ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240850

NeurCAM: Interpretable Neural Clustering via Additive Models

Nakul Upadhya and Eldan Cohen

nakul.upadhya@mail.utoronto.ca, ecohen@mie.utoronto.ca
University of Toronto, Toronto, Canada

Abstract. Interpretable clustering algorithms aim to group similar data points while explaining the obtained groups to support knowledge discovery and pattern recognition tasks. While most approaches to interpretable clustering construct clusters using decision trees, the interpretability of trees often deteriorates on complex problems where large trees are required. In this work, we introduce the Neural Clustering Additive Model (NeurCAM), a novel approach to the interpretable clustering problem that leverages neural generalized additive models to provide fuzzy cluster membership with additive explanations of the obtained clusters. To promote sparsity in our model's explanations, we introduce selection gates that explicitly limit the number of features and pairwise interactions leveraged. Additionally, we demonstrate the capacity of our model to perform text clustering that considers the contextual representation of the texts while providing explanations for the obtained clusters based on uni- or bi-word terms. Extensive experiments show that NeurCAM achieves performance comparable to black-box methods on tabular datasets while remaining interpretable. Additionally, our approach significantly outperforms other interpretable clustering approaches when clustering on text data.

1 Introduction

As Machine Learning (ML) has become more prevalent in society in recent years, the need for trustworthy models that stakeholders can audit has increased dramatically. One desirable aspect of trustworthiness in ML is that the approaches utilized are constrained so that their predictive mechanisms are innately understandable to humans. As a result, they are much easier to troubleshoot and more practical for real-world usage [51].

One stream of interpretable machine learning is interpretable clustering [69]. By using algorithms capable of providing innate explanations of cluster compositions, interpretable clustering methods have found great success in fields such as market segmentation [2], climate science [61], and healthcare [41, 17]. Most approaches to this task involve the use of decision trees to build clusters [5, 15, 16, 18, 56, 14]. However, the size of decision trees heavily influences their interpretability [63, 40], and complex problems may necessitate larger, less interpretable trees.

Another innately interpretable architecture, the Generalized Additive Model (GAM) [22], has found great success as an interpretable approach in many high-stakes classification and regression tasks [27, 53]. A recent line of work has focused on developing Neural GAMs that enjoy better scalability and are able to learn more expressive, yet interpretable, additive models [1, 48, 10, 25]. Despite these benefits, GAMs have not been utilized for clustering.

In this work, we introduce the **Neur**al **C**lustering **A**dditive **M**odel (NeurCAM), an interpretable clustering approach that constructs clusters via Neural Generalized Additive Models. Our approach explains how input features influence cluster assignment by modeling the relationship between features and clustering assignments through additive shape functions. Our contributions are as follows:

1. We present a novel approach for interpretable clustering that leverages neural GAMs to provide fuzzy cluster membership. Our approach can leverage deep representations of the data for clustering while still producing explanations in the original feature space. To our knowledge this is the first work to utilize GAMs to provide interpretable clustering.
2. We introduce a mechanism that allows users to explicitly constrain the number of single-feature and pairwise interaction shape functions our model utilizes therefore encouraging sparsity in the final explanations, a key quality of interpretable models [52].
3. Through experimentation on a variety of datasets, we demonstrate NeurCAM's effectiveness at creating high-quality clusters when using disentangled representations and also showcase the interpretability provided by additive explanations.
4. We demonstrate the capabilities of NeurCAM to perform interpretable text clustering by leveraging transformer-based embeddings in the objective. This allows us to provide uni-word and bi-word explanations while still taking structural and contextual information of the document into account.

The rest of our paper is organized as follows. In Section 2 we outline our desiderata for the interpretable clustering task and discuss prior work that aligns with these objectives. In Section 3 we define GAMs and what makes them interpretable. In Section 4, we describe the components of our approach, NeurCAM, and in Section 5 demonstrate its performance and interpretability.

2 Interpretable Clustering

Our approach for interpretable clustering consists of developing an intrinsically interpretable out-of-sample mapping from samples to clusters. In this section we describe what encompasses this approach, the benefits of achieving it, and discuss existing approaches for interpretable clustering.

2.1 Interpretable Out-of-Sample Mapping

Out-of-Sample Mapping (OSM) in clustering refers to the task of assigning a given sample $\boldsymbol{x} \in \mathbb{R}^D$ (potentially unseen during training) to a particular cluster using a mapping that is *agnostic of the clustering cost function used* [18]. In particular, OSM allows us to separate the representation of the data used to construct the mapping from samples to clusters from the representation used in the clustering cost function. Previous work has shown that various transformed representations of the original data, such as spectral embeddings, learned embeddings, or PCA, can lead to better clustering performance [66, 60, 11, 14]. However, such representations are not human-understandable, making it difficult for practitioners to gain insights into the generated clusters. We therefore propose to construct *interpretable* OSM where the mapping is based on interpretable feature representation, while the clustering cost is independently defined over a transformed representation.

2.2 Model-Based Interpretability

We advocate the use of *intrisically interpretable models* to create the mapping. Formally, our goal is to develop an approach that satisfies the requirements for intrinsic model-based interpretability posed by Murdoch et al. [43]: *modularity, sparsity,* and *simulability*:

- **Modularity:** A ML model can be considered modular if a user can interpret a meaningful portion of its prediction making process independently from other parts of the network [43].
- **Sparsity:** Sparsity is achieved by limiting the number of non-zero parameters that limit the components a user must analyze to understand model behavior. Understandably, sparse models are easier for practitioners to understand, and hence easier to trust in high-stakes applications [43, 52].
- **Simulability:** An approach is said to be simulable if a human is able to reasonably internally simulate the entire decision-making process [43]. This requirement synergizes with the prior two requirements, as a model often needs to be sparse and modular for a practitioner to be able to recreate its predictions.

In contrast to model-based interpretability, one may opt to use an unintepretable model (e.g. deep neural networks) to assign samples to clusters and apply *post hoc* methods to explain clustering decisions. Although useful in many cases, post hoc approaches face a key problem in practice, where users often have to perform analysis on multiple post hoc explanations to identify which method they should trust [21]. Furthermore, arbitrary post hoc explanations can often be constructed for a given model to obfuscate true biases in the modeling procedure, reducing the trustworthiness of the approach [57].

2.3 Existing Approaches

The most prevalent approach to model-based interpretable clustering involves using unsupervised decision trees to partition the space [5, 15, 16, 56, 14, 18]. The most relevant approaches to our work are the soft clustering tree (SCT) [14] and the tree-alternating optimization (TAO) clustering [18]. Both approaches utilize a decision tree to map points to clusters and support OSM with seperate representations. The SCT utilizes an axis-aligned soft-decision tree trained through continuous optimization methods (including mini-batch gradient descent) [14], while TAO iteratively refines a tree that approximates a K-means clustering via alternating optimization [18].

Despite their interpretable nature, tree-based approaches face a substantial interpretability-performance trade-off as complex problems require large trees to represent cluster boundaries adequately, reducing the sparsity of these approaches. Growing the number of leaves in a tree has been shown to significantly increase the difficulty users face when trying to understand the decision pathways of the model [40]. Furthermore, increasing the number of features used in the decision tree significantly increases the time it takes users to analyze the tree and understand what features are essential for predictions [63].

Other interpretable clustering approaches include the construction of clusters using polytope machines [34] and rectangular rules [13].

Some notable post-hoc approaches include Kauffmann et al. [28] who explains a neural clustering via layerwise relevance propagation [4], Lawless and Gunluk [33] who profiles assignments via Polyhedral Descriptions, and Carrizosa et al. [8] who profiles assignments via prototypical examples. Additionally, Guan et al. [19] proposed to cluster text using deep embeddings and then performed a post hoc approximation of the clustering via a logistic regression on a bag-of-words representation. TELL [46] and IDC [62] both cluster via a layer in a Neural Network. Similar to post hoc methods, these approaches do not satisfy our desiderata for model-based interpretability (Section 2.2).

3 Generalized Additive Models

In this work, we propose to construct an interpretable cluster map using a Generalized Additive Model. Given a D-dimensional input $\boldsymbol{x} = \{x_i\}_{i=1}^D, \boldsymbol{x} \in \mathbb{R}^D$, univariate shape functions f_i corresponding to the input features x_i, bivariate shape functions f_{ij} corresponding to the features x_i and x_j, and link function $g(\cdot) : \mathbb{R}^D \to \mathbb{R}$, the predictions of a GAM and GA^{2}M are defined as follows:

$$\textbf{GAM} : g(\boldsymbol{x}) = f_0 + \sum_{i=1}^{D} f_i(x_i) \tag{1}$$

$$\textbf{GA}^2\textbf{M} : g(\boldsymbol{x}) = f_0 + \sum_{i=1}^{D} \left(f_i(x_i) + \sum_{j>i}^{D} f_{ij}(x_i, x_j) \right) \tag{2}$$

In recent years, many powerful GAM models have been proposed, the primary difference between them being how shape functions are constructed. Some notable examples include the Explainable Boosting Machine [37], which uses tree ensembles trained using a cyclical gradient boosting algorithm, and NODE-GAM [10], which leverages layers of ensemble oblivious neural decision trees.

Additionally, many works have proposed representing the shape function of the GAMs through MLPs such as Neural Additive Models (NAM) [1], which trains an MLP for each feature, and Neural Basis Models [48] which extend NAM by constraining all features to utilize a common MLP backbone except for the last layer, which is unique to each feature.

Interpretability of GAMs GAM and GA^{2}Ms satisfy the model-based interpretability requirements previously posed. GAMs force the relationship between the features in the model to be additive, resulting in a *modular* model [43]. This modularity allows the contribution of each feature or interaction to the prediction to be visualized as a graph or heatmap, allowing humans to *simulate* how a GAM works by querying the different graphs and adding the results together [38, 9]. This enables decision makers in high-stakes fields such as healthcare to easily understand the explanations provided

by GAM and GA^{2}M shape functions [23, 1]. Technical stakeholders have also shown a preference for GAMs over post hoc explanations such as SHAP values [39], as they reduce the cognitive load needed to grasp a model's decision mechanisms, enhancing stakeholders' confidence in the deployed ML system [29]. Furthermore, stakeholders have shown a preference for additive explanations over tree explanations when both utilize a similar number of features [63]. To further ensure that the explanations provided by NeurCAM are *sparse*, we propose selection gates that allow users to enforce a cardinality constraint on the number of features and interactions used.

4 NeurCAM

In this work, we consider the following problem: Let $X = \{\boldsymbol{x}_n\}_{n=1}^N$ be a set of N data points with $\boldsymbol{x}_n$ being a D-dimensional feature vector $\boldsymbol{x}_n \in \mathbb{R}^D$. We aim to decide fuzzy cluster assignments $w_{n,k} \in [0,1]$, $\sum_k w_{n,k} = 1.0$ for each point $\boldsymbol{x}_n$ and cluster $k \in 1, 2, \ldots, K$. Our approach to this problem involves utilizing a Neural GAM to obtain the assignments (Section 4.1) and train this GAM through a combination of a fuzzy clustering loss and self-supervised regularization (Section 4.2).

4.1 Model Architecture

In the following section, we describe how we obtain the cluster assignments via an interpretable neural GAM. For succinctness, we drop the sample index n in this section.

4.1.1 Neural Basis Model

For our additive model, we leverage a Neural Basis Model (NBM) [48] which we describe here for thoroughness.

The NBM operates by projecting each feature $x_i, i = 1, 2, \ldots, D$ onto B basis functions that are shared between all features. The projection function $\boldsymbol{b}(\cdot) : \mathbb{R} \to \mathbb{R}^B$ is represented by an MLP backbone with a single input and B outputs. The projection of each feature is then reconstructed into the final shape function for each cluster via a linear combination, with each feature having its own set of weights for each cluster $\boldsymbol{\lambda}_{i,k} \in \mathbb{R}^B$. More concretely, the prediction from a single feature x_i is as follows:

$$\boldsymbol{b}(x_i) = \text{MLP}(x_i) \tag{3}$$

$$f_{i,k}(\boldsymbol{x}) = \boldsymbol{\lambda}_{i,k} \cdot \boldsymbol{b}(x_i) \tag{4}$$

The resultant logits assigned to each cluster $h_k(\boldsymbol{x}), k = 1, 2, \ldots, K$ is the sum of the contributions from each feature:

$$g_k(\boldsymbol{x}) = \sum_i^D f_{i,k}(\boldsymbol{x}) \tag{5}$$

These logits are transformed via the softmax operation to obtain the final fuzzy assignment weights:

$$w_k(\boldsymbol{x}_n) = \frac{\exp\left(g_k(\boldsymbol{x})\right)}{\sum_{k'=1}^K \exp\left(g_{k'}(\boldsymbol{x})\right)} \tag{6}$$

Pairwise interaction can be represented in a similar manner using an MLP with two inputs instead of one.

4.1.2 Feature Selection Gates

A common problem across MLP-based GAMs and GA^{2}Ms is the variable explosion problem. As the dimensionality of the dataset increases, the number of single-feature and pairwise shape functions rapidly increases as well. This is especially true for GA^{2}Ms as the number of pairwise interactions grows quadratically with the dimensionality of the dataset. Previous MLP-based approaches have no inherent mechanisms to limit pairwise interactions and instead utilize heuristics based on residual analysis of a fitted GAM to select interactions to use in the GA^{2}M [1, 48]. However, this approach does not translate to the unsupervised setting as there are no residuals due to the absence of true labels.

Instead, we propose to *learn* what features and interactions are in our model through feature selection gates. More concretely, NeurCAM learns C independent shape functions. The feature used by shape function $c \in 1, \ldots, C$ is selected via a selection function $s_c(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$. A common way to represent such a function is the product of the feature vector and a one-hot selection vector $\boldsymbol{F}_c$ where a variable is selected if its corresponding selection logit $\tilde{F}_c \in \mathbb{R}^D$ is the largest:

$$s_c(\boldsymbol{x}) = \boldsymbol{F}_c \cdot \boldsymbol{x} \tag{7}$$

$$F_{c,i} = \begin{cases} 1 & i = \text{argmax}(\tilde{\boldsymbol{F}}_c) \\ 0 & \text{else} \end{cases} \tag{8}$$

To train this selection as part of our network, we *temporarily* relax the one-hot vector to be represented by entmax$_\alpha$ [47], a sparse version of softmax that allows elements to become exactly zero if the logits are sufficiently small:

$$s_c(\boldsymbol{x}) = \text{entmax}_\alpha(\tilde{\boldsymbol{F}}_c/T) \cdot \boldsymbol{x} \tag{9}$$

Here, $T > 0$ is a temperature annealing parameter that controls the sparsity of the distributions obtained from the entmax$_\alpha$ operations, with smaller values of T resulting in sparser distributions. As $T \to 0$, the resultant distribution will become one-hot.

At the start of training, we set $T = 1.0$, making the gate a weighted mixture of features. After a number of warm-up training epochs, we anneal T by a factor of $\epsilon \in (0, 1)$ until $\text{entmax}_\alpha(\tilde{\boldsymbol{F}}_c/T) \cdot \boldsymbol{x} = x_{\text{argmax}(\tilde{\boldsymbol{F}}_c)}$, allowing $s_c(\cdot)$ to serve as a proper feature selection gate and making NeurCAM a valid GAM once again. The factor ϵ is a hyperparameter that controls how gradual the tempering is, with higher epsilon values resulting in a slower annealing process and lower values resulting in a faster annealing.

We modify the NBM to include these selection gates to obtain the additive model architecture used by NeurCAM whos prediction mechanisms are as follows:

$$\boldsymbol{b}(\boldsymbol{x}) = \text{MLP}(s_c(\boldsymbol{x})) \tag{10}$$

$$f_{c,k}(\boldsymbol{x}) = \boldsymbol{\lambda}_{c,k} \cdot \boldsymbol{b}(\boldsymbol{x}) \tag{11}$$

$$g_k(\boldsymbol{x}) = \sum_{c=1}^C f_{c,k}(\boldsymbol{x}) \tag{12}$$

A softmax is then applied to $g_1(\boldsymbol{x}), g_2(\boldsymbol{x}) \ldots, g_k(\boldsymbol{x})$ to obtain the soft cluster assignments $\boldsymbol{w}(\boldsymbol{x})$ like in Equation (6).

The number of shape functions serves as an upper bound for the number of features utilized by the model, and the features used in the model's explanations can be limited by setting $C < D$. The choice of C is problem and stakeholder specific.

While we apply this mechanism on an NBM, these selection gates can be applied to other Neural GAM models such as the Neural Additive Model [1].

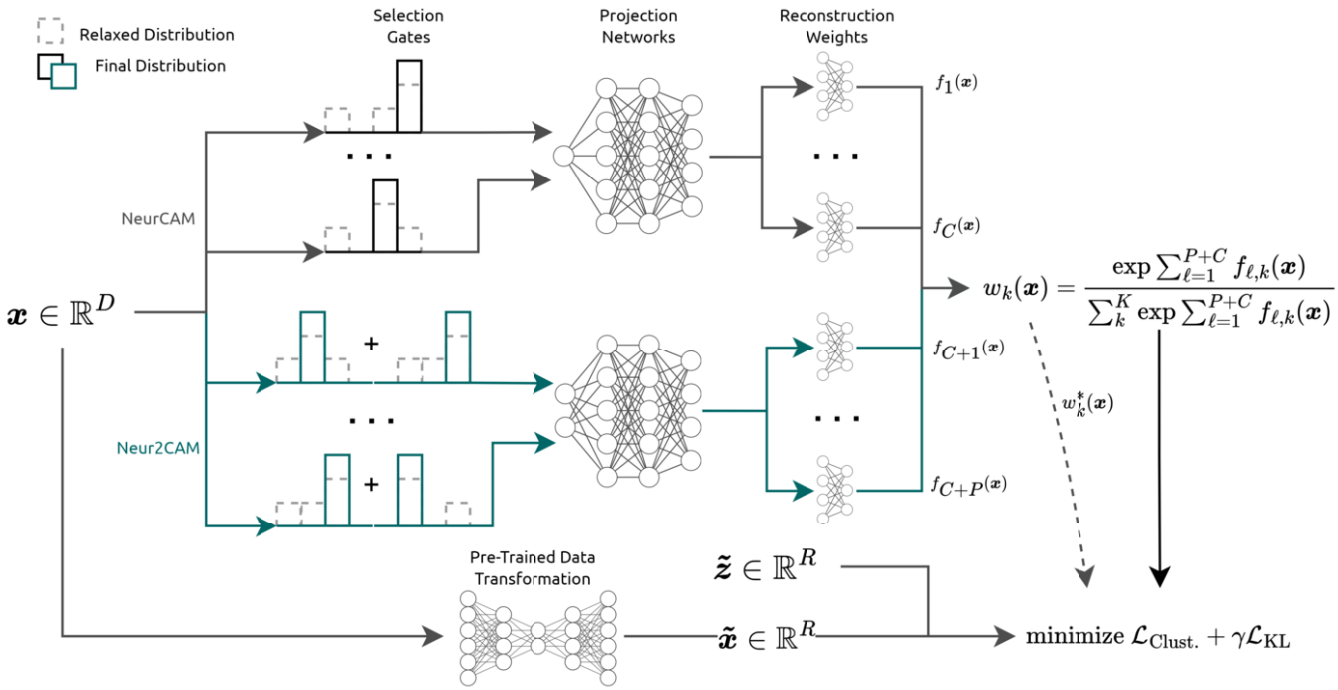

Figure 1. Our proposed approach. We leverage multiple shape functions that each pick a feature via a selection gate. The final prediction is the sum of the individual shape function contributions. The black shape functions represents NeurCAM and the additional blue pairwise shape functions represent Neur2CAM.

Extending to Neur2CAM: To extend NeurCAM to allow pairwise interactions, we introduce P additional shape functions whose selection gates allow for two features. More concretely, the pairwise interaction used by shape function $p \in C+1, \ldots, C+P$ is chosen by selection function $s_p^2(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}^2$, defined as:

$$s_p^2(\boldsymbol{x}) = \begin{bmatrix} \text{entmax}_\alpha(\tilde{\boldsymbol{F}}_{p,0}/T_2) \\ \text{entmax}_\alpha(\tilde{\boldsymbol{F}}_{p,1}/T_2) \end{bmatrix} \boldsymbol{x} \tag{13}$$

Like the single-order case, the pairwise temperature annealing parameter T_2 is set to 1.0 during the warmup phase and is annealed to near zero afterwards. All pairwise shape function share a common two-input MLP backbone that projects a pair of features into B outputs, with each shape function having its own set of reconstruction weights used to build the final shape functions. In theory, our approach can extend NeurCAM to interactions of any order, but we only explore up to pairwise to maintain a high degree of interpretability.

4.2 Training NeurCAM

In this section we describe the loss function and procedure used to train NeurCAM.

4.2.1 Fuzzy Clustering Loss

We employ a loss function inspired by Fuzzy C-Means [6]:

$$\mathcal{L}_{\text{Clust}} = \sum_{n=1}^{N} \sum_{k=1}^{K} w_k(\boldsymbol{x}_n)^m ||\boldsymbol{x}_n - \boldsymbol{z}_k||^2 \tag{14}$$

Here $m \geq 1.0$ is a hyperparameter controlling the fuzziness of the clustering. Our loss departs from Fuzzy C-Means in two main ways. In constrast to Fuzzy C-Means where the cluster assignment weights are free variables, our assignment weights are parameterized by an interpretable GAM (Equation (6)).

Additionally, instead of the centroids of the clusters being calculated by the clustering assignments [6], we opt to make the centroids $\boldsymbol{z}_k \in \mathbb{R}^D, k = 1, \ldots, K$ free variables to aid in the optimization of our network.

4.2.2 Disentangling Representations

The mapping constructed by NeurCAM is agnostic to how the distance between samples and the cluster centroids are defined. As such, we decouple the representations of the data in our loss function into the interpretable representation $\boldsymbol{x}$ and the transformed representation $\tilde{\boldsymbol{x}} \in \mathbb{R}^R$.

NeurCAM maps a sample to a given cluster using the interpretable feature set. However, the distance from each sample to the centroids of the clusters will be calculated using the transformed representation:

$$\mathcal{L}_{\text{Clust}} = \sum_{n=1}^{N} \sum_{k=1}^{K} w_k(\boldsymbol{x}_n)^m ||\tilde{\boldsymbol{x}}_n - \tilde{\boldsymbol{z}}_k||^2 \tag{15}$$

It is important to note that the learned centroids $\tilde{\boldsymbol{z}}_k \in \mathbb{R}^R$ are in the same space as the transformed representation $\tilde{\boldsymbol{x}}$.

4.2.3 Self-Supervised Regularization

Empirically, we observe that while our model is able to achieve a high-quality clustering at the end of the warm-up period, the selec-

tion gate annealing process often significantly degrades the ability of our network to directly optimize the clustering loss (Equation (15)) and can lead to poor local optima. To mitigate this degradation, we propose to take advantage of the clustering discovered at the end of the warm-up period, denoted by $\boldsymbol{w}^*(\boldsymbol{x}_n) \in \mathbb{R}^K$, to guide the optimization process after the annealing process starts.

When we start annealing the selection gates' temperatures T and T_2 toward zero, we add a regularization term that penalizes the KL-divergence between the current mapping $\boldsymbol{w}(\boldsymbol{x}_n)$ and the mapping $\boldsymbol{w}^*(\boldsymbol{x}_n)$ discovered at the end of the warm-up phase.

$$\mathcal{L}_{\text{KL}} = \sum_{n=1}^{N}\sum_{k=1}^{K} w_k^*(\boldsymbol{x}_n)\log\frac{w_k^*(\boldsymbol{x}_n)}{w_k(\boldsymbol{x}_n)} \tag{16}$$

The final objective utilized after the warmup phase is as follows:

$$\text{minimize } \mathcal{L}_{\text{Clust}} + \gamma\mathcal{L}_{\text{KL}} \tag{17}$$

Where γ is a parameter that controls the weight of the KL term. The complete pseudocode of NeurCAM's training procedure can be found in Algorithm (1).

Algorithm 1: NeurCAM Training Pseudocode

Input: $X, \tilde{X}, K, \gamma, \alpha, \epsilon, E_{\text{warmup}}, E_{\text{total}}$
$\tilde{\boldsymbol{z}} \leftarrow$ InitializeCentroids($\tilde{X}$)
$\theta_{\text{GAM}} \leftarrow$ RandomInitModelParams()
$\theta \leftarrow \tilde{\boldsymbol{z}} \cup \theta_{\text{GAM}}$
$T \leftarrow 1.0$
for $E = 1, 2, \ldots, E_{total}$ **do**
  if $E = E_{warmup}$ **then**
    $\theta^* \leftarrow \theta$
  end
  for $(X_{batch}, \tilde{X}_{batch}) \in (X, \tilde{X})$ **do**
    $\boldsymbol{w}_{\text{batch}} \leftarrow$ ForwardPass($X_{\text{batch}}, \theta, T$)
    if $E > E_{warmup}$ **then**
      $\boldsymbol{w}^*_{\text{batch}} \leftarrow$ ForwardPass($X_{\text{batch}}, \theta^*, 1.0$)
      $\mathcal{L}_{\text{KL}} \leftarrow \text{KL}(\boldsymbol{w}^*_{\text{batch}} || \boldsymbol{w}_{\text{batch}})$
    else
      $\mathcal{L}_{\text{KL}} \leftarrow 0$
    end
    $\mathcal{L}_{\text{Clust}} \leftarrow$ CalculateClustLoss($\tilde{X}_{\text{batch}}, \boldsymbol{w}_{\text{batch}}, \tilde{\boldsymbol{z}}$)
    $\mathcal{L} \leftarrow \mathcal{L}_{\text{clust}} + \gamma\mathcal{L}_{\text{KL}}$
    $\theta \leftarrow \theta - \alpha\nabla\mathcal{L}$
  end
  if $\exists c\ s_c(\boldsymbol{x}) \neq \boldsymbol{x}_{argmax(\tilde{\boldsymbol{F}}_c)}$ **then**
    $T \leftarrow \epsilon T$
  end
end

5 Experiments and Evaluation

In this section, we highlight the benefits of our approach in various real-world datasets. We first demonstrate our ability to generate high-quality clusters using disentangled representations on tabular data sets. We later extend our approach to text clustering. In addition, we validate our training scheme via an ablation study and provide an analysis of the interpretability of NeurCAM.

5.1 Experimental Details

We run experiments with two variants of our approach. NeurCAM (NCAM) includes only single-feature shape functions, and Neur2CAM (N2CAM), which extends NeurCAM to include pairwise interaction.

For our interpretable benchmarks, we consider approaches that provide model-based interpretability and are capable of disentangling representations, namely the recently proposed Soft Clustering Trees (SCT) [14] and the axis-aligned TAO Clustering Tree (TAO) [18]. For both of these tree-based approaches, we consider two different depths. We first consider a highly interpretable shallow tree with a depth of five (SCT/TAO-5). This choice also guarantees that the number of leaves is greater than the number of clusters across all datasets. For a more expressive, but less sparse, baseline, we also consider trees with a depth of seven (SCT/TAO-7). As a representative of black-box clustering methods, we compare with Mini-Batch K -Means (mKMC) [54], a scalable variant of K-Means.

Evaluation Metrics As all of our utilized datasets come with known labels, we assess the clustering results using three external evaluation measures: Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and Unsupervised Clustering Accuracy (ACC). We also report the Inertia (Iner.) normalized by the number of datapoints in the dataset. For our fuzzy models (NeurCAM and SCT), we calculate all metrics using the hard clustering decision obtained by selecting the cluster with the highest fuzzy weight for each data point at inference time.

Rand index (RI) [49] measures agreements between two partitions of the same dataset P_1 and P_2 with each partition representing $\binom{n}{2}$ decisions over all pairs, assigning them to the same or different clusters and is defined as follows:

$$\text{RI}(P_1, P_2) = \frac{a+b}{\binom{n}{2}} \tag{18}$$

Where a is the number of pairs assigned to the same cluster, and b is the number of pairs assigned to different clusters. ARI [24] is a correction for RI based on its expected value:

$$\text{ARI} = \frac{\text{RI} - \mathbb{E}(\text{RI})}{\max(\text{RI}) - \mathbb{E}(\text{RI})} \tag{19}$$

An ARI score of zero indicates that the cluster assignment is no better than a random assignment, while a score of 1 indicates a perfect match between the two partitions.

Normalized Mutual Information measures the statistical information shared between distributions [59], normalized by the average entropy of the two distributions. This metric is defined as follows:

$$\text{NMI}(P_1, P_2) = \frac{\text{MI}(P_1, P_2)}{\text{mean}(H(P_1), H(P_2))} \tag{20}$$

Where $H(\cdot)$ is the entropy of a given distribution and $\text{MI}(P_1, P_2)$ is the mutual information between P_1 and P_2.

Unsupervised clustering accuracy [68] measures the best agreement between the cluster label c_n and the ground-truth l_n.

$$\text{ACC} = \max_{map \in M} \frac{1}{N}\sum_{n=1}^{N} \mathbb{1}\{l_n = map(c_n)\} \tag{21}$$

M is the set of all possible one-to-one mappings from clusters to ground-truth labels.

Inertia is defined as the sum of the squared distances from the representation of the datapoints and the centroids of the cluster they are assigned to:

$$\text{Inertia} = \sum_{n=1}^{N}\sum_{k=1}^{K} w_{n,k}||\tilde{\boldsymbol{x}}_n - \boldsymbol{z}_k|| \tag{22}$$

$$w_{n,k} \in \{0,1\},\ \sum_{k=1}^{K} w_{n,k} = 1\ \forall n = 1,2,\ldots,N \tag{23}$$

Training and Implementation Details All models and benchmarks are implemented in Python. The SCT and NeurCAM are implemented in PyTorch [44] and we utilize the Mini-Batch K-Means implementation found in the Scikit-Learn package [45]. For TAO, the implementation from the original paper is not publicly available and we implemented the approach following the details outlined by Gabidolla and Carreira-Perpiñán [18].

NeurCAM is trained using the Adam [31] optimizer with plateau learning rate decay. For Neur2CAM, the pairwise gate temperature parameter T_2 is fully annealed before T starts its annealing procedure. To initialize the centroids $\tilde{\boldsymbol{z}}$, we utilize centroids obtained from Mini-Batch K-Means clustering. Training details and hyperparameters for NeurCAM and the other approaches can be found in the supplementary materials [64].

As our model and all benchmarks may converge to a locally optimum solution, we perform five runs with different random seeds and select the run with the lowest Inertia value for the hard clustering.

Extracting Shape Graphs: To extract the final shape graphs for each feature and interaction in NeurCAM, we query the predictions from each inidividual shape function and then combine the shape functions that have selected the same feature (or interaction) to obtain the final shape graphs:

$$f_{i,k}(\boldsymbol{x}) = \sum_{c=1}^{C} \mathbb{1}_i(\boldsymbol{F}_c) f_{c,k}(\boldsymbol{x}) \tag{24}$$

$$f_{i,j,k}(\boldsymbol{x}) = \sum_{p=C+1}^{C+P} \mathbb{1}_{i,j}(\boldsymbol{F}_{p,0}, \boldsymbol{F}_{p,1}) f_{p,k}(\boldsymbol{x}) \tag{25}$$

Here $\mathbb{1}_i(\cdot)$ is an indicator function on whether feature i is selected and $\mathbb{1}_{i,j}(\cdot,\cdot)$ is an indicator function on whether both features i and j were selected in the selection vectors (regardless of the order). Following Agarwal et al. [1] and Radenovic et al. [48], we set the average cluster activation (logits) of each feature's shape function to zero by subtracting the mean activation. For the pairwise shape graphs of Neur2CAM, we adopt GA^{2}M purification to push interaction effects into main effects if possible [35]. To derive feature-importance from our model, we follow Lou et al. [37] and take the average absolute area under the shape graph.

5.2 Clustering Tabular Data

We demonstrate the ability of our model to create high-quality cluster assignments on tabular tasks by testing it on six datasets from the UCI repository [30]: Adult, Avila, Gas Drift, Letters, Pendigits, and Shuttle. All datasets were standardized by removing the mean and scaling to unit variance. Information about these datasets can be found in our supplementary materials [64].

In this set of experiments we set the number of single-feature shape functionss equal to the number of features. For Neur2CAM, we set the number of pairwise shape functions equal to the number of single-feature shape functions to maintain a high degree of interpretability. We set $m = 1.05$ for our loss function (Equation (15)).

Representations Utilized: For our interpretable representation $\boldsymbol{x}$, we utilize the original feature space provided by the datasets. For the representation in our loss function $\tilde{\boldsymbol{x}}$, we consider two different deep transformations:

- Denoising AutoEncoder (DAE): We cluster on the embeddings from a pre-trained DAE [65] with dropout corruption and a bottleneck of size 8.
- SpectralNet (Spectral): We cluster on embeddings from a SpectralNet [55], a deep-learning based approximation of Spectral Clustering [66]. The embedding dimension is equal to the number of clusters in the dataset.

NeurCAM, the SCT, and TAO make clustering decisions using the original, interpretable feature space, while Mini-Batch K-Means directly clusters in the transformed space.

5.2.1 Results

Table 1 provides a summary of the comparison between our approach and our baselines on the tabular data sets. A detailed breakdown of the results by dataset can be found in the supplementary materials [64].

We observe that our approaches on SpectralNet embeddings consistently outperforms baselines in average rank on external metrics across all datasets. This trend remains consistent when looking at the average value across data sets, as well.

When considering Inertia for SpectralNet embeddings, we achieve the best average value and average rank across the interpretable methods. Although TAO-7 is able to obtain a slightly lower Inertia value on DAE embeddings, we still achieve a lower average rank, indicating that our approaches result in lower Inertia more often.

5.3 Clustering Text Data

Our approach can be utilized for any task where a human-understandable tabular representation can be extracted. We demonstrate this ability by extending our approach to perform text clustering on 4 text datasets: AG News[70], DBPedia [3], 20 Newsgroups [32], and Yahoo Answers [70]. For AG News, DBPedia, and Yahoo, we follow the approach taken in the previous literature [60, 67] and sample 1,000 points from each class.

To create our interpretable representation, we remove the punctuation, lowercase, and tokenize all datapoints. We then lemmatize all tokens using the WordNet [42] lemmatizer available in the Natural Language Toolkit (NLTK) [7] and remove English stopwords, corpus-specific stopwords that appear in more than 99% of the datapoints, and rare words that appear in less than 1% of the documents. Finally, we then calculate term-frequency in each datapoint and normalize the representation so that the L_2 norm of the resultant vector is equal to 1.0.

For the representation in our loss function (Equation (15)), we leverage embeddings from the MPNet pretrained transformer [58] available in the Sentence Transformers package [50]. This model has a representation size of 768, a maximum sequence length of 384, and uses mean pooling to construct its embedding.

To retain model sparsity, we opt to only use 128 single feature shape functions and 128 pairwise shape functions (when applicable). We set $m = 1.025$ for our loss function (Equation (15)).[1]

[1] On the text datasets, we empirically observed that $m = 1.05$ resulted in

Table 1. Summarized results from our tabular clustering experiments. The best overall results are highlighted in bold, while the best interpretable results are underlined. The dashed line separates interpretable and non-interpretable approaches.

		ARI ↑		NMI ↑		ACC ↑		Iner. ↓	
		AE	Spectral	AE	Spectral	AE	Spectral	AE	Spectral
Average Scores	NCAM	0.192	0.261	0.303	0.385	0.524	0.561	2.509	1.995
	N2CAM	0.190	**0.272**	0.295	**0.394**	0.517	**0.568**	2.399	1.994
	TAO-5	0.185	0.240	0.287	0.347	0.521	0.515	2.960	4.181
	TAO-7	0.189	0.257	0.291	0.371	0.526	0.518	**2.315**	2.517
	SCT-5	0.159	0.190	0.276	0.314	0.452	0.479	3.552	5.460
	SCT-7	0.157	0.198	0.267	0.315	0.469	0.476	3.416	4.443
	mKMC	0.211	0.250	0.301	0.380	0.512	0.520	2.682	**1.579**
Average Rank	NCAM	9.50	4.17	8.17	3.33	7.00	4.67	2.83	3.17
	N2CAM	8.83	**3.00**	7.83	**3.00**	7.67	**3.50**	2.67	3.50
	TAO-5	9.67	6.50	9.50	5.83	6.83	7.83	4.67	4.50
	TAO-7	9.50	5.67	9.50	4.33	7.00	7.00	3.00	3.17
	SCT-5	9.83	6.00	10.83	6.33	9.83	9.50	6.50	6.50
	SCT-7	11.33	8.50	12.00	7.33	10.17	9.67	5.83	5.33
	mKMC	7.00	4.00	8.17	3.33	7.33	6.00	**2.33**	**1.67**

Table 2. Summarized results from our text clustering experiments. The best overall results are highlighted in bold, while the best interpretable results are underlined. The dashed line separates interpretable and non-interpretable approaches.

		ARI ↑	NMI ↑	ACC ↑	Iner. ↓
Average Values	NCAM	0.359	0.463	0.584	696.717
	N2CAM	0.455	0.546	0.625	686.311
	TAO-5	0.052	0.236	0.286	737.509
	TAO-7	0.081	0.272	0.346	730.358
	SCT-5	0.127	0.233	0.333	738.614
	SCT-7	0.134	0.224	0.320	737.754
	mKMC	**0.491**	**0.570**	**0.659**	**674.071**
Average Rank	NCAM	3.00	3.00	2.75	3.0
	N2CAM	1.75	**1.50**	2.00	2.0
	TAO-5	6.25	6.00	6.00	6.0
	TAO-7	5.25	4.75	4.75	4.5
	SCT-5	5.50	5.00	5.50	6.0
	SCT-7	5.00	6.25	5.75	5.5
	mKMC	**1.25**	**1.50**	**1.25**	**1.0**

5.3.1 Results

Table 2 provides a summary of the comparison between our approaches and our baselines on the text datasets. A detailed breakdown of the results by dataset can be found in the supplementary materials [64].

We observe that our approaches significantly outperform the interpretable baselines. Neur2CAM and NeurCAM obtain the first and second place, respectively, when considering the interpretable models across both external and internal metrics in both average value and average rank. More concretely, NeurCAM has a 167.9% higher ARI than the SCT of depth seven and a 4.6% lower Inertia than TAO with depth seven. The introduction of pairwise interactions makes this gap even more drastic, with Neur2CAM having both a higher average ARI (26. 6% higher) and a lower average Inertia (1.5% lower) than NeurCAM.

As expected, given the restrictions we imposed on our feature space, our interpretable approaches do not outperform the black-box model. However, despite significantly limiting the number of terms and interactions used,[2] we were able to obtain more than 92% of the performance of the black-box model (Mini-Batch K-Means) across all external evaluation metrics. More specifically, Neur2CAM achieves an ARI equal to 92.7% of the ARI, 95. 9% of the NMI and 94. 8% of the ACC achieved by Mini-Batch K-Means. Furthermore, Neur2CAM achieves an average Inertia value within 1.8% of Mini-Batch K-Means.

degenerate solutions, therefore a smaller m value was used.

[2] We use a maximum of 128 interactions out more than 50,000 possible.

5.4 Ablation Analysis

Table 3. Results of our ablation analysis. Full is our approach with both terms in the loss function. No CL denotes ablating the clustering loss term and No KL denotes ablating the KL-Divergence term.

		NeurCAM		Neur2CAM	
		Loss	Iner.	Loss	Iner.
Average	Full	**83.454**	**696.717**	**82.646**	**686.311**
	No CL	84.254	698.432	83.012	686.824
	No KL	83.568	702.358	83.031	691.754

To demonstrate the benefit of the loss terms used in the second phase of our training, we introduce two ablations of NeurCAM. The first ablation removes the clustering loss (Equation (15)) in the second phase of training, making NeurCAM to optimize only the KL divergence between NeurCAM and its relaxation. The second ablation keeps the clustering loss term, but instead ablates the KL-Divergence loss (Equation (16)) so that NeurCAM only optimizes the clustering loss throughout its training. We perform this analysis on our text datasets and report both the loss (Equation (15)) and the Inertia of the run that achieves the minimum Inertia across five seeds. We observe that including both terms in our loss function consistently results in lower values across both metrics.

5.5 Interpretability of NeurCAM

In this section, we highlight the interpretability of NeurCAM and connect it back to the model-based interpretability desiderata presented in Section 2.2.

5.5.1 Controllable Sparsity

NeurCAM allows users to explicitly control the number of features and interactions used to construct the clusters through our selection

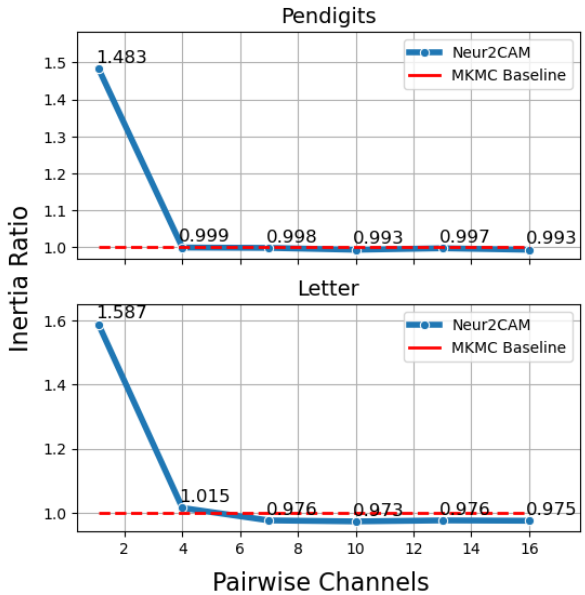

Figure 2. Cost Ratio of Neur2CAM compared to MKMC as more pairwise shape functions are added.

gate mechanism (Equations (9) and (13)). In many cases, users can significantly limit the number of shape functions used, improving the model sparsity while still generating high-quality clusters. To demonstrate this capability, we ablate the number of selection gates utilized. We report the ratio between the Inertia of Neur2CAM (minimum across five random seeds) and the Inertia of Mini-Batch K-Means (also across five random seeds) and plot the cost ratio as we vary the number of pairwise shape functions. To isolate the impact of using interactions, we set the number of single feature shape functions to zero for this experiment. In Figure 2, we observe that we are able to achieve a loss value comparable to Mini-Batch K-Means with only 4 shape functions, or at most 4 pairwise interactions, when clustering on both the Pendigits and Letters dataset.

5.5.2 *Modularity and Simulability*

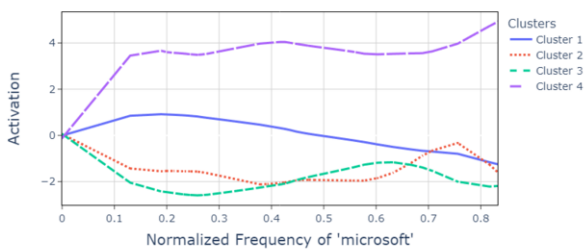

Figure 3. Shape Graphs for "Microsoft" learned when clustering the AG News Dataset

The learned shape functions for each feature and interaction in NeurCAM are *modular* and can be independently analyzed, allowing stakeholders to understand the impact individual features have on the end prediction as well as *simulate* predictions by querying the different shape functions. As an example, Figure 3 we displays the shape function learned by NeurCAM when clustering the AG News Text Dataset for the term "Microsoft". When examining this shape graph, we can see that when Microsoft is present in a sample, the network allocates more weight to Cluster 4 and reduces the weight to the other clusters. When examining the alignment between the clusters and the ground-truth labels, we observed that cluster 4 consists primarily of news titles related to technology, indicating that the presence of Microsoft in a sample results in NeurCAM mapping the sample to the "technology" cluster. A full example of NeurCAM's explanations can be found in our supplementary material [64].

6 Conclusion

In this work, we present NeurCAM, the first approach to interpretable clustering that uses neural GAMs. Our experiments showcase our ability to produce high-quality clusters on tabular data that are comparable to black-box approaches while using a limited number of features and interactions chosen by our proposed selection gate mechanism. We also demonstrate NeurCAM's ability to utilize deep representations of the data while still providing explanations in a human interpretable feature space on both tabular and text datasets.

NeurCAM is a powerful tool for large-scale clustering tasks for knowledge discovery and can be used as a foundation for further research on interpretable clustering via Generalized Additive Models. Potential extensions of our work involve the joint performance of clustering and representation learning by integrating approaches such as DEC / IEC [68, 20] or VADE [26]. Furthermore, other possible directions include designing specialized GAMs to focus on learned interpretable features from images, such as prototypes [12], or time series, such as shapelets [36].

References

[1] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton. Neural additive models: Interpretable machine learning with neural nets. *NeurIPS*, pages 4699–4711, 2021.

[2] A. Aouad, A. N. Elmachtoub, K. J. Ferreira, and R. McNellis. Market segmentation trees. *M&SOM*, 25(2):648–667, 2023.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC*, pages 722–735, 2007.

[4] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7), 2015.

[5] D. Bertsimas, A. Orfanoudaki, and H. Wiberg. Interpretable clustering: an optimization approach. *Machine Learning*, 110:89–138, 2021.

[6] J. C. Bezdek, R. Ehrlich, and W. Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984.

[7] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[8] E. Carrizosa, K. Kurishchenko, A. Marín, and D. R. Morales. Interpreting clusters via prototype optimization. *Omega*, 107:102543, 2022.

[9] C.-H. Chang, S. Tan, B. Lengerich, A. Goldenberg, and R. Caruana. How interpretable and trustworthy are gams? In *KDD*, pages 95–105, 2021.

[10] C.-H. Chang, R. Caruana, and A. Goldenberg. Node-gam: Neural generalized additive model for interpretable deep learning. In *ICLR*, 2022.

[11] M.-C. Chang, P. Bus, and G. Schmitt. Feature extraction and k-means clustering approach to explore important features of urban identity. In *ICMLA*, pages 1139–1144. IEEE, 2017.

[12] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. This looks like that: deep learning for interpretable image recognition. In *NeurIPS*, 2019.

[13] J. Chen, Y. Chang, B. Hobbs, P. Castaldi, M. Cho, E. Silverman, and J. Dy. Interpretable clustering via discriminative rectangle mixture model. In *IEEE ICDM*, pages 823–828, 2016.

[14] E. Cohen. Interpretable clustering via soft clustering trees. In *CPAIOR*, pages 281–298, 2023.

[15] R. Fraiman, B. Ghattas, and M. Svarc. Interpretable clustering using unsupervised binary trees. *ADAC*, 7:125–145, 2013.

[16] N. Frost, M. Moshkovitz, and C. Rashtchian. Exkmc: Expanding explainable k-means clustering. *arXiv preprint arXiv:2006.02399*, 2020.

[17] R. Fu, W. Li, J. Chen, and M. Han. Recognizing single-trial motor imagery eeg based on interpretable clustering method. *Biomedical Signal Processing and Control*, 63:102171, 2021.

[18] M. Gabidolla and M. Á. Carreira-Perpiñán. Optimal interpretable clustering using oblique decision trees. In *KDD*, pages 400–410, 2022.

[19] R. Guan, H. Zhang, Y. Liang, F. Giunchiglia, L. Huang, and X. Feng. Deep feature-based text clustering and its explanation. *IEEE TKDE*, 34 (8):3669–3680, 2020.

[20] X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *IJCAI*, pages 1753–1759, 2017.

[21] T. Han, S. Srinivas, and H. Lakkaraju. Which explanation should i choose? a function approximation perspective to characterizing post hoc explanations. In *NeurIPS*, pages 5256–5268, 2022.

[22] T. J. Hastie. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

[23] S. Hegselmann, T. Volkert, H. Ohlenburg, A. Gottschalk, M. Dugas, and C. Ertmer. An evaluation of the doctor-interpretability of generalized additive models with interactions. In *MLHC*, pages 46–79, 2020.

[24] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

[25] S. Ibrahim, G. Afriat, K. Behdin, and R. Mazumder. Grandslamin'interpretable additive modeling with structural constraints. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou. Variational deep embedding: an unsupervised and generative approach to clustering. In *IJCAI*, pages 1965–1972, 2017.

[27] T. Karatekin, S. Sancak, G. Celik, S. Topcuoglu, G. Karatekin, P. Kirci, and A. Okatan. Interpretable machine learning in healthcare through generalized additive model with pairwise interactions (ga2m): Predicting severe retinopathy of prematurity. In *Deep Learning and Machine Learning in Emerging Applications*, pages 61–66, 2019.

[28] J. Kauffmann, M. Esders, L. Ruff, G. Montavon, W. Samek, and K.-R. Müller. From clustering to cluster explanations via neural networks. *IEEE TNNLS*, 2022.

[29] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. Wortman Vaughan. Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning. In *CHI*, pages 1–14, 2020.

[30] M. Kelly, R. Longjohn, and K. Nottingham. The uci machine learning repository. https://archive.ics.uci.edu, 2017.

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

[32] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[33] C. Lawless and O. Gunluk. Cluster explanation via polyhedral descriptions. In *ICML*, pages 18652–18666, 2023.

[34] C. Lawless, J. Kalagnanam, L. M. Nguyen, D. Phan, and C. Reddy. Interpretable clustering via multi-polytope machines. In *AAAI*, pages 7309–7316, 2022.

[35] B. Lengerich, S. Tan, C.-H. Chang, G. Hooker, and R. Caruana. Purifying interaction effects with the functional anova: An efficient algorithm for recovering identifiable additive models. In *AISTATS*, pages 2402–2412, 2020.

[36] J. Lines, L. M. Davis, J. Hills, and A. Bagnall. A shapelet transform for time series classification. In *KDD*, pages 289–297, 2012.

[37] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In *KDD*, pages 623–631, 2013.

[38] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker. Accurate intelligible models with pairwise interactions. In *KDD*, pages 623–631, 2013.

[39] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *NeurIPS*, 2017.

[40] M. Luštrek, M. Gams, S. Martinčić-Ipšić, et al. What makes classification trees comprehensible? *Expert Systems with Applications*, 62: 333–346, 2016.

[41] L. Manduchi, M. Hüser, M. Faltys, J. Vogt, G. Rätsch, and V. Fortuin. T-dpsom: An interpretable clustering method for unsupervised learning of patient health states. In *CHIL*, page 236–245, 2021.

[42] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[43] W. J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Definitions, methods, and applications in interpretable machine learning. *National Academy of Sciences*, 116(44):22071–22080, 2019.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, pages 8024–8035, 2019.

[45] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *JMLR*, 12:2825–2830, 2011.

[46] X. Peng, Y. Li, I. W. Tsang, H. Zhu, J. Lv, and J. T. Zhou. Xai beyond classification: Interpretable neural clustering. *JMLR*, 23(1):227–254, 2022.

[47] B. Peters, V. Niculae, and A. F. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, 2019.

[48] F. Radenovic, A. Dubey, and D. Mahajan. Neural basis models for interpretability. In *NeurIPS*, pages 8414–8426, 2022.

[49] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[50] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP*, 11 2019.

[51] C. Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[52] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16, 2022.

[53] A. Sarica, A. Quattrone, and A. Quattrone. Explainable boosting machine for predicting alzheimer's disease from mri hippocampal subfields. In *BI*, pages 341–350, 2021.

[54] D. Sculley. Web-scale k-means clustering. In *WWW*, pages 1177–1178, 2010.

[55] U. Shaham, K. Stanton, H. Li, B. Nadler, R. Basri, and Y. Kluger. Spectralnet: Spectral clustering using deep neural networks. In *ICLR*, 2018.

[56] P. Shati, E. Cohen, and S. McIlraith. Optimal decision trees for interpretable clustering with constraints. In *IJCAI*, pages 2022–2030, 2023.

[57] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *AIES*, pages 180–186, 2020.

[58] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. *NeurIPS*, pages 16857–16867, 2020.

[59] A. Strehl and J. Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *JMLR*, 3(Dec):583–617, 2002.

[60] A. Subakti, H. Murfi, and N. Hariadi. The performance of bert as data representation of text clustering. *Journal of big Data*, 9(1):1–21, 2022.

[61] K. Sun, T. Lan, Y. M. Goh, S. Safiena, Y.-H. Huang, B. Lytle, and Y. He. An interpretable clustering approach to safety climate analysis: Examining driver group distinctions. *Accident Analysis & Prevention*, 196: 107420, 2024.

[62] J. Svirsky and O. Lindenbaum. Interpretable deep clustering for tabular data. In *Forty-first ICML*, 2024.

[63] S. Tan, G. Hooker, P. Koch, A. Gordo, and R. Caruana. Considerations when learning additive explanations for black-box models. *Machine Learning*, pages 1–27, 2023.

[64] N. Upadhya and E. Cohen. NeurCAM: Interpretable Neural Clustering via Additive Models. *arXiv preprint arXiv:2408.13361*, 2024.

[65] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[66] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

[67] Z. Wang, H. Mi, and A. Ittycheriah. Semi-supervised clustering for short text via deep representation learning. In *CoNLL*, 2016.

[68] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *ICML*, pages 478–487, 2016.

[69] H. Yang, L. Jiao, and Q. Pan. A survey on interpretable clustering. In *2021 40th Chinese Control Conference (CCC)*, pages 7384–7388. IEEE, 2021.

[70] X. Zhang, J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. *NeurIPS*, 2015.