SigmaCam: Exact Decision Boundary Extraction for DNNs with Smooth Nonlinearities

Nikkat Afrin[†], Mohammadzubair Khan[†], Likun Xia[‡], Yading Yuan[§] and Ming Ma[†*]
[†]Department of Graduate Computer Science and Engineering, Katz School of Science and Health
Yeshiva University, New York, NY, USA
[‡]Capital Normal University, Beijing, China
[§]Department of Radiation Oncology
Columbia University Irving Medical Center, New York, NY, USA

***Abstract*—Understanding how a trained deep neural network (DNN) classifies input data is critical for interpretability and trust in AI systems. Existing tools such as SplineCam can visualize theoretically exact decision boundaries for neural networks with piecewise polynomial activation functions. However, these methods often struggle with more commonly used smooth activations such as Sigmoid, SiLU, and their combinations. In this paper, we introduce a novel framework termed SigmaCam using a recursive algorithm that is both computationally efficient and capable of generating theoretically exact 2D decision boundaries for any Multi-Layer Perceptron (MLP) employing smooth activations. Our approach extends previous works to a broader class of activation functions and allows for the visualization of decision boundaries on the input domain of any dimension, analogous to the capabilities of SplineCam, but now applicable to widely used smooth nonlinearities. We demonstrate the effectiveness of our method by applying it to a variety of network architectures, activation function combinations (e.g., Sigmoid and SiLU), and datasets. The experiments show that our method can accurately capture complex decision boundaries, providing insights into model behavior and aiding in model interpretability. The proposed algorithm enables the analysis of networks that were previously difficult to handle, broadening the scope of exact decision boundary visualization tools beyond piecewise polynomial activations.**

***Index Terms*—Decision boundaries, deep neural networks, interpretability, Sigmoid, SiLU, recursive algorithms, activation functions.**

I. Introduction

Understanding the decision-making process of neural networks is vital for improving model interpretability, trust, and transparency in artificial intelligence systems. Neural networks are often considered "black boxes" due to the complexity of their internal decision-making mechanisms, leading to challenges in explaining their predictions. Visualization of decision boundaries provides a powerful tool for interpreting the behavior of neural networks and understanding how they partition input space for classification.

Recent methods, such as SplineCam [1], have provided exact decision boundary visualization capabilities for networks with piecewise polynomial activation functions. These tools rely on mathematical properties of spline-based activations to construct decision boundaries precisely. However, SplineCam's reliance on Continuous Piece-Wise Linear (CPWL) properties restricts its applicability in some scenarios. State-of-the-art models predominantly utilize smooth nonlinear activations such as Sigmoid [2], SiLU [3], and Swish [4], which pose additional challenges for decision boundary extraction due to their continuous and differentiable nature. Existing visualization techniques often resort to approximations, which may not capture subtle details of decision boundaries, particularly for networks with complex architectures and high-dimensional inputs. For example, the synthetic toroidal dataset provides a challenging scenario for the decision boundary extraction due to inherent nonlinear structures and high-dimensional nature, as shown in Figure 1.

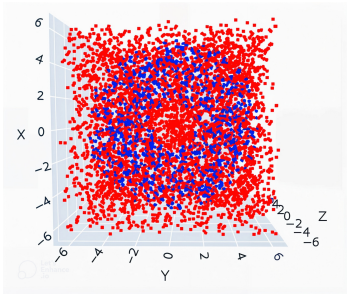

Fig. 1. Visualization of a 3D toroidal dataset, showcasing its intricate structure, which presents challenges for decision boundary extraction.

Our work focuses on three activation functions: ReLU, Sigmoid, and SiLU. ReLU has historically been favored for its simplicity, efficiency, and compatibility with methods such as SplineCam that rely on piecewise-linear properties. In contrast, smooth activations such as Sigmoid and SiLU have gained traction because of their natural probabilistic outputs and improved gradient flow [4]. Although many alternatives (e.g., GELU, Mish, ELU variants, SELU) have emerged since 2017, we contend that the core properties needed for robust decision boundary extraction are already well captured by this select group. ReLU's piecewise-linearity facilitates theoretical analysis, while Sigmoid and SiLU (also known as Swish when its parameter is fixed to 1) provide smooth gradient propagation and reliable output calibration. A recent survey confirms that only a few activation functions offer the unique combination of attributes required for challenging tasks such as exact decision boundary extraction [5].

In this work, we propose a novel framework termed Sig-

*Corresponding author: ming.ma@yu.edu

maCam based on a recursive algorithm designed to address these limitations by enabling exact decision boundary extraction for Multi-Layer Perceptrons (MLPs) with smooth nonlinear activation functions. In essence, SigmaCam works by projecting the input space onto a two-dimensional manifold, creating a structured grid within this space, and then mapping these points back into the original high-dimensional domain. By recursively computing network activations for these grid points and evaluating against classification thresholds, SigmaCam precisely extracts and visualizes decision boundaries. Our approach extends the theoretical foundations of methods such as SplineCam to cover a broader class of activation functions, offering improved generality and precision. Unlike heuristic-based approaches [6], our method guarantees exactness in identifying the points in the input space where classification decisions change, thereby providing a more accurate representation of the model's decision regions.

In this paper, our main contributions are as follows:

- We propose a novel framework termed SigmaCam for exact decision boundary extraction in DNN with smooth nonlinear activation functions, including Sigmoid and SiLU.
- We introduce a unified and computationally efficient recursive algorithm, leveraging matrix-based operations to handle complex networks and high-dimensional inputs. The algorithm accommodates smooth activations, and incorporates geometric techniques, high-dimensional embeddings, and disentangled representations, thereby enhancing interpretability and scalability.
- We validate our approach through extensive experiments on synthetic datasets and three different real-world datasets (including both nature images and medical images), demonstrating its superior capability to accurately recover intricate decision boundaries.

II. Related Work

A. Decision Boundary Visualization Methods

Decision boundary visualization has advanced significantly to enhance AI interpretability. Early approaches utilized sampling techniques and gradient-based heuristics to approximate boundaries in high-dimensional spaces [7], [8]. These foundational methods, however, struggled with accurately capturing complex geometries, particularly in high-dimensional or nonlinear datasets where dimensionality reduction often led to information loss [8]. Discriminative dimensionality reduction techniques were introduced to address this by shaping projections based on class labels, improving visualization but lacking robustness across diverse real-world data distributions [9].

B. Feature Extraction and Exact Boundary Methods

Initial work on feature extraction focused on decision boundaries, demonstrating that boundary-based feature matrices could reduce dimensionality without sacrificing classification accuracy [7]. Nonetheless, these methods were limited to parametric classifiers and became computationally infeasible for larger neural networks. SplineCam [1] marked a significant advancement by enabling exact decision boundary visualization in networks with continuous piecewise-linear (CPWL) activations such as ReLU and leaky ReLU. It precisely computed boundary geometries without approximations, facilitating insights into network generalization and architecture choices. However, SplineCam's reliance on CPWL properties restricted its applicability to networks with smooth nonlinear activations such as Sigmoid and SiLU, which introduce more complex boundary structures [10].

C. Non-linear Dimensionality Reduction and Dynamic Visualization

To preserve class boundary topology, recent methods integrate non-linear dimensionality reduction techniques like t-SNE and Fisher metrics [8]. These approaches enhance class separability and compactness in low-dimensional spaces, improving visualization robustness against data distribution variations [9]. Image-based visualization tools [11], [12] project high-dimensional inputs onto 2D planes using t-SNE or PCA, enabling intuitive observation of decision regions, confusion zones, and class overlaps. Rodrigues et al. [11] introduced dense mapping with color-coded outputs to depict confidence and separations, though reliant on the fidelity of dimensionality reduction algorithms [13].

Dynamic visualization tools such as PaletteViz [14] and latent space cartography [15] facilitate interactive exploration of high-dimensional data. PaletteViz employs Pareto optimization to reveal trade-offs in decision-making, while latent space cartography maps semantic dimensions within embeddings, bridging model representations with interpretable decision frameworks.

D. Interactive and User-centric Visualization Tools

Interactive platforms such as ActiVis [16] and the What-If Tool (WIT) [17] enhance user engagement by allowing dynamic exploration of neural networks. ActiVis visualizes model architectures and neuron activations for instance and subset-level analysis, whereas WIT supports hypothetical scenario testing, counterfactual generation, and fairness evaluations, fostering transparency and trust in AI systems. Additionally, tools such as ilastik [18] and JAABA [19] empower domain experts to interactively train classifiers for specific tasks, highlighting the importance of human-in-the-loop approaches in refining decision-making processes [20].

E. Advances in Visualization and Interpretability

Recent work has integrated visualization with interpretability through methods such as counterfactual reasoning [21], decision boundary maps [22], and robust explainability frameworks [23]. Techniques such as CROP [24] utilize continuous piecewise-affine properties to enforce robust boundaries. Our recursive algorithm extends these efforts by accommodating smooth activations, incorporating geometric techniques [25], high-dimensional embeddings [26], and disentangled representations [27], thereby enhancing interpretability and scalability [28]. Additionally, approaches such as Affinitree [29] and

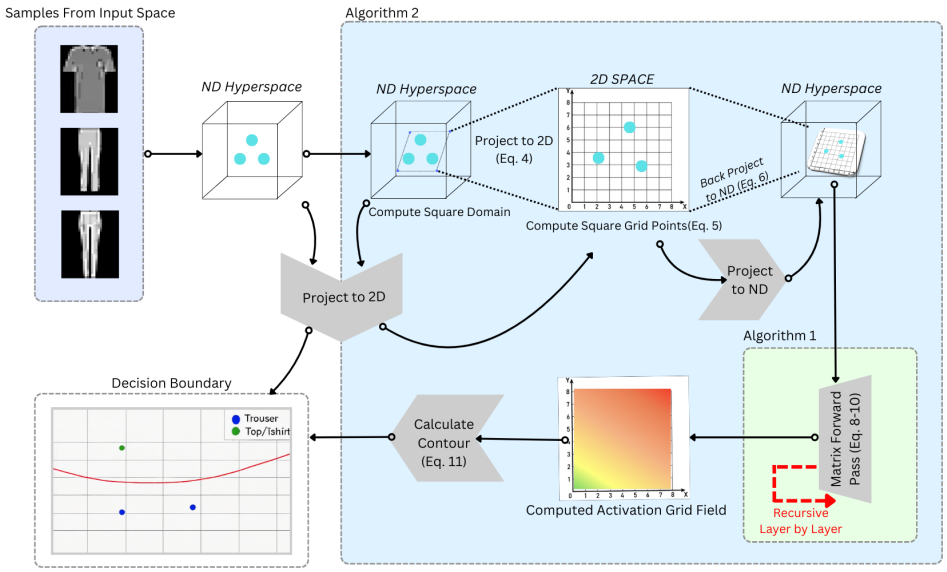

Fig. 2. Pipeline of the proposed method, detailing the stages from data preparation to decision boundary extraction. Each step highlights the respective tasks, including input initialization, projection, grid generation, matrix forward pass, decision boundary evaluation, and final visualization.

GIME [30] demonstrate the critical role of decision boundary analysis in understanding model behavior, further supported by neural network visualization [13] and initialization studies [31]. Despite progress, challenges in scalability, precision, and computational efficiency remain, particularly for large-scale or high-dimensional models. The computational demands of t-SNE-based projections [11] and the resolution limitations of heatmap techniques [32] restrict real-time application utility. Moreover, qualitative insights from tools such as Palette-Viz [14] often lack robust quantitative validation [13].

III. Methodology

A. Activation Functions

Activation functions play a crucial role in the expressiveness and learning capability of neural networks. They introduce nonlinearity into the model, enabling the network to learn complex patterns in the data. In this study, we focus on three widely used activation functions: Rectified Linear Unit (ReLU), Sigmoid, and Sigmoid Linear Unit (SiLU).

1) Rectified Linear Unit (ReLU): The ReLU activation function is defined as:

$$\text{ReLU}(x) = \max(0, x) \tag{1}$$

ReLU activates a neuron only if the input is positive; otherwise, it outputs zero. This sparsity in activation helps in reducing computational complexity and improving convergence rates during training.

2) Sigmoid: The Sigmoid function maps any real-valued input to a value between 0 and 1, making it suitable for binary classification tasks. It is mathematically expressed as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2}$$

While Sigmoid functions provide smooth gradients, they are prone to the vanishing gradient problem, especially in deep networks, which can hinder the learning process.

3) Sigmoid Linear Unit (SiLU): SiLU, also known as the Swish activation function, combines the properties of ReLU and Sigmoid to offer a smoother and more flexible activation. It is defined as:

$$\text{SiLU}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \tag{3}$$

SiLU allows small negative values, which can improve the flow of gradients during training and enhance the model's ability to capture complex relationships in the data.

B. Overview of SigmaCam

Figure 2 presents the comprehensive pipeline of the SigmaCam methodology for exact decision boundary extraction in deep neural networks with smooth nonlinearities. The process initiates with the initialization of the high-dimensional input domain, which is then projected onto a two-dimensional plane using a projection matrix. A structured grid is generated within this 2D space and subsequently back-projected into the original high-dimensional space. These grid points undergo a matrix-based forward pass through the trained Multilayer Perceptron (MLP), where activations are computed recursively at each layer. The final activations are evaluated against a predefined threshold to delineate the decision boundary. The extracted boundary is visualized as a clear contour on the 2D projection, effectively illustrating the model's learned classification regions. This pipeline underscores the efficiency and precision of SigmaCam in mapping complex decision boundaries within deep neural architectures.

Given a high-dimensional input domain $\mathcal{D} \subset \mathbb{R}^d$, we define the 2D projection as:

$$\mathcal{D}_{2D} = P(\mathcal{D} - x_0), \tag{4}$$

where $P \in \mathbb{R}^{2\times d}$ is the projection matrix, and $x_0 \in \mathbb{R}^d$ is the centroid of the data. The structured grid $\mathcal{G}_{2D}$ within the 2D projection is generated as:

$$(X, Y) = \text{meshgrid}(x_{\text{range}}, y_{\text{range}}), \tag{5}$$

where (X, Y) represent grid coordinates in 2D. These grid points are then back-projected into the original d-dimensional space as:

$$\mathcal{G} = P^{\dagger}(X, Y) + x_0, \tag{6}$$

where $P^{\dagger}$ denotes the pseudo-inverse of P. These back-projected points are subsequently fed into the neural network for boundary computation.

C. Recursive Decision Boundary Computation

Consider an L-layer multilayer perceptron (MLP) where layer l transforms its input $\mathbf{z}^{(l-1)}$ (with $\mathbf{z}^{(0)} \equiv \mathbf{x}$ representing the input) according to the following equation:

$$\mathbf{z}^{(l)} = \sigma(W^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}), \tag{7}$$

where $\sigma(\cdot)$ denotes a smooth nonlinear activation function, such as Sigmoid or SiLU.

To determine the decision boundary for a binary classification task, we identify all $\mathbf{x} \in \mathbb{R}^d$ satisfying $f(\mathbf{x}) = \tau$, where $f(\mathbf{x})$ represents the network output (typically a scalar probability) and τ is a threshold value, commonly set to $\tau = 0.5$. Unlike existing methods that rely on inverting functions or analytically solving equations at each layer, our approach propagates matrices forward through the network layers, thereby enabling efficient computation of decision boundaries.

The visual progression of the algorithm is illustrated in Figure 3. Each stage—from initializing the input domain to projecting high-dimensional decision boundaries onto interpretable two-dimensional planes—is depicted. These visuals demonstrate how structured grids and recursive computations facilitate the exact extraction of decision boundaries, even in complex models.

Explicitly, given an input $\mathbf{x} = \mathbf{z}^{(0)}$, activations for an L-layer neural network are recursively computed by:

$$\mathbf{z}^{(1)} = \sigma(W^{(1)}\mathbf{x} + b^{(1)}) \tag{8}$$

$$\mathbf{z}^{(2)} = \sigma(W^{(2)}\mathbf{z}^{(1)} + b^{(2)}) \tag{9}$$

$$\vdots$$

$$\mathbf{z}^{(L)} = \sigma(W^{(L)}\mathbf{z}^{(L-1)} + b^{(L)}), \tag{10}$$

with decision boundary defined by the set:

$$\{\mathbf{x} \mid \mathbf{z}^{(L)} = \tau\}, \tag{11}$$

where τ is a decision threshold, typically 0.5 for binary classification.

Algorithm 1 Matrix-Based Decision Boundary Extraction

Require: Trained MLP model $\{W^{(l)}, b^{(l)}\}_{l=1}^{L}$, activation σ, grid points $\mathbf{X}$, layer index $li = 0$.
Ensure: Decision boundary values Z for visualization.

```
function ComputeBoundary(X, {W^(l), b^(l)}_{l=1}^L, li)
    if li == L then                          ▷ Check if at last layer
        return X                             ▷ Final output activations
    end if
    Z ← σ(W^(li) X + b^(li))                 ▷ Matrix activation
    return ComputeBoundary(Z, {W^(l), b^(l)}_{l=1}^L, li + 1)
end function
```

D. Algorithms

The decision boundary extraction process involves several key steps, as detailed below:

Step 1: Matrix Forward Pass. Beginning with an input grid of candidate points within the domain, activations at each layer are computed via a matrix forward pass, as described in Algorithm 1:

$$\mathbf{z}^{(l)} = \sigma(W^{(l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}), \tag{12}$$

where $W^{(l)}$ and $\mathbf{b}^{(l)}$ represent the weight matrix and bias vector for layer l, respectively.

Step 2: Recursive Activation Computation. The algorithm recursively computes activations for each subsequent layer until reaching the final output layer. At this stage, the activations are compared against the threshold τ to delineate the decision boundary. This step is detailed in Algorithm 1.

Step 3: Contour Extraction. A two-dimensional projection of the input domain facilitates the visualization of the decision boundary. For each point in the 2D grid, activations are propagated through the network, and the output is evaluated. The decision boundary contour is generated using Algorithm 2, highlighting the model's learned partitions of the input space. The result is visible in the fifth panel of Figure 3 over a synthetic dataset.

Step 4: Efficiency. By utilizing matrix operations and mesh grids, the algorithm efficiently computes forward activations while avoiding unnecessary inversions or redundant calculations.

These steps collectively enable the exact extraction of decision boundaries in deep neural networks with smooth nonlinearities, as depicted in Figure 3. The proposed algorithm efficiently computes decision boundaries for MLPs through a matrix-based forward pass approach, as outlined in Algorithms 1 and 2. By circumventing analytical inversions, the method achieves both scalability and precision.

The centroid x_0 and projection matrix P are computed explicitly as follows. Given training data points $X = \{\mathbf{x}_i\}_{i=1}^{N} \subset \mathbb{R}^d$:

$$x_0 = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i, \quad C = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - x_0)(\mathbf{x}_i - x_0)^T \tag{13}$$

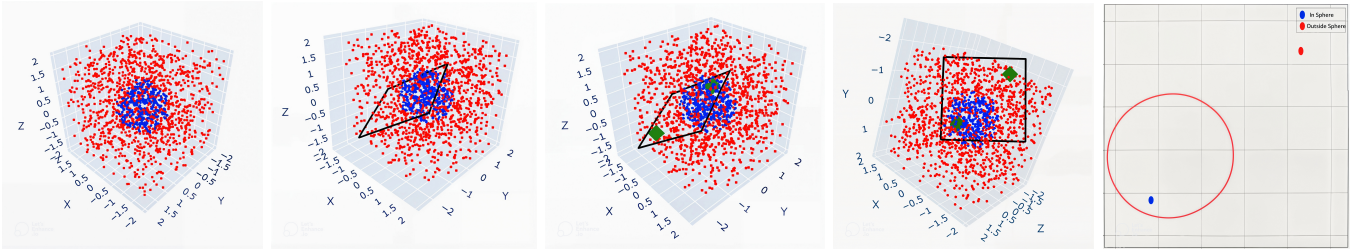

Fig. 3. Stages of decision boundary computation. The first panel displays the labeled dataset, with blue points inside the sphere and red points outside. In the second panel, a square domain is created within the three-dimensional input space, positioned based on the data distribution. The third panel highlights two critical green points used to define the square domain: one green point lies inside the sphere, and the other lies outside. These points are selected to ensure the square domain effectively captures the decision boundary. The fourth panel presents a view normal to the plane of the square domain in the input space, illustrating the black square that encompasses the green points. This square intersects the sphere and delineates the area of interest. Finally, the fifth panel shows the decision boundary cross-section generated by our algorithm, clearly separating the classes within the square domain.

Algorithm 2 2D Decision Boundary Visualization

Require: Input domain $\mathcal{D} \subset \mathbb{R}^d$, projection matrix P, trained model $\{W^{(l)}, b^{(l)}\}_{l=1}^{L}$, threshold τ.
Ensure: 2D contour plot of the decision boundary.

```
function VisualizeBoundary(D)
    Compute Projection Matrix P, Centroid x_0
    P, x_0 ← ComputeProjection(D)
    Project D to 2D: D_2D ← P(D − x_0)
    Create grid in D_2D: (X, Y)
    Back-project grid to R^d: G ← P^†(X, Y) + x_0
    Evaluate model: Z ← ComputeBoundary(G, W^(l), b^(l))
    Plot contour Z = τ
end function
```

where C is the covariance matrix of data. To project data onto a 2D plane, we compute the eigen-decomposition:

$$C = U \Lambda U^T, \tag{14}$$

where the projection matrix P consists of the top two eigenvectors corresponding to the largest eigenvalues:

$$P = [u_1 \quad u_2]^T, \quad u_1, u_2 \in U. \tag{15}$$

The 2D grid $\mathcal{D}_{2D}$ is explicitly defined by uniformly spacing points in the ranges:

$$x_{\text{range}} = [\min(X_P[:,1]), \max(X_P[:,1])], \tag{16}$$
$$y_{\text{range}} = [\min(X_P[:,2]), \max(X_P[:,2])], \tag{17}$$

where $X_P = P(X - x_0)$ are projected training points, ensuring the grid fully encompasses the embedded data distribution.

E. Handling Arbitrary Input Dimensionalities

Analogous to SplineCam, our methodology extends to higher-dimensional input spaces by employing projection or slicing techniques. Specifically, the input domain is either projected onto lower-dimensional manifolds or sliced into two-dimensional sections, to which the recursive algorithm is subsequently applied.

The implementation leverages structured grid generation functions (e.g., `np.meshgrid`) and back-projects points to coordinates defined within the input domain. As described in Algorithm 2, this process effectively maps high-dimensional decision boundaries onto interpretable two-dimensional manifolds. Figures 3 and 6 illustrate this technique, demonstrating how the decision boundary is projected and visualized on lower-dimensional spaces while retaining the key characteristics of the underlying model.

Figure 4 highlights this capability on a spiral dataset, where the algorithm reveals intricate decision geometries shaped by the network's nonlinear transformations. Similarly, Figures 4 and 8 show the method's adaptability to diverse datasets, including medical image embeddings and latent spaces from neural networks trained with different activation functions.

Furthermore, Figure 6 presents an application to MNIST digits "3" and "2," where the model's learned decision boundaries are superimposed onto a 2D embedding space, effectively capturing the class separations. The recursive algorithm in Algorithm 2 facilitates this visualization by computing and back-projecting points onto the embedding manifold.

Through this process, a d-dimensional space is accurately transformed into a two-dimensional manifold, resulting in high-fidelity decision boundary plots that reflect the network's decision-making logic. This capability is further exemplified in Figure 9, where the decision boundary derived from chest X-ray embeddings demonstrates clinically significant separations between normal and pneumonia cases.

F. Integration with Common Datasets and Architectures

Our code demonstrations cover scenarios including:

- **Synthetic Functions:** We start with simple, synthetic datasets such as linear classifiers or synthetic boundaries defined by $x_1 + x_2 = 1$. Here, the recursive method verifies the exactness and matches the known analytical decision lines.
- **Spirals:** By training MLPs on well-known 2D classification benchmark (spiral distribution), we illustrate how the algorithm recovers exact complex nonlinear boundaries. The code trains Sigmoid- and SiLU-activated MLPs of varying depth and width, then executes the recursive process to trace decision boundaries.

- **Higher-Dimensional Inputs:** For a 3D sphere classification problem, we show that by selecting 2D cross-sections and applying the algorithm, one obtains exact contour lines on these slices.
- **Real world Data (MNIST variants):** We also demonstrate how the algorithm can be applied to dimensionality-reduced inputs from real-world datasets (e.g., MNIST or MedMNIST). By reducing the dimension or focusing on specific 2D embeddings, the recursive steps recover meaningful decision curves that reflect the network's learned classification strategy.

IV. Experiments

We conducted extensive experiments to validate our approach's accuracy, versatility, and computational performance. The experiments combine the recursive method with fully connected neural network architectures, activation functions, and datasets, guided by Algorithms 1 and 2 provided earlier.

A. Experimental Setup

We use PyTorch as the primary deep learning framework. Models are defined as small MLPs (2–5 layers) with input dimensions ranging from 2D (for direct visualization) to higher dimensions (with projected slices). The experiments employ:

- **Activation Functions:** Sigmoid and SiLU, as well as their mixtures, are tested to confirm the general applicability beyond piecewise polynomial activations.
- **Optimizers and Losses:** Standard optimizers such as Adam and losses such as binary cross-entropy (BCE) or mean squared error (MSE) are used for network training.
- **Data Generation and Preprocessing:** Synthetic datasets (spheres, spirals) are generated using custom functions (e.g., `make_spiral`). Real datasets (MNIST, Fashion-MNIST, MedMNIST) are loaded via torchvision and medmnist libraries, reduced to binary tasks (e.g., digit "2" vs digit "3" classification) and, if necessary, projected into 2D domains.

All experiments run on commodity hardware (CPUs and optional GPUs).

B. Results on Synthetic Benchmarks

Spiral Dataset: The spiral classification problem is more challenging due to highly nonlinear, winding boundaries. We trained a Sigmoid-based MLP and a SiLU-based MLP on a spiral dataset of 10,000 points. Our approach generated exact contour lines that elegantly trace the spiral arms where class decisions flip. This confirms that even in complex topologies, we can rely on the recursive steps to reveal the intricate geometry of the decision frontier.

In the spiral classification task (Figure 4), our approach demonstrates the interplay between activation choice and decision boundary complexity. While both networks achieve good accuracy, the SiLU-activated network's boundary is more elegant and clearly aligns with the underlying spiral geometry, affirming that our method provides a clear window into the internal workings of different model configurations.

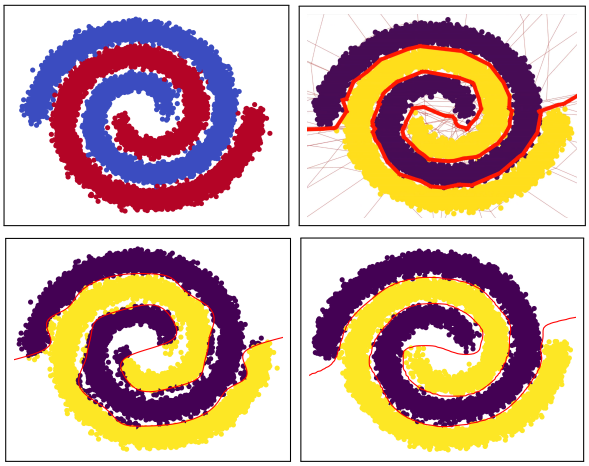

Fig. 4. Visualization of the spiral dataset and comparison of decision boundary using SigmaCam for different activation functions. Top Left: Original labeled spiral dataset, with red and blue points representing two distinct classes. Top Right: Decision boundary generated by a ReLU-activated neural network with five hidden layers, extracted using the SigmaCam methodology. The red contour lines highlight the exact decision boundary separating the two classes. Bottom Left: Decision boundary generated using a Sigmoid-activated MLP, showing smooth but distinct class separations. Bottom Right: Decision boundary generated using a SiLU-activated MLP, demonstrating more natural and fluid transitions between classes. The red contour lines indicate the exact decision boundary extracted using the recursive algorithm.

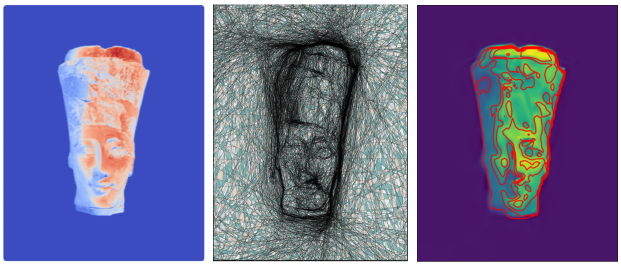

Fig. 5. Decision boundary visualization on a 2D latent embedding derived from facial image data. The left panel displays class distributions in a smooth embedding space, while the right panel overlays the exact decision boundary contours (red lines) computed by our recursive method on a SiLU activated network using SigmaCAM. Notice how the contours faithfully follow the subtle class distinctions present in the latent representation, highlighting the model's nuanced understanding of facial features. As compared to that, the middle image is generated using SplineCam on the same architecture with ReLU activation. The class separation uses lines and is relatively less smooth as compared to how beautifully the SiLU network captures the separations.

Figure 4 underlines the high fidelity of our extracted boundaries. When the trained MLP attains high accuracy on a complex dataset like the spiral, the exact boundary lines precisely match the class transitions, enabling practitioners to confirm that the network's internal representation aligns intuitively with the data's true geometry.

As illustrated in the Figure 4, varying the activation function significantly alters the shape and smoothness of decision boundaries. The SiLU-activated network's boundaries are notably more fluid and continuous than those formed by ReLU, supporting the claim that our exact boundary computation can visually expose the influence of specific activation functions on model interpretability.

As seen in Figure 5, our approach can be applied beyond simple synthetic datasets. Even when working with embeddings of high-dimensional image data (such as faces), the algorithm recovers highly detailed decision boundaries that trace complex class separations in the latent space.

C. Results on Real-World Benchmarks

To move beyond purely synthetic data, we evaluated our method using subsets of the MNIST, FashionMNIST, and MedMNIST datasets, including BreastMNIST and PneumoniaMNIST. These datasets serve as benchmarks for assessing the interpretability and generalizability of our approach.

The MNIST dataset consists of 70,000 grayscale images of handwritten digits (0 through 9), each with a resolution of 28×28 pixels. For our experiments, we focused on binary classification tasks, such as distinguishing between the digits "2" and "3." The dataset was split into 60,000 training images and 10,000 test images. To simplify the experiments and reduce dimensionality, we extracted or learned 2D embeddings using methods such as Principal Component Analysis (PCA) or autoencoder-based latent representations. A small Multi-Layer Perceptron (MLP) was then trained on these embeddings to classify the data. Our recursive algorithm was subsequently applied to the embedded space, where it extracted exact decision boundaries.

The FashionMNIST dataset provides 70,000 grayscale images of clothing items across 10 categories, such as "Trousers" and "Tops," with each image having a resolution of 28×28 pixels. For our experiments, we filtered the dataset to focus on a binary classification task, distinguishing between the categories "Trousers" (label 1) and "Tops" (label 0). This subset contained 12,000 training samples and 2,000 test samples after filtering. The images were normalized and relabeled for binary classification, and a small MLP with SiLU activations was trained on the dataset. Our approach was then used to visualize decision boundaries in the input space, revealing the network's ability to distinguish between the two visually distinct clothing categories.

MedMNIST provides a collection of lightweight medical image datasets designed for evaluating machine learning models. We utilized two datasets:

- **BreastMNIST:** This dataset contains 780 grayscale breast ultrasound images for classifying benign and malignant lesions. Images are resized to 28×28 pixels, with a standard train-validation-test split of 546, 78, and 156 samples, respectively.
- **PneumoniaMNIST:** This dataset includes 5,856 grayscale chest X-ray images for distinguishing between normal and pneumonia-affected cases. Images are downsampled to 28×28 pixels, with 4,708 training samples, 524 validation samples, and 624 test samples.

Despite the inherently high-dimensional nature of these datasets, our method successfully demonstrated its ability to extract interpretable decision boundaries in 2D. By training a small MLP on the 2D embeddings and applying our recursive algorithm, we generated clear contour plots that highlighted the subtle classification cues learned by the model. These plots provide critical insights into how the network distinguishes between visually or diagnostically similar classes.

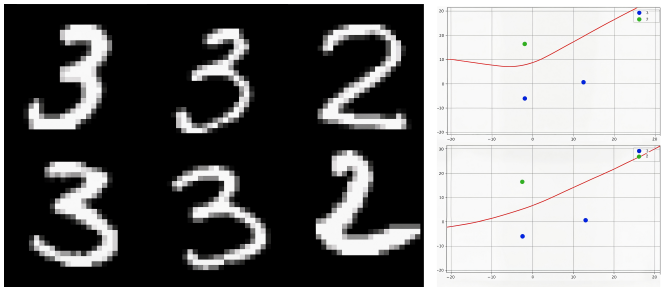

Fig. 6. Decision boundary visualization for discriminating between MNIST digits, e.g., "3" and "2". Left: Example digit images show the raw pixel inputs that the model learns to separate. Right: In a 2D embedding space, the model's decision boundary (red line) is superimposed onto embedded samples of these digits (blue and green markers). This illustrates precisely where the model draws the line between two visually similar classes. Such visualization helps confirm that latent embeddings maintain meaningful class distinctions, which are now made visible through exact boundary extraction.

Fig. 7. Decision boundary analysis on FashionMNIST data projected into two dimensions. Left: Selected FashionMNIST samples such as "Trousers" and "Tops" are displayed in their pixel form, illustrating the complexity and variability within each class. Right: A contour plot shows the learned boundary (in red) separating these classes at the decision threshold. The color-coded points (blue, green) represent embedded samples, helping us see how the network places different clothing items in its latent space and where it draws the line between categories.

For example, Figure 6 showcases decision boundary visualization for discriminating between MNIST digits "3" and "2". Similarly, Figure 8 visualizes the class separations in BreastMNIST and PneumoniaMNIST, illustrating the model's ability to capture clinically relevant patterns in latent embeddings. Finally, Figure 7 highlights the boundaries learned on FashionMNIST data, demonstrating how our approach provides interpretable insights even for datasets with subtle visual differences. These results validate the effectiveness of our recursive algorithm in providing a deeper understanding of model behavior on real-world datasets.

Figure 6 highlights the model's capacity to differentiate between two similar digits. While 3" and 2" share general shape characteristics, the decision boundary in the latent space emerges as a clean curve separating clusters of digit samples. By visualizing this boundary, we gain confidence in the model's internal logic and interpretability.

As shown in Figure 7, when applying our technique to

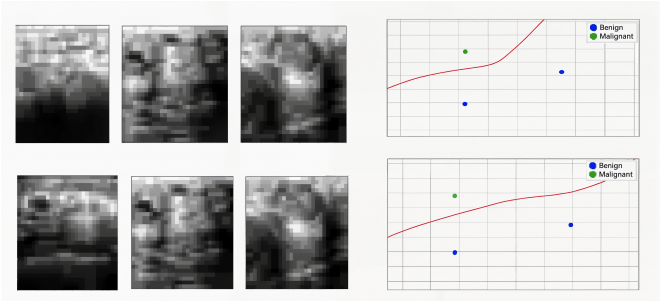

Fig. 8. Visualizing class decision boundaries in a latent embedding of BreastMNIST image data. On the left side, we show original medical image samples (ultrasound scans) projected into a 2D embedding space. On the right side, the extracted decision boundary (red contour line) is overlaid on a scatter plot of embedded, labeled points (colored markers). Each marker represents a particular sample or label of interest, allowing us to see precisely where the model distinguishes between underlying classes (e.g., normal vs. abnormal tissues). This visualization validates that even complex medical image data, once projected, can be analyzed to reveal interpretable decision boundaries aligned with clinical conditions.

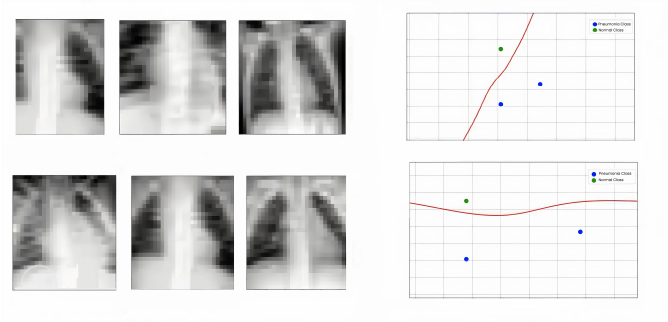

Fig. 9. Visualization of decision boundaries for a pneumonia (PneumoniaMNIST) detection model in a reduced-dimension embedding space. On the left, representative chest X-ray patches from "Normal" and "Pneumonia" cases are shown. On the right, we overlay the class boundary contour over embedded sample points. The contour line at $f(\mathbf{x}) = 0.5$ indicates where the model transitions from predicting Normal (blue points) to Pneumonia (green points), enabling clinicians and researchers to confirm that the network's learned latent representation aligns with the underlying pathological differences in the chest X-ray patterns.

FashionMNIST embeddings, we can discern precisely where the model distinguishes between apparel categories. The contour line closely matches intuitive separations, confirming that even relatively simple MLPs, when embedded and visualized in 2D, produce coherent decision boundaries between items like trousers and tops.

Figure 8 demonstrates the method's applicability to medical image embeddings. The original image samples are shown alongside the exact contour lines defining class separations. This allows clinical experts to confirm that the network partitions the embedded feature space in a medically meaningful manner.

In Figure 9, our method reveals how the model's internal feature representation separates pneumonia-infected samples from healthy ones. The extracted contour demonstrates a smooth, clinically relevant partition in the embedding space, suggesting that the model's latent representation captures the distinctive radiological signatures of pneumonia.

D. 3D Sphere Classification and Higher-Dimensional Generalization

To demonstrate the applicability of our decision boundary visualization technique in higher-dimensional spaces, we experimented with a synthetic 3D sphere dataset. This dataset consists of points randomly sampled within a cube, with the classification label based on whether the point is inside or outside the sphere of radius 1. Specifically, we generated 5,000 samples, using a batch size of 32, where the points inside the sphere were labeled as 1 (inside), and the points outside were labeled as 0 (outside). The data was processed using a custom data loader, and the labels were assigned based on the Euclidean distance from the origin.

As higher-dimensional decision boundaries are more complex to visualize, we used a slicing and projection technique. Although direct visualization of the decision boundary in 3D is difficult, we effectively visualized the decision boundary in 2D by taking cross-sections of the 3D sphere. The cross-sectional slices of the sphere allowed us to visualize the decision boundary as a 2D contour plot. Figure 3 shows this process, where the fifth panel represents a decision boundary visualization after slicing the 3D sphere dataset.

We then trained a small fully connected neural network (MLP) with a SiLU activation function to classify points in the 3D sphere dataset. The model was trained using binary cross-entropy loss, and after convergence, we applied our recursive boundary extraction technique to visualize the decision boundary for the model's predictions. Figure 6 and Figure 8 further illustrate the generalizability of our approach, showing how we apply the same techniques to both simpler datasets like MNIST and more complex, high-dimensional medical datasets like MedMNIST.

The slicing and projection technique used here for visualizing decision boundaries in higher-dimensional spaces is crucial for understanding and interpreting models trained on more complex data. By extracting 2D slices from a high-dimensional space, we ensure that the decision boundary can be visualized while maintaining its accuracy and fidelity to the original data's structure. This technique extends the capabilities of traditional decision boundary visualizations, enabling its use on higher-dimensional classification tasks.

E. Comparisons with SplineCam

We compared our exact solution SigmaCam to the state-of-the-art method SplineCam where SplineCam applies (e.g., models using certain piecewise polynomial activations). For activations such as Sigmoid and SiLU, where SplineCam is not directly applicable, we demonstrated the advantages of our method in producing cleaner, more accurate decision boundaries.

On tasks where SplineCam works, both methods produced visually indistinguishable results, validating the correctness of our exact boundary extraction approach. For example, Figure 3 shows the boundary computed on a 3D sphere dataset, demonstrating how our method extracts decision boundaries through slicing and projection. Similarly, Figure 4 highlights

TABLE I
PERFORMANCE COMPARISON OF SIGMACAM AND SPLINECAM ON THE 2D SPIRAL DATASET

Epoch	ReLU (SplineCam)		Sigmoid (SigmaCam)		SiLU (SigmaCam)		SiLU + Sigmoid (SigmaCam)	
	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)	Accuracy	Time (s)
50	0.99825	866.12	0.73625	80.47	0.99840	100.03	0.99855	94.63
100	0.99845	1098.87	0.79905	80.59	0.99875	85.37	0.99825	82.03
150	0.99865	1368.40	0.92275	81.39	0.99880	86.43	0.99885	81.70
200	0.99850	1666.79	0.97985	82.14	0.99880	84.64	0.99870	85.80
Total Time	-	5000.18	-	324.59	-	356.47	-	344.16

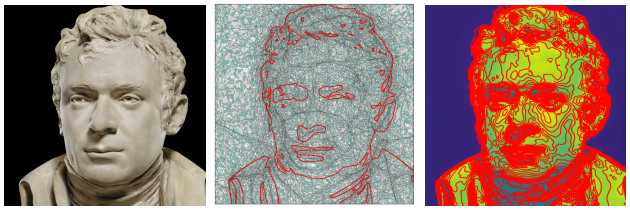

Fig. 10. Comparison of decision boundaries and output landscapes for MLPs using ReLU (middle) and SiLU (right) activations. The ReLU-based model yields a piecewise-linear decision boundary, resulting in more jagged contour lines. In contrast, the SiLU-based model produces smoother transitions and more naturally contoured boundaries, reflecting how different nonlinearities influence the geometry of learned decision regions.

the application of our recursive algorithm on the spiral dataset, where complex geometries are accurately captured.

In Figure 10, we extend this analysis to 2D implicit neural representations (INRs) and show how different activation functions influence the geometry of decision boundaries and output landscapes. The ReLU-based model (middle) produces jagged, piecewise-linear boundaries, while the SiLU-based model (right) achieves smoother, naturally contoured boundaries. These results emphasize how the recursive approach adapts seamlessly to varying nonlinearities, maintaining high fidelity in boundary extraction.

By focusing on exact boundary computation, our method avoids the approximation errors and irregularities associated with gradient-based approaches. This robustness makes it a valuable alternative to existing visualization methods, particularly in cases where SplineCam is either inapplicable or less effective due to activation function constraints.

F. Computational Performance

Table I presents a comparison of the computational performance of SigmaCam and the method SplineCam on the 2D Spiral dataset. SigmaCam was evaluated using Sigmoid, SiLU, and SiLU + Sigmoid activation functions, while SplineCam was assessed with the ReLU activation function. All models maintained identical network architectures, including size and width, and were trained for the same number of epochs to ensure a fair comparison.

SigmaCam demonstrates superior efficiency when utilizing smooth activation functions such as Sigmoid and SiLU, achieving high accuracy with significantly lower computation times compared to SplineCam. Specifically, SigmaCam completes the task within approximately 80 to 100 seconds across all epochs for Sigmoid and SiLU activations. In contrast, SplineCam with ReLU activation maintains similar accuracy levels but requires substantially longer computation times, ranging from 866 to 1,666 seconds, resulting in a total computation time of 5,000 seconds.

Furthermore, we introduced an additional configuration combining SiLU and Sigmoid activations (Final Hidden layer Sigmoid activated), denoted as SiLU + Sigmoid in Table I. This configuration consistently outperformed the standard SiLU activation in terms of both accuracy and computational efficiency. The SiLU + Sigmoid model achieved higher accuracy across all epochs while maintaining competitive execution times, ranging from 81.70 to 94.63 seconds, with a total computation time of 344.16 seconds—faster than both standalone SiLU (356.47 s) and Sigmoid (324.59 s) while achieving superior classification accuracy.

These results underscore the effectiveness of SigmaCam in handling smooth activation functions with minimal computational overhead. The enhanced performance is attributed to its optimized implementation, which leverages vectorized operations and compute accelerated linear algebra. While SplineCam remains a robust tool for ReLU activations, the significant increase in computation time highlights the advantages of SigmaCam for applications requiring efficient visualization of decision boundaries. Moreover, the PE+Sigmoid configuration demonstrates that augmenting the input with periodic features can further improve classification accuracy on challenging, non-linear datasets. Our experiments also extend to evaluating runtime and memory usage across various configurations. Despite the inherent complexity of exact boundary extraction compared to approximate sampling, SigmaCam's recursive approach remains efficient. For input domains sized 100×100 or larger, boundary computations are typically completed within seconds to a few minutes on standard hardware, demonstrating the practicality of SigmaCam for large-scale applications.

V. CONCLUSION AND FUTURE WORK

We have introduced a novel recursive algorithm for visualizing decision boundaries of MLPs employing smooth activation functions. Our approach extends the theoretical exactness and domain coverage of methods such as SplineCam to activation

types previously unsupported. Future work includes extending the method to larger architectures and investigating efficient parallelization strategies. Also, we will develop hybrid frameworks integrating visual analytics with mathematical rigor to enhance neural decision boundary understanding across diverse applications. Additionally, addressing the complexities introduced by smooth activation functions remains a critical area for advancing exact decision boundary extraction methods [33], [34], alongside exploring interpretability and embedding-based solutions [35].

REFERENCES

[1] A. I. Humayun, R. Balestriero, G. Balakrishnan, and R. Baraniuk, " SplineCam: Exact Visualization and Characterization of Deep Network Geometry and Decision Boundaries ," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2023, pp. 3789–3798. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR52729.2023.00369

[2] C. Lee and D. Landgrebe, "Decision boundary feature extraction for neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 75–83, Jan 1997.

[3] V. Shah and N. Youngblood, "Leveraging Continuously Differentiable Activation Functions for Learning in Quantized Noisy Environments," *arXiv e-prints*, p. arXiv:2402.02593, Feb. 2024.

[4] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *arXiv e-prints*, p. arXiv:1710.05941, Oct. 2017.

[5] M. Gustineli, "A survey on recently proposed activation functions for Deep Learning," *arXiv e-prints*, p. arXiv:2204.02921, Apr. 2022.

[6] Y. Li, L. Ding, and X. Gao, "On the Decision Boundary of Deep Neural Networks," *arXiv e-prints*, p. arXiv:1808.05385, Aug. 2018.

[7] C. Lee and D. Landgrebe, "Feature extraction based on decision boundaries," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 388–400, April 1993.

[8] A. Schulz, A. Gisbrecht, and B. Hammer, "Using discriminative dimensionality reduction to visualize classifiers," *Neural Processing Letters*, vol. 42, no. 1, pp. 27–54, 2015.

[9] M. Becker, J. Lippel, A. Stuhlsatz, and T. Zielke, "Robust dimensionality reduction for data visualization with deep neural networks," *Graphical Models*, vol. 108, p. 101060, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1524070320300035

[10] J. L. Valerdi, "On Minimal Depth in Neural Networks," *arXiv e-prints*, p. arXiv:2402.15315, Feb. 2024.

[11] F. C. M. Rodrigues, R. Hirata, and A. C. Telea, "Image-based visualization of classifier decision boundaries," in *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, Oct 2018, pp. 353–360.

[12] N. Gong, Z. Wang, S. Chen, G. Liu, and S. Xin, "Decision boundary extraction of classifiers," *Journal of Physics: Conference Series*, vol. 1651, no. 1, p. 012031, nov 2020. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1651/1/012031

[13] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci, "Visualizing high-dimensional data: Advances in the past decade," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 3, p. 1249–1268, Mar. 2017. [Online]. Available: https://doi.org/10.1109/TVCG.2016.2640960

[14] A. K. A. Talukder and K. Deb, "Paletteviz: A visualization method for functional understanding of high-dimensional pareto-optimal data-sets to aid multi-criteria decision making," *IEEE Computational Intelligence Magazine*, vol. 15, no. 2, pp. 36–48, May 2020.

[15] Y. Liu, E. Jun, Q. Li, and J. Heer, "Latent space cartography: Visual analysis of vector space embeddings," *Computer Graphics Forum*, vol. 38, no. 3, pp. 67–78, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13672

[16] M. Kahng, P. Y. Andrews, A. Kalro, and D. H. Chau, "ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models," *arXiv e-prints*, p. arXiv:1704.01942, Apr. 2017.

[17] J. Wexler, M. Pushkarna, T. Bolukbasi, M. Wattenberg, F. Viegas, and J. Wilson, "The What-If Tool: Interactive Probing of Machine Learning Models," *arXiv e-prints*, p. arXiv:1907.04135, Jul. 2019.

[18] S. Berg, D. Kutra, T. Kroeger, C. N. Straehle, B. X. Kausler, C. Haubold, M. Schiegg, J. Ales, T. Beier, M. Rudy, K. Eren, J. I. Cervantes, B. Xu, F. Beuttenmueller, A. Wolny, C. Zhang, U. Koethe, F. A. Hamprecht, and A. Kreshuk, "ilastik: interactive machine learning for (bio)image analysis," *Nature methods*, vol. 16, no. 12, p. 1226—1232, December 2019. [Online]. Available: https://doi.org/10.1038/s41592-019-0582-9

[19] M. Kabra, A. A. Robie, M. Rivera-Alba, S. Branson, and K. Branson, "Jaaba: interactive machine learning for automatic annotation of animal behavior," *Nature methods*, vol. 10, no. 1, p. 64—67, January 2013. [Online]. Available: https://doi.org/10.1038/nmeth.2281

[20] S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza, "Power to the people: The role of humans in interactive machine learning," *AI Mag.*, vol. 35, no. 4, p. 105–120, Dec. 2014. [Online]. Available: https://doi.org/10.1609/aimag.v35i4.2513

[21] J.-T. Sohns, C. Garth, and H. Leitte, "Decision boundary visualization for counterfactual reasoning," *Computer Graphics Forum*, vol. 42, no. 1, pp. 7–20, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14650

[22] F. C. M. Rodrigues, M. Espadoto, R. Hirata, and A. C. Telea, "Constructing and visualizing high-quality classifier decision boundary maps," *Information*, vol. 10, no. 9, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/9/280

[23] A. Grushin, J. Nanda, A. Tyagi, D. Miller, J. Gluck, N. C. Oza, and A. Maheshwari, *Decoding the Black Box: Extracting Explainable Decision Boundary Approximations from Machine Learning Models for Real Time Safety Assurance of the National Airspace*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2019-0136

[24] A. Humayun, J. Casco-Rodriguez, R. Balestriero, and R. Baraniuk, "Provable instance specific robustness via linear constraints," *2nd AdvML Frontiers Workshop at International Conference on Machine Learning 2023*. [Online]. Available: https://par.nsf.gov/biblio/10466270

[25] M.-C. Brandenburg, G. Loho, and G. Montúfar, "The Real Tropical Geometry of Neural Networks," *arXiv e-prints*, p. arXiv:2403.11871, Mar. 2024.

[26] Z. Zhong, W. Wang, B. Lévy, J. Hua, and X. Guo, "Computing a high-dimensional euclidean embedding from an arbitrary smooth riemannian metric," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3197517.3201369

[27] P. Esser, R. Rombach, and B. Ommer, "A Disentangling Invertible Interpretation Network for Explaining Latent Representations," *arXiv e-prints*, p. arXiv:2004.13166, Apr. 2020.

[28] X. Li, H. Xiong, X. Li, X. Wu, X. Zhang, J. Liu, J. Bian, and D. Dou, "Interpretable Deep Learning: Interpretation, Interpretability, Trustworthiness, and Beyond," *arXiv e-prints*, p. arXiv:2103.10689, Mar. 2021.

[29] M. Schlüter and B. Steffen, "Affinitree: A compositional framework fornbsp;formal analysis andnbsp;explanation ofnbsp;deep neural networks," in *Tests and Proofs: 18th International Conference, TAP 2024, Milan, Italy, September 9–10, 2024, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 148–167. [Online]. Available: https://doi.org/10.1007/978-3-031-72044-4_8

[30] H. Dong, B. Liu, D. Ye, and G. Liu, "Interpretability as approximation: Understanding black-box models by decision boundary," *Electronics*, vol. 13, no. 22, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/22/4339

[31] S. Adam, D. A. Karras, and M. N. Vrahatis, *Revisiting the Problem of Weight Initialization for Multi-Layer Perceptrons Trained with Back Propagation*. Berlin, Heidelberg: Springer-Verlag, 2009, p. 308–315. [Online]. Available: https://doi.org/10.1007/978-3-642-03040-6_38

[32] W. Samek, A. Binder, G. Montavon, S. Bach, and K.-R. Müller, "Evaluating the visualization of what a Deep Neural Network has learned," *arXiv e-prints*, p. arXiv:1509.06321, Sep. 2015.

[33] K. Biswas, S. Kumar, S. Banerjee, and A. K. Pandey, "SMU: smooth activation function for deep networks using smoothing maximum technique," *arXiv e-prints*, p. arXiv:2111.04682, Nov. 2021.

[34] T. Szandała, "Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks," *arXiv e-prints*, p. arXiv:2010.09458, Oct. 2020.

[35] I. Arkoudi, R. Krueger, C. L. Azevedo, and F. C. Pereira, "Combining discrete choice models and neural networks through embeddings: Formulation, interpretability and performance," *Transportation Research Part B: Methodological*, vol. 175, p. 102783, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S019126152300108X