VR-ML: Visualization of Machine Learning Models and their Evolution in Virtual Reality

Roy Oberhauser[0000-0002-7606-8226]
Computer Science Dept.
Aalen University
Aalen, Germany
e-mail: roy.oberhauser@hs-aalen.de

Abstract – **Machine Learning (ML) models can be difficult to grasp, and especially for novices or infrequent users. Between its inputs and outputs, various ML models can seem abstract and mathematical and not readily understandable and explorable, especially across diverse models. To make diverse ML models accessible and support comprehension and analysis for further stakeholders, this paper contributes VR-ML, a solution concept for immersive multi-model ML visualization in Virtual Reality (VR). A prototype is used to exhibit its feasibility, demonstrating support for diverse ML model visualizations including Neural Networks, Random Forest, and Naive Bayes. A case-based evaluation provides insights into its capabilities and potential for visualizing diverse ML models immersively and concurrently, including loading, building or configuring, training, comprehension, analysis, exploration, scalability, and highlights collaboration opportunities.**

Keywords – virtual reality; visualization; machine learning; artificial neural networks; naive bayes; random forest.

I. INTRODUCTION

This paper extends our prior work VR-ANN [1], incorporating VR viewpoint support for multiple Machine Learning (ML) models, illustrated with Artificial Neural Networks (ANN), Naive Bayes (NB), and Random Forest (RF) models, and exhibiting support for additional capabilities such as model evolution, history, and comparison.

ML is a subset of Artificial Intelligence (AI) [2] that focuses on having machines learn from data, improving their performance over time without reprogramming. A learning algorithm can optimize its model's parameters to improve its performance, which can improve pattern detection, predictions, decisions, etc. Various models can be applied in the area of ML. Artificial NNs (ANN) [3], referred to as just Neural Networks (NN) in this paper, are inspired by biological NNs and consist of nodes (referred to as artificial neurons) connected via weighted links or edges (i.e., synapses), with other nodes and are aggregated into layers. Numbers are used to represent signals, while a node's activation function determines a neuron's output based on its inputs and their associated link weights. Between the input and output layer are intermediate (hidden) layers. Dense layers are fully-connected to the preceding layer. Dropout layers address overfitting by randomly setting a fraction of the input units to that layer to 0 during training. Hidden layers are any that are neither input nor output layers. Deep Learning (DL) is a branch of ML utilizing deep NNs having multiple hidden layers. Convolutional Neural Networks (CNNs) are a type of DL network especially relevant for image processing. Feedforward NNs (FNNs) are a type of NN where all neurons are fully connected to the next layer with unidirectional information flow. Recurrent NNs (RNNs) are a type of NN for processing sequential data where its order matters. Naive-Bayes (NB) [4] is a supervised probabilistic ML classification algorithm, based on Bayes' theorem, that assumes features are independent and assigns probabilities. Application areas include medical diagnosis, recommendation engines, spam filtering, software defect detection, and sentiment analysis. Random Forest (RF) classification [5] is a supervised ML ensemble method that creates multiple decision trees during training using different random subsets of the data and features, and then combines predictions to produce a final classification based on the majority. Applications areas of RF include credit scoring, medical diagnosis, and fraud detection, and weather forecasting [6]. Each has its strengths: NNs can capture complex nonlinear patterns in large datasets, NB is simple, fast, and effective for high-dimensional data with approximate independence, and RFs offer robust performance with good accuracy and resistance to overfitting. Yet each also has its liabilities: NNs require more data, tuning, and computation and are difficult to interpret, NB assumes strong independence that affects accuracy when features are correlated, while RF can become computationally expensive and less interpretable than NB, while often not achieving the accuracy of well-tuned NNs for complex tasks.

Yet the complexity of ML models can be difficult for various stakeholders to understand, analyze, and trace their evolution. Even seemingly slight configuration changes can affect performance, and a visual comparison of model types could be beneficial in understanding performance differences. An immersive visualization of ML models can make such models accessible for stakeholders and support comprehension, analysis, and learning. While various tools support 2D, there has been relatively little investigation into the potential offered by Virtual Reality (VR). To address the visualization of multiple heterogeneous ML models and their evolution, this paper proposes and investigates an immersive visualization solution concept for diverse ML model types, which, due to VR's spatial scalability, can readily depict multiple models and their evolution concurrently.

In prior work, we investigated the application of VR to various other areas. A selection of our prior VR-related contributions include: in the Software Engineering (SE)

space, VR-SDLC [7] models development lifecycles, VR-Git [8] models Git repositories, VR-DevOps [9] models Continuous Development pipelines, VR-SBOM [10] models Software Bill of Materials (SBOM) and software supply chains, and VR-EA+TCK [11] and VR-EvoEA+BP [12] exemplify enterprise modeling and business processes. Our VR-ANN [1] focused only on ANN models. This paper contributes a solution concept for immersive multi-ML model and model evolution visualization of in VR. A prototype is used to exhibit its feasibility, demonstrating support for diverse ML model visualizations including NN, NB, and RF. A case-based evaluation investigates its capabilities and potential for supporting multiple model and model evolution depiction, comprehension and analysis.

The remainder of this paper is structured as follows: the next section discusses related work. In Section III, our solution is described. Section IV presents our realization and is followed by our evaluation in Section V. And finally, a conclusion is provided.

II. Related Work

In their survey of the application of XR to AI, Hirzle et al. [13] screened 2619 publications (2017-2021) and reviewed 311 in depth. They found only seven papers that applied XR to AI problems (2.3%), five of which visualize AI methods for immersive analytics, or to improve the understanding of neural networks for non-expert users by visualizing them in VR (the rest are discussed below). The authors state XR "methods are promising to facilitate the interaction with neural networks for novices." The 2025 survey by Yim and Su [14] of K-12 learning tools for AI examined 46 papers, but and makes no mention any XR/VR tool. The 2021 survey by Reiners et al. [15] identified 36 papers that combined XR and AI, of which only two were non-domain-specific and related to visualizing DL in XR (discussed below). The survey on AI in VR by Inkarbekov et al. [16] identified a research gap for more user-friendly, intuitive, and adaptable VR tools that can accommodate complex and high-dimensional AI models. From these surveys we conclude that VR-based visualization of AI has not been thoroughly investigated nor readily adopted and remains relatively unexplored.

VR-related NN visualization work includes Bellgardt et al. [17], who depict convolutional ANNs in VR as node-link diagrams by stacking circular layers for a robotic image processing case. InteractML [18] is a node-based tool for creative practitioners to train an ML model for movement interactions in VR using real-time gesture demonstrations. AIive [19] uses a force-directed graph visualization and sonification to enable VR users to manipulate NN parameters via virtual hands and auditory feedback. Towards non-experts, Meissler et al. [20][21] depict a simplified CNN model in 3D in a closed room virtual environment, with information in 2D anchored to different areas of the room. Schreiber and Bock [22] use the Unreal Engine to display a NN in 3D, whereby connections between layers are not visualized. While VR is mentioned in the title, there is no mention of VR or immersion in the paper, which focuses on 3D. Queck et al. [23] create a virtual room in VR with areas providing CNN information. VR4DL [24] can train and test CNNs with a focus on biomedical image classification; it is not intended to be generic for arbitrary CNNs, and is tailored to users with little to no knowledge of ML. DeepVisionVR [25] visualizes CNNs in VR with a focus on image processing. Common 2D NN model visualization tools include (TensorFlow) Deep playground, TensorBoard, Netron, Comet, and neptune.ai.

NB visualization work includes Možina et al. [26], who propose 2D nomograms. Balfer and Bajorath [27] propose an interactive 2D pie chart visualization with feature groups as slices, points representing a feature colored by influence (negative or positive), pole distance corresponding to absolute log odds ratio, and point size for likelihood. For single user assessment in VR simulators, Ferreira et al. [28] propose a double-weighted fuzzy hypergeometric NB network. However, VR visualization aspects are not directly addressed.

RF visualization work includes Welling et al. [29], who propose 3D forest floor visualizations that depict feature contributions and utilize color gradients. Nsch et al. [30] propose 2D botanically-inspired colored binary tree visualizations. iForest [31] is an interactive 2D visualization for analyzing RF models and predictions. ReFINE [32] consists of several 2D RF visualization views: proximity, interactions, and prototypes.

In summary, most prior work has focused on 2D ML visualization and typically focused on a single model. Where immersive VR has been used in conjunction with ML, it is often regarding ML-based assessment of users during simulation and training sessions (such as in medicine). Hence, VR-based immersive ML model visualization with multi-model and evolution support has not been sufficiently investigated and remains relatively unexplored.

III. Solution Concept

Our VR-ML solution concept is shown (highlighted in blue) in the AI area (green) relative to our other prior VR solutions in our conceptual map of Figure 1.

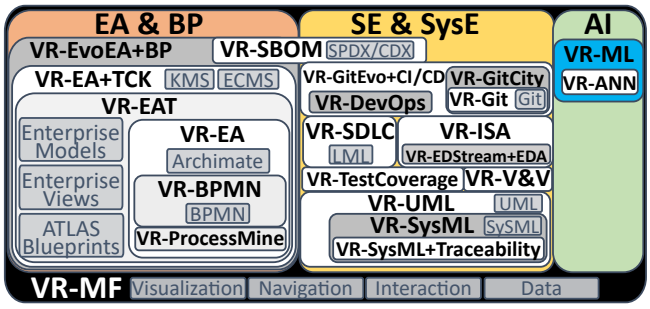

Figure 1. Conceptual map of our various published VR solution concepts with VR-ML highlighted in blue.

Our generalized VR Modeling Framework (VR-MF, black area, detailed in [33]) is the foundation, which provides a domain-independent hypermodeling framework, which addresses the VR aspects of visualization, navigation, interaction, and data integration. Our VR-based solutions specific to the Enterprise Architecture (EA) and Business Process (BP) space (EA & BP, red area) include: VR-EA [33] for mapping EA models to VR, VR-BPMN [34] for BPMN models, VR-EAT for enterprise repository integration, VR-

EA+TCK [11] for knowledge and content integration, and VR-EvoEA+BP [12] for EA evolution and business process animation, VR-ProcessMine, and VR-SBOM [10]. Solutions in the SE and Systems Engineering (SE & SysE, orange area) areas include: VR-Git [8], VR-GitCity [35], VR-DevOps [9], VR-GitEvo+CI/CD [36], VR-V&V (Verification and Validation), VR-TestCoverage, VR-SDLC [3], VR-ISA for informed software architectures, and VR-UML and VR-SysML for software and systems modeling.

A. Solution Concept Grounding in VR-Related Research

As previously described in [36], our rationale for integrating VR into our solution concept stems from existing VR research in fields we consider relevant to modeling, analysis, and collaboration, of which several are summarized here. Akpan and Shanker [37] in their systematic meta-analysis demonstrate that VR and 3D provide marked advantages in discrete event modeling, including model development, analysis, and verification and validation, with common findings pointing to the positive impact of 3D/VR model analysis and V&V. Across 23 studies on 3D analysis, 95% found 3D more effective and conducive to enhanced analysis compared to 2D, notably when assessing model behavior or conducting what-if scenarios; there was broad agreement that 3D/VR effectively communicates results to decision-makers with clarity and conviction; 74% of 19 papers concluded that 3D/VR significantly enhances model development tasks, benefiting team support, and sharpening precision and clarity. In exploring VR applicability for analytical tasks within an information architecture, Narasimha et al. [38] conducted a card sorting collaboration study. Their findings indicated VR matched or exceeded in-person card sorting for certain variables, surpassing both traditional and video-based settings. Qualitative insights on awareness suggested that during collaboration, participants maintained awareness of tasks, people, and the environment, mimicking in-person interactions, with positive perceptions of VR. These findings indicate that VR facilitates a sense of presence and collaboration akin to face-to-face settings. Fonnet and Prie's [39]survey of Immersive Analytics (IA) reviewed 177 papers and found that for complex graph and spatial data, IA offers advantages over non-IA methods, though for multi-dimensional data, benefits vary with the task. They highlight that while IA supports the exploration of extensive data environments, the underutilization of context-aware navigation techniques is problematic, despite their importance to users. Müller et al. [40] compared VR vs. 2D for a software analysis task, finding no significant decrease for VR in comprehension and analysis time. While interaction time was less efficient, VR improved the user experience, was more motivating, less demanding, more inventive/innovative, and more clearly structured.

Consequently, we infer that an immersive, context-rich VR experience holds significant promise for comprehensively visualizing 3D models, while also enhancing comprehension, awareness, analysis, and the inclusion of and collaboration with stakeholders.

B. Visualization in VR

Our solution concept leverages extensible *VR Viewpoints* (VR:VP) to support different forms of immersive visualization for different ML model types and stakeholder concerns. Viewpoints are generic and provide conventions, rules, or templates governing view types, whereas ML views are then specific to a certain ML model and address specific stakeholder concerns. The VR viewpoints offer clarity about how to support the visualization of a certain ML type (kind), while not hindering further additional VR viewpoints to support additional ML types. Our solution concept is extensible, in that it can theoretically support an unlimited set of viewpoints and ML model types, from 1-to-1 through M-to-N relations. Each unique ML model type is mapped to at least one VR Viewpoint. A VR Viewpoint can either be unique to that ML model type, reused (aggregating/supporting multiple types), or adapted (templated and specialized from another similar viewtype, perhaps conceptually superclassed or abstracted). Hence, viewpoints can aggregate ML model types, can readily reuse visualization support from other viewtypes. Moreover, additional crosscutting or specialized ML viewpoints are feasible. Given the unlimited visual space of VR, this enables a much larger scale of multi-ML-model visual support for comprehension, analysis, and collaboration.

To assist with visually delineating the various models from one another, an invisible (glass) 3D Bounding Box (BB) showing outlines is used.

For simplification and for the sake of brevity, we will focus on three common ML types and associated viewpoints to convey our concept, but the concept is extensible with additional VR viewpoints for any given ML model.

VR Viewpoint Neural Network (VR:VP:NN): This viewpoint is concerned with portraying NN models, offering support for depicting construction, configuration, layers, connections, execution, prediction, training, data, versioning, evolution, etc. in VR-based views. To better delimit the context of a NN model in a multi-NN VR space, each NN model is visualized within an outlined 3D transparent Boundary Box (BB). This enables further additional portfolio of NNs to be visualized contemporaneously. Contained within the NN BB are outlined transparent 3D BBs, each representing a NN layer, which vertically delineates a set of spherical nodes (neurons). Lines are used as connectors to indicate which nodes are connected between layers.

VR Viewpoint Naive-Bayes (VR:VP:NB): This viewpoint is concerned with portraying a NB model, offering support for depicting classification, data, configuration, etc. in VR-based views. Visualization utilizes a standard 3D top-down hierarchically-layered tree (model, classes, features).

VR Viewpoint Random Forest (VR:VP:RF): This viewpoint is concerned with portraying RF models, offering support for depicting trees, data, configuration, etc. in VR-based views. As any tree in a RF can be structured differently with different depths, for RF model visualization, vertical 2D trees are radially separated and dispersed, and trees utilize vertical space expanding further out radially, analogous to an explosion.

To support model evolution and version tree visualization, we chose to use the intuitive analogy of growing trees with the root at the bottom and newer versions spreading upwards, with connectors between BBs showing relations. To help visually differentiate changes, green is used to indicate any additions to a model on the new version, while red is used to mark what no longer exists in the next version. Furthermore, panels at the front of a BB are used to retain version-specific information for that BB, enabling detailed parameter, data, or metrics comparisons between versions. These can be collapsed or expanded individually using the affordance or for all via the VR-Tablet. The anchored panels offer clear context association to their model when multiple models are involved, while the VR-Tablet can be used to adjust overall parameters.

C. *Navigation in VR*

Dual navigation modes are incorporated in our solution: default gliding controls for fly-through VR, while teleporting instantly places the camera at a selected position in space. While teleporting can be potentially disconcerting, it may reduce the likelihood of VR sickness.

D. *Interaction in VR*

User-element interaction is supported primarily through VR controllers and the incorporation of a VR-Tablet. The VR-Tablet provides detailed context-specific element information. Various tabs in the VR-Tablet enable support for loading, training, executing, or configuring models and viewing related metrics. Since for our examples no text entry and keyboard were required, a virtual keyboard was not included. However, our implementation can be readily enhanced with a virtual keyboard for text entry using laser pointer key selection, as demonstrated in our other VR solutions. A small control sphere is placed as an affordance on the edge of a BB for dragging, collapsing (to reduce visual clutter), or expanding a BB.

IV. Realization

For our prototype realization, the VR visualization aspects were implemented using Unity in C#, referred to as our frontend, as shown in Figure 2.

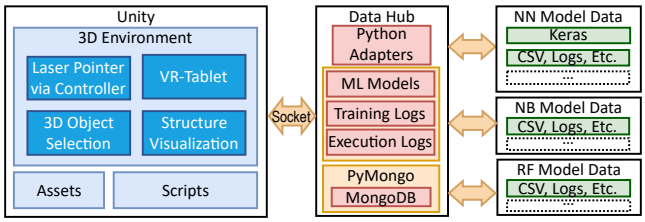

Figure 2. VR-ML logical architecture.

It connects via a Socket to our backend Data Hub, which is based on the hexagonal (or ports-and-adapters) design pattern and is implemented in Python. The Data Hub integrates and stores all data via PyMongo to a MongoDB database in JSON/BSON format utilizing Python adapters to utilize and convert the various ML models, APIs, and data formats to our internal format. This permits the frontend to quickly render its various visualizations. Docker images and containers can be readily used on the backend if available for the various ML models. Various formats and serialization are supported and used, including PKL, CSV, JSON, and Joblib.

For realizing *VR:VP:NN* support, the Keras API was used as a high-level DL API primarily due to its popularity and flexibility, since it supports multiple backends, such as TensorFlow, PyTorch, OpenVINO, and JAX (Python library for high-performance ML). Initially for prototyping, the Sequential model with various layer types placed in sequence is supported, but support for additional models can be readily added. Data can be imported from or exported to the common keras format (.keras extension), which is a zip archive containing JSON-based configuration, H5-based state file containing layers and weights, model weights, and metadata. Additionally, the CSV file format is supported. Further formats can be readily supported via adapters. The models can either be pretrained, loaded, and executed in VR (load and execute), or can be (re)configured and trained in a VR session via our Model Builder support mode.

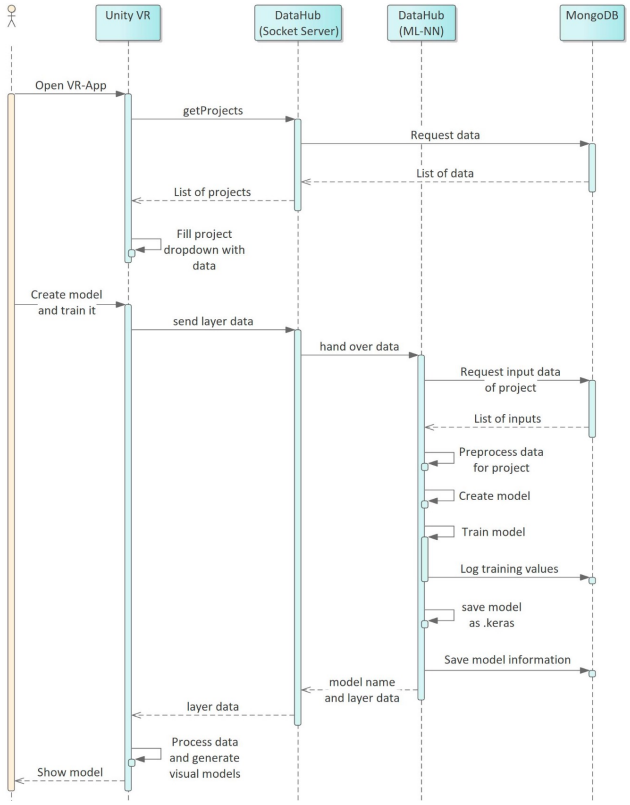

Figure 3. Example VR-ML sequence diagram for a NN training session.

To exemplify the internal dynamics of the architecture, a sequence diagram for a VR-centric NN training session is shown in Figure 3. Initially, the list of available NN projects is retrieved from the Data Hub. If the user creates a new model and starts a training session, then the layer data is sent to the Data Hub, the training inputs are retrieved, and a model is created and trained via the Keras API, training values are logged, and the model is saved or exported. The acquired layer data is returned to Unity via the socket, and is visualized as a model, which can be further analyzed.

Our VR-Tablet supports interaction in VR. This is exemplified for a VR:VP:NN context in the bottom right in Figure 4. The tab groups Load, Training, Execution, Display options, and Graphs are shown, as is the scrollbar. Here, under the selected Display options tab, the epoch slider is shown as well as options to show or color connections. Also shown in this figure, the model layer BBs top side provide layer information (number and type) and metrics (number of neurons) and containing spheres as neurons evenly spaced along a single vertical plane with slots for the next best-fitting square matrix, filled from the bottom left (upper right may have empty slots), while dropout layers are depicted as an opaque slab. Connectors are shown by default in blue; the diameter of the lines indicate the relative weighting to the next layer. In execution, the most active (top five) routes as nodes and connectors are colored (darker to lighter) green. Connectors can optionally be colored (iterated list of 32 colors) to more easily follow a connector between nodes.

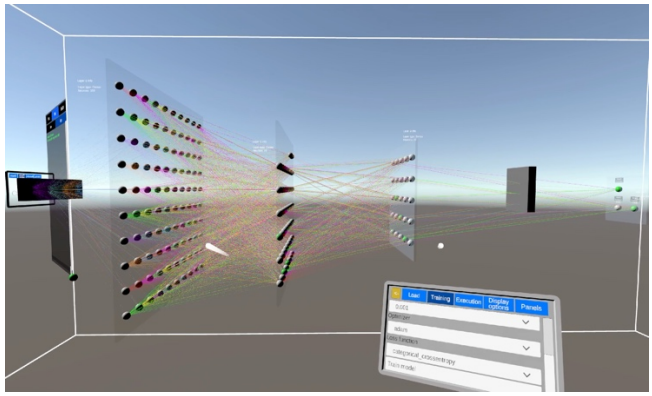

Figure 4. Boundary Box (BB) enclosing a ML model (NN), with the VR-Tablet showing tab groups and input options.

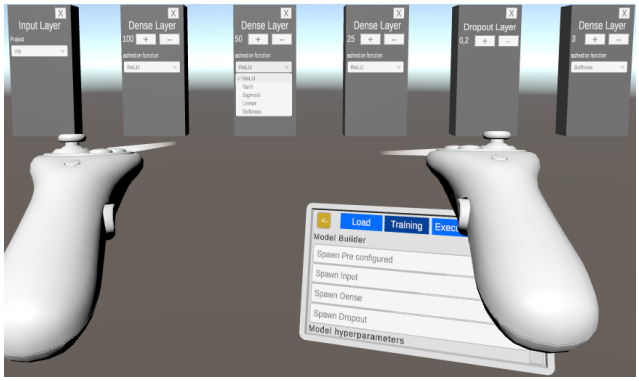

Figure 5. VR:V:NN:MB Model Builder VR View for configuring a NN.

VR:VP:NN also includes support for a Model Builder VR View (VR:V:NN:MB). For this, opaque 3D boxes represent layers ordered from left to right, each of which offers appropriate options based on type, as shown in Figure 5. To avoid requiring text input and simplify interaction, the default values can be adjusted with plus/minus buttons and dropdown lists offer selection options. Layers can be removed by an X button at the top right, and new layer types can be inserted via the VR-Tablet. The ordering of the layers can be adjusted by dragging the boxes. RNN support could be readily included but was not yet implemented. All applicable parameters can be selected and configured in the VR-Tablet, including epochs, learning rate, optimizer, loss function, etc.

VR:VP:RF support in the Python adapters was implemented utilizing the sklearn.ensemble RandomForestClassifier. For RF visualization, the root node of each 2D vertical tree is radially equidistantly dispersed on an invisible horizontal circle near the center of the BB, as shown in Figure 6. Each RF tree can be structured differently and is vertically depicted from the center with nodes and colored connectors. Gray connectors represent RF sequence connection, in the trees white(black) connectors indicate a condition was fulfilled(unfulfilled), blue is used to indicate a selected node for the "Node Inspector", and green is used for paths to follow a prediction. Additional helper nodes are provided near the corners of the BB to gather values from the various trees and support analysis.

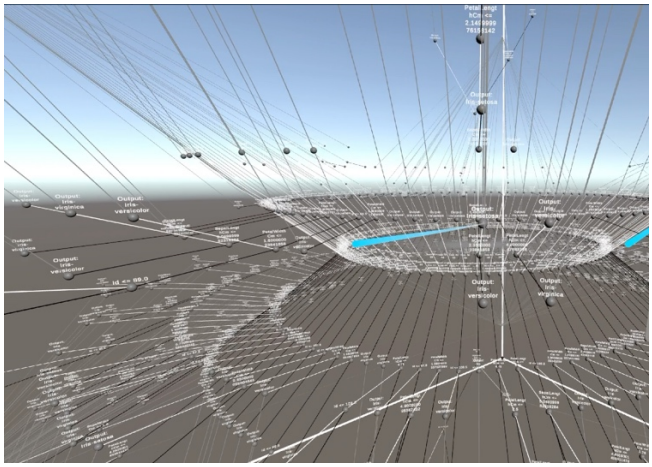

Figure 6. RF model view depicted as vertical trees in radial layout.

VR:VP:RF also includes support for a Model Builder VR View (VR:V:RF:MB), where training aspects for previous trainings is shown as blocks and the associated training parameters are shown and can be adjusted in the VR-Tablet, as in Figure 7. New training parameters can be (individually) set and reset, and training started with a button, as seen in Figure 8.

Figure 7. RF Model Builder VR View (VR:V:RF:MB) showing in the VR-Tablet previous training parameters.

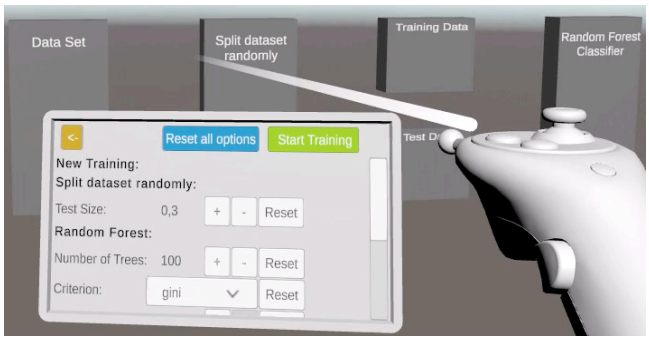

Figure 8. RF Model Builder VR View (VR:V:RF:MB) showing in the VR-Tablet new training parameters.

VR:VP:NB support in the Python adapters was implemented with sklearn.naive_bayes, including support for the NB variants: Gaussian, Bernoulli, Multinomial, Complement, and Categorical. NB visualization was realized as 3D hierarchical layered tree as shown in Figure 9. Blue is used to indicate the currently selected node for node inspection in the VR-Tablet. If it is a class, then the associated features are marked in yellow.

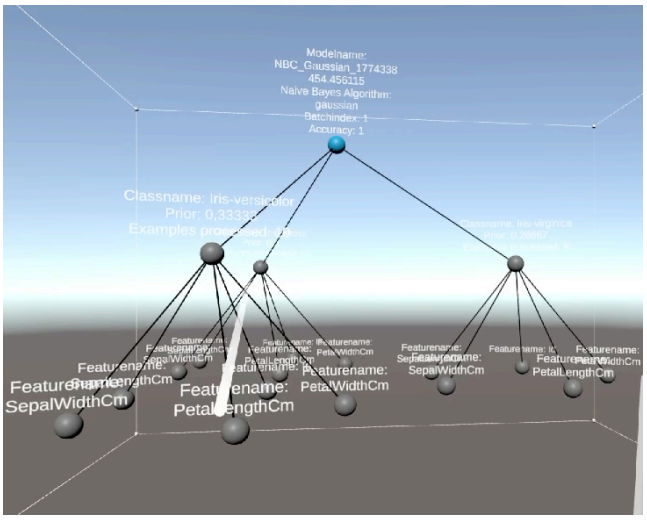

Figure 9. NB model view depicted as 3D hierarchical layered tree.

To support evolution and version differentiation and analysis, chose to anchor information panels and an embedded control panel to the front of a BB. This was done to minimize the risk of user errors when many BBs with only slight differences are involved. The left information panel relates to data flow and operational status (train, test, predict) with subtabs (data, logs) for input and output data and any generated logs. The right panel supports analysis of hyperparameters and performance/metric graphs (accuracy, loss). Furthermore, at the front of each BB an embedded control tablet offers hyperparameter adjustment, and triggering or retraining or predictions directly. That way it is obvious for which BB any hyperparameter change, training, or execution is being enacted.

V. Evaluation

The evaluation of our VR-ML solution concept is based on the design science method and principles [41], in particular a viable artifact, problem relevance, and design evaluation (utility, quality, efficacy). A scenario-based case study is used for each of the implemented VR viewpoints (VR:VP:NB, VR:VP:RF, VR:VP:NN), model evolution, model scaling, and concurrent multi-model support.

For the evaluation dataset, an Iris flower dataset on Kaggle [42] was used. For this, 50 samples each of three Iris species (Setosa, Versicolor, Virginica) are classified in the output based on four properties: SepalLengthCm, SepalWidthCm, PetalLengthCm, PetalWidthCm. One species is linearly separable from the other two, but they are not linearly separable from each other.

A. VR:VP:NB Case

This case demonstrates and assesses VR Viewpoint support for NB models. A NB project and model are loaded via the Load tab in the VR-Tablet. In the Training tab, the training parameters (if already trained) are shown, and training data and applicable parameters can be configured natively in VR, and a training session started via the "Start training" button, as shown in Figure 10.

Figure 10. NB training configuration via VR-Tablet.

The NB is depicted as a 3D tree structure, shown in Figure 11. A single project node at the top offers a clear starting point for navigation to the possible classes (as well as current project parameters), the classes on the mid-level, and the features determining a class on the bottom level. These can be explored to comprehend and analyze a classification result. As shown in the VR-Tablet, the Inspector tab offers additional node-specific information.

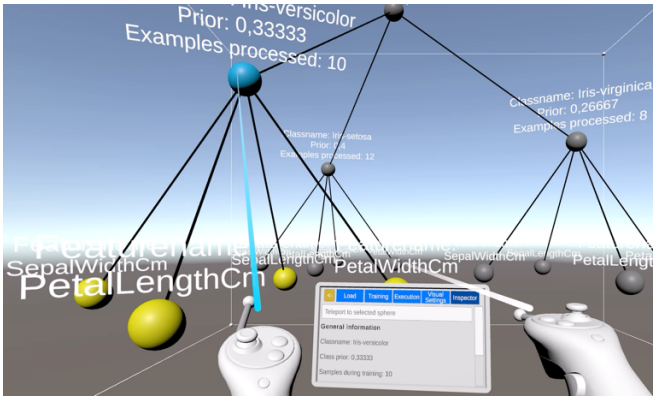

Figure 11. NB node selection colors selected class node blue and corresponding features yellow (Iris-versicolor); node info shown on VR-Tablet.

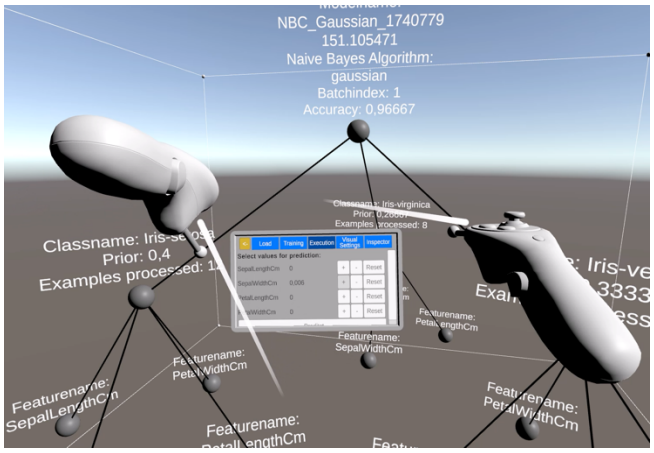

Figure 12. NB input parameters can be natively adjusted before prediction.

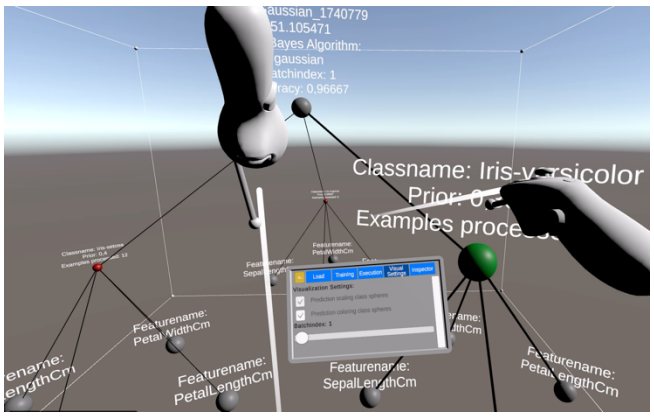

Figure 13. Execution on inputs (left) and result in the output layer (right).

For execution support, a tab in the VR-Tablet supports manually adjusting input parameters, as shown in Figure 12. Prediction results can be visualized and explored, as shown in in Figure 13. Furthermore, visual settings are configurable as shown here: node scaling to indicate probability, and node colorization (green for the most likely class, red for the others).

B. VR:VP:RF

This case demonstrates and assesses VR Viewpoint support for RF models. A RF project is selected and loaded via the VR-Tablet. In the Model Builder View, various training options are selected, as was shown in Figure 8. The resulting model is then depicted with input in the center, as shown in Figure 14. The input is connected (in grey) to the root of every tree radially. An individual tree can be explored, as shown in Figure 15. Nodes are labeled with the tree and condition, gray connectors represent an RF sequence connection, white/black for condition fulfilled/unfulfilled.

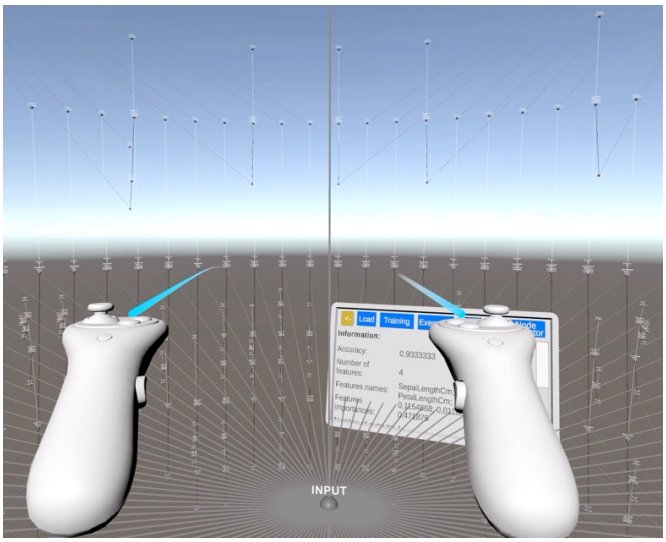

Figure 14. RF model depiction as radially dispersed 2D vertical trees.

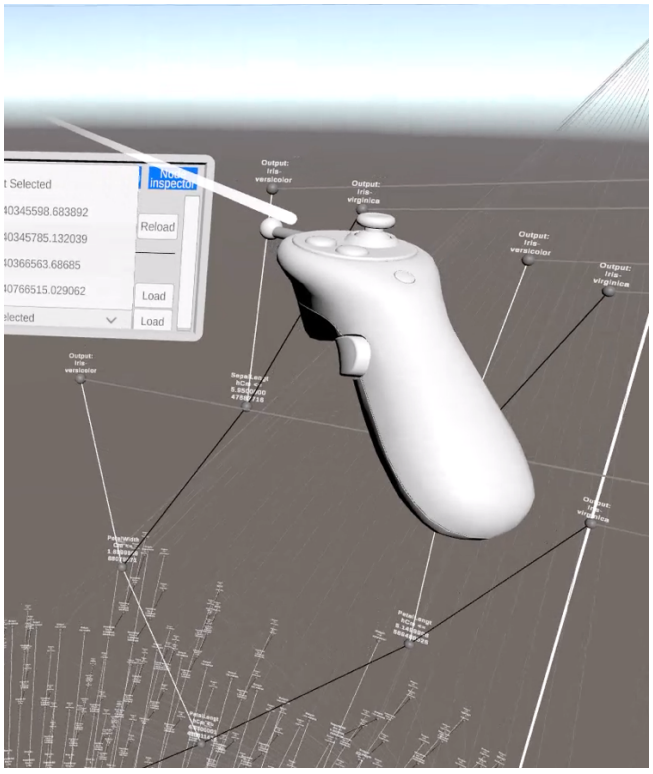

Figure 15. RF tree, each node labeled with tree number and condition with connectors to rest of tree (white if fulfilled, black unfulfilled).

Helper nodes placed near corners of the BB are used to aggregate the tree results via grey connectors to nodes, such as the output collector shown in Figure 16. The majority vote node and resulting final output node are shown in Figure 17.

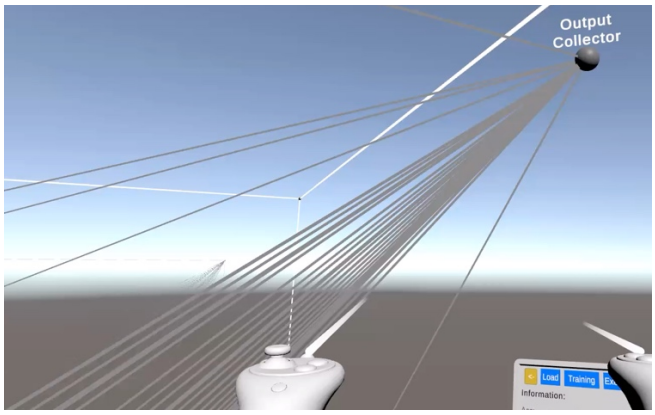

Figure 16. Collector nodes aggregate tree results to support analysis.

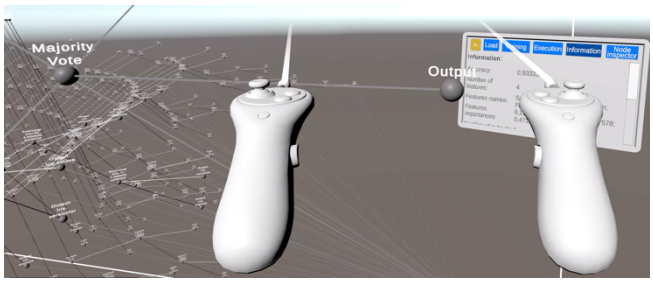

Figure 17. RF depicting majority vote and output nodes.

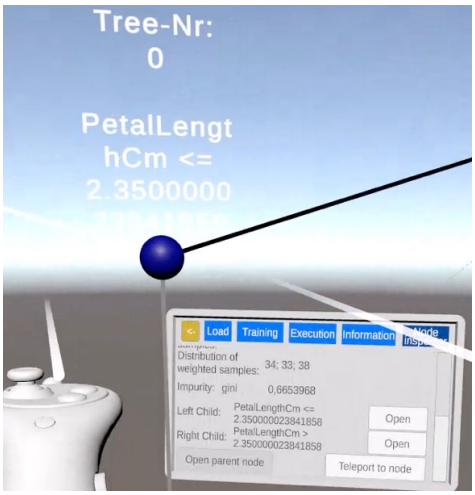

Figure 18. Analyzing RF: selected node information.

To support analysis, an individual node can be selected (blue) and inspected in the VR-Tablet, as shown in Figure 18. It offers information on and navigation to children and parent nodes as well as teleporting.

To support prediction and execution analysis in VR natively, inputs parameters can be adjusted and then the "Run Prediction" button pressed, as shown in Figure 19. Green nodes and branches show which were active. For this case, setosa was predicted by 99 trees, while 1 tree predicted versicolor. To determine which tree, one can iterate thru the tree index to see each tree prediction (80 was setosa), as seen in Figure 21, whereas tree 79 was unique in predicting versicolor, as shown in Figure 22.

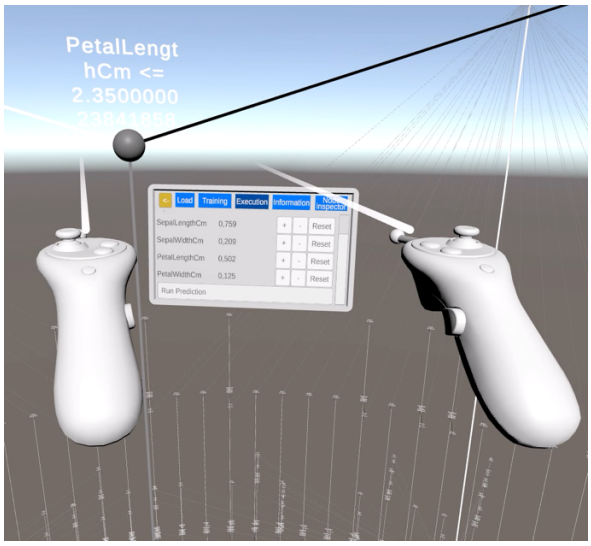

Figure 19. RF: Adjusting inputs in VR to run a prediction.

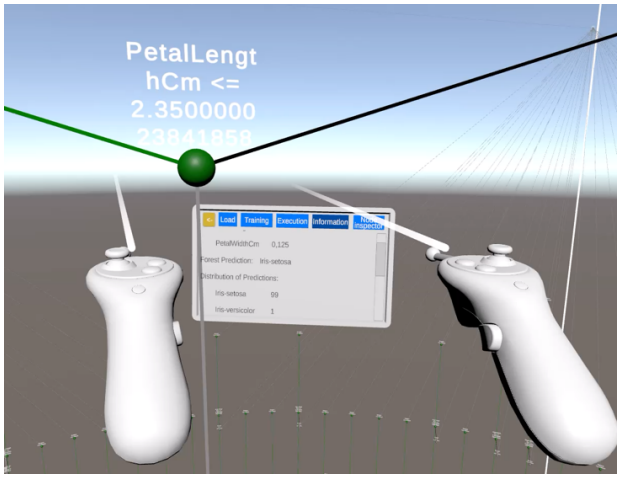

Figure 20. RF execution results in VR-Tablet and green nodes/connectors.

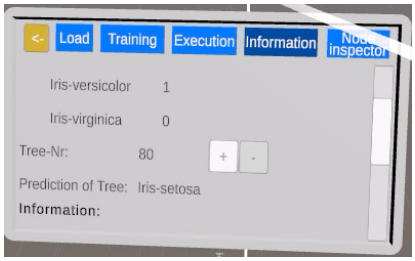

Figure 21. RF analysis: tree 80 predicts Iris-setosa.

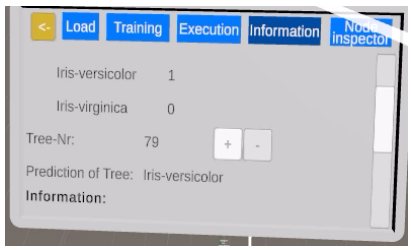

Figure 22. RF analysis: tree 79 predicts Iris-versicolor.

C. VR:VP:NN Case

This case demonstrates and assesses VR Viewpoint support for NNs.

1) Comparison of 2D vs. VR-ML Visualization

A typical 2D NN visual representation is shown via the TensorFlow Playground in Figure 23. A VR View was shown in Figure 4. In both mediums, a NN can be readily investigated and analyzed. Much of the information seen in the 2D playground is available immersively in VR by interacting with the VR-Tablet, elements (layer, nodes), or panels anchored to the BB. Hence, there are no significant impediments to analysis in VR vs. 2D. Given the many NN connections, we contend it to be more straightforward to immersively follow a connector separated spatially in 3D space.

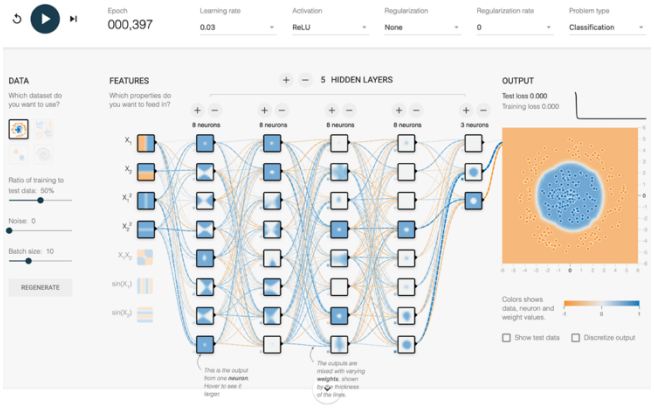

Figure 23. Screenshot of a NN model in TensorFlow playground [43].

2) Build and Train NN Model Support

VR:V:NN:MB support for building and configuring NNs was shown in Figure 5. Via the VR-Tablet, a preconfigured existing model can be loaded and layer types flexibly inserted, removed, or adjusted. After NN training, optionally triggered natively within VR, the training results can be loaded.

3) NN Analysis Support

The model, with all its layer types and neurons, is visualized within a BB as was shown in Figure 4. This helps to readily in comprehend of the total number of layers and their type, size, and ordering. Furthermore, each layer is labeled with its name, type, and size, as shown in Figure 24.

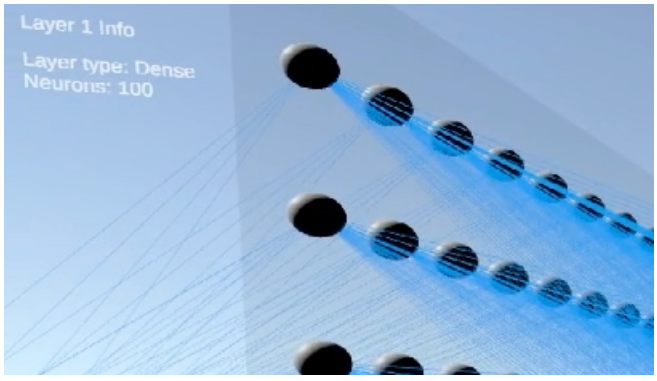

Figure 24. NN layer labels.

To support analysis, optional panels anchored to the front of a BB offer additional model information and insights, as shown in Figure 25. This provides better contextual clarity than a single VR-Tablet alone would offer when multiple models are involved, especially with regard to supporting evolution and version differentiation.

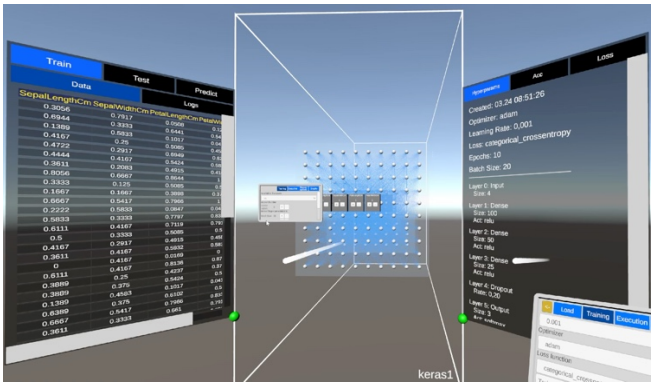

Figure 25. Information panels: data and operational state left (train, test, predict) with subtabs (data, logs), on the right hyperparameters and graphs (accuracy, loss).

The left scrollable information panel relates to data flow and operational status (Train, Test, Predict) with subtabs (Data or Logs). Test Data input (e.g., LengthCm_PetalWidth) and output columns (e.g., true_Species, predicted_Species) are shown in Figure 26. To support prediction analysis, the left panel offers a Predict tab and Data subtab with inputs and outputs as seen in Figure 27. Log output for training, testing, and prediction, can be analyzed, as seen for the Train tab Logs subtab shown in Figure 28.

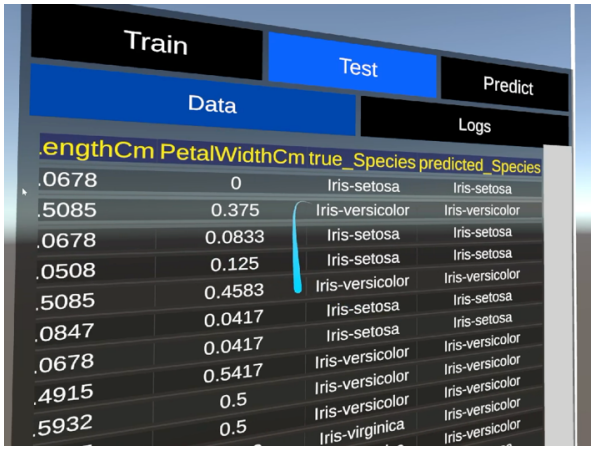

Figure 26. Left panel Test tab with subtab Data.

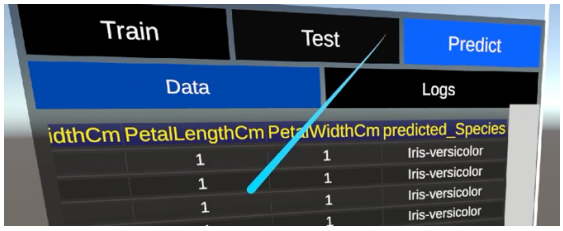

Figure 27. Left panel Predict tab with subtab Data.

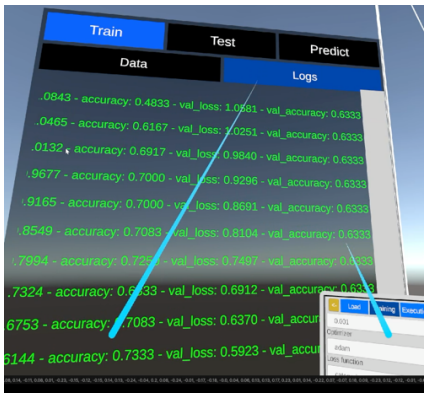

Figure 28. Right panel Train tab with Logs subtab.

After NN execution or prediction, the top five most frequent activation paths (pathways, routes) are indicated via (darker-to-lighter) green nodes and connectors. Within the BB, the model highlights the top five activation nodes and routes in shades of green. Input values for native VR inputs can be adjusted on the front of the BB with +/- buttons, and a prediction run via the front BB control panel, as seen in Figure 29. The output can be viewed the Predict tab Data subtab, or within the BB in the output layer, as shown in Figure 30.

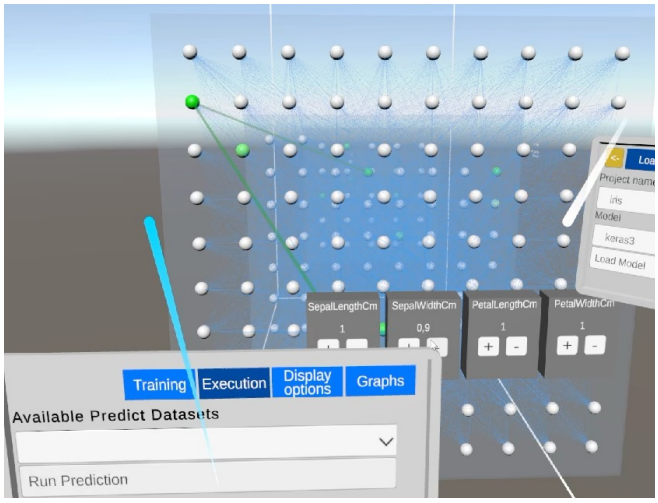

Figure 29. Predict values input values model in BB highlights top five activation nodes/routes in green.

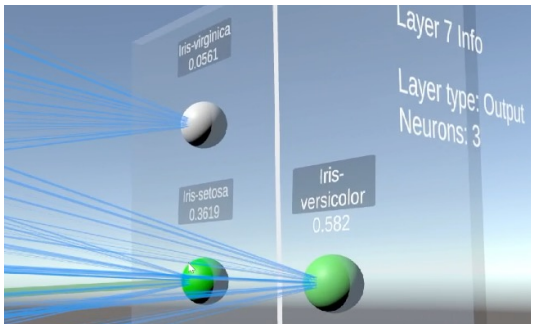

Figure 30. Prediction result output (setosa, 0.3896) depicted on output layer (higher-weighted are larger), top activation paths are highlighted in green.

On the right panel, the Hyperparameters tab supports analysis of hyperparameters, as shown in Figure 31. Furthermore, performance and metric graphs can be viewed in the Accuracy and Loss tabs, as shown in Figure 32. Furthermore, at the front of each BB an embedded control tablet offers hyperparameter adjustment, and triggering or retraining or predictions directly.

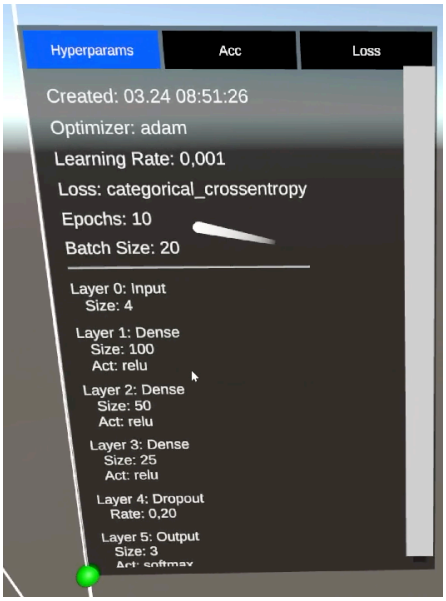

Figure 31. Right panel tabs: Hyperparameters.

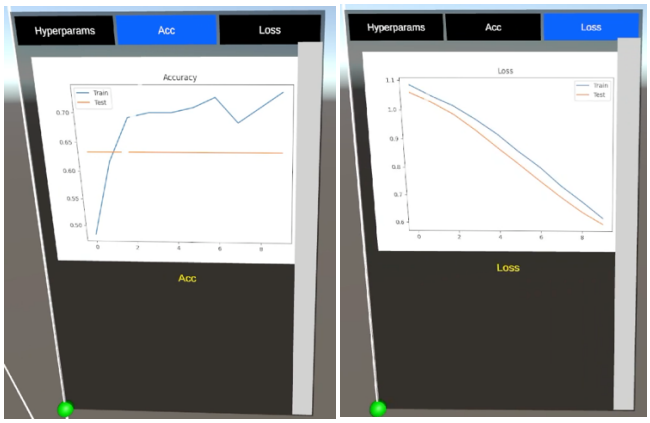

Figure 32. Right panel tabs: Accuracy graph (left) and Loss graph (right).

To support immersive analysis, connections between neurons can be optionally randomly colored for differentiation. To support analyzing a single node, selecting a single node in the model will cause only connections related to that node to be shown and all others to be hidden (to reduce visual distractions), as seen for colorized connections in Figure 33. Furthermore, the weight values of all the connected input nodes are displayed above each input node in the previous layer (higher-weighted connections are visualized larger), and metadata about the selected node is depicted. Also, the VR-Tablet offers an Epoch slider, allowing one to adjust the epoch and see the weights adjust accordingly.

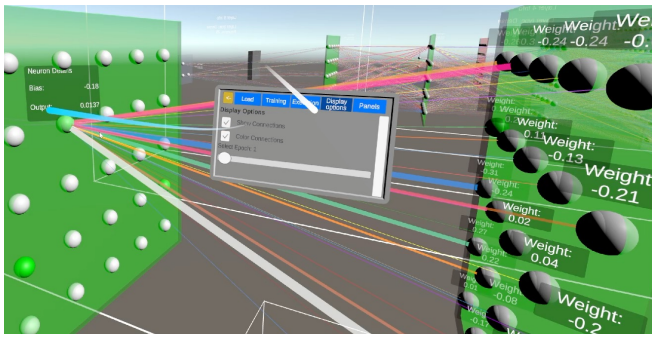

Figure 33. Single node selection displays relevant (colorized) connections with input weight values from previous layer (output connections not seen).

D. *Model Evolution Support*

Model evolution capability support is showcased via NNs to demonstrate its feasibility and illustrate realization features, but can be equivalently realized for other model types.

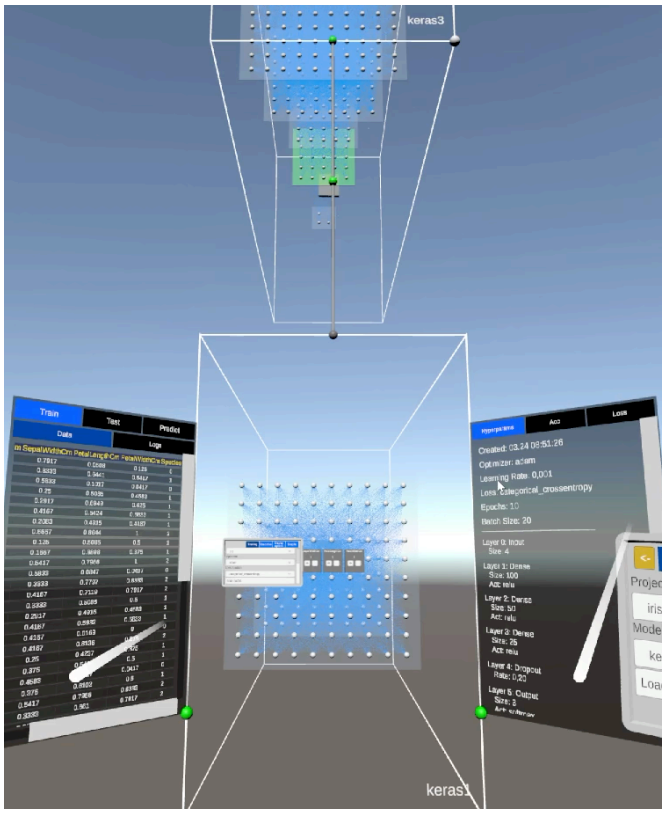

Figure 34. NN model evolution, green/red layers indicate addition/removal.

After changing various parameters in the initial (keras1) model in VR and pressing "Train Model", a new model (keras3) appears above it with a parent-child connector to visual indicate model basis for changes as shown in Figure 34. As seen, a green panel indicates an additional layer vs. the previous (its basis, lower BB) model. Furthermore, the panels also indicate any textual differences to the basis in yellow, as shown in Figure 35. A red panel visually indicates a panel removed in later models (higher BB), as seen in Figure 36.

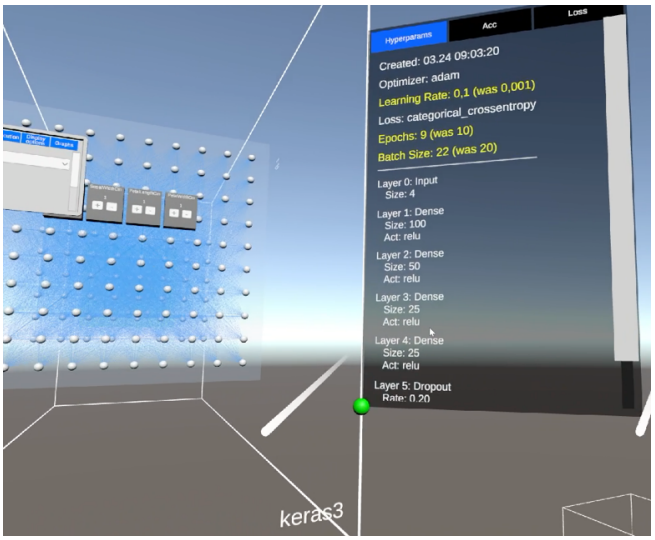

Figure 35. Information panels: text differences to prior version are depicted in yellow.

E. *Scaling Support*

To evaluate the scaling ability of VR to support large NN models, a mid-sized model that consists of ten (8 hidden) layers with 432 neurons counts (4, 100, 50, 25, 100, 50, 25, 50, 25, dropout, 3) was used. A large model consisting of 982 neurons across ten (8 hidden) layers (4, 100, 200, 25, 100, 200, 100, 50, 200, 3) with the connectors was colored. Our principle is to initially depict a model's reality with its inherent complexity, and then let the user decide which items to hide. Via immersive navigation, selection of an element of interest, and display filtering (such as turning off connectors or colorization), visual overload can be addressed. Based on the stakeholder's interest and intentionality, comprehension or issue analysis for large models can be supported and stakeholder collaboration opportunities with a common immersive model utilized.

F. *Concurrent Multi-Model Support*

To demonstrate the visual support for analysis capabilities that VR offers, loading another model starts a separate visual evolution tree, as shown in Figure 37. The capability to load and compare diverse ML models, such as comparing a RF to a NB model, is shown in Figure 38. This can support comprehension and analysis of model differences, output difference analysis, and offer decision support for potential optimizations and model choices. Furthermore, model evolution trees can be concurrently loaded with other models, as seen in Figure 39. This supports the optimization and traceability of models, which can support governance and model transparency for larger organizations managing a plethora of models.

VI. Conclusion

This paper described our VR-ML contribution, a solution concept towards immersive visualization of multiple diverse ML models and their evolution concurrently in VR. The VR-ML solution concept utilizes VR Viewpoints to provide ML

model type-specific guidance for immersive visualization, configuration or building, training, prediction, analysis, and interaction. Our prototype demonstrates its feasibility by realizing various immersive model viewpoints, specifically NB, RF, and NN. The case-based evaluation provides insights into the solution concept's capabilities and potential for immersively supporting VR Viewpoints using NB, RF, and NN, model evolution support, scaling, and concurrent multi-model support. This makes ML models accessible to diverse stakeholders to better comprehend, explore, analyze, and collaborate on diverse ML models and their evolution.

Future work includes realizing support for additional ML models (e.g., RNN, CNN), dynamic enhancements, support for additional ML frameworks and ML formats, and a comprehensive empirical study.

ACKNOWLEDGMENT

The author would like to thank Daniel Godeck, Marc Jaenke, Nils Jaenke, and Lukas Roufos for their assistance with the design, implementation, and screenshots.

REFERENCES

[1] R. Oberhauser, "VR-ANN: Visualization of Artificial Neural Network Models in Virtual Reality," In: Proceedings of The Second International Conference on AI-based Systems and Services (AISyS 2025), IARIA, 2025, pp. 60-65.

[2] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Global Edition, 4th ed. Pearson Education, 2021.

[3] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," Journal of microbiological methods, 43(1), 2000, pp.3-31. doi: 10.1016/S0167-7012(00)00201-3

[4] I. Wickramasinghe and H. Kalutarage, "Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation," Soft computing, 25(3), 2021, pp.2277-2293.

[5] T. K. Ho, "Random decision forests," In: Proceedings of 3rd international conference on document analysis and recognition, vol. 1, IEEE, 1995, pp. 278-282.

[6] A. Ziegler and I. R. König, "Mining data with random forests: current options for real-world applications," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 4(1), Wiley, 2014, pp. 55-63. doi: 10.1002/widm.1114

[7] R. Oberhauser, "VR-SDLC: A Context-Enhanced Life Cycle Visualization of Software-or-Systems Development in Virtual Reality," In: Business Modeling and Software Design (BMSD 2024), LNBIP, vol 523, Springer, Cham, 2024, pp. 112-129, doi: 10.1007/978-3-031-64073-5_8.

[8] R. Oberhauser, "VR-Git: Git Repository Visualization and Immersion in Virtual Reality," 17th Int'l Conf. on Software Engineering Advances (ICSEA 2022), IARIA, 2022, pp. 9-14.

[9] R. Oberhauser, "VR-DevOps: Visualizing and Interacting with DevOps Pipelines in Virtual Reality," Nineteenth International Conference on Software Engineering Advances (ICSEA 2024), IARIA, 2024, pp. 43-48.

[10] R. Oberhauser, "VR-SBOM: Visualization of Software Bill of Materials and Software Supply Chains in Virtual Reality," In: Business Modeling and Software Design (BMSD 2025), LNBIP, vol 559, Springer, Cham, 2025, pp. 52-70, doi: 10.1007/978-3-031-98033-6_4.

[11] R. Oberhauser, M. Baehre, and P. Sousa, "VR-EA+TCK: Visualizing Enterprise Architecture, Content, and Knowledge in Virtual Reality," In: Business Modeling and Software Design (BMSD 2022), LNBIP, vol 453, Springer, 2022, pp. 122-140. doi: 10.1007/978-3-031-11510-3_8.

[12] R. Oberhauser, M. Baehre, and P. Sousa, "VR-EvoEA+BP: Using Virtual Reality to Visualize Enterprise Context Dynamics Related to Enterprise Evolution and Business Processes," In: Business Modeling and Software Design (BMSD 2023), LNBIP, vol 483, Springer, 2023, pp. 110-128, doi: 10.1007/978-3-031-36757-1_7.

[13] T. Hirzle et al., "When XR and AI Meet - A Scoping Review on Extended Reality and Artificial Intelligence," In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), ACM, Article 730, 2023, pp. 1–45. doi: 10.1145/3544548.3581072.

[14] I. Yim and J. Su, "Artificial intelligence (AI) learning tools in K-12 education: A scoping review," Journal on Computers in Education, Vol. 12, pp. 93–131, 2025, doi: 10.1007/s40692-023-00304-9.

[15] D. Reiners, M. R. Davahli, W. Karwowski, and C. Cruz-Neira, "The combination of artificial intelligence and extended reality: A systematic review," Frontiers in Virtual Reality, 2, 721933, 2021, doi: 10.3389/frvir.2021.721933

[16] M. Inkarbekov, R. Monahan, and B. A. Pearlmutter, "Visualization of ai systems in virtual reality: A comprehensive review," arXiv preprint, 2023, arXiv:2306.15545.

[17] M. Bellgardt, C. Scheiderer, and T. W. Kuhlen, "An Immersive Node-Link Visualization of Artificial Neural Networks for Machine Learning Experts," 2020 IEEE International Conf. on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2020, pp. 33-36, doi: 10.1109/AIVR50618.2020.00015.

[18] C. Hilton et al., "InteractML: Making machine learning accessible for creative practitioners working with movement interaction in immersive media," In: Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology (VRST '21), ACM, Article 23, 2021, pp. 1–10. doi: 10.1145/3489849.3489879.

[19] Z. Lyu, J. Li, and B. Wang, "AIive: Interactive Visualization and Sonification of Neural Networks in Virtual Reality," 2021 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2021, pp. 251-255, doi: 10.1109/AIVR52153.2021.00057.

[20] N. Meissler, A. Wohlan, N. Hochgeschwender, and A. Schreiber, "Using Visualization of Convolutional Neural Networks in Virtual Reality for Machine Learning Newcomers," 2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR), IEEE, 2019, pp. 152-1526, doi: 10.1109/AIVR46125.2019.00031.

[21] A. Wohlan, N. Hochgeschwender, and N. Meissler, "Visualizing Convolutional Neural Networks with Virtual Reality," In: Proc. 25th ACM Symposium on Virtual Reality Software and Technology (VRST '19). ACM, Article 100, pp. 1–2, 2019, doi: 10.1145/3359996.3364817

[22] A. Schreiber and M. Bock, "Visualization and exploration of deep learning networks in 3D and virtual reality," In: Human-Computer Interaction International 2019 (HCII 2019) Posters, Proc. 21st International Conference on HCI (HCII 2019), CCIS, Springer International Publishing, 2019, pp. 206-211.

[23] D. Queck, A. Wohlan, and A. Schreiber, „Neural Network Visualization in Virtual Reality: A Use Case Analysis and Implementation," In: Human Interface and the Management of Information: Visual and Information Design (HCII 2022), LNCS, Springer, 2022, vol 13305, pp. 384-397, doi: 10.1007/978-3-031-06424-1_28.

[24] K. VanHorn and M. C. Çobanoğlu, "Democratizing AI in biomedical image classification using virtual reality," Virtual Reality, 26(1), pp. 159-171, 2021., doi: 10.1007/s10055-021-00550-1

[25] C. Linse, A. Hammam, and T. Martinetz, "A walk in the black-box: 3D visualization of large neural networks in virtual

reality," Neural Computing and Applications, vol. 34, no. 23, pp. 21237-21252, 2022.

[26] M. Možina, J. Demšar, M. Kattan, and B. Zupan, “Nomograms for visualization of naive Bayesian classifier,” In: European conference on principles of data mining and knowledge discovery, Springer Berlin Heidelberg, 2004, pp. 337-348.

[27] J. Balfer and J. Bajorath, “Introduction of a methodology for visualization and graphical interpretation of Bayesian classification models,” Journal of chemical information and modeling, 54(9), 2014, pp. 2451-2468.

[28] J. Ferreira, L. S. Machado, and R. M. de Moraes, “A new double weighted fuzzy hypergeometric naive bayes network and its application for user’s assessment in virtual reality simulators,” International Journal of Computational Intelligence Systems, 18(1), 2025, pp. 90-107.

[29] S. Welling, H. Refsgaard, P. Brockhoff, and L. Clemmensen (2016). Forest floor visualizations of random forests. arXiv preprint arXiv:1605.09196.

[30] R. H. Nsch, P. Wiesner, S. Wendler, and O. Hellwich, "Colorful Trees: Visualizing Random Forests for Analysis and Interpretation," 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2019, pp. 294-302, doi: 10.1109/WACV.2019.00037.

[31] X. Zhao, Y. Wu, D. L. Lee, and W. Cui, "iForest: Interpreting Random Forests via Visual Analytics," in IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 1, pp. 407-416, 2019, doi: 10.1109/TVCG.2018.2864475.

[32] N. Kuznetsova, M. Westenberg, K. Buchin, K. Dinkla, and S. J. Van Den Elzen, “Random forest visualization,” Eindhoven University of Technology, 2014.

[33] R. Oberhauser and C. Pogolski, "VR-EA: Virtual Reality Visualization of Enterprise Architecture Models with ArchiMate and BPMN," In: Business Modeling and Software Design (BMSD 2019), LNBIP, vol. 356, Springer, Cham, 2019, pp. 170-187, doi: 10.1007/978-3-030-24854-3_11.

[34] R. Oberhauser, C. Pogolski, and A. Matic, "VR-BPMN: Visualizing BPMN models in Virtual Reality," In: Shishkov, B. (ed.) Business Modeling and Software Design (BMSD 2018), LNBIP, vol. 319, Springer, 2018, pp. 83–97, doi: 10.1007/978-3-319-94214-8_6.

[35] R. Oberhauser, “VR-GitCity: Immersively Visualizing Git Repository Evolution Using a City Metaphor in Virtual Reality,” International Journal on Advances in Software, 16 (3 & 4), 2023, pp. 141-150, ISSN: 1942-2628.

[36] R. Oberhauser, “VR-GitEvo+CI/CD: Visualizing the Evolution of Git Repositories and CI/CD Pipelines in Virtual Reality,” International Journal on Advances in Software, 18 (1 & 2), 2025, pp. 82-96, ISSN: 1942-2628.

[37] I. J. Akpan and M. Shanker, "The confirmed realities and myths about the benefits and costs of 3D visualization and virtual reality in discrete event modeling and simulation: A descriptive meta-analysis of evidence from research and practice," Computers & Industrial Engineering, vol. 112, pp. 197-211, 2017.

[38] S. Narasimha, E. Dixon, J. W. Bertrand, and K. C. Madathil, "An empirical study to investigate the efficacy of collaborative immersive virtual reality systems for designing information architecture of software systems," Applied Ergonomics, vol. 80, pp. 175-186, 2019.

[39] A. Fonnet and Y. Prie, "Survey of immersive analytics," IEEE Transactions on Visualization and Computer Graphics, vol. 27, no. 3, pp. 2101-2122, 2019.

[40] R. Müller, P. Kovacs, J. Schilbach, and D. Zeckzer, "How to master challenges in experimental evaluation of 2D versus 3D software visualizations," In: 2014 IEEE VIS International Workshop on 3Dvis (3Dvis), IEEE, 2014, pp. 33-36.

[41] A. R. Hevner, S. T. March, J. Park, and S. Ram, “Design science in information systems research,” MIS Quarterly, 28(1), 2004, pp. 75-105.

[42] R. Jain. Visualization -> Machine learning -> Deep learning. [Online]. Available from: https://www.kaggle.com/code/ranjeetjain3/visualization-machine-learning-deep-learning 2026.05.09

[43] TensorFlow Playground. [Online]. Available from: https://playground.tensorflow.org 2026.05.0

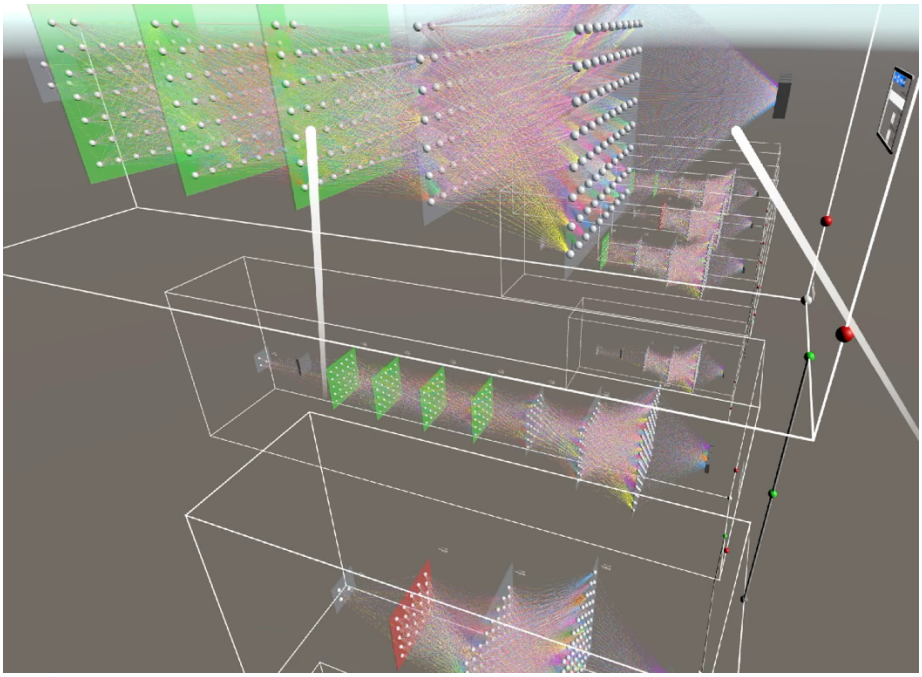

Figure 36. NN evolution view side perspective with colorized connectors. Red/green panels indicate removal/addition.

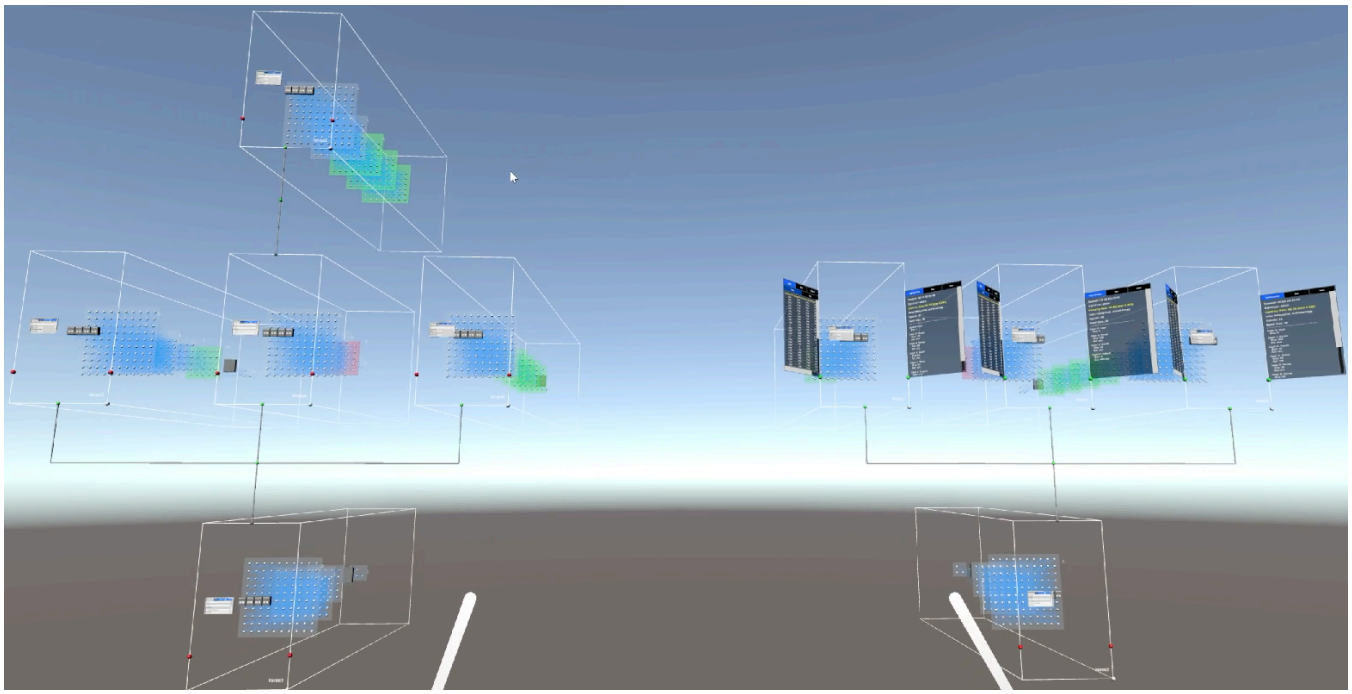

Figure 37. Loading another model (keras2 on right) starts a new evolution tree once any changes are made. Panels can be optionally turned on/off.

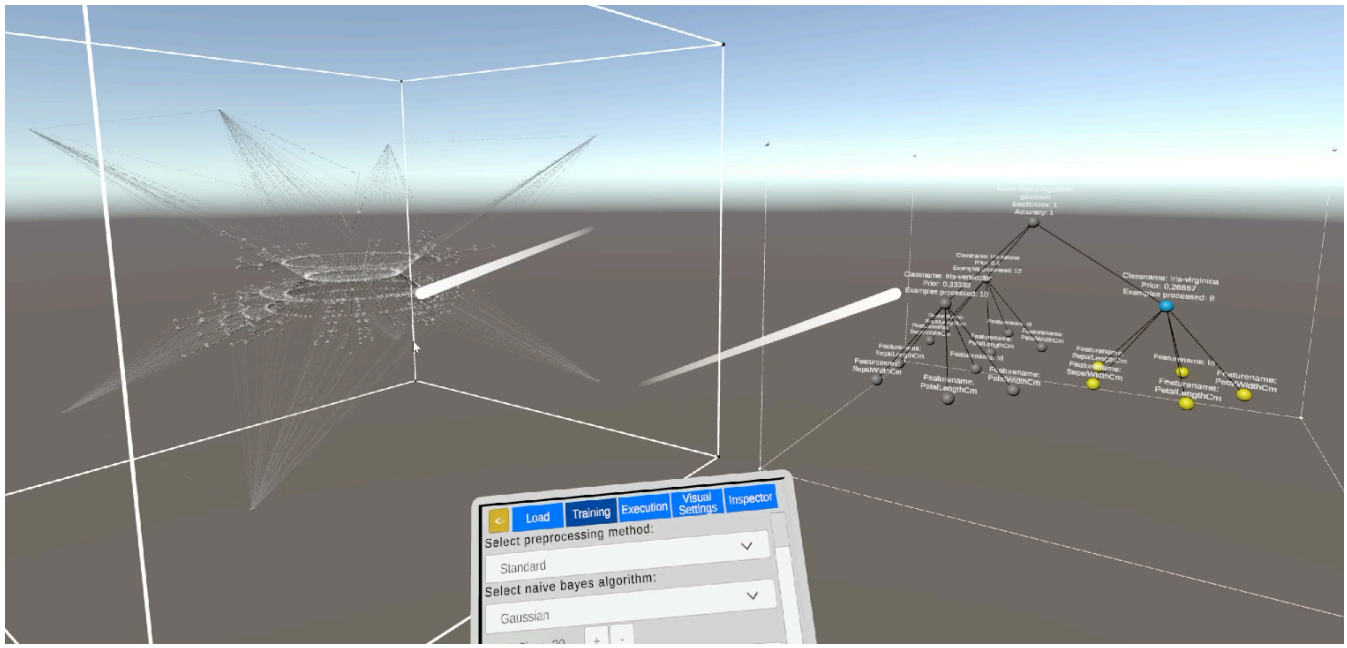

Figure 38. Concurrent dual heterogeneous models (RF left, NB right).

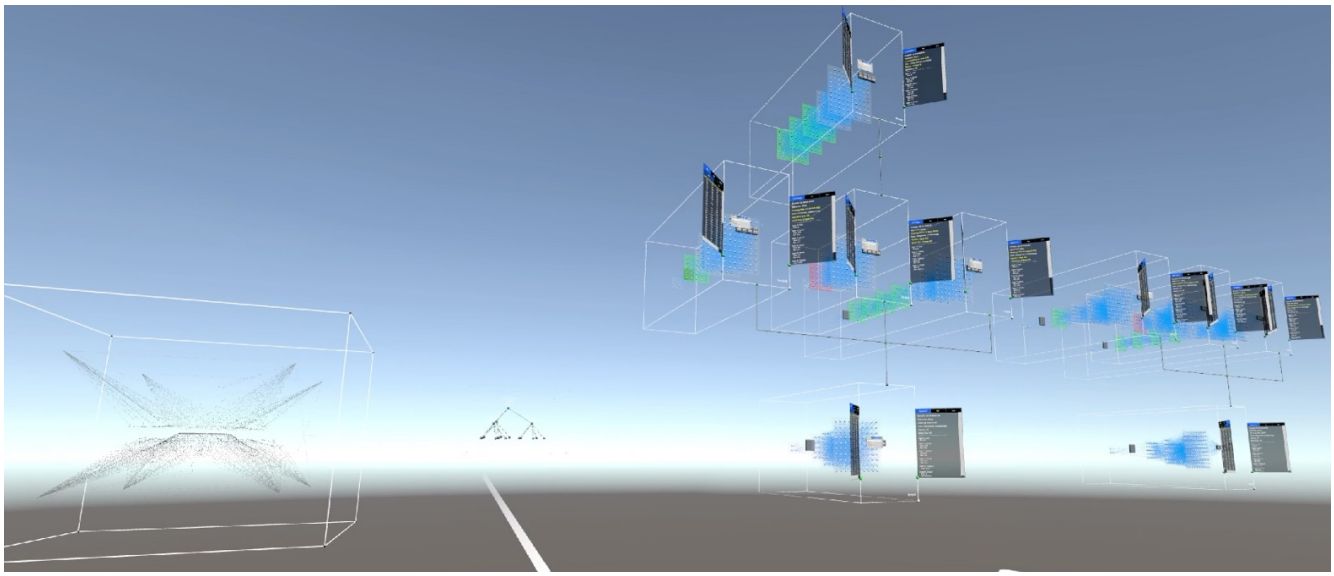

Figure 39. Diverse ML models (RF, NB, NN) visualized concurrently while depicting homogeneous model type history and evolution (two models, right).