# ImmersiveSHAP: Immersive Analytics Visualization System for XAI using SHAP Scatter Plot

J. Montilla-López, Daniel Valencia, Jovani A. Jimenez-Builes, and Gustavo Ramirez-Gonzalez

***Abstract*—This study presents ImmersiveSHAP, an immersive analytics visualization system for explainable artificial intelligence that leverages a modular pipeline to render SHAP scatter dependence plots in virtual reality. The research contribution is the integration of explainable artificial intelligence (SHAP Python library) with immersive analytics in a virtual reality environment (Unity). The system integrates a Python-based preprocessing module that handles data loading, model training, and SHAP explanation computation, and a Unity-based rendering and interaction module, implemented within a client–server architecture. The WebSocket protocol establishes communication between the Python server and the Unity client. The system extends traditional 2D SHAP plots into interactive 3D visualizations, designed to support immersive analytics with post hoc model explanations. Furthermore, a technical validation used the Iris, Breast Cancer, and California Housing datasets, covering point clouds from $N = 150$ to $N = 20{,}640$, and deployed the system on a Meta Quest 3. Results identify operational constraints, showing stable performance on small-to-medium datasets ($N \leq 2{,}000$) with an average frame rate of approximately 70 FPS, close to the device's refresh rate target and within acceptable ranges for virtual reality. These results indicate the system's viability as a baseline architecture for immersive visualization of SHAP-based explanations.**

**Link to graphical and video abstracts, and to code:**
**https://latamt.ieeer9.org/index.php/transactions/article/view/10596**

***Index Terms*—Immersive analytics, Explainable Artificial Intelligence, SHAP, Virtual Reality, Immersive visualization.**

## I. Introduction

EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI) is a branch of Artificial Intelligence (AI) research that makes the actions and decisions of automated systems understandable to humans [1]. XAI addresses the "black-box" problem, in which complex AI algorithms achieve high predictive accuracy but provide no intuitive rationale for their internal decision-making processes [2]. By fostering transparency, accountability, and trust, XAI enables the deployment of AI in mission-critical domains like healthcare, finance, and defense [3].

XAI methods are commonly categorized as ante-hoc or post-hoc, and further classified as model-agnostic or model-specific, operating at both global and local levels [4]–[6]. Among output formats, visual explanations play a central role because they convey complex relationships in an interpretable form [7]. Ante-hoc approaches use structures such as decision trees or fuzzy inference systems to provide transparent global representations [5]. Post-hoc methods generate explanations after training, including local techniques such as saliency maps, class activation maps, Layer-wise Relevance Propagation [8], and Local Interpretable Model-Agnostic Explanations (LIME) [9], as well as global methods such as Partial Dependence Plots (PDP) [10], and Individual Conditional Expectation plots [11]. The Shapley Additive exPlanations (SHAP) method [12] unifies local and global explainability by offering different plots for individual predictions and ranking global feature importance.

Despite their utility, most XAI visualizations rely on two-dimensional (2D) representations. These approaches often suffer from visual clutter, overplotting, and limited support for representing complex relationships, requiring users to reconstruct higher-dimensional structures mentally [13], [14]. These limitations motivate the exploration of alternative visualization paradigms.

Immersive Analytics (IA) addresses these limitations by leveraging stereoscopic vision, motion tracking, and three-dimensional (3D) environments to provide an alternative paradigm for data interaction [15]. This paradigm enables users to explore data spatially and inspect structures that remain difficult to see in 2D representations [16], [17]. As an interdisciplinary field, IA integrates data visualization, human–computer interaction (HCI), and extended reality (XR) technologies to enable immersive data exploration and analysis [15]. Through virtual reality (VR) and augmented reality (AR), embodied and interactive environments can facilitate the exploration of complex multidimensional data [18]. In data visualization contexts, IA can function as an interface for exploring and managing multidimensional datasets in 3D environments [19]. Prior work in VR supports immersive visualization techniques, including scatter plots for large-scale data exploration [20], and scatter plots for dimensionality-reduced data for perceptual tasks [21]. Techniques like interactive 3D visualizations, node-link graphs, and immersive representations enable intuitive data exploration and manipulation. For example, [16] proposes a VR environment for deep learning (DL) visualization to navigate the layers of convolutional neural networks. Similarly, [22] and [23] explore collaborative environments in AR and tools such as Immersive Parallel Coordinate Plots for multidimensional data visualization in VR. Additionally, [17] integrates DL frameworks with real-time interaction capabilities, offering insights into neural networks using Shapley values to enhance interpretability.

The associate editor coordinating the review of this manuscript and approving it for publication was Adriana Tellez (*Corresponding author: Daniel Valencia*).

J. Montilla-López, Daniel Valencia, and Gustavo Ramirez-Gonzalez are with Universidad del Cauca, Colombia (e-mails: julianamontillalo@unicauca.edu.co, danielvalencia@unicauca.edu.co, and gramirez@unicauca.edu.co). Jovani A. Jimenez-Builes is with the Universidad Nacional de Colombia, Medellín, Colombia (e-mail: jajimen1@unal.edu.co).

IA also presents several challenges, including visual scalability, complexity of interaction, and the lack of standardized frameworks to assess technical accuracy and human interpretability [24]. Rendering large-scale models may require substantial computational resources, and 3D visualizations remain susceptible to occlusion and depth distortion [16], [17]. Despite these limitations, the ability to navigate an unbounded workspace in XAI may reduce the semantic gap between the behavior of mathematical models and human reasoning [25].

Although IA has demonstrated effectiveness in multiple fields [26], [27], its application to XAI remains underexplored. As discussed in [28], most approaches rely on 2D representations, and only limited efforts explore immersive post-hoc explanations. To address this gap, [28] proposes a conceptual approach for a visualization pipeline based on a client–server architecture that uses Python for computation and Unity for rendering. Prior work has adopted similar architectures to decouple processing and visualization, thereby improving performance and interoperability [16], [17], [29], and has demonstrated the feasibility of immersive visualization of abstract data, including large-scale scatter plots in VR environments [21]. The conceptual approach in [28] uses random data with a maximum of 100 points for model training, SHAP value calculation, and point cloud generation to demonstrate the feasibility of WebSocket communication between a Python-based SHAP library and a Unity visualization. The study focuses on SHAP due to its theoretical foundation in cooperative game theory and its ability to provide local and global explanations that are theoretically consistent [2], [12]. In contrast to LIME, which may yield unstable local approximations, SHAP provides consistent attribution of characteristics [6]. The systematic review in [20] indicates that immersive visualization systems have been extensively explored in all domains. However, these systems focus on visualization performance or perceptual evaluation rather than on integrating XAI pipelines with immersive environments. In contrast, the current research focuses on XAI & IA integration through a modular system combining SHAP-based explanations with immersive analytics visualization, and on its technical validation using real datasets.

Building on this foundation, the current research develops, deploys, and technically validates ImmersiveSHAP, an immersive analytics visualization system that implements a visualization pipeline on a head-mounted display (HMD). Whereas the preliminary study in [28] focused on architectural feasibility, the current research uses Python to train models, calculate SHAP values, and generate point clouds. It builds a 3D VR environment in Unity with spatial axes and colormaps that render a SHAP scatter plot. This work targets SHAP scatter plots, which encode both local and global explainability but remain constrained by static 2D representations that introduce visual clutter, limit interaction, and limit the exploration of complex feature relationships. SHAP scatter plots display a feature and SHAP values on the horizontal and vertical axes, and a second feature with color variation; these plots can be related to a 3D representation. ImmersiveSHAP supports spatial exploration of feature contributions and interactions, which may support the identification of patterns, distributions, and relationships that are difficult to observe in 2D views. By mapping data onto three spatial axes, the visualization supports dispersion, clustering, and trend analysis through spatial perception and interaction.

The ImmersiveSHAP validation focuses on three aspects: (i) system latency from visualization configuration to visual display, (ii) scalability in terms of the maximum number of rendered points while sustaining a frame rate close to the 72 Hz refresh rate of the VR display, which remains above commonly recommended minimum thresholds for VR performance, and (iii) interaction stability, evaluated through the responsiveness of mechanisms such as hovering and tooltips. Results indicate that the system maintains an average performance of approximately 70.17 FPS when rendering up to 2,000 points, which remains sufficiently close to the target to maintain stable system performance under immersive conditions.

## II. Background

### A. SHapley Additive exPlanations

SHAP is a unified framework for interpreting machine learning predictions by assigning an importance value to each input feature for a given outcome. It builds on concepts from game theory, modeling the prediction task as a cooperative game in which each feature acts as a player, and the model output represents the total payoff. Using Shapley values, SHAP quantifies each feature's average marginal contribution across all possible coalitions. This formulation yields an additive attribution method that satisfies local accuracy, missingness, and consistency, enabling both local explanations for individual instances and global explanations across datasets [6], [12].

SHAP supports both model-agnostic and model-specific explanations. KernelSHAP approximates Shapley values for arbitrary models, while TreeSHAP and DeepSHAP provide efficient implementations for tree-based models and deep neural networks, respectively [2], [12]. Despite its theoretical grounding, SHAP requires careful interpretation, as assumptions such as feature independence may introduce bias when features are correlated or exhibit complex nonlinear interactions. These properties make SHAP particularly suitable for integration into visualization systems that require both local and global explainability.

SHAP provides a Python application programming interface (API) for computing and visualizing feature attributions [30]. The core component, the explainer, encapsulates the model's prediction function and a background dataset to estimate conditional expectations. Users instantiate an appropriate Explainer (e.g., TreeExplainer or KernelExplainer) and pass input samples to compute each feature's marginal contribution. The API returns an Explanation object integrating SHAP values, the model's expected output, and feature data, enabling direct use in visualization tools for interpretable analysis.

A key visualization used in this work is the SHAP dependence scatter plot, generated via the shap.plots.scatter function, visualizes how a specific feature influences a model's predictions across a dataset [30]. Constructed from an Explanation object containing feature values and SHAP attributions, each point represents an instance, with the value of

the feature on the horizontal axis and its SHAP value on the vertical axis quantifying the contribution of the feature. The vertical dispersion at similar feature values highlights interaction effects, while an optional histogram shows the empirical distribution of the feature, supporting the analysis of linear or non-linear relationships [31]; SHAP formalizes interactions using the Shapley interaction index, decomposing contributions into main and pairwise effects. The API colors the points according to the strongest interacting feature to make the dependencies explicit. By capturing instance-level attributions, SHAP dependence plots extend classical PDP, providing a finer-grained and more expressive representation of model behavior [30].

### B. Virtual Reality in Immersive Analytics

VR is an HCI paradigm to immerse, navigate, and interact within computer-generated 3D environments. VR systems rely on stereoscopic rendering, motion tracking, and real-time interaction for spatial exploration of digital content [26]. These systems aim to induce a sense of presence through immersion, allowing users to respond to virtual stimuli while remaining aware of their artificial nature [20], [32]. High-quality immersion depends on accurate head tracking, consistent mapping between physical movements and virtual actions, and stable real-time rendering [20], [32]. Maintaining stable rendering performance, typically above 60 FPS with low latency, is commonly associated with smooth interaction and reduced cybersickness [33], [34].

VR system evaluations commonly use performance metrics such as CPU/GPU/RAM utilization, and error rates [33]. Efficient rendering aims to reduce draw calls and latency spikes, while scalability for large datasets relies on GPU parallelization, compute shaders, level-of-detail techniques, and frustum culling [35], [36]. Modular architectures and API-driven designs support extensibility and integration with external data processing pipelines [16], [29]. Tracking accuracy analyzes record spatial and temporal data to evaluate positional and angular precision during interaction tasks [34].

VR hardware has evolved from room-scale Cave Automatic Virtual Environment systems to portable HMDs [32]. Real-time engines such as Unity and Unreal Engine support software development for device integration, interaction, and rendering capabilities [20], [35], [36]. Unity supports XR development through the XR Interaction Toolkit, which enables object manipulation, locomotion, and interface control, and integrates with HMDs through OpenXR [17].

### C. Head-Mounted Displays

HMDs are wearable devices that provide stereoscopic visualization and six degrees of freedom (6-DOF) tracking, enabling real-time immersion and interaction in 3D environments. Modern standalone HMDs integrate sensing, computation, and display capabilities within a single device, supporting mobile immersive applications [20], [26].

These devices deploy immersive visualization systems under real-time rendering and interaction constraints, where computational resources, latency, and display refresh rates directly affect user experience and system performance.

## III. Immersive Analytics Visualization System Design

This section presents ImmersiveSHAP, an immersive analytics visualization system that renders a 3D SHAP dependence scatter plot using Python, Unity, and Meta Quest 3. The source code developed in this study is available in the ImmersiveSHAP GitHub repository [37].

### A. System Architecture

We propose ImmersiveSHAP, an immersive analytics visualization system that follows the conceptual design introduced in [28], and establishes an extended modular pipeline composed of four modules: Preprocessing, Communication, Rendering, and Visualization & Immersive Analytics. Fig. 1 illustrates the architecture supporting the system's design and deployment. Each module comprises specialized submodules that address specific stages of the workflow, collectively enabling the generation, transmission, rendering, and immersive exploration of SHAP-based explanations. The pipeline is defined independently of specific implementation technologies, supporting portability across different rendering engines and programming environments.

The system adopts a distributed client–server architecture using WebSocket communication to integrate Python-based analytical processing with immersive visualization in Unity. Similar architectural decoupling has been used in prior immersive systems to improve performance and interaction stability [16], [17], [29]. As shown in Fig. 2, the Python server performs data loading, model training, and SHAP explanation generation, while the Unity client manages rendering and user interaction. This separation offloads computationally intensive tasks to the backend, allowing the HMD to focus on low-latency rendering, motion tracking, and interaction. The system is not fully autonomous because it depends on the external server for preprocessing and explanation generation. This dependency constitutes a design limitation of the current implementation.

*1) Preprocessing Module:* This module performs data preparation, model training, and SHAP-based explanation generation using Python. The system supports XGBoost regression and classification models trained on the Iris [38], Breast Cancer [39], and California Housing [40] datasets, covering multiclass, binary, and regression tasks. The request_manager submodule coordinates the workflow and serves as the module's entry point, receiving deserialized requests from the WebSocket server and orchestrating dataset inspection, data loading, model training, explanation computation, and plot generation.

The resource_inspector submodule centralizes metadata for datasets, models, colormaps, and supported plot types, enabling semantic validation prior to execution. `shap.Explainer()` computes explanations and produces a `shap.Explanation` object from which feature values, SHAP attributions, and metadata are extracted and organized into a structured format for export to Unity. The module also generates a 2D SHAP dependence scatter plot for comparison. Although the pipeline is extensible, the current

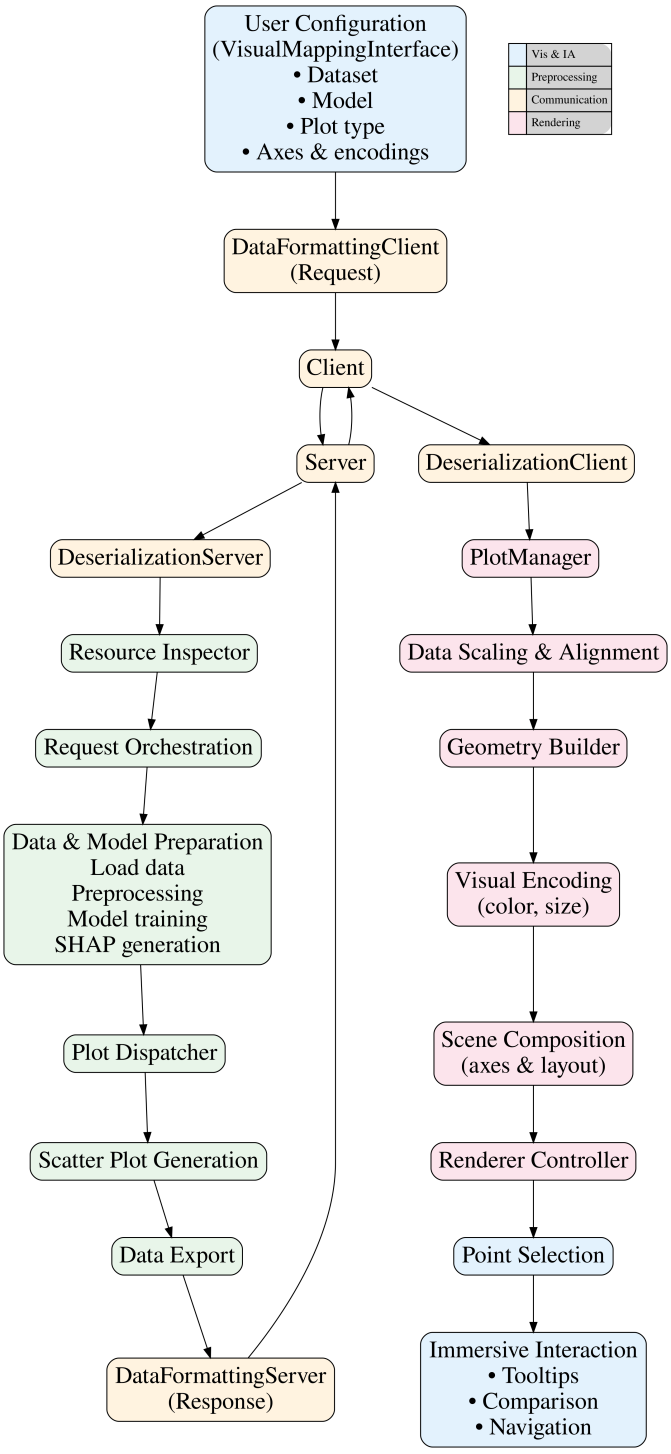



Fig. 1. ImmersiveSHAP Visualization pipeline.

validation focuses on SHAP dependence scatter plots, and the plot_dispatcher submodule selects the appropriate plot type based on the input configuration.

The system preserves the semantics of the standard 2D SHAP scatter plot, which maps feature values to the X-axis and SHAP values to the Y-axis, while encoding a secondary interacting feature through color. This structure is extended into 3D VR by assigning the interaction feature to both color and depth (Z-axis). This mapping preserves the original X–Y relationships, such that a frontal orthographic view reproduces the corresponding 2D plot. The additional spatial dimension may reduce overplotting and provide alternative perspectives for analyzing feature interactions, clustering patterns, and distribution dispersion. This approach complements conventional 2D visualizations, as simpler tasks (e.g., single-feature trend inspection) may remain more efficient in planar representations.

The module supports both regression and classification tasks, including binary and multiclass settings, enabling class-specific attributions when applicable. Interaction variables can be specified by the user or automatically selected using `shap.utils.approximate_interactions()`, which provides a heuristic ranking of candidate feature interactions. This functionality enables a consistent mapping of interaction features to the Z-axis while preserving alignment with the underlying SHAP explanation.

*2) Communication Module:* This module enables bidirectional data exchange between the Python-based preprocessing environment and the Unity-based visualization system through a WebSocket client–server architecture. The Python component operates as the server, serializing explanatory data in JSON, while the Unity application acts as the client, issuing visualization requests and receiving responses. Depending on the deployment configuration, communication occurs over localhost or a local wireless network, preserving data integrity and semantic consistency while supporting efficient request–response interactions.

*3) Rendering Module:* The rendering module transforms explanatory data, such as SHAP values, feature values, and associated metadata, into 3D visualizations using Unity. It focuses exclusively on data-to-visual mapping and rendering, without handling interaction logic or user interface components. Its submodules handle data ingestion; filtering through clipping and downsampling; scaling and normalization into 3D space; geometric construction using instanced primitives (e.g., sphere-based prefabs); visual encoding of attributes such as color, size, and transparency; and the generation of labeled reference axes.

Additional components manage scene configuration, including spatial layout, cameras, and lighting, while the RendererController activates and configures rendering profiles for personal computer (PC) and Meta Quest deployments. The PlotManager acts as the central orchestrator, coordinating data flow, rendering requests, and scene updates by executing submodules sequentially to ensure consistent rendering of 3D explanatory plots.

*4) Visualization & Immersive Analytics Module:* This module enables immersive exploration, interaction, and comparative analysis of 3D visualizations in a VR environment using HMDs such as Meta Quest 3. It supports navigation, point-level selection, and detail-on-demand mechanisms, providing visual feedback (e.g., color changes) and contextual tooltips directly within the 3D space. These are triggered when the user hovers over a point and remain fixed upon selection. Each point is defined as an interactable object, enabling detection through raycasting.

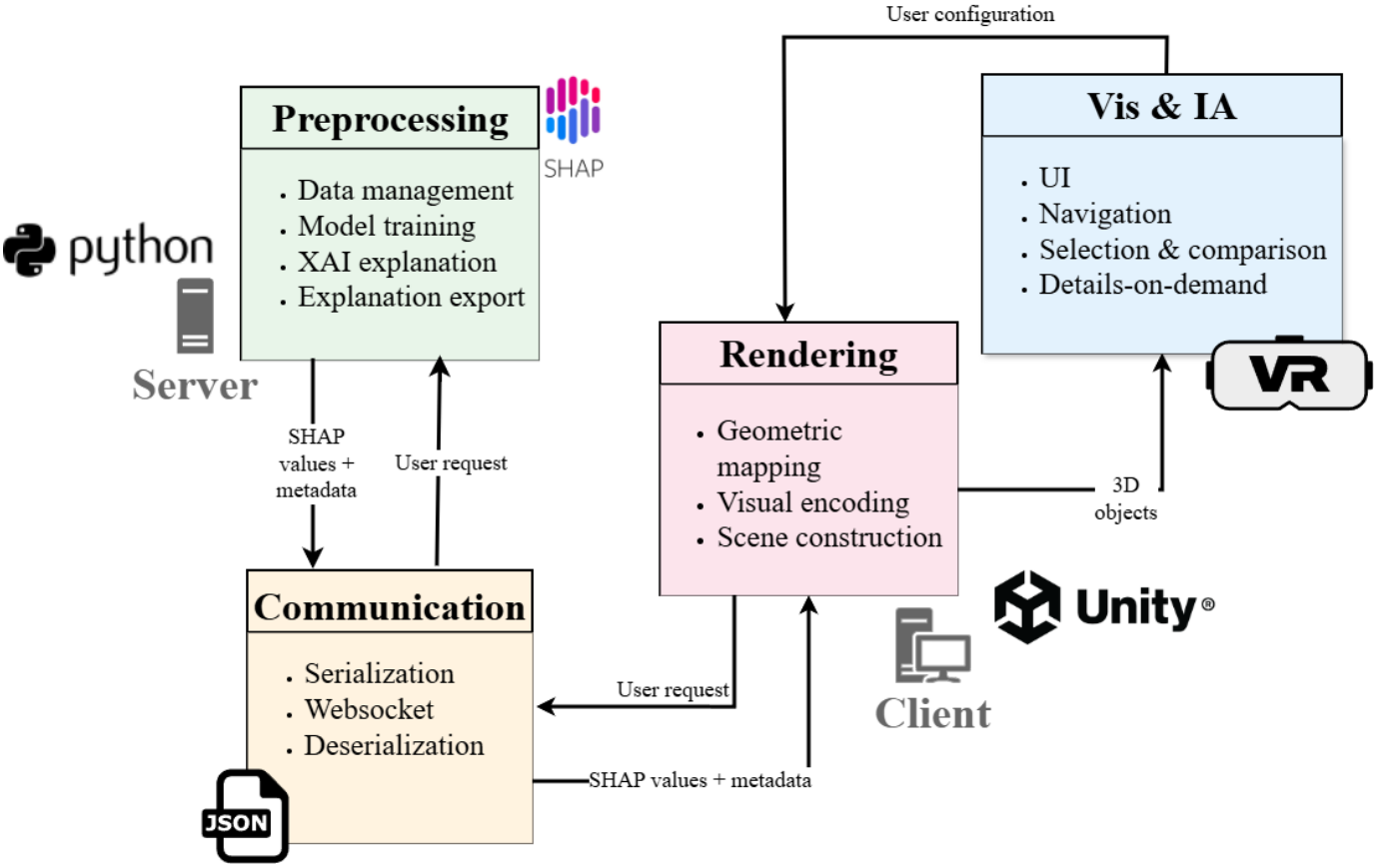



Fig. 2. ImmersiveSHAP system architecture.

These interaction mechanisms enable point-level inspection of SHAP-based explanatory data. A configurable user interface allows users to select datasets, models, plot types, and colormaps, supporting flexible immersive analysis.

Fig. 3 presents example outputs of the ImmersiveSHAP system for the California Housing dataset, illustrating the extension from conventional 2D representations to immersive 3D visualization and interaction.

### *B. Materials*

System development and testing used a workstation equipped with an AMD Ryzen 7 7735HS processor (3.20 GHz), 16 GB of RAM (15.2 GB usable), a 512 GB SSD, and an NVIDIA GeForce RTX 4050 Laptop GPU with 6 GB of dedicated memory, complemented by integrated AMD Radeon Graphics (486 MB). The system was deployed on a Meta Quest 3 standalone headset, powered by a Qualcomm Snapdragon XR2 Gen 2 SoC and 8 GB of LPDDR5 RAM. The device features dual RGB LCD displays with a resolution of 2064 $\times$ 2208 pixels per eye, pancake optics, and a 72 Hz refresh rate for these experiments. Interaction uses two Touch Plus controllers (6-DoF), and connectivity relies on Wi-Fi 6E.

The server-side communication module is implemented in Python using the WebSockets library (version 15.0.1). On the client side, in Unity, communication uses the NativeWebSocket library [41].

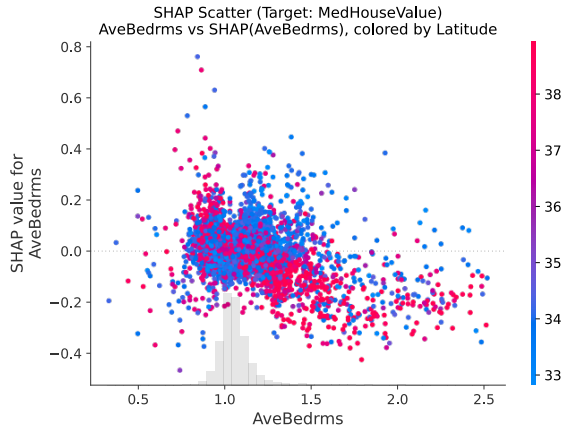
The preprocessing module is implemented in Python 3.11.9 using SHAP 0.48, NumPy 2.2.6, Matplotlib 3.10.3, scikit-learn 1.7.0, and XGBoost 3.0.2. Rendering and interaction are implemented in Unity version 6000.0.29f1 using the Universal Render Pipeline (URP), which supports lightweight rendering on standalone XR devices.

The system integrates the XR Interaction Toolkit (version 3.0.8), including XR Origin, interactors, and interaction providers. XR Plugin Management uses OpenXR for device compatibility. Controller profiles for Meta Quest devices are enabled, and Starter Assets support input configuration. The XR Device Simulator supports PC-based testing. Interaction relies on ray-based selection and controller input.

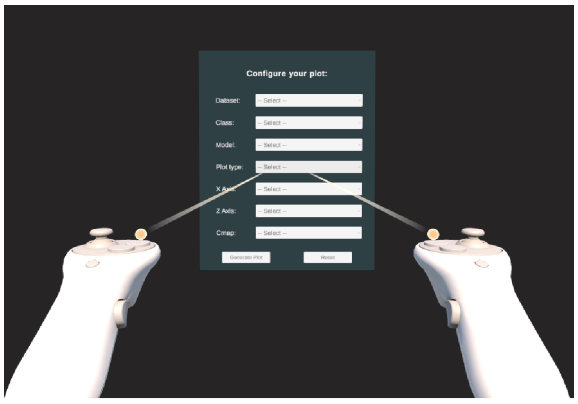
For deployment, the application was built for Android using Unity's Build and Run option, with OpenXR enabled, the IL2CPP scripting backend, ARM64 architecture, a minimum API level of 29 (Android 10), and ASTC texture compression.

Standalone operation requires the PC and the HMD to connect to the same wireless network. The `websockets.serve(handler, "0.0.0.0", 8765)` configures the Python server to listen on all network interfaces and accept connections from devices on the local network. The Unity client connects using the PC's IPv4 address and the same port (8765).

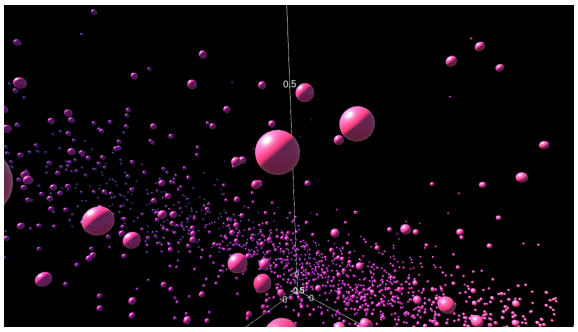
This distributed configuration offloads computational tasks, such as model inference and SHAP generation, to the backend while preserving HMD resources for rendering and interaction.




(a) 2D SHAP scatter plot




(b) User interface



(c) 3D SHAP visualization


(d) Interaction in VR

Fig. 3. ImmersiveSHAP views for the California Housing dataset: (a) 2D SHAP scatter plot, (b) user interface, (c) 3D visualization, and (d) VR interaction with tooltips.

Although the application runs in standalone mode on the Meta Quest 3, it depends on the external server for data processing and is therefore not fully autonomous.

### C. Metrics

For this study, we evaluated the system using the Iris, Breast Cancer, and California Housing datasets to assess performance across varying point cloud sizes, corresponding to the number of GameObjects instantiated in the rendered scatter plot. Each experiment was repeated three times under identical conditions, using the same feature assignments for the X, Y, and Z axes, the same colormap, and the same AI model to ensure consistent and reliable results.

We performed a technical validation to quantify end-to-end latency, from initial plot configuration to final visual rendering, and to identify the system's scalability limits. In this context, scalability is the maximum point density the system can support before significant performance degradation or visual instability occurs. The dataset size directly determines the number of points rendered in the VR environment. The Iris dataset was used as a baseline (representing a low-density scenario) to estimate the system's minimum scalability overhead. Conversely, the California Housing dataset enables evaluations of stability and scalability at medium-to-high point densities. For this dataset, we varied the number of rendered points from 2,000 to 20,640 (the full dataset size) to validate the system's scalability and determine at what point its performance degrades.

Table I summarizes the experimental configuration, including the dataset size, the features assigned to the x and z axes, the selected target, the model used, and the number of points rendered per test (N).

TABLE I
IMMERSIVESHAP EXPERIMENTAL CONFIGURATION

| Dataset | Task | Size | Configuration |
|---|---|---|---|
| Iris | Multiclass | 150 | Model: XGBCls.<br>Target: versicolor<br>X: Petal length - Z: Sepal width<br>$N = 150$ |
| Breast Cancer | Binary | 569 | Model: XGBCls.<br>Target: benign<br>X: Mean area - Z: Worst smooth.<br>$N = 569$ |
| California Housing | Regression | 20,640 | Model: XGBReg.<br>Target: MedHouseValue<br>X: HouseAge - Z: AveOccup<br>N = {2,000, 5,000, 10,000, 15,000, 20,640} |

The three core dimensions of performance assessment are:

- *Latency Response*: Including Network Latency (Round Trip Time (RTT)), Server Processing Time, and Client Rendering Time (Unity instantiation).
- *Visual Rendering Performance*: Analyzing Average Frame Rate (FPS) and 1% Low FPS (jitter detection).
- *Hardware Resource Efficiency*: Monitoring CPU/GPU utilization, power consumption, and system temperature on the Meta Quest 3 standalone hardware.

The Meta Quest Developer Hub (MQDH) [42] collects system-level telemetry using the OVR Metrics Tool and Android Logcat [43] for high-precision Unity timestamps, and an asynchronous server-side log for Python processing metrics. Table II lists the measured performance metrics.

TABLE II
PERFORMANCE AND SYSTEM METRICS

| Category | Metric | Description |
|---|---|---|
| Latency | End-to-End Latency (Total_Wait_ms) | Total time from request to visual confirmation. |
| | Network Latency (RTT) | Round-trip delay excluding server processing. |
| | Server Processing (Train + SHAP) | ML training and SHAP explanation time. |
| | Client Rendering (Unity_Engine_ms) | 3D instantiation and rendering time. |
| | Data Parsing (Unity_Parsing_ms) | JSON deserialization on the headset. |
| Visual Perf. | Avg Frame Rate (Avg_FPS) | Sustained rendering smoothness (target: 72 Hz). |
| | 1% Low FPS | Extreme frame drops (1st percentile). |
| | Critical Frame Latency (CFL_ms) | Worst-case frame time (visual stutter). |
| | Geometry Complexity | Average vertex count per frame. |
| Hardware | CPU Utilization | Average CPU load (%). |
| | GPU Utilization | Rendering workload (%). |
| | Memory Footprint | RAM usage (PSS, MB). |
| | Energy Consumption | Instantaneous power draw (W). |
| | Thermal Stress | Battery temperature (°C). |

## IV. RESULTS AND DISCUSSION

This section presents the technical validation of the proposed system, focusing on latency, visual rendering performance, and hardware efficiency metrics. Table III summarizes the experimental results.

To analyze system latency, we decomposed the end-to-end pipeline into three stages: server-side computation, network communication, and client-side rendering. This decomposition allows identifying whether delays originate from Python-based model training and SHAP computation, network RTT between Python and Unity, or Unity instantiation and rendering. The results show that server-side latency increases with dataset size due to the computational cost of calculating SHAP values. In contrast, network latency depends on payload size and remains stable during transmission, as the dataset is transferred in a single response. The maximum observed average RTT was 545.08 ms for the full California Housing dataset, indicating that the asynchronous WebSocket architecture supports high-density data transfer. Client-side rendering time increases with the number of points, from 28.67 ms for 150 points (Iris dataset) to 4,415.67 ms for 20,640 points (California Housing dataset).

Overall, server-side SHAP computation is the primary contributor to end-to-end latency, while network communication and client-side processing (parsing and rendering) have a comparatively lower impact. Fig. 4 reports the latency components and payload sizes for each dataset.

To evaluate scalability, we analyzed frame rate degradation as the number of rendered points increased, relating $N$ to Avg_FPS and total wait time. The system maintains

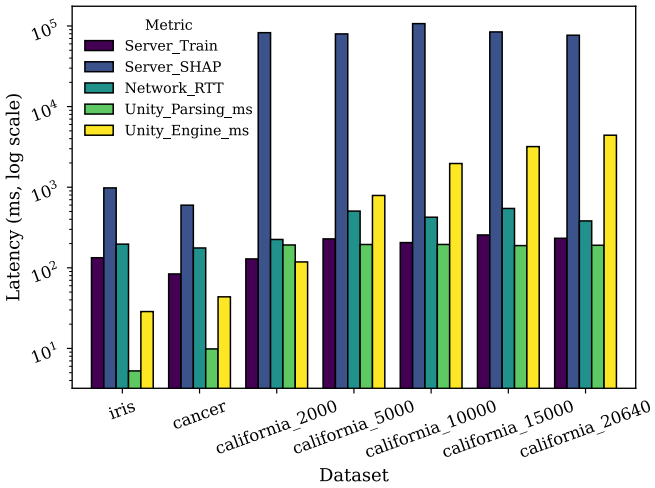



Fig. 4. End-to-end latency decomposition (Pipeline stages).

performance near the device refresh target (72 FPS) up to $N = 2{,}000$ (70.17 FPS). Beyond this point, performance declines, reaching 48.47 FPS at $N = 5{,}000$, below commonly reported VR thresholds (≈60 FPS). These results suggest a practical scalability limit of around 2,000 rendered GameObjects on the Meta Quest 3. End-to-end latency remains high due to server-side SHAP computation (average total wait of 88,778.98 ms for larger datasets). The divergence between stable latency and decreasing FPS indicates that on-device GameObject instantiation is a bottleneck. A non-linear FPS increase at $N = 15{,}000$ (64.09 FPS) is observed, which may be associated with the reactive resource management and clock-scaling mechanisms of the standalone HMD under sustained load, and performance does not recover to the target level. These scalability limitations affect temporal stability. To verify this, we analyzed system behavior as the number of rendered points increased, using both average FPS and 1% Low FPS, which captures short-term frame rate drops associated with rendering irregularities (Fig. 5).

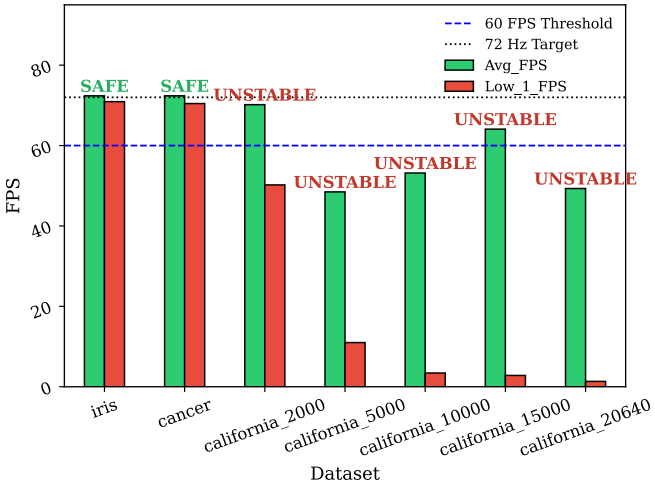



Fig. 5. Stability validation (avg FPS & 1%Low FPS).

At low point densities, the system maintains an average FPS

TABLE III
END-TO-END LATENCY, VISUAL PERFORMANCE, AND HARDWARE UTILIZATION METRICS PER DATASET (MEAN ± STD)

| Dataset | Total Wait (ms) | | Avg_FPS | | 1% Low FPS | | CPU Util (%) | | GPU Util (%) | | Power (W) | | Temp (°C) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std | mean | std |
| Iris | 2004.4 | 180.5 | 72.37 | 0.05 | 70.93 | 0.13 | 26.81 | 2.79 | 24.98 | 0.85 | 2.37 | 0.26 | 33.10 | 5.24 |
| Cancer | 1459.6 | 20.2 | 72.37 | 0.07 | 70.45 | 0.45 | 35.60 | 2.37 | 39.33 | 3.20 | 2.83 | 0.10 | 37.88 | 2.14 |
| California_2000 | 77877.1 | 37555.7 | 70.17 | 0.94 | 50.21 | 5.91 | 50.60 | 0.82 | 71.65 | 2.21 | 3.19 | 0.12 | 40.13 | 0.12 |
| California_5000 | 82558.2 | 24657.6 | 48.47 | 5.68 | 10.99 | 3.60 | 55.59 | 3.86 | 81.18 | 7.56 | 3.69 | 0.50 | 38.79 | 1.80 |
| California_10000 | 111083.9 | 5993.0 | 53.17 | 5.00 | 3.41 | 1.95 | 55.39 | 0.63 | 83.65 | 3.08 | 3.10 | 0.23 | 39.54 | 1.37 |
| California_15000 | 89485.5 | 16790.0 | 64.09 | 6.42 | 2.82 | 0.74 | 60.92 | 0.66 | 76.79 | 5.24 | 2.70 | 0.34 | 38.94 | 0.70 |
| California_20640 | 82890.1 | 22805.0 | 49.31 | 2.53 | 1.33 | 0.58 | 67.00 | 6.64 | 84.31 | 2.84 | 4.01 | 0.78 | 38.61 | 1.74 |

close to the device target, with a slight reduction observed at $N = 2{,}000$ (70.1 FPS). As the number of rendered points increases to $N = 5{,}000$, average FPS decreases to 48.47, accompanied by a drop in the 1% Low FPS to 10.99, indicating increased frame time variability. For $N \geq 10{,}000$, both metrics continue to degrade, with a widening gap that reflects the growing computational and rendering load. These results suggest that GameObject-based rendering introduces scalability constraints on standalone VR devices beyond approximately 5,000 points.

To determine the main causes of instability, we analyzed the effect of geometric complexity on GPU behavior relating Avg_Vertices with GPU_Load metrics (Fig. 6).

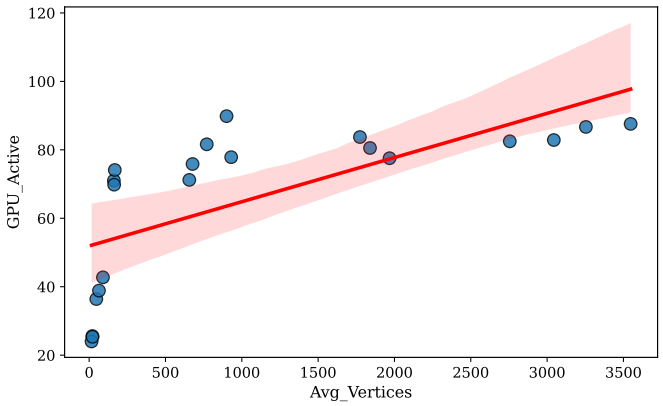



Fig. 6. Rendering impact (Geometry complexity vs GPU utilization).

GPU_Load rises from a baseline of 20–25%, reflecting inherent VR system overhead, to over 85% as vertex density increases, confirming that point-cloud rendering imposes substantial graphics costs. Regression analysis shows a clear scaling trend, while the confidence interval at higher vertex counts indicates greater variability in performance. At high densities, GPU utilization nears saturation, operating close to the Meta Quest 3's thermal and processing limits. These findings indicate that geometric complexity is a key factor causing GPU stress and frame-rate drops in high-density visualizations.

The impact of interaction on system stability, measured by relating CFL to the 1% Low FPS metric, shows an inverse relationship (Fig. 7). A critical threshold occurs at approximately 16 ms, beyond which interaction spikes caused by ray–interactor collisions reduce performance from around 70 FPS to below 10 FPS (Table III), resulting in severe stuttering. High-CFL samples cluster during periods of intensive interaction, suggesting main-thread blocking due to synchronous raycasting and tooltip updates. Although GPU rendering remains stable, CPU stalls delay frame submission, producing perceptual judder. The widening confidence interval further indicates unstable behavior under interaction load. Overall, these results indicate that synchronous interaction tasks on the Unity main thread, particularly ray-interactor collisions, constitute the primary bottleneck affecting visual stability during exploration.

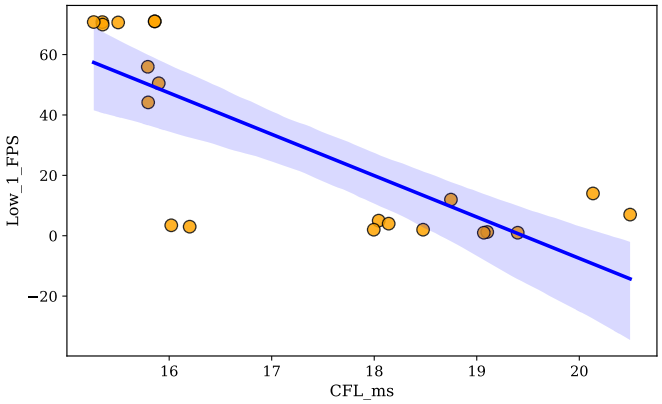



Fig. 7. Interaction Impact (CFL_ms vs 1% Low FPS).

To identify the underlying cause of this interaction-induced instability, we analyzed the processor cost of transitioning from an idle pipeline state (where the application is running, but points have yet to appear) to an active interactive visualization, in which spheres appear in 3D space, and the user begins interacting with them. Fig. 8 illustrates this interaction cost by isolating the hardware stress jump between phases. CPU utilization remains stable at 20–23% across all tests during the idle phase, indicating that communication and engine initialization remain efficient and independent of data size. In contrast, the active interaction phase increases CPU load, which scales with the number of rendered points, rising from 26.81% for 150 points (Iris) to 67% for $N = 20{,}640$ points (California Housing), leading to a thermal increase of up to 39.54°C. The sustained high-density interaction pushes the hardware toward its thermal throttling threshold. Since server-side processing completes before this phase, the additional overhead is due to Unity main-thread tasks: object hierarchy management and ray–interactor collision detection.

### A. Discussion

The system implements a 3D extension of the SHAP dependence scatter plot that preserves the semantics of the original

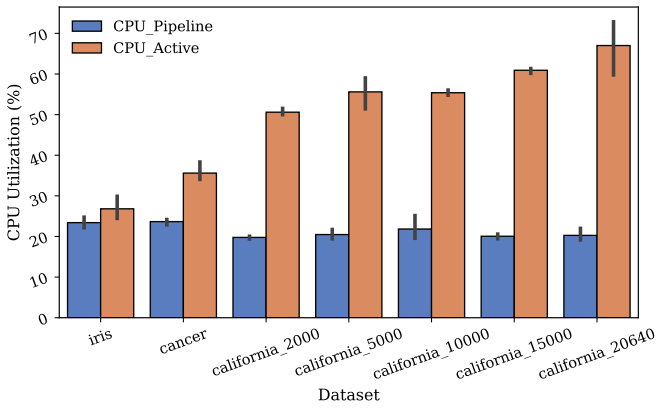



Fig. 8. CPU load expansion (Pipeline & Active Phase).

2D representation: the X-axis encodes feature values, the Y-axis encodes SHAP values, and the interaction maps to both color and depth (Z-axis). This design supports the exploration of multivariate relationships, including clustering patterns, distribution dispersion, and potential non-linear interactions, while maintaining consistency with the original explanation. However, simpler analytical tasks may remain more efficient in 2D representations. The additional spatial dimension reduces visual overlap and supports localized inspection through spatial separation, but it also introduces challenges, including occlusion and perceptual distortions inherent to immersive environments. Rendering constraints, data transfer latency, and interaction precision can further affect performance. The modular architecture supports extensibility to other SHAP plots, although this work focuses on SHAP dependence scatter plots, as extending additional visualizations requires redefining their semantic mapping in 3D.

From a systems perspective, the results show that the main constraints on performance are CPU-bound interactions and object management. The analysis shows that the system is primarily limited by CPU processing and interaction-related overhead, with the Unity main thread being the primary bottleneck. GPU utilization increases predictably with visual density but does not cause significant performance degradation, while network transmission is less significant. A key finding is the significant drop in FPS, reflected in the gap between the average FPS and the 1% Low FPS (31.40 FPS). This gap indicates that interaction-heavy tasks, such as raycast-based selection and dynamic tooltips, exceed the 13.8 ms frame time budget required for stable performance near the 72 Hz device target. Consequently, frame instability can occur even when the average frame rate stays within nominal performance ranges.

The results characterize the system's operational range for datasets with $N \leq 2{,}000$. At this density, the system achieves an average of 70.17 FPS, remaining close to the 72 Hz refresh rate target of the Meta Quest 3 and above, as well as to the commonly reported minimum technical performance thresholds for VR systems ($\approx$ 60 FPS). With 71.65% GPU utilization, it reflects a well-balanced trade-off between visual fidelity and hardware capacity. Although the 1% Low FPS drops to 50.21, falling below stable rendering performance ranges, the thermal and energy profiles remain stable, averaging 40.13°C and 3.19 W, respectively. These metrics suggest stable thermal and energy behavior without triggering thermal throttling. These results further suggest that residual instability arises from interaction logic and CPU-bound thread management (the overhead of Unity's GameObject system), confirming that the current architecture operates consistently within the identified density limits.

The identified bottlenecks suggest opportunities for optimization. The current implementation relies on the synchronous instantiation of individual GameObjects, in which each data point includes a mesh, a renderer, and a collider for interaction. Alternative approaches could improve performance at higher data densities. GPU instancing or point-cloud rendering may shift workload to the GPU by enabling multiple meshes to be rendered in a single draw call using positional and color buffers. Data-oriented approaches, such as Unity's ECS/DOTS, may organize data in contiguous memory and enable parallel processing across CPU cores. Additionally, compute shaders combined with spatial indexing structures (e.g., octrees) may offload ray–point intersection to the GPU while restricting interaction checks to relevant regions. Together, these strategies may reduce CPU overhead and improve performance in future iterations of the system.

## V. Conclusions and Future Works

This work focuses on XAI & IA integration, developing ImmersiveSHAP, an immersive analytics visualization system that extends traditional 2D SHAP scatter dependence plots into interactive 3D visualizations in VR. The system integrates Python-based SHAP computation with Unity-based rendering and interaction through a modular client–server architecture. The WebSocket protocol enables communication between the Python server and the Unity client, supporting both localhost (on the same computer) and wireless deployment on HMDs such as the Meta Quest 3. The current research focuses on SHAP dependence scatter plots. While the pipeline is extensible, generalization to other SHAP visualizations remains future work.

The proposed system demonstrates the technical viability of delivering precomputed SHAP-based explanations from a preprocessing backend to an immersive visualization environment, thereby enabling interactive exploration without incurring significant network overhead. Experimental validation shows that the system operates as a technically validated reference architecture for XAI & IA visualization, supporting the synchronized delivery and rendering of post hoc explanations in VR environments.

The results indicate that the system maintains stable performance on datasets with $N \leq 2,000$, achieving an average frame rate of 70.17 FPS, which remains close to the VR device's 72 Hz refresh target. Although the recorded 1% Low FPS of 50.21 at this density suggests potential short-term instability, it remains near acceptable performance levels. At this operating point, the system reaches 71.65% GPU utilization, signaling an efficient balance between visual quality

and hardware capacity. Thermal and energy profiles remain stable, averaging 40.13°C and 3.19 W, respectively, supporting sustained operation without evidence of thermal throttling or significant battery impact. Identifying interaction-driven bottlenecks provides a clear path to refine synchronous logic and improve performance in future iterations.These findings define a practical scalability boundary and highlight the importance of technical validation in XAI & IA visualization systems.

From a methodological perspective, this work provides a quantitative characterization of system-level performance constraints and establishes a validated baseline for XAI & IA visualization. This technical validation represents a necessary step toward reliable user-centered evaluation, ensuring that future studies assess interpretability rather than artifacts introduced by system limitations.

Future work could address current system limitations by refining interaction logic, improving performance at higher point densities, and extending support to additional SHAP visualizations. Within the validated operational range ($N \leq 2{,}000$ points), the system demonstrates stable performance, providing a technical basis for future proof-of-concept user-centered evaluations. These evaluations may assess user perception, cognitive load, interpretability, and task performance under controlled conditions.

## REFERENCES


[1] D. Gunning, "Darpa's explainable artificial intelligence (xai) program," in *Proceedings of the 24th International Conference on Intelligent User Interfaces*, ser. IUI '19. New York, NY, USA: Association for Computing Machinery, 2019, p. ii, DOI: 10.1145/3301275.3308446.

[2] S. Ali, T. Abuhmed, S. El-Sappagh, K. Muhammad, J. M. Alonso-Moral, R. Confalonieri, R. Guidotti, J. Del Ser, N. Díaz-Rodríguez, and F. Herrera, "Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence," *Information Fusion*, vol. 99, p. 101805, 2023, DOI: 10.1016/j.inffus.2023.101805.

[3] G. Vilone and L. Longo, "Notions of explainability and evaluation approaches for explainable artificial intelligence," *Information Fusion*, vol. 76, pp. 89–106, 2021, DOI: 10.1016/j.inffus.2021.05.009.

[4] M. R. Islam, M. U. Ahmed, S. Barua, and S. Begum, "A systematic review of explainable artificial intelligence in terms of different application domains and tasks," *Applied Sciences*, vol. 12, no. 3, 2022, DOI: 10.3390/app12031353.

[5] A. Saranya and R. Subhashini, "A systematic review of Explainable Artificial Intelligence models and applications: Recent developments and future trends," *Decision Analytics Journal*, vol. 7, 2023, DOI: 10.1016/j.dajour.2023.100230.

[6] A. B. Arrieta, N. Díaz-Rodríguez, J. D. Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 6 2020, DOI: 10.1016/j.inffus.2019.12.012.

[7] G. Vilone and L. Longo, "Classification of explainable artificial intelligence methods through their output formats," *Machine Learning and Knowledge Extraction*, vol. 3, pp. 615–661, 9 2021, DOI: 10.3390/make3030032.

[8] 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*, vol. 10, no. 7, pp. 1–46, 07 2015, DOI: 10.1371/journal.pone.0130140.

[9] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 1135–1144, DOI: 10.1145/2939672.2939778.

[10] J. H. Friedman, "Greedy function approximation: A gradient boosting machine." *Annals of Statistics*, vol. 29, pp. 1189–1232, 2001, DOI: 10.1214/aos/1013203451.

[11] J. B. A. G. A. Kapelner and E. Pitkin, "Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation," *Journal of Computational and Graphical Statistics*, vol. 24, pp. 44–65, 2015, DOI: 10.1080/10618600.2014.907095.

[12] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017, DOI: 10.48550/arXiv.1705.07874.

[13] G. Alicioglu and B. Sun, "A survey of visual analytics for explainable artificial intelligence methods," *Computers & Graphics*, vol. 102, pp. 502–520, 2022, DOI: 10.1016/j.cag.2021.09.002.

[14] R. Yu and L. Shi, "A user-based taxonomy for deep learning visualization," *Visual Informatics*, vol. 2, pp. 147–154, 2018, DOI: 10.1016/j.visinf.2018.09.001.

[15] T. Chandler, M. Cordeil, T. Czauderna, T. Dwyer, J. Glowacki, C. Goncu, M. Klapperstueck, K. Klein, K. Marriott, F. Schreiber, and E. Wilson, "Immersive analytics," in *2015 Big Data Visual Analytics (BDVA)*, 2015, pp. 1–8, DOI: 10.1109/BDVA.2015.7314296.

[16] C. Linse, H. Alshazly, and T. Martinetz, "A walk in the black-box: 3d visualization of large neural networks in virtual reality," *Neural Computing and Applications*, vol. 34, pp. 21 237–21 252, 12 2022, DOI: 10.1007/s00521-022-07608-4.

[17] A. Aamir, M. Tamosiunaite, and F. Wörgötter, "Caffe2unity: Immersive visualization and interpretation of deep neural networks," *Electronics*, vol. 11, no. 1, 2022, DOI: 10.3390/electronics11010083.

[18] C. Maathuis, M. A. Cidota, D. Datcu, and L. Marin, "Integrating explainable artificial intelligence in extended reality environments: A systematic survey," *Mathematics*, vol. 13, no. 2, 2025, DOI: 10.3390/math13020290.

[19] M. Inkarbekov, R. Monahan, and B. A. Pearlmutter, "Visualization of ai systems in virtual reality: A comprehensive review," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 8, 2023, DOI: 10.14569/ijacsa.2023.0140805.

[20] E. H. Korkut and E. Surer, "Visualization in virtual reality: a systematic review," *Virtual Real.*, vol. 27, no. 2, p. 1447–1480, Jan. 2023, DOI: 10.1007/s10055-023-00753-8.

[21] J. A. Wagner Filho, M. F. Rey, C. M. D. S. Freitas, and L. Nedel, "Immersive visualization of abstract information: An evaluation on dimensionally-reduced data scatterplots," in *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2018, pp. 483–490, DOI: 10.1109/VR.2018.8447558.

[22] M. Benk, R. P. Weibel, S. Feuerriegel, and A. Ferrario, ""is it my turn?": Assessing teamwork and taskwork in collaborative immersive analytics," *Proc. ACM Hum.-Comput. Interact.*, vol. 6, 11 2022, DOI: 10.1145/3555580.

[23] S. Bobek, S. K. Tadeja, Łukasz Struski, P. Stachura, T. Kipouros, J. Tabor, G. J. Nalepa, and P. O. Kristensson, "Virtual reality-based parallel coordinates plots enhanced with explainable ai and data-science analytics for decision-making processes," *Applied Sciences*, vol. 12, 2022, DOI: 10.3390/app12010331.

[24] B. Ens, B. Bach, M. Cordeil, U. Engelke, M. Serrano, W. Willett, A. Prouzeau, C. Anthes, W. Büschel, C. Dunne, T. Dwyer, J. Grubert, J. H. Haga, N. Kirshenbaum, D. Kobayashi, T. Lin, M. Olaosebikan, F. Pointecker, D. Saffo, N. Saquib, D. Schmalstieg, D. A. Szafir, M. Whitlock, and Y. Yang, "Grand challenges in immersive analytics," in *Conference on Human Factors in Computing Systems - Proceedings*. Association for Computing Machinery, 5 2021, DOI: 10.1145/3411764.3446866.

[25] R. Hackathorn and T. Margolis, "Immersive analytics: Building virtual data worlds for collaborative decision support," in *2016 Workshop on Immersive Analytics (IA)*, 2016, pp. 44–47, DOI: 10.1109/IMMERSIVE.2016.7932382.

[26] A. Fonnet and Y. Prie, "Survey of immersive analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, pp. 2101–2122, 3 2021, DOI: 10.1109/TVCG.2019.2929033.

[27] M. Kraus, J. Fuchs, B. Sommer, K. Klein, U. Engelke, D. Keim, and F. Schreiber, "Immersive analytics with abstract 3d visualizations: A survey," *Computer Graphics Forum*, vol. 41, pp. 201–229, 2 2022, DOI: 10.1111/cgf.14430.

[28] J. Montilla-López and G. Ramírez-González, "Análisis inmersivo para inteligencia artificial explicable: un enfoque conceptual," in *2025 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2025, pp. 1–6, DOI: 10.1109/COLCOM66267.2025.11185789.

[29] F. A. Pedroso and P. D. P. Costa, "Immvis: Bridging data analytics and immersive visualisation," in *VISIGRAPP 2021 - Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and*

*Computer Graphics Theory and Applications*, vol. 3. SciTePress, 2021, pp. 181–187, DOI: 10.5220/0010256001810187.
[30] S. Lundberg, "Shap: A game theoretic approach to explain the output of any machine learning model," 2018, gitHub repository. Accessed: 2026-01-09. [Online]. Available: https://github.com/shap/shap
[31] A. V. Ponce-Bobadilla, V. Schmitt, C. S. Maier, S. Mensing, and S. Stodtmann, "Practical guide to shap analysis: Explaining supervised machine learning model predictions in drug development," *Clinical and Translational Science*, vol. 17, no. 11, p. e70056, 2024, DOI: 10.1111/cts.70056.
[32] T. Dwyer, K. Marriott, T. Isenberg, K. Klein, N. Riche, F. Schreiber, W. Stuerzlinger, and B. H. Thomas, *Immersive Analytics: An Introduction*. Cham: Springer International Publishing, 2018, pp. 1–23, DOI: 10.1007/978-3-030-01388-2_1.
[33] M. O. Anwer and E. Zagrouba, "A comprehensive evaluation framework for Virtual Reality applications in biological data visualization: Metrics, validation, and expert assessment," Jul. 2025, DOI: 10.21203/rs.3.rs-7132029/v1.
[34] A. Geris, B. Cukurbasi, M. Kilinc, and O. Teke, "Balancing performance and comfort in virtual reality: A study of fps, latency, and batch values," *Software: Practice and Experience*, vol. 54, no. 12, pp. 2336–2348, 2024, DOI: 10.1002/spe.3356.
[35] M. Cordeil, A. Cunningham, B. Bach, C. Hurter, B. H. Thomas, K. Marriott, and T. Dwyer, "Iatk: An immersive analytics toolkit," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2019, pp. 200–209, DOI: 10.1109/VR.2019.8797978.
[36] R. Sicat, J. Li, J. Choi, M. Cordeil, W.-K. Jeong, B. Bach, and H. Pfister, "Dxr: A toolkit for building immersive data visualizations," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, pp. 715–725, 1 2019, DOI: 10.1109/TVCG.2018.2865152.
[37] J Montilla-López, "Immersiveshap," 2026, gitHub repository. [Online]. Available: https://github.com/JulianaMontillaLopez/ImmersiveSHAP
[38] R. A. FISHER, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936, DOI: 10.1111/j.1469-1809.1936.tb02137.x.
[39] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, "Nuclear feature extraction for breast tumor diagnosis," in *Biomedical Image Processing and Biomedical Visualization*, R. S. Acharya and D. B. Goldgof, Eds., vol. 1905, International Society for Optics and Photonics. SPIE, 1993, pp. 861 – 870, DOI: 10.1117/12.148698.
[40] R. Kelley Pace and R. Barry, "Sparse spatial autoregressions," *Statistics & Probability Letters*, vol. 33, no. 3, pp. 291–297, 1997, DOI: 10.1016/S0167-7152(96)00140-X.
[41] E. Dreyer, "Nativewebsocket: A simple, fast websocket library for unity and webgl," 2026, gitHub repository. Accessed: 2026-01-09. [Online]. Available: https://github.com/endel/NativeWebSocket
[42] Meta Horizon Developers, "Meta horizon documentation: Unity — meta quest developer hub (mqdh)," 2026, official online documentation. Accessed: 2026-01-09. [Online]. Available: https://developers.meta.com/horizon/documentation/unity/ts-mqdh/
[43] Unity Technologies, "Android logcat package documentation (com.unity.mobile.android-logcat@1.4)," 2026, official Unity documentation. Accessed: 2026-01-09. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.mobile.android-logcat@1.4/manual/index.html

**J Montilla-López** received the B.Sc. degree in Industrial Automation Engineering in 2019. She is currently a M.Sc. student in telematics engineering at the Universidad del Cauca in Colombia. Her research interests include power systems, AI, data visualization, and VR.

**Daniel Valencia** received the B.Sc. degree in Physical Engineering in 2016 and Industrial Automation Engineering in 2018, and M.Sc. degree in Automatic in 2025, from Universidad del Cauca, Colombia. His research interests include expertise in control, mathematical modeling, machine vision, and discrete-event systems.

**Jovani A. Jimenez-Builes** received the B.Sc. degree in Computer Education from the Universidad de Medellín, Medellín, Colombia, in 1997, the M.Sc. degree in Systems Engineering in 2002, and the Ph.D. degree in Systems Engineering in 2006, both from the Universidad Nacional de Colombia, Medellín. He also holds an M.Ed. degree from the Universidad de Medellín and M.Sc. and Specialization degrees in Neuropsychology and Education from UNIR. He is currently a Full Professor with the Facultad de Minas, Universidad Nacional de Colombia, Medellín, and is pursuing a Ph.D. degree in Management at Universidad EAFIT. His research interests include artificial intelligence in education, educational robotics, systems engineering, and e-learning.

**Gustavo Ramirez-Gonzalez** received the B.Sc. degree in Electronic and Telecommunications Engineering and the M.Sc. degree in Telematics Engineering from the Universidad del Cauca, Colombia, in 2001, and the Ph.D. degree in Telematics Engineering from Universidad Carlos III de Madrid, Spain, in 2010. He is currently a Professor and a Researcher in the Department of Telematics at the University of Cauca. His research interests include image processing, secure communication, ML, and the IoT.