028	0.004	0.029	0.006
18	0.047	0.005	0.028	0.007	0.027	0.011
19	0.045	0.002	0.026	0.007	0.025	0.003
20	0.044	0.003	0.025	0.006	0.024	0.006

K Hard cases

Across solution concepts (Shapley, Banzhaf and Least core), discontinuous jumps in the solution space yield large errors (see Section 4.1.2 and Figures 5, L.4). In this section, we highlight failure cases for the Least core. In contrast to the power indices, a solution in the least core is obtained by solving a Linear Program under several hard constraints. Thus, a solution to a weighted voting game is only feasible when, for each winning coalition $C \in C^{\text{win}}$, we have that

$$v(C) = \sum_{i=1}^{C} p_i \geq 1 - \varepsilon \tag{K.1}$$

where p_i is the payoff of player i in coalition C. In words, each winning coalition must obtain joint reward that is at least as good as what they would have obtained on their own. If this criteria is not met, the solution is infeasible: some players will have an incentive to deviate from the grand coalition.

Majority of the model predictions fail to meet hard constraints. Figure K.1 depicts the percentage of feasible solutions over the game sizes. For small player games ($n < 8$), between 50 and 5.5 % of the model predictions are feasible). The larger the number of players, the fewer model predictions satisfy the hard constraints. From eight player games and above, the number of feasible predictions ranges between 0.2 and 10 %. A preliminary analysis shows no relationship between infeasible solutions and larger errors in the payoffs nor the LCV.

The fact that the model is not able to predict many feasible solutions in these large games is unsurprising. The set of winning coalitions increases approximately exponentially in n. As such, the more players in a game, the more hard constraints need to be satisfied in order for a solution to be feasible per game. All hard constraints need to be met in order for a solution to be feasible. We expect that including a penalty term for infeasible predictions during training will help to improve this metric and hope to address this in future work.

When the least core is a set, simplex based methods tend to find corner solutions, which leads to ambiguity in the solutions. Further analysis of individual weighted voting games reveals that the most problematic games are cases where the least core contains multiple correct solutions, but the full joint payoff is allocated to a single player. In other words, payoffs are divided such that one player receives one, and all the others zero. To illustrate what is happening here, consider a simple four player game that produced the largest error (Figure K.2). We focus our analysis on the game on the left, but our insights apply to a wide range of n-player games in the dataset. This particular

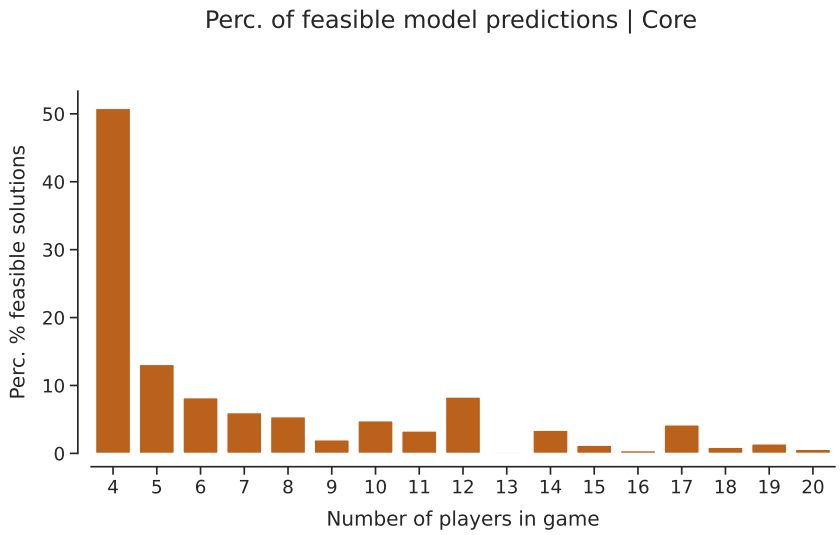


Figure K.1: **Percentage of feasible solutions for each $n-$player game**. We count the number of solutions that meet the hard constraints by validating equation K.1 for each winning coalition C in the set of winning coalitions across 1000 samples per $n-$player game.

game has the following weights and quota (rounded to one decimal place for simplicity)

$$\mathbf{w} = (2.8, 1.6, 6.6, 1.5); \quad q = 12.1 \tag{K.2}$$

where we can see that no player is a winning coalition by itself (all w_i's ¡ q). In fact, the only winning coalition in this game is the grand coalition, that is,

$$C^{\text{min}} = \{1, 2, 3, 4\} \tag{K.3}$$

The solution to this weighted voting game is:

$$\mathbf{p} = (1, 0, 0, 0) \quad \text{with} \quad \text{LCV} = 0 \tag{K.4}$$

where player one receives the full joint payoff, even while its' weight is smaller than the weight of player three ($2.8 < 6.6$). The reason why we obtain this solution comes down to two facts. First of all, the least core is a set, and can therefore contain multiple correct solutions. Secondly, simplex based methods return corner solutions. Thus, in cases where several players are required to form a winning coalition, the solver will allocate the joint payoff to an arbitrary player that belongs to this critical subset. Figure K.3 depicts this visually for a simple two player game. Here, we have the WVG $\mathbf{w} = (1, 1)$ with $q = 2$, so the only winning coalition is the grand coalition. Solving this game results in either of the corner solutions: the payoff vector $\mathbf{p} = (1, 0)$ or $\mathbf{p} = (0, 1)$ with a LCV $= 0$. However, if we define α as the solution of a player one, any combination $\mathbf{p} = (\alpha, 1 - \alpha)$ is a feasible solution in the least core.

Obtaining the full set of solutions for this game can be done empirically. In principle, we can take the payoff vector and LCV above, generate the set of permutations of length two (player pairs) and check whether the solution is still feasible if we give a small value δ from one player (whose payoff is not zero) to any other player. However, this approach does not scale to larger games. It would be more efficient to derive rules for acquiring the full least core set from a single solution.

We limited our analysis above to four-player games. However, we observe the above behavior irrespective of the number of players in the game. Therefore, addressing this issue is critical for improving the predictive performance of our models.

L Applying Our Models to Voting In The European Council

In this section we apply our trained models to predict voting power of member states in the EU Council. First, we take the weights from four countries in the EU voting council [31], see Table 7, and define the quota as the majority vote (50 % of the total weights + 1). Our (weighted) voting

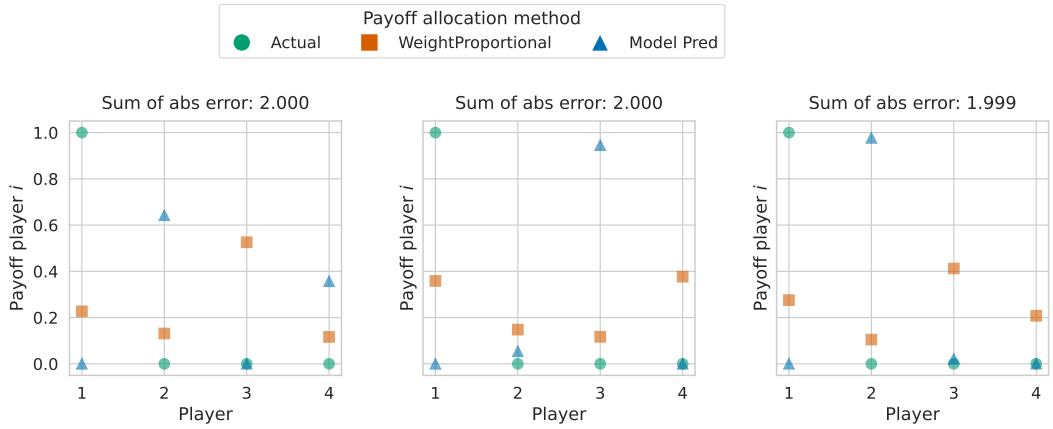


Figure K.2: **Examples of games where the joint payoff is allocated to a single player**. We depict the actual solutions (green dots), payoffs allocated according to the weight proportional heuristic (see eq. J, orange squares) and model predictions (blue triangles).

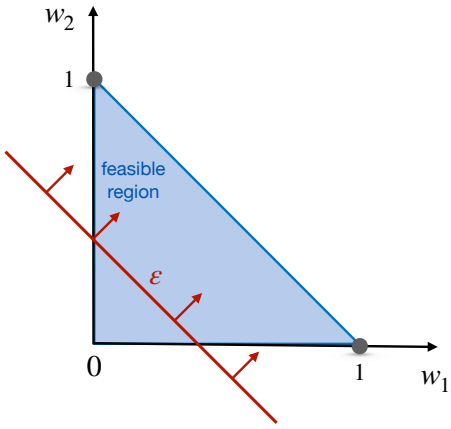


Figure K.3: **Visual depiction of the corner solutions and feasible region of a two-player weighted voting game**. Take the weighted voting game $\mathbf{w} = (1, 1)$ with $q = 2$. Simplex based methods tend to return corner solutions, in this case $\mathbf{p} = (1, 0)$ or $\mathbf{p} = (0, 1)$ with a LCV $= 0$.

game consists of the member states Hungary, Netherlands, Poland and Ireland with the weights $\mathbf{w} = (12, 13, 27, 7)$, and quota $q = 30,5$.

Following, we divide the weights by the quota and feed the normalized weights (see Section 3.1). We contrast the model predictions against the ground-truth solutions and find that the Mean Mean Absolute Error (MMAE) between our model predictions and actual solutions per member state is 0.003 (Banzhaf), 0.002 (Shapley), 0.017 and 0.007 for the Least core payoffs and excess respectively. Figure L.1 compared the payoff allocations for the Shapley, Banzhaf and Core solution concepts. We observe that the contribution measured by the Power indices (Shapley and Banzhaf) is less than the weight, while the Core attributes a larger share of the joint payoff to this member state.

Next, we perform a sensitivity analysis. In such an analysis, we are interested in the interdependence of member states' voting power. Holding all other game parameters constant, we increment the weight of Hungary ($w_{\text{Hungary}} = 7$) with values of one, until it exceeds the quota of 30,5. Figure L.2 depicts how the member states' voting power (in Shapley value) changes as Hungary gains absolute power (as we increase the weight). The predicted payoff for Hungary is depicted by the red triangles, the model predictions for the other states in blue, and the actual solutions by the green dots. We observe that there are two transition points in the solution space. One transition point occurs when $w_{\text{Hungary}} = 17.5/18.5$. Hungary can now form a winning coalition both with the Netherlands ($w_{\text{Netherlands}} = 12$) and with Poland ($w_{\text{Poland}} = 13$), which raises its power in the voting

Table 7: Comparison of voting weights in the Council of the European Union [31].

Member state	Population	Population Perc.	Weights	Weights Perc.
Germany	82.54m	16.5%	29	8.4%
France	59.64m	12.9%	29	8.4%
UK	59.33m	12.4%	29	8.4%
Italy	57.32m	12.0%	29	8.4%
Spain	41.55m	9.0%	27	7.8%
Poland	38.22m	7.6%	27	7.8%
Romania	21.77m	4.3%	14	4.1%
Netherlands	17,02m	3.3%	13	3.8%
Greece	11.01m	2.2%	12	3.5%
Portugal	10.41m	2.1%	12	3.5%
Belgium	10.36m	2.1%	12	3.5%
Czech Rep.	10.20m	2.1%	12	3.5%
Hungary	10.14m	2.0%	12	3.5%
Sweden	8.94m	1.9%	10	2.9%
Austria	8.08m	1.7%	10	2.9%
Bulgaria	7.85m	1.5%	10	2.9%
Denmark	5.38m	1.1%	7	2.0%
Slovakia	5.38m	1.1%	7	2.0%
Finland	5.21m	1.1%	7	2.0%
Ireland	3.96m	0.9%	7	2.0%

and accordingly its' share of the joint payoff. We notice that our model is able to predict accurate payoffs overall, with a spike in absolute error per player at this transition point.

The second transition point occurs when $w_{\text{Hungary}} = 30.5 = q$. Hungary is now also a winning coalition by itself, thereby obtaining a larger share of the joint payoff. We observe that its' Shapley value increases to approximately 0.6, obtaining the more than half of the total reward. Our model is able to capture this transition almost perfectly (MMAE < 0.01).

We perform a similar analysis, this time perturbing the quota. Figure L.3 depicts the same game with the original weights fixed, increasing the value of the quota from small to the sum of the weights (notice that this is the same experiment we performed in Section 4.1.2). The value of the quota determines the set of winning coalitions, thereby dictating the payoff of each player. As such, perturbing in this game yields a large number of discontinuous jumps in the member states solutions compared to our previous analysis perturbing the weight of Hungary. Our model is able to capture the overall changes in the solutions but has difficulty with capturing large jumps in payoffs for small quota increments.

Our variable models models have been trained on games up to ten players and can be applied to instantly predict the payoffs of member states in larger councils. To illustrate this, we take all member states in Table 7 and use our variable models to predict the payoffs for each, again setting the quota as the majority vote. We report the prediction quality of our models with respect to the ground truth (in Mean Mean Absolute Error), which is 0.013 for Shapley, 0.004 for Banzhaf, and 0.05 for the least core. Figure L.4 compares the solutions across member states and solution concepts.

While there is room for improvement in terms of prediction quality, our approach provides a useful tool for obtaining quick estimates of the voting power in medium size games ($10 \leq n < 50$).

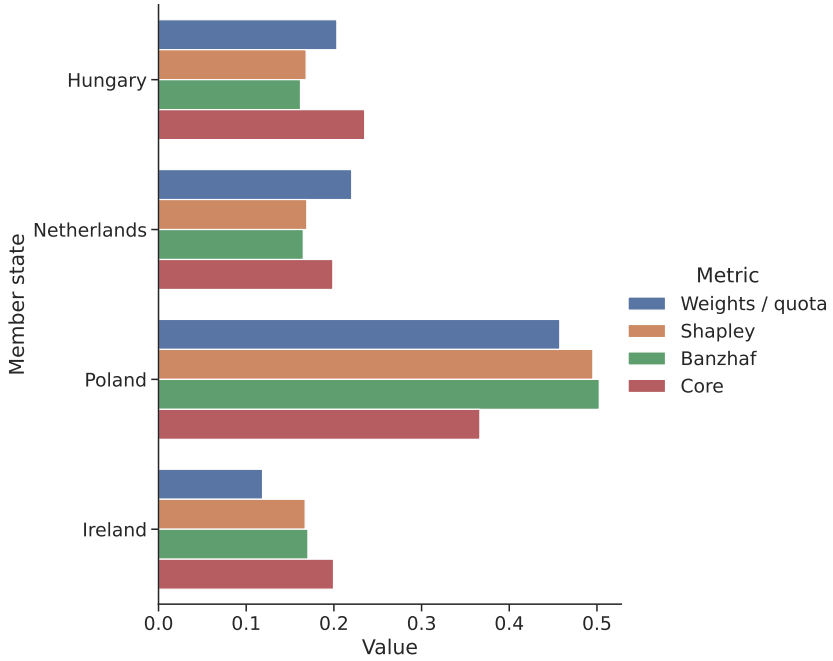


Figure L.1: **Allocating payoffs to member states in the EU Council.** A comparison of how absolute power (the normalized weights $(\mathbf{w}/q)$) relates to the allocated share of the joint payoff.

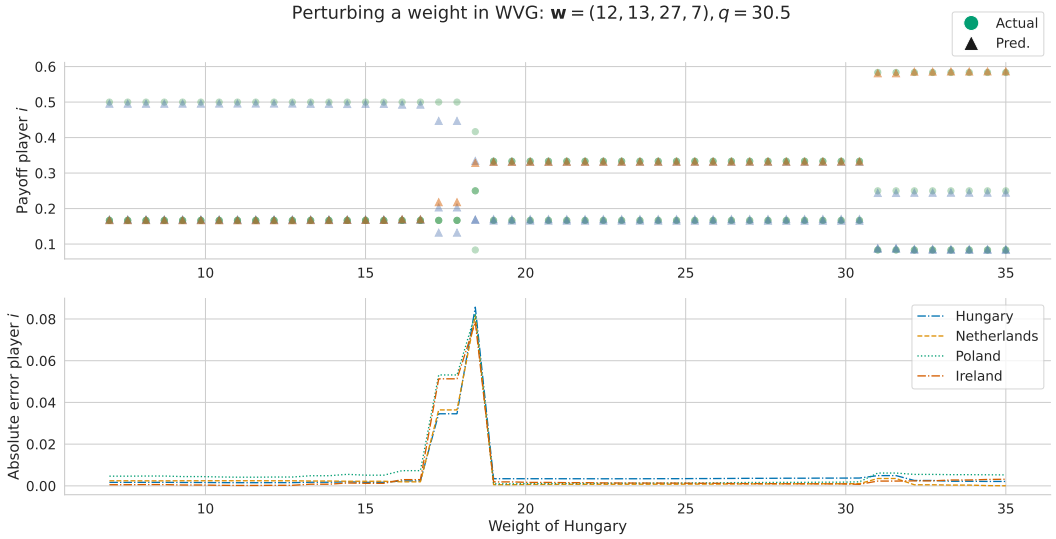


Figure L.2: **Perturbing the weight of a member state in a voting game.** *Top*: Model predictions and actual payoffs for a four-player voting game with weight vector $\mathbf{w} = (12, 13, 27, 7)$. textitBottom: Absolute errors between the individual member state payoffs.

Furthermore, while approximation methods exist for Shapley and Banzhaf, there are currently no effective methods for estimating payoffs in the least core. This solution concept is more challenging as the solutions are obtained by solving a Linear Program with a hard constraints. Thus, solving a game with 15 or even 20 players contains on the order of 2^{15} or 2^{20} constraints, which slow down the solvers dramatically. Here, we can apply the same technique, solve for the small/medium size-games, then leverage our neural network approach to tackle a higher numbers of players. In other words, provided a solver that can handle up to M players, we can train a neural network on this number of players, allowing us to extend the number of players even beyond M, possibly up to $2M$.

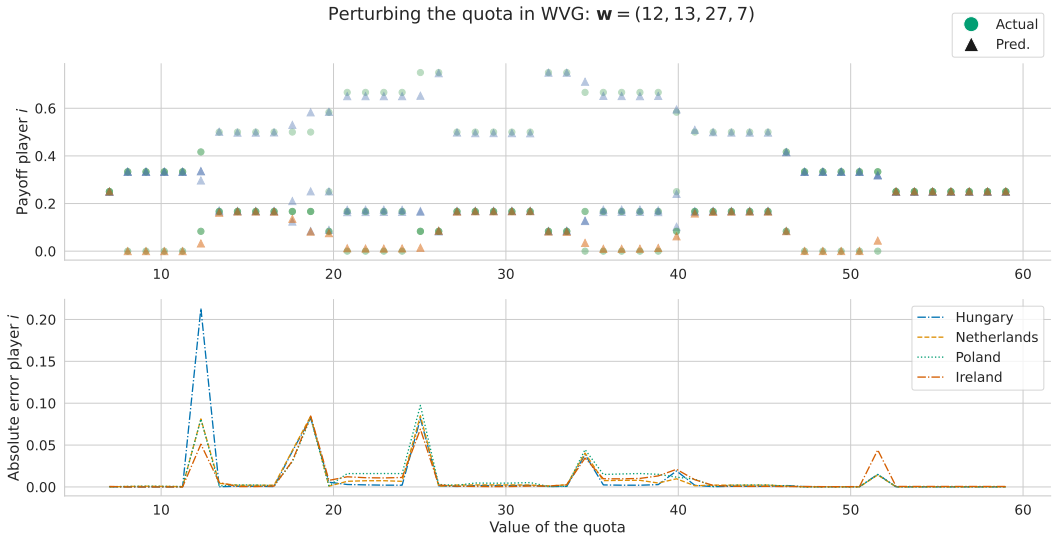


Figure L.3: **Perturbing the quota in a voting game.** *Top*: Model predictions and actual payoffs for a four-player voting game with weight vector $\mathbf{w} = (12, 13, 27, 7)$. *Bottom*: Absolute errors between the individual member state payoffs. Note that these are a large number of transition points in the solutions.

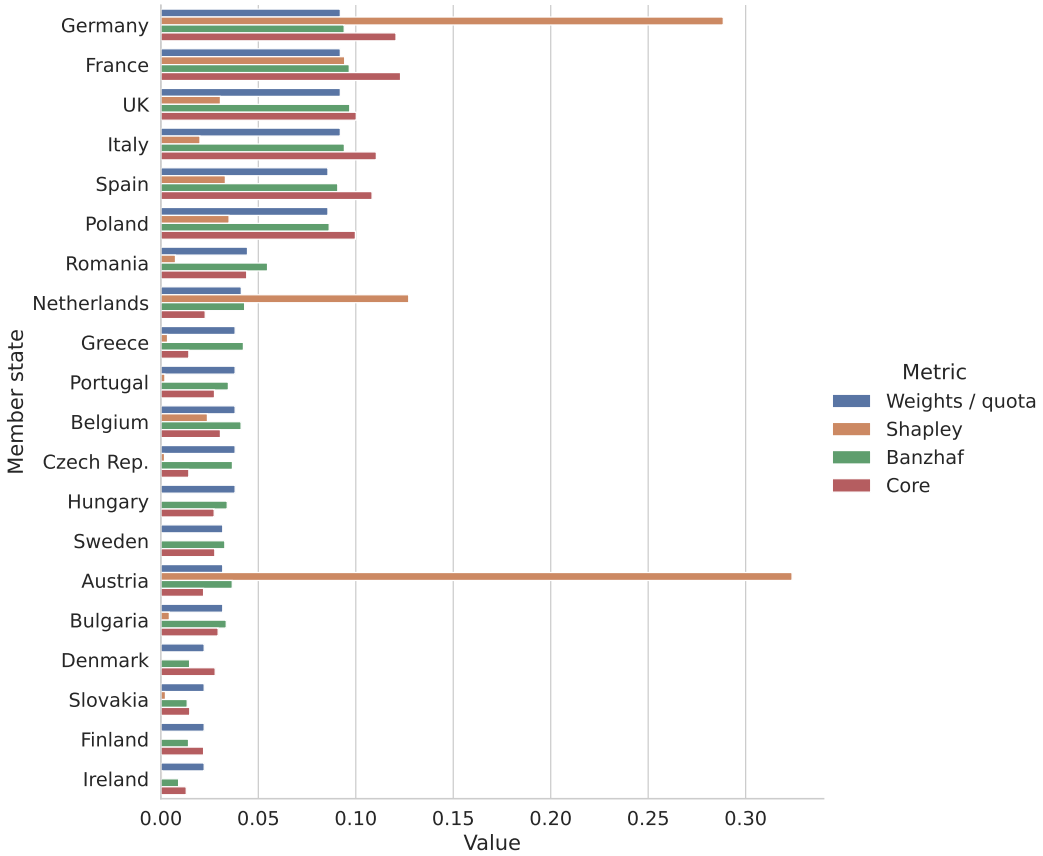


Figure L.4: **Allocating payoffs to member states in the EU council with variable models**. A comparison of the allocated payoffs over solution concepts.

M Real World Applications of Weighted Voting Games

Weighted voting games, and cooperative game theory in general, provide valuable tools for reasoning about the allocation of collective outcomes to individual agents. As such, their applicability span numerous domains. In this section, we highlight several real-world applications of our work. In all examples, our framework can be applied to both speed up and enhance the scalability. Dinar et al. [18] utilize solution concepts from cooperative game theory – including the Core, the Shapley Value, the Nucleolus and Nash – to allocate water resources to competing parties in the presence of scarcity. More recently, Mirzaei-Nodoushan et al. [38] also performed an extensive evaluation of both cooperative and non-cooperative game theoretic approach for water management in trans-boundary river basins.

Moreover, cooperative game theory has proven useful in analyzing the costs of joint activities [54]. Vázquez-Brage et al. [50] combine a model of airport games with the Shapley value to fairly distribute the aircraft landing fees among airlines. When an airport is build, the size of the runway is chosen to match the largest designed airplaine. The authors address how to fairly divide costs between airlines provided the different designs and sizes of their planes.

Bistaffa et al. [11] applied cooperative game theory to analyze the social ridesharing problem. Mobility platforms such as Lyft allow their users to share their trip with nearby users in real-time, providing a more environmentally friendly and cost effective alternative to commuting alone. They study how the costs of the ride should be divided among passengers. Another application is in supply chain management, where Fiestras-Janeiro et al. [22] study how to coordinate actions between suppliers, manufactorers, retailers and customers.