International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 6, Nov-Dec 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

An Interactive Visualizer for Machine Learning and Deep Learning Algorithms

Syed Zeeyan*, Jayashree M**, Yashas T.M***, Zaid Ilyasi****
*(Information Science, VTU, and Bangalore, India
Email: syedzeeyan45@gmail.com)
**(Information Science, VTU, and Bangalore, India
Email: Jayashree.raju2006@gmail.com)
***(Information Science, VTU, and Bangalore, India
Email: yashastm100@gmail.com)
****(Information Science, VTU, and Bangalore, India
Email: zais22ise@gmail.com)

----------**********----------

Abstract:

The pedagogical transition from mathematical derivations to practical intuition in Machine Learning (ML) and Deep Learning (DL) presents a substantial barrier for novice practitioners. Traditional educational resources rely heavily on static diagrams, which fail to capture the temporal dynamics of stochastic gradient descent or the evolving topology of decision boundaries during training. To address this disconnect, we architected the Machine Learning Visualizer, a comprehensive Single Page Application (SPA) built on the React.js ecosystem.This platform moves beyond isolated algorithm demonstrations by implementing a client-side execution engine that runs training loops directly in the browser, ensuring zero-latency feedback. The system features interactive modules for Linear Regression, Decision Boundaries (visualizing K-Nearest Neighbors, SVM, and Logistic Regression), and Deep Learning architectures (including MLP for XOR problems and CNNs for digit recognition). A primary contribution of this work is the development of a novel Model Comparison Dashboard, which enables students to execute multiple algorithms simultaneously on identical, user-generated datasets. This paper details the system's component-based architecture, the optimization of render cycles for real-time visualization, and the utility of side-by-side metric analysis (MSE, Accuracy, Loss) in demonstrating the trade-offs between model complexity and overfitting.

Keywords — **Machine Learning, Deep Learning, Algorithm Visualization, Educational Technology, Neural Networks, CNN, Linear Regression, Decision Boundaries, React, Interactive Learning.**

----------**********----------

I. INTRODUCTION

As Machine Learning curricula evolve, the pedagogical focus has shifted from mathematical derivation to intuitive debugging. Students often treat algorithms as 'black boxes,' struggling to visualize the internal state changes during training. This project, the Machine Learning Visualizer, addresses this disconnect by leveraging a **client-side React architecture**. By moving the computational logic from the server to the browser, we enable a zero-latency feedback loop, transforming abstract equations into tangible, manipulatable UI components. To facilitate this exploration, the application is structured around four core educational modules:

Linear Regression: To understand the fundamentals of fitting a line to data by visualizing the minimization of residuals.

Decision Boundaries: To compare how different classifiers (KNN, SVM, Logistic Regression) partition data in a 2D space.

Neural Networks: To demystify the "black box" by showing a network learn the classic XOR problem, visualizing changing weights and activations in real-time.

CNN Visualizer: To illustrate the entire pipeline of a Convolutional Neural Network, from input image to classification, showing intermediate feature maps and pooling layers.

Finally, the tool introduces a unique **Model Comparison Dashboard**, allowing users to run different models on the same data and evaluate their performance side-by-side. This paper discusses the related work in algorithm visualization, details the implementation of our system, and presents the key features of each module.

CLASSIFICATION OF VISUALIZED ALGORITHMS

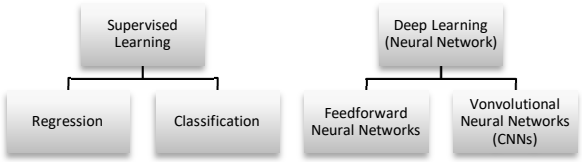

Fig. 1 Classification of Algorithms in the Visualizer

The application structures its educational modules into two distinct pedagogical paradigms, shifting the learner's focus from spatial geometry to architectural flow.

1. Geometric Intuition (Supervised Learning): These modules treat learning as a spatial optimization problem on a 2D plane.

Regression: Users engage with **Curve Fitting**, observing how minimizing geometrical distance (residuals) yields the optimal continuous prediction.

Classification: Users engage with **Space Partitioning**, manipulating data points to observe how different mathematical kernels (Linear vs. RBF) distinctively slice the feature space.

2. Structural Intuition (Deep Learning): These modules shift focus to the internal signal propagation within a network graph.

Feedforward Networks: The focus is on **Weight Dynamics**, where students analyze how error signals (gradients) travel backward to adjust connection strengths.

Convolutional Neural Networks (CNNs): The focus is on **Feature Transformation**, demonstrating how high-dimensional tensor data is progressively distilled into abstract semantic labels.

II. RELATED WORK

A. The Role of Interactivity in Algorithm Learning Our pedagogical approach is grounded in the extensive body of research surrounding Algorithm Visualization (AV), which suggests that the medium of instruction is less critical than the mode of engagement. Foundational meta-analyses by Hundhausen et al. [6] and empirical studies by Saraiya et al. [5] argue that "learner agency" is the primary driver of cognitive retention. Their findings indicate that passive viewing of high-quality animations yields minimal benefit compared to active manipulation, where students modify input parameters and immediately observe the algorithmic consequences. This "constructivist" strategy has been successfully implemented in deterministic domains like sorting, pathfinding, and scheduling by tools such as HalVis [2], AlgoViz [4], and AlgoRhythm [10]. These systems effectively link code execution to visual output [9], establishing a mental model of cause-and-effect. However, while modern web-based platforms have successfully lowered the barrier to entry for these standard Computer Science algorithms [3] [7] [11], there remains a significant scarcity of tools that apply this "interactive control" philosophy [8] to the stochastic and probabilistic nature of Machine Learning, where outputs are not always deterministic.

B. Fragmentation in Classical ML **Tools** The current landscape of classical Machine Learning (ML) education is characterized by a disjointed ecosystem of resources. For instance, interactive articles by Wilber [16] and the "Pocket Guide" [17] provide excellent, isolated demonstrations of specific concepts like Mean Squared Error (MSE) minimization in Regression. Similarly, regarding classification tasks, resources like GeeksforGeeks [22] and Hussain [15] rely on static or semi-static 2D plots to illustrate how Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) partition feature space. Sallam et al. [18] expanded this visualization to multi-dimensional boundaries, yet these tools typically exist as standalone tutorials or

isolated weblets. This forces students to context-switch between different interfaces to learn related concepts. Our work directly addresses this fragmentation by aggregating these distinct solvers into a unified Model Comparison Dashboard. Unlike existing tools that treat these models in isolation, our platform enables the direct, simultaneous benchmarking of linear vs. non-linear classifiers on identical, user-generated datasets. This aggregation significantly reduces cognitive load and fills a critical gap in unified educational environments. Future iterations will aim to incorporate Decision Trees [20] and unsupervised Clustering [21] to further round out the classical suite.

C. Demystifying Deep Learning Architectures Deep Learning presents a unique pedagogical hurdle due to the inherent opacity of neural architectures—the so-called "black box" problem. Smilkov et al [13] set the industry standard for visualizing this internal state with the "Neural Network Playground," which uses a node-link diagram to visualize backpropagation [19] in real-time. In the realm of computer vision, the "CNN Explainer" by Wang et al. [14] effectively deconstructs the feature extraction pipeline, allowing users to see intermediate tensor transformations. While advanced visualization techniques have been applied to Generative Adversarial Networks (GANs) [23] and Reinforcement Learning agents [24] to improve model interpretability [25], these tools rarely interface with classical methods. A novice student using these isolated tools cannot easily compare *why* a heavy-weight Neural Network outperforms (or underperforms) a simple Logistic Regression model on the same dataset. Our system bridges this engineering gap. By placing Deep Learning models side-by-side with Classical algorithms, we provide a practical environment for visualizing the bias-variance trade-off, allowing students to empirically validate model selection decisions rather than relying on theoretical assumptions.

III. FLOW OF PROPOSED WORK

The educational efficacy of the Machine Learning Visualizer is intrinsically linked to its high degree of interactivity. Unlike traditional static textbook examples, which present the "final state" of a trained model, our system implements a constructivist approach where the user is the active agent in the learning process. By leveraging an event-driven architecture, the application transforms abstract mathematical variables into tangible, manipulatable controls.

This interactivity allows for "Exploratory Learning," where students can form a hypothesis (e.g., "What happens if I increase the learning rate?") and immediately test it against the model. From a technical perspective, this is achieved by binding user interface elements (sliders, toggles, and canvas clicks) directly to the algorithm's internal state. Any modification triggers a re-calculation of the model's loss function and decision boundary in real-time, providing instant visual feedback. We define these controllable elements as "attributes"—mutable hyperparameters or data inputs that drive the algorithmic logic. To ensure the tool addresses specific learning outcomes, the set of available attributes is strictly tailored to the mathematical nature of each specific module:

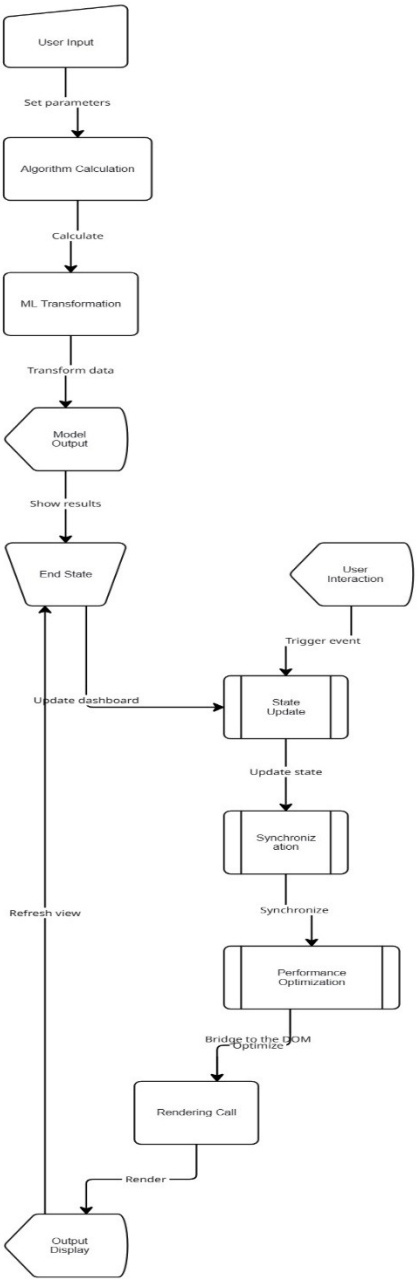

Fig. 2 Flow Chart

a. Linear Regression Visualizer: The core "attributes" are the data points (X, Y), which the user adds by clicking. Users can also directly manipulate the model parameters, which are the Manual Slope (m) and Manual Intercept (b), to see how they affect the Total Error (Sum of Squared Residuals).

$$\mathrm{SSR}(m,b) = \sum_{i=1}^{N} \left(y_i - (mx_i + b)\right)^2 \qquad (1)$$

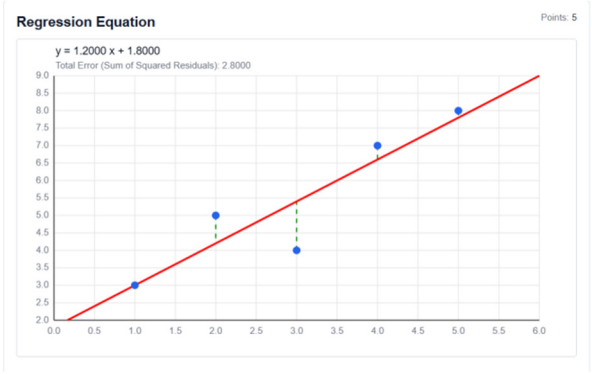

Fig. 3 Linear Regression Visualization

Technical Implementation Strategy:

The application is engineered as a Single Page Application (SPA) using React.

State Management: We utilized React Hooks (useState, useEffect) to manage the training loop. A key engineering challenge was updating the DOM with new weights without freezing the UI thread during intensive calculations (like CNN convolution).

Rendering: Graphs are rendered using a high-performance charting library (e.g., Recharts), allowing for 60fps updates of decision boundaries as the model trains.

Algorithm Implementation:

For Linear Regression, rather than relying on server-side calls, the system calculates the slope (m) and intercept (b) in the browser using the closed-form Ordinary Least Squares method. This provides instant feedback as the user clicks the canvas to add points.

For the **CNN Visualizer**, visualizing the network in the browser requires mapping the pixel data of the input (e.g., MNIST digit) through the kernel filters directly in JavaScript. We visualize the intermediate feature maps to show exactly what the network 'sees' after the pooling layer.

b. Decision Boundary Visualizer: Users can create a 2D dataset by adding data points (X, Y) and assigning them to Class A or Class B. They can then select the "attribute" of the model itself: Logistic Regression, Support Vector Machine (SVM), or K-Nearest Neighbors (KNN). For each model, they can tune its

key parameters, such as adjusting the polynomial weights for Logistic Regression or the K-value for KNN.

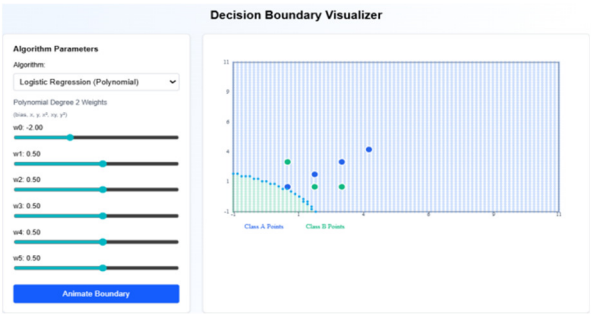

Fig. 4 Logistic Regression Visualization

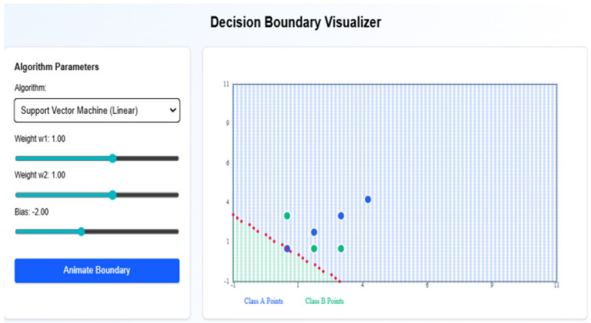

Fig. 5 SVM Visualization

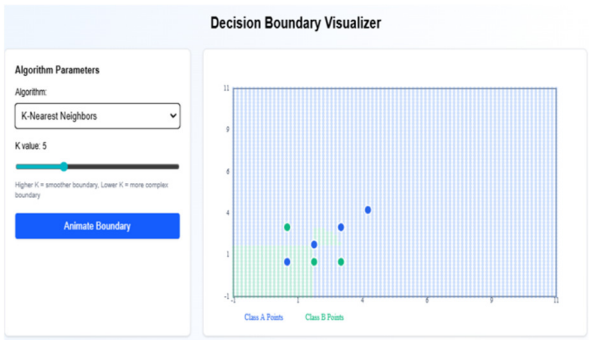

Fig. 6 KNN Visualization

c. Neural Network (XOR) Visualizer: Interactivity here is focused on controlling the learning process. The user can Start/Stop Training and adjust the Training Speed slider. The visualized "attributes" are the network's internal states, such as the changing weights of each connection and the activation values of the hidden and output neurons.

$$w_{\text{new}} = w_{\text{old}} + \eta \cdot \delta \cdot a_{\text{input}} \qquad (2)$$

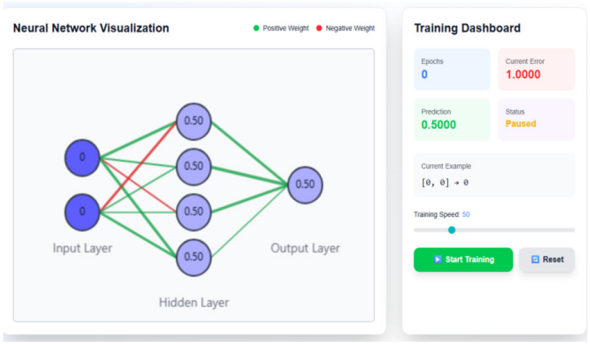

Fig. 7 Neural Network Visualization

d. CNN Visualizer: The primary user-selected attribute is the input digit (0-9). The module then visualizes the entire pipeline of "attributes" (feature maps) as they are extracted and transformed by each layer, including Conv Layer 1, Pool Layer 1, Conv Layer 2, Pool Layer 2, and the Fully Connected layer.

$$O_{i,j} = \sum_{k=1}^{F} \sum_{l=1}^{F} I_{i+k-1,j+l-1} \cdot K_{k,l} \qquad (3)$$

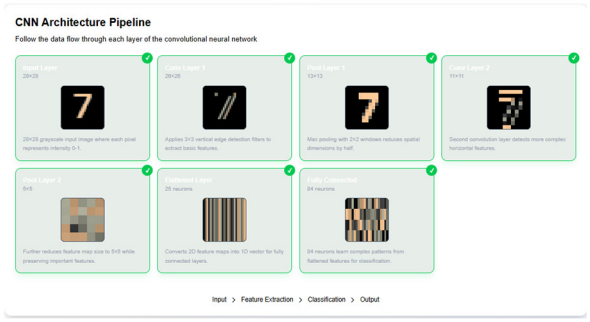

Fig. 8 CNN Visualization

e. Model Comparison Dashboard: Similar to the regression module, the key user-controlled attributes are the (X, Y) data points that form the shared dataset. The system then displays the calculated attributes for each model, such as MSE, Loss, and Accuracy.

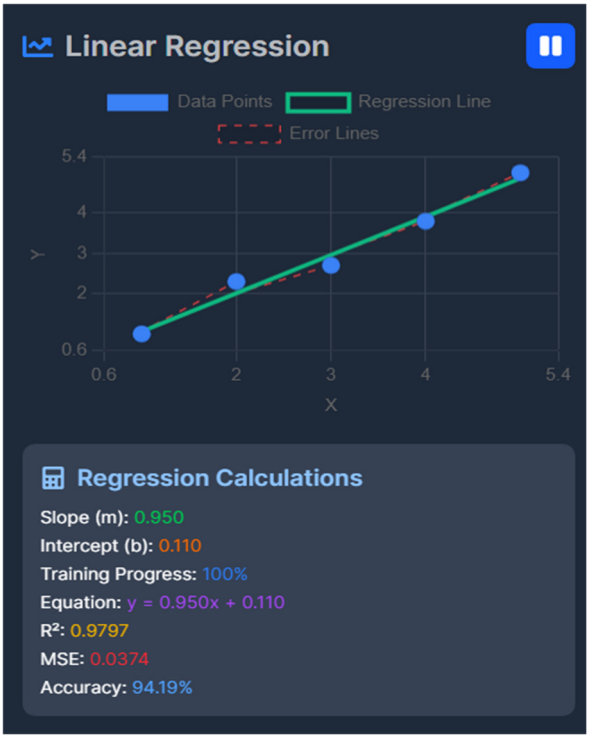

Fig. 9 Comparison Dashboard (Linear Regression)

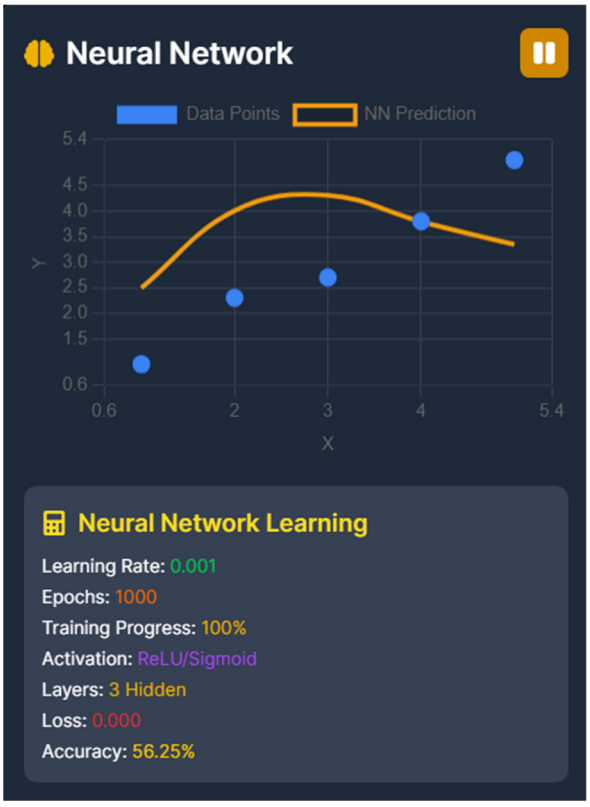

Fig. 10 Comparison Dashboard (Neural Network)

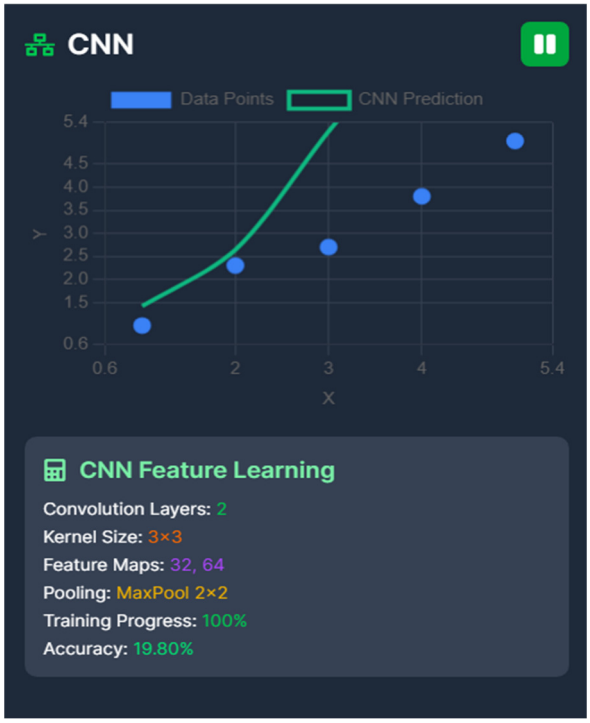

Fig. 11 Comparison Dashboard (CNN)

IV. VISUALIZATION ARCHITECTURE & IMPLEMENTATION

Unlike traditional educational tools that rely on pre-rendered videos or server-side Python backends, our system is architected as a client-side Single Page Application (SPA). We migrated the mathematical logic directly into the browser's JavaScript engine. This design choice eliminates server latency, enabling a reactive loop where the visualization updates instantly (at 60 frames per second) as users manipulate hyperparameters.

A. Supervised Learning Modules

Support Vector Machine (SVM): To visualize the "Maximum Margin" concept, we avoided static plotting libraries in favor of a custom Scalable Vector Graphics (SVG) implementation. The core challenge was rendering the hyperplane dynamically. As the user drags a data point on the canvas, the application re-runs the optimization algorithm (Sequential Minimal Optimization) in the background thread. The resulting decision boundary and margins are rendered as SVG paths that animate to their new positions, visually demonstrating how a single outlier vector can aggressively shift the optimal hyperplane.

$$w^T x + b = 0 \qquad (4)$$

K-Nearest Neighbors (KNN): Visualizing "lazy learning" required a dynamic approach to spatial analysis. We implemented a grid-based rendering system using the **HTML5 Canvas API**. When a user hovers their cursor over a specific coordinate, the system calculates the Euclidean distance to every data point in the set in real-time. We then render connecting lines to the 'K' nearest neighbors, dynamically changing the color of the cursor's location to reflect the majority class. This provides an immediate visual proof of how local neighborhood density determines classification, rather than a global formula.

$$D(x_1, x_2) = \sqrt{\sum_{j=1}^{d} (x_{2,j} - x_{1,j})^2} \qquad (5)$$

Logistic Regression: While Linear Regression fits a straight line, Logistic Regression requires visualizing the "Sigmoid S-curve" probability distribution. We utilized a **pixel-shader approach** to render the background gradients. The system calculates the probability score $(p = 1/(1 + e^{-z}))$ for every pixel in the coordinate space, mapping values from 0 to 1 to a color gradient (e.g., Blue to Red). This allows students to visually identify the "uncertainty zone" $(where \$p \approx 0.5\$)$ and observe how changing the learning rate affects the steepness of this transition boundary.

$$p = \sigma(z) = \frac{1}{1+e^{-z}} \qquad (6)$$

B. Deep Learning (Neural Networks)

Neural Network (XOR): The backpropagation process is often abstract to students. We "white-boxed" this algorithm by mapping the internal weight matrices directly to the DOM. The magnitude of each weight is programmatically bound to the stroke-width and opacity of the SVG lines connecting the neurons. As the network trains, users can physically see the connections strengthening or fading. This visual feedback allows students to detect phenomena like the "Vanishing Gradient" problem—observed when weights stop changing color despite high epochs—without needing to inspect console logs.

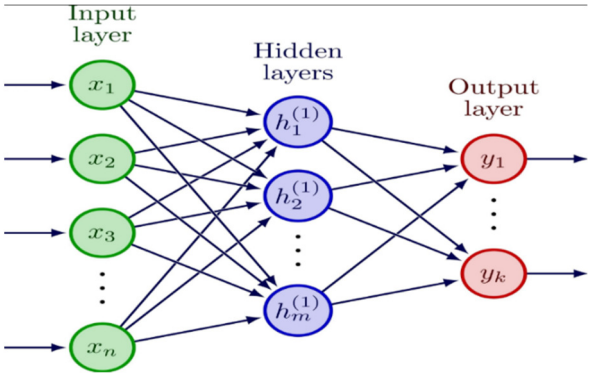

Fig. 12 Neural Network Model

CNN Pipeline (Digit Recognition): The pedagogical objective here was to demystify the "feature extraction" phase, which often confuses students. From an engineering standpoint, this involved intercepting the intermediate 3D tensor outputs (Height × Width × Channels) after every convolution and pooling operation. We developed a custom rendering engine that normalizes these raw matrix values into 0-255 grayscale pixel intensities and paints them onto a grid of dynamic HTML5 Canvas elements.

This visualization exposes the hierarchical nature of computer vision. In the initial layers, students can visually verify that the kernels are acting as simple "Edge Detectors," highlighting vertical or horizontal lines. In the deeper layers (after Max Pooling), the visualization demonstrates the concept of abstraction, where the feature maps become less recognizable to the human eye but richer in semantic meaning. By hovering over specific kernels, users can trace how a specific pixel pattern in the input digit (e.g., the curve of a '9') activates a specific neuron in the final Fully Connected layer, effectively bridging the gap between raw pixel data and classification logic.

V. COMPARISON OF THE RESULTS

The Model Comparison Dashboard (Fig 9, 10) reveals the inherent trade-offs in model selection through real-time metrics.

A. Complexity vs. Performance As shown in Table 1, while the Neural Network achieved a higher accuracy (56.25%) compared to the CNN (19.80%) in early training epochs, the CNN's feature extraction capabilities allow it to scale better with complex image data over time.

B. Visualizing Underfitting Users can visually observe the Linear Regression model failing to capture non-linear data patterns, resulting in a high MSE of 0.0375. This provides a visual proof of 'underfitting,' where the straight regression line cannot bend to accommodate the curvature of the dataset."

TABLE I

COMPARISON OF METRICS

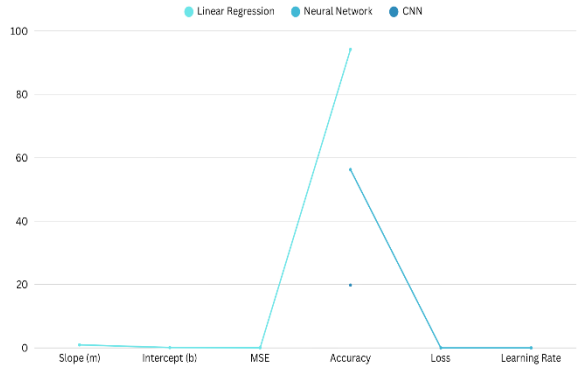

Fig. 13 Comparison Graph

This direct comparison of metrics like Accuracy, MSE, and Loss provides powerful intuition. Users can instantly see how a simple model like Linear Regression has a high error (MSE) on non-linear data points, while a Neural Network can achieve a lower loss and higher accuracy, demonstrating the trade-off between model complexity and performance.

VI. CONCLUSION

By shifting the computation from server-side Python scripts to client-side JavaScript, the Machine Learning Visualizer provides immediate feedback, essential for the 'trial-and-error' learning process. The project successfully demonstrates that complex deep learning concepts, including Backpropagation and Convolutions, can be demystified through direct manipulation. Unlike static textbook examples, this tool allows students to break the models to understand how they work. Future iterations will focus on integrating a backend for saving user-generated datasets and expanding the library to include Unsupervised Learning models like K-Means.

Furthermore, the introduction of the Model Comparison Dashboard addresses a critical gap in current educational tools: the ability to visualize the "bias-variance trade-off" in real-time. By observing a Linear Regression model underfitting on non-linear data while a Neural Network successfully converges, students gain a visceral understanding of why model architecture matters. Ultimately, this work establishes that the modern browser is a viable, powerful environment for scientific visualization, effectively removing the barrier of "environment setup" that often discourages beginners from entering the field.

Key Metrics Displayed	Linear Regression	Neural Network	CNN
Slope (m)	**0.95**	-	-
Intercept (b)	0.110	-	-
MSE	0.0375	-	-
Accuracy	94.19	56.25	19.80
Loss	-	0.000	-
Learning Rate	-	0.001	-

VII. FUTURE SCOPE

While the current Single Page Application (SPA) successfully handles small-scale datasets, the next phase of development focuses on architectural scalability and broader algorithmic coverage:

Hybrid Architecture & Persistence: To support custom data uploads and session saving, we plan to migrate from a purely client-side architecture to a MERN stack (MongoDB, Express, React, Node) solution. This will allow students to save their custom datasets and trained model weights, enabling a "build-and-resume" workflow for classroom assignments.

Advanced Visualization Techniques: As we introduce more computationally intensive

algorithms like **Random Forests** (to visualize ensemble voting) and **K-Means Clustering** (to visualize centroid convergence), we intend to explore **WebGL** or **WebAssembly (WASM)**. This would offload heavy matrix calculations from the main JavaScript thread, ensuring the UI remains responsive even with larger datasets.

Curriculum Integration: Beyond software features, the next logical step is measuring pedagogical efficacy. We aim to deploy this tool in an introductory Data Science course to empirically measure if "visual intuition" leads to better performance in debugging code-based assignments compared to traditional theoretical instruction.

REFERENCES

[1] IEEE UP Section Conference (2021). Design and Analysis of Algorithms: Making learning easier with visual animations, games, and interactive tools focused on pathfinding, sorting, and CPU scheduling.

[2] Hansen, S., et. al. (2002). Designing Educationally Effective Algorithm Visualizations: HalVis system integrates animations into meaningful learning environments, improving student outcomes. Journal of Visual Languages & Computing.

[3] Thakkar, A., et. al. (2022). Sorting Algorithm Visualizer: A web application for pictorial and interactive demonstration of sorting algorithms to simplify learning. 2022 Intl. Conference on Cyber Resilience.

[4] Gupta, A. S., & Vyawahare, M. (2023). AlgoViz: Algorithm Visualization tool using Python libraries PyGame and Tkinter for step-by-step sorting, searching, and pathfinding visualizations. 2023 ICNTE.

[5] Saraiya, P., et. al. (2004). Effective Features of Algorithm Visualizations: Empirical study finds user control and clarity key for learning. Proceedings of the 35th SIGCSE Technical Symposium.

[6] Hundhausen, C. D., et. al. (2002). A Meta-Study of Algorithm Visualization Effectiveness: Reviews 24 studies and emphasizes learner engagement's impact on outcomes. Journal of Visual Languages & Computing.

[7] Singh, A. P. (2024). Sorting and Path Finding Algorithm Visualizer: Software tool for interactive exploration and real-time visualization of sorting and pathfinding algorithms with complexity explanations. International Journal of Scientific and Innovative Computing.

[8] Urquiza-Fuentes, J., & Vel´azquez-Iturbide, J. ´ A. (2009). A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems: Reviews 18 systems and 33 studies on visualization effectiveness. ACM TOCE, 9(2), Article 9.

[9] Ghandge, A. B., et. al. (2021). AlgoAssist: Algorithm Visualizer and Coding Platform for Remote Classroom Learning: Combines visualization, coding, and assessment to enhance remote algorithm education. ICCCSP 2021.

[10] Trivedi, A., et. al. (2023). AlgoRhythm A Sorting and Path-finding Visualizer Tool: Interactive visualization with integrated coding environment for enhanced learning engagement. 2023 Intl. Conf. on Cloud Computing, Data Science & Engineering.

[11] Prabhakar, G., et. al. (2022). Analysis of Algorithm Visualizer to Enhance Academic Learning: Offline, easy-to-use downloadable tool with interactive GUI and speed control. 2022 Intl. Conf. on Innovative Practices in Technology and Management.

[12] Lanjewar, R., et. al. (2022). Algovis– An Enhanced Way to Visualize Algorithms: Website for animated step-by-step visualization of data structures and algorithms with custom inputs. EDP Sciences ITM Conferences.

[13] Smilkov, D., et. al. (2016). "A Neural Network Playground." An interactive, in-browser tool for visualizing and understanding neural networks. [Online].

[14] Wang, Z. J., et. al. (2020). "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization." IEEE Transactions on Visualization and Computer Graphics (Proc. VIS 2020).

[15] Hussain, A. (2009). "Decision Boundaries and Classification Performance Of SVM And KNN Classifiers For 2-Dimensional Dataset." Industrial Electronic Seminar.

[16] Wilber, J. (2022). "Linear Regression - MLU-Explain." An interactive, visual-first article explaining linear regression models, focusing on coefficients and Mean-Squared Error (MSE).

[17] Pocket Guide to Machine Learning. (n.d.). "Interactive Guide to Linear Regression." A web-based interactive guide that allows users to adjust slope and intercept to "fit" a line to data.

[18] Sallam, A. M., et. al. (2015). "Visualizing multi-dimensional decision boundaries in 2D." Proc. 9th International Conference on Computer Engineering & Systems (ICCES).

[19] IBM. (n.d.). "What is Backpropagation?" Provides a conceptual and mathematical explanation of the backpropagation algorithm, which is the core learning process visualized in our Neural Network (XOR) module.

[20] Ando, S., et. al. (2024). "Guided Decision Tree: A Tool to Interactively Create Decision Trees Through Visualization." MDPI, Applied Sciences.

[21] Fournier-Viger, P. (2023). "K-Means Interactive Demo in your browser." An interactive, step-by-step visualization tool for the K-Means clustering algorithm.

[22] GeeksforGeeks. (2025). "Visualizing Support Vector Machines (SVM) using Python." A practical tutorial explaining the concepts of SVM, kernels, and the visualization of their decision boundaries.

[23] Bau, D., et. al. (2019). "GAN Dissection: Visualizing and Understanding Generative Adversarial Networks." Proc. ICLR 2019.

[24] Deshpande, S., et. al. (2020). "Interactive Visualization for Debugging RL." A tool for interpreting and debugging reinforcement learning (RL) agents. arXiv preprint arXiv:2008.07331.

[25] Chatzimparmpas, A., et. al. (2020). "A survey of surveys on the use of visualization for interpreting machine learning models." IEEE Transactions on Visualization and Computer Graphics.

[26] Ahluwalia, M., & Kumar, V. (2020). "A review on K-Means clustering and its variants." International Conference on Intelligent Computing and Control Systems (ICICCS).

[27] Liu, Y., & Li, H. (2018). "Visualizing Decision Tree and Random Forest." arXiv preprint arXiv:1805.04262.

[28] Olah, C. (2015). "Visual Explanations: Understanding Neural Networks Through Deep Visualization." Distill.pub.

[29] Nielsen, M. A. (2015). "A visual proof of backpropagation." In Neural Networks and Deep Learning, Determination Press.

[30] ML Cheatsheet. (2019). "Logistic Regression: An Interactive Visualization." [Online].

