Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 126 (2018) 234–243

Procedia Computer Science

www.elsevier.com/locate/procedia

22nd International Conference on Knowledge-Based and Intelligent Information & Engineering Systems

Analysing Neural Network Topologies: a Game Theoretic Approach

Julian Stier[a,*], Gabriele Gianini[b,c], Michael Granitzer[a], Konstantin Ziegler[a]

[a] *University of Passau, Innstrasse 42, 94032 Passau, Germany*
[b] *Università degli Studi di Milano, via Bramante 65, Crema, IT26013, Italy*
[c] *EBTIC, Khalifa University of Science, Technology & Research, Hadbat Al Zaafran 127788, Abu Dhabi (UAE)*

Abstract

Artificial Neural Networks have shown impressive success in very different application cases. Choosing a proper network architecture is a critical decision for a network's success, usually done in a manual manner. As a straightforward strategy, large, mostly fully connected architectures are selected, thereby relying on a good optimization strategy to find proper weights while at the same time avoiding overfitting. However, large parts of the final network are redundant. In the best case, large parts of the network become simply irrelevant for later inferencing. In the worst case, highly parameterized architectures hinder proper optimization and allow the easy creation of adverserial examples fooling the network.

A first step in removing irrelevant architectural parts lies in identifying those parts, which requires measuring the contribution of individual components such as neurons. In previous work, heuristics based on using the weight distribution of a neuron as contribution measure have shown some success, but do not provide a proper theoretical understanding.

Therefore, in our work we investigate game theoretic measures, namely the Shapley value (SV), in order to separate relevant from irrelevant parts of an artificial neural network. We begin by designing a coalitional game for an artificial neural network, where neurons form coalitions and the average contributions of neurons to coalitions yield to the Shapley value. In order to measure how well the Shapley value measures the contribution of individual neurons, we remove low-contributing neurons and measure its impact on the network performance.

In our experiments we show that the Shapley value outperforms other heuristics for measuring the contribution of neurons.

Keywords: Neural networks, Shapley value, Topology optimization, Neural network pruning

* Corresponding author
E-mail address: julian.stier@uni-passau.de

10.1016/j.procs.2018.07.257

1. Introduction

The architecture of an Artificial Neural Network (ANN) strongly influences its performance [16, 10, 25]. However, designing the structure of an artificial neural network is a complex task requiring expert knowledge and extensive experimentation. Usually fully connected layers are used yielding to a high number of parameters. While well-chosen optimization strategies allow to identify proper parameterization in general, larger architectures (i) have an increased risk of overfitting the training data, (ii) use more computational resources and (iii) are more affected by adversarial examples.

Identifying optimal architectures for an ANN is a NP-complete optimization problem. Solutions can be categorized into bottom-up, top-down or mixed methods. Bottom-up methods start from small architectures or none at all and gradually add more components (e.g. layers, neurons or weights). An example for a bottom-up method is a grid-search [1] for probing different number of hidden layers and number of neurons per layer. Top-down methods start with larger architectures and remove low-contributing components, yielding a significantly smaller, *pruned* architecture.

Well known top-down methods such as *optimal brain damage* [18] or *skeletonization* [20] utilize different heuristics, like for example the weight of a connection, and different search strategies, like for example greedy search, in order to identify components to remove. Both methods, top-down and bottom-up, conduct a non-exhaustive search in a huge parameter space. Such a search requires proper importance measures, i.e. measures to identify the importance of individual components. However, most importance-measures do not rely on a well-formed theory but are defined in an ad-hoc manner, thereby limiting its applicability.

When viewing neurons in a neural network as competing and collaborating individuals, Game Theory can provide a possible theoretical background for properly selecting the most important "player" in a game. In particular, coalitional games (also known as cooperative games) allow to view groups of neurons as coallitions competing with other coalitions. Measures like for example the Shapley value [24] allow to determine the payoff for an individual, thereby determining its contribution to the coalition.

1.1. Contributions

In our work we utilize the Shapley value as importance-measure to determine the contribution of neurons in a network. We transform an ANN into a coalitional game between neurons, giving us access to well-studied game theoretic results. The possibilities of this transformation process are discussed the first time, despite former existing experiments involving the Shapley value, and treating the ANN as a coalitional game is separated from pruning ANNs.

Given the coalitional game from this transformation, we can estimate the Shapley value for every neuron reflecting its individual contribution to the overall ANN architecture. As the Shapley value requires forming all possible coalitions, we suggest a sampling procedure for obtaining Shapley value approximations. The suggested sampling parameters are justified, even for non-uniform Shapley value distributions.

Finally, we use the Shapley value in a top-down pruning strategy and compare it to other heuristical pruning measures based on the weights between neurons. We show that the Shapley value provides a more robust estimate for the importance of a neuron yielding to a better performance of an ANN for the same model size than weight-based heuristics. While pruning with Shapley values was previosuly only shown in small problem domains [19] and with one single strategy based on Shapley values, this work presents results on image and text classification with multiple strategies, including Shapley values.

2. Related work

Bergstra et. al claim "grid search and manual search are the most widely used strategies for hyper-parameter optimization" [1]. However, there exist various destructive and constructive approaches to obtain better network topologies.

Concerning destructive approaches, *optimal brain damage* [18] "uses information-theoretic ideas to derive [..] nearly optimal schemes for adapting the size of a[n artificial] neural network". For this, it uses second-derivative information and tries to minimize a composed cost function of training error and a measure of network complexity.

In *Skeletonization* [20], Mozer presents a destructive technique in which he prunes network components by means of their relevance. The relevance is basically measured as the difference of the training error with and without the specific network component. With respect to pruning strategies in section 4, this technique is similar to the payoff function used in obtaining Shapley values.

Another second-order derivative method is presented with *optimal brain surgeon* [9] and improves the previous optimal brain damage method by pruning multiple weights based on their error change and immediately applying changes to remaining weights.

In [11, 12] Keinan et. al present the multi-perturbation Shapley value analysis (MSA) using Shapley value with "a data set of multi-lesions or other perturbations". Shapley value is used to analyse contributions of components in biological neural networks. Different choices of network components as aspect of the coalitional game are considered in [13], chapter 9.

Based on the multi-perturbation Shapley value analysis, Cohen et. al [4] present a Contribution-Selection algorithm (CSA) "using either forward selection or backward elimination" [4]. Furthermore, Kötter et. al use the "Shapley value principle [..] to assess the contributions of individual brain structures" [14]. They even find "strong correlation between Shapley values" and properties from graph theory such as "betweenness centrality and connection density" [14].

Leon [19] uses Shapley value to optimize artificial neural network topologies by pruning neurons with minimal value or below a threshold in relation to the average value of the Shapley value distribution. The method is applied on the XOR-, Iris-, energy efficiency, ionosphere, and Yacht hydrodynamics problems which all yield test set accuracies above 0.9 with at most four neurons in their networks' hidden layer.

Schuster and Yamaguchi [23] investigate a complementary approach, where the interaction of two neurons in an artificial neural network is seen as a non-cooperative game.

3. A Game on Topologies

In a coalitional game, different subsets of a population of players generate different payoffs. The payoff for a subset (*coalition*) depends only on the participating players and a central question is how to value a single players "contribution".

3.1. The Shapley Value

The predominant solution concept for coalitional games is the *Shapley value* [24]. Let U be a set of n players, $\mathcal{P}(U)$ its powerset, and let $v\colon \mathcal{P}(U) \to \mathbb{R}$ be a set function which assigns a payoff $v(S)$ to every subset $S \subseteq U$ of players. The Shapley value "can be interpreted as the expected marginal contribution of player i" [22]. For player $i \in U$, it is given by

$$\phi_v(i) = \frac{1}{n!} \sum_{S \subseteq U \setminus \{i\}} (|S|!(n - |S| - 1)!) \cdot (v(S) - v(S - i)) \tag{1}$$

$$= \frac{1}{|U|!} \sum_{\pi \in \Pi} \left(v(P_i^\pi \cup \{i\}) - v(P_i^\pi) \right), \tag{2}$$

where Π is the set of all permutations of U and $P_i^\pi = \{j \in U : \pi(j) < \pi(i)\}$.

Imagine a **simple example** with three collaborating players $U = \{A, B, C\}$ contributing to a common goal such as selling a product. The payoff of players is only known coalition-wise and thus the payoff function $v(S)$ could be given as: $\{(\emptyset, 0), (\{A\}, 1), (\{B\}, 2), (\{C\}, 2), (\{A, B\}, 4), (\{A, C\}, 3), (\{B, C\}, 3), (\{A, B, C\}, 5)\}$. Then the Shapley value for each player results in $\phi_v(A) = 1.5$, $\phi_v(B) = 2$ and $\phi_v(C) = 1.5$ which can be scaled to $\phi_v(A) = 0.3$, $\phi_v(B) = 0.4$ and $\phi_v(C) = 0.3$, given the fact that the maximum payoff is 5. Player B can then be interpreted as most contributing player to the game and both players A and C are contributing less.

Due to its exponential computational complexity, Shapley values are approximated with a Monte Carlo method. The *subset definition* (1) for $\phi_v(i)$ is approximated with random subsets $R \subset U$:

$$\phi_v^R(i) = \frac{1}{\sum_{S \in R} \omega_S} \sum_{S \in R} \omega_S \cdot (v(S) - v(S - i)), \tag{3}$$

with $\omega_S = |S|!(n - |S| - 1)!$. Analogously, the *permutation definition* (2) is approximated with r random permutations Π^R:

$$\phi_v^{\Pi^R}(i) = \frac{1}{r} \sum_{\pi \in \Pi^R} \left(v(P_i^\pi \cup \{i\}) - v(P_i^\pi) \right). \tag{4}$$

3.2. Designing the Game

The idea of assigning each player a value, given a set function which defines the payoff of a coalition of players, can be transferred to neural networks. For this, a set of players U and the payoff function v must be defined.

The **set of players** U can consist of any mutually excluding structural components of the network. Choosing the concrete structural components defines the *perspective* in which the game is played. Because the structure can be arbitrarily broken into players, perspectives are categorized into homogeneous and non-homogeneous perspectives.

Homogeneous perspectives consist exclusively of structurally equivalent components. Non-homogeneous perspectives are not further considered in this paper as they are not directly intuitive and introduce unnecessary complexity. An example for a homogeneous perspective is the set of players representing each neuron in the hidden layer of a feed-forward network with one hidden layer. Neurons of the input layer or several layers of a multilayer neural network provide other perspectives.

To define the **payoff function** v of the coalitional game of an ANN, any evaluation measure or error value of the network could be considered. Error values such as the training error are unbounded which might be undesirable for later analysis. Evaluation measures such as the accuracy are usually bounded and the cross-entropy accuracy in particular is used in this work. Usually, coalitional games in game theory can be combined based on their super-additive payoff functions. However, both choices – error values or evaluation measures – do not provide the super-additivity property. This disables deriving desirable properties of symmetry, efficiency and additivity for the Shapley value as proven by Shapley for games with such super-additive payoffs [24]. It is not possible to combine multiple coalitional games on artificial neural networks without finding a super-additive payoff function.

The accuracy[1] as an evaluation measure of ANNs is used to construct the payoff value of the coalitional game. Looking at the accuracy it can be stated:

1. The payoff for the *grand coalition* is not necessarily the maximum possible payoff value.
2. The maximum value of the payoff is not known in an analytical way prior to computing all values for every possible coalition.
3. The accuracy is not super-additive, meaning there are coalitions $S, T \subseteq U$ with $v(S) < v(S \cap T) + v(S - T)$. It is not even monotone as there might exist neural networks with fewer network components but still larger accuracy.

Nonetheless, given a network evaluation measure m such as the accuracy, a payoff value can be defined as following:

$$v(S) := m(S) - m(\emptyset),$$

with $S \subseteq U$ and $m(T)$ denoting the evaluated measure of the network with only players contained in T. Usually, $m(\emptyset)$ should be at least above the naïve expectation of the classification or regression problem. Therefore, it can be assumed that $m(\emptyset) > 0$ and e.g. for a classification problem of k classes $m(\emptyset) > \frac{1}{k}$ (the evaluated model should be better than random guessing).

[1] Number of correctly classified instances given a test or validation set of e.g. 10000 observations.

Pruning Strategies	
Name	Description
$SVbottom(k)$	Prune k players with smallest Shapley value.
$SVbottom(p)$	Prune players with Shapley value below p: $\phi(i) < p \cdot \frac{1}{\lvert S \rvert}$.
$SVbucket(p)$	Prune n players with smallest Shapley values such that $\sum_{i=0}^{n} \phi(i) < p$.
$random(k)$	Prune k random players.
$Wbottom(k)$	Prune k players with smallest norm of their weights; e.g. $\text{norm}(n) = \sqrt{\sum_{j \in E_n^{in}} (w_{j,n})^2}$.

Table 1. Pruning strategies gathered and analysed in the course of this work. The strategy used by Leon [19] is $SVbottom(p)$, but with differences in technical detail.

This definition can produce negative values, as well. In fact, $v(S)$ is in range $[-1, 1]$ instead of $[0, 1]$ [2]. As stated in [2], "the meaning of the sign is clear. For positive values, the corresponding criterion has to be considered, in average, as a benefit, conversely, for negative values, it represents a cost".

As another example, Leon [19] uses a compound metric of the correlation coefficient and an error measure but does not explain this choice in-depth.

3.3. Strategies to obtain network topologies

A top-down method derives a neural network structure from an initial (potentially large) root model in a *derivation process*. The result of such a process is a trained ANN model with potentially as few players (e.g. neurons) as possible, but at least fewer players than the initial root model. One important step of the derivation process is the pruning of concrete structural components, which depends on the chosen *pruning strategy*. Each strategy defines a derivation rule, optional requirements[2] and a stopping criterion. The derivation rule selects a set of players to remove in each derivation step. The stopping criterion decides if further pruning is possible or if the overall process terminates.

This work examined three different families of strategies: random-based, weight-based and Shapley-value-based strategies. An overview of strategies is listed in table 1.

Strategies which select a fixed number of players for pruning within one step can be considered naïve or non-dynamic. They mostly prune too few players in early steps and too many in late steps with few players left.

Strategies based on random selections with a fixed size are an example for such naïve strategies. A random-based strategy with k players prunes k randomly selected players. Any strategy claiming to use a well-founded heuristic to select players for pruning must compete against random selection.

A lot of existing strategies are based on information of the network's weights. Three of such weight-based strategies are analysed in Wang et al [27]. Their first strategy is based on "$\sigma(R)$ score [..] generalized from" approaches gathered by Thimm et al [26]. Those gathered approaches include smallest weight pruning *min(w)* and sensitivity of a network to removal of a weight by monitoring sum of all weights changes during training. The third strategy of Wang et al "uses the average value of absolute weights sum" of a neuron.

Here, a naïve pruning strategy based on weights is chosen as baseline comparison. The strategy prunes k players with the smallest norm of their weights.

For a given neuron n its norm is calculated as $\text{norm}(n) = \sqrt{\sum_{j \in E_n^{in}} (w_{j,n})^2}$. with E_n^{in} being the set of incoming connections to neuron n and $w_{j,n}$ the weight for the connection from neuron j to neuron n.

If the game perspective defines one player as one neuron, simply $\text{norm}(n)$ can be considered for the strategy. For multiple neurons treated as one player, one might sum this norm over all neurons of the concerned player. With a

[2] Requirements of a strategy include the computation of statistical values such as weight norm for $Wbottom(k)$ or Shapley value for $SVbottom(p)$.

fine-grained perspective on weights, one can use the related weights of the player for the root of summed, squared weights.

Based on the game design given in section 3.2 three types of pruning strategies based on Shapley values are proposed: A naïve strategy is given as $SVbottom(k)$ which prunes k players with lowest Shapley value in analogy to $random(k)$ and $Wbottom(k)$.

More dynamically, $SVbottom(p)$ prunes a player i given its Shapley value is below a threshold given by factor p and the current average contribution if it would be uniformly distributed:

$$\phi(i) < p \cdot \frac{1}{|S|}$$

This approach is similar to the one used by Leon in which "the maximum Shapley value threshold to eliminate network elements is $\theta = \theta_s \cdot a_s$ where $\theta_s \in \{0, 0.1, 0.25\}$ and a_s is the average Shapley value of all existing network elements."[19]. The dynamic threshold in both methods can be considered highly similar as the expectation of Shapley values evidently matches the expectation of a uniform distribution. Computing the expectation $\frac{1}{|S|}$ is neat and advantages of using an average of approximated Shapley values could not be found. Like in the method of Leon, if no player below this threshold is found, "the one with the minimum value becomes the candidate for elimination."[19].

The third strategy, $SVbucket(p)$, prunes players $i \in T$ such that $\underset{T \subset U}{\operatorname{argmax}}(|T|)$ and

$$\sum_{i \in T} \phi(i) < p$$

In other words, $SVbucket(p)$ collects all players $i \in T$ with smallest Shapley value as long as their sum of Shapley values stays below a bucket value p. Again, if no player matches this criterion, the one with the smallest Shapley value is selected.

3.4. Approximating the Shapley value

Calculating the exact Shapley value requires averaging over all 2^N possible coalition, which is computational too expensive. In fact, it is NP-complete [5]. To overcome this limitation, the Shapley value is usually approximated through random sampling, as proposed in [6]. Because there has been recent focus on other methods for approximating the Shapley value such as sampling-based polynomial calculations [3] or structured random sampling [7] and to identify the applicability of random sampling for our approach, we condcuted preliminary experiments with randomly generated coalitional games (*Randomly Pertubated Uniform game*) and the well-known *United Nations Security Council game* [22]. While the first games evaluate the approximation errors in case of almost uniformly distributed contributions with small perturbations, the second game addresses a non-uniform distribution of contributions. Our experimental results show, that at least 100 random samples are required for the permutation definition and at least 500 random samples are required for the subset definition. The results go along with experiments in [6] in which in "most cases, the error is less than 5%.". Due to space constraints, we do not present further details here.

4. Experiments

To assess the expressiveness of Shapley values we used them in context of pruning and compared them to methods with different heuristics. For this assessment, we conducted the following experiments:

1. **MNIST pruning** Pruning *MNIST* models in an iterative top-down manner based on different strategies including ones based on Shapley values.
2. **Pruning evaluation** An evaluation of Shapley-value-based pruning by comparing it with random selections of models obtained by grid search.
3. **20newsgroups pruning** Pruning of larger *20newsgroups* models for comparison with previous insights and proof of scale.

In order to understand the effect of the Shapley value, we stated several questions which can be summarized as following:

- Which strategy requires the least number of steps?
- Can we define a lower bound for the game size given a problem and a threshold for the evaluation measure?
- How stable are the examined strategies?

The first question addresses *how many steps* each strategy requires before the performance drops below a given threshold $\theta \in (0, 1)$ for the evaluation measure. The number of steps directly influences the overall number of training epochs and thus reflects a computational cost. Given θ, we also looked at which strategy found the *least number of players* and if it could find this minimum repeatedly. The found minimum across several strategies was compared to an exhaustive grid search (*pruning evaluation*) to assess if there can be better performing models with less players found. Strategies were calculated repeatedly to estimate their stability in terms of expectation and their standard deviation. We did not only compare the found minimums, but also watched the pruning strategies along each step to compare how much contributional value (sum of Shapley values) was removed by non-Shapley-value-based strategies and if there can be any patterns found.

In the following, we briefly outline the experimental setup and results obtained for the experiments.

4.1. MNIST models pruning

In the first experiment we evaluate our approach on a feed-forward network with a single hidden layer of size h trained on the MNIST [17] dataset using cross-entropy error. Each neuron in the hidden layer is represented by a player in U such that $|U| = h$. We choose an initial hidden layer size $h = 40$ which forms our root-model. After training the root-model for $T_0 = 20$ epochs, we apply the different pruning strategies outlined above. Estimating the payoff $v(S)$ over a coallition of players, we calculate the accuracy over the test set for this coalition, thereby removing all neurons not in S.

Based on the selected pruning strategy we remove the least-contributing neurons resulting in a new model. The new model is trained for $T_1 = 2$ epochs in order to compensate for the removed neurons. We measure the validation set accuracy of the new model as estimator for the goodness of the pruning strategy. If important neurons would have been pruned, we expect a lower accuracy than when pruning unimportant nodes.
Each experiment for a single pruning strategy is repeated independently twenty times in order to account for random effects.

4.1.1. Results

Figures 1a and 1b depict pruning walks of selected Shapley-value-based strategies in comparison with other strategies from 1.

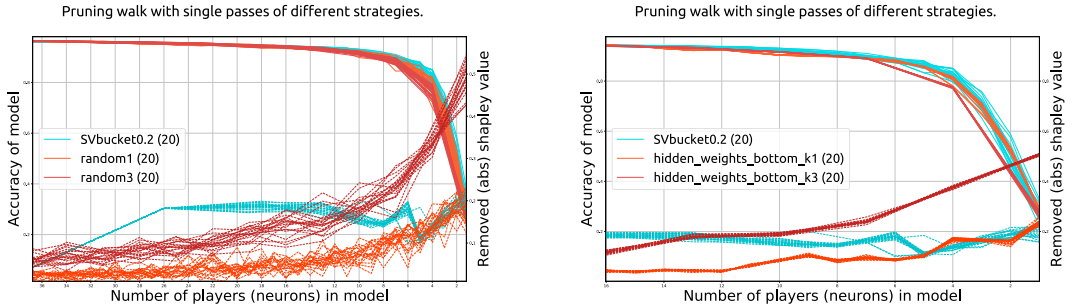
The number of required steps for a single strategy is not directly viewable in this visualization. For this, take a look at the listed tables such as table 2a. Strategies with a fixed number of players to prune in each step, the number of steps can be directly obtained by dividing the number of initial players with the number of pruned players. In case of a model-search with an accuracy threshold this number of steps does not apply as the strategy reaches the stopping criterion earlier.

Almost all strategies perform pretty similar for a number of players larger than twelve. This fact is one indicator for the conclusion that contributions of a single player can be taken over by other players as long as the capacity of the network suffices to solve the problem. A shift in the internal solving methodology of the network can not be identified.

Continuous lines depict the accuracy curve of each model. Dotted lines are their respective sum of removed Shapley value.

Figure 1a compares **SVbucket**(0.2) with **random**(1) and **random**(3). The average number of steps for SVbucket(0.2) is 12.05 with a minimum of 12 and a maximum of 13 steps. Despite the algorithmic approach of strategies, randomness of the derivation process of obtaining and retraining a smaller model and the approximation error for Shapley values reflect slightly in these values. Random(k) have 40 and 14 fixed steps, respectively.

It can be observed that below twelve players the Shapley-value-based strategy is able to stay above random pruning. Note, how the removed Shapley value of each strategy stays below a threshold of 0.2 or increased with decreasing number of players, respectively.



(a) **SVbucket**(0.2) **vs. random**(k): fewer steps and superior in moment of sudden accuracy decay: Shapley-value-based pruning is clearly preferrable to random guessing.

(b) **SVbucket**(0.2) **vs Wbottom**(k)**:** Weights are obviously no direct indicator for well contributing players. At least when considering pruning, a strategy based on weights is inferior to one based on Shapley values.

Fig. 1. Exemplary pruning walks. Comparing $SVBucket(p)$ with $random(k)$ and $Wbottom(k)$.

Figure 1b compares **SVbucket**(0.2) with **Wbottom**(1) and **Wbottom**(3). The weight-based strategies take 40 and 14 fixed steps, respectively. It can be clearly seen that the Shapley-value-based strategy stays above the weighted-based one. A weight-based strategy is questionable as low weights are assumably no indicator for contributional value of a single player.

SVbottomP(0.5) takes 31.35 steps on average, its maximum is 33 and its minimum 30. While SVbucket(0.2) starts with pruning a larger amount of players (fitting into a bucket of $p = 0.2$) SVbottomP(0.5) slightly increased the amount of Shapley value to be removed in each step as the average Shapley value increases. Before the moment of sudden accuracy decay (below eight players) one could argue SVbottomP(0.5) to be superior to SVbucket(0.2) – it prunes fewer players. However, it also takes more steps and SVbucket(0.2) is able to jump pretty far down within a few steps which saves a lot of retraining epochs.

In search for a minimum number of required players to stay above an accuracy threshold of $\theta = 0.9$ some statistical values for SVbucket(p) and SVbottomP(p) are given in tables 2a and 2c. Average number of steps (obtained model versions) and possible outliers in those values are easy to compare runtimes required for a grid search conducted in the following experiment. It can be clearly seen that one of the described top-down strategies is able to find a value near the minimum within a significant lower required training epochs.

4.2. *MNIST models grid search*

For hyperparameters *number of training epochs* and *number of players* a grid search is conducted and repeated 200 times. Each result is an accuracy value obtained from a MNIST feedforward network, trained with the according number of epochs and number of given players (hidden neurons).

Comparing the grid search results to the MNIST experiments with different strategies, it gets obvious to state that a top-down pruning strategy is more efficient under assumption of obtaining Shapley values in constant or small linear time. In fact, approximating Shapley values with 500 samples only takes 500 inference steps which are computationally cheap in comparison to training epochs. Unlike training, approximating Shapley value can be fully distributed and the parallel inference computations only need to be summed. This parallelization is used in the underlying framework. Grid search finds comparable models and minimum number of players with a technically more simple method.

SVbucket0.1	
avg # steps	19.65
max # steps	21
min # steps	19
avg # epochs	59.3
max # epochs	62
min # epochs	58
avg # found	6.95
max # found	8
min # found	6

(a) A low bucket value of 0.1 for pruning requires a lot of steps.

SVbucket0.2	
avg # steps	12.05
max # steps	13
min # steps	12
avg # epochs	44.1
max # epochs	46
min # epochs	44
avg # found	8.05
max # found	10
min # found	7

(b) A compromise between pruning too fast and still having few retraining epochs.

SVbottomP0.5	
avg # steps	31.35
max # steps	33
min # steps	30
avg # epochs	82.7
max # epochs	86
min # epochs	80
avg # found	7.25
max # found	8
min # found	7

(c) SVbottomP0.5

SVbottomP0.7	
avg # steps	15.75
max # steps	19
min # steps	13
avg # epochs	51.5
max # epochs	58
min # epochs	46
avg # found	7.25
max # found	8
min # found	6

(d) SVbottomP0.7

Table 2. **SVbucket**-strategy: The table shows average, maximum and minimum number of steps to reach the stopping criterion (no more neurons to prune). It also shows statistics for the total number of required epochs during the destructive iterative approach. The average, maximum and minimum number of neurons to stay above the threshold obtained by all repeated strategies is denoted as "found".
Starting with a pre-trained root model with 20 training epochs, SVbucket results in an almost constant number of required epochs. Depending on the used parameter, the strategy finds an average of seven, eight or nine minimum number of players to stay above a threshold of 0.9 in accuracy within a total maximum number of required training epochs of 60, 45 and 36.
SVbottomP-strategy: With a total number of required training epochs of 52 and 83 the strategy is able to find a minimum number of seven required players to stay above a threshold of 0.9 in accuracy for classifying MNIST. Twenty repeated runs are performed for each strategy to obtain the statistics and confirms the stability of the method.

With means of computational costs however, contribution-based pruning methods are faster and might provide more insights into ANNs in future.

4.3. 20newsgroups models pruning

The third experiment is set up equivalent to the first but based on the *20newsgroups* dataset [15]. Evaluation measures are not directly comparable to MNIST. However, statistical results for all strategies of the MNIST experiment could be reproduced. The accuracy threshold was set to $\theta = 0.72$. Repetitions of SVbucket(0.2) produced an average of 9.2 players required in minimum for θ.

5. Future Work

Theoretical aspects: 1) Properties like symmetry or efficiency might contribute to *reducing the complexity* of computing Shapley values. 2) The classic Shapley value implies that all players of a game can form a meaningful coalition. Concerning neural networks, you might not have sets where every subset is meaningful. The game could be extended to account for network and not just coalition structure. Myerson [21] augmented the classic cooperative game by adding a network structure and obtained a *communication game* in which the role of the network defines which coalitions can actually operate.

Algorithmic aspects: 1) Destructive, constructive or hybrid approaches could be navigated by Shapley values. 2) Simulated annealing suggests that greedy steps can get you stuck in local minima. Mixing random pruning and Shapley value guided pruning would be an example of another heuristic for topology optimization.

6. Conclusion

Shapley value as a solution concept for coalitional games can be applied to ANNs to obtain values of contribution for components such as neurons.

The idea was separated into a game on topologies to obtain Shapley values as measuring values and pruning strategies as one possible application. The methodology was generalised to different perspectives on neural networks.

Choices in course of constructing a game on topologies – such as the perspective or the payoff function – were discussed. It was discovered that the properties symmetry, efficiency and additivity of the Shapley value can not easily be derived. This led to further research questions, e.g. "is it possible to construct a super-additive payoff function based on neural networks?".

The question of usefulness of the Shapley value was approached within the scope of pruning methods for neural networks. On the one hand, it could be shown that methods based on the Shapley value have a longer power of endurance in terms of their accuracy when pruned. On the other hand, only models with already severely reduced components showed significant differences between strategies.

The hypothesis for this observation is that the analysed problems are still low in their complexity while very large models just offer a large amount of redundant degrees of freedom. Reducing these parameters then, it can be observed that the model compensates the loss without dropping significantly in accuracy as long as there are still a large amount of parameters. This goes along with findings in compressing neural networks via pruning, e.g. as in Han et. al [8]. Neural networks are able to compensate neural loss if they undergo a retraining phase.

Shapley values give a hint of the importance of different network components. However, the competence of such an component can be taken over by equivalent components when pruned.

References

[1] Bergstra, J., Bengio, Y., 2012. Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305.

[2] Cardin, M., Giove, S., et al., 2007. On non-monotonic Choquet integrals as aggregation functions. Technical Report.

[3] Castro, J., Gómez, D., Tejada, J., 2009. Polynomial calculation of the shapley value based on sampling. Computers & Operations Research 36, 1726–1730.

[4] Cohen, S., Dror, G., Ruppin, E., 2007. Feature selection via coalitional game theory. Neural Computation 19, 1939–1961.

[5] Deng, X., Papadimitriou, C.H., 1994. On the complexity of cooperative solution concepts. Mathematics of Operations Research 19, 257–266.

[6] Fatima, S.S., Wooldridge, M., Jennings, N.R., 2007. A randomized method for the shapley value for the voting game, in: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, ACM. p. 157.

[7] Hamers, H., Husslage, B., Lindelauf, R., Campen, T., et al., 2016. A New Approximation Method for the Shapley Value Applied to the WTC 9/11 Terrorist Attack. Technical Report.

[8] Han, S., Pool, J., Tran, J., Dally, W., 2015. Learning both weights and connections for efficient neural network, in: Advances in neural information processing systems, pp. 1135–1143.

[9] Hassibi, B., Stork, D.G., Wolff, G.J., 1993. Optimal brain surgeon and general network pruning, in: Neural Networks, 1993., IEEE International Conference on, IEEE. pp. 293–299.

[10] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780.

[11] Keinan, A., Hilgetag, C.C., Meilijson, I., Ruppin, E., 2004a. Causal localization of neural function: the shapley value method. Neurocomputing 58, 215–222.

[12] Keinan, A., Sandbank, B., Hilgetag, C.C., Meilijson, I., Ruppin, E., 2004b. Fair attribution of functional contribution in artificial and biological networks. Neural Computation 16, 1887–1915.

[13] Keinan, A., Sandbank, B., Hilgetag, C.C., Meilijson, I., Ruppin, E., 2006. Axiomatic scalable neurocontroller analysis via the shapley value. Artificial Life 12, 333–352.

[14] Kötter, R., Reid, A.T., Krumnack, A., Wanke, E., Sporns, O., 2007. Shapley ratings in brain networks. Frontiers in neuroinformatics 1, 2.

[15] Lang, K., 1995. Newsweeder: Learning to filter netnews, in: Proceedings of the 12th international conference on machine learning, pp. 331–339.

[16] LeCun, Y., Bengio, Y., et al., 1995. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks 3361, 1995.

[17] LeCun, Y., Cortes, C., Burges, C.J., 1998. The mnist database of handwritten digits.

[18] LeCun, Y., Denker, J.S., Solla, S.A., Howard, R.E., Jackel, L.D., 1989. Optimal brain damage., in: NIPs, pp. 598–605.

[19] Leon, F., 2014. Optimizing neural network topology using shapley value. Proceedings of the 18th International Conference on System Theory Control and Computing 18.

[20] Mozer, M.C., Smolensky, P., 1989. Skeletonization: A technique for trimming the fat from a network via relevance assessment .

[21] Myerson, R.B., 1977. Graphs and cooperation in games. Mathematics of operations research 2, 225–229.

[22] Roth, A.E., 1988. Introduction to the shapley value. Essays in honor of Lloyd S. Shapley 2.

[23] Schuster, A., Yamaguchi, Y., 2010. Application of game theory to neuronal networks. Advances in Artificial Intelligence 2010, 2.

[24] Shapley, L., 1953. A value for n-person games. Contributions to the Theory of Games 2.

[25] Srivastava, R.K., Greff, K., Schmidhuber, J., 2015. Highway networks. arXiv preprint arXiv:1505.00387 .

[26] Thimm, G., Fiesler, E., 1995. Evaluating pruning methods, in: Proceedings of International Symposium on Artificial Neural Networks, pp. 20–25.

[27] Wang, Z., Zhu, C., Xia, Z., Guo, Q., Liu, Y., 2017. Towards thinner convolutional neural networks through gradually global pruning. arXiv preprint arXiv:1703.09916 .