IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, July 2024

Neural Network Visualizer Web App with Python

Ms. Divya K[1] and Dr. Annu Sharma[2]
Department of Masters of Computer Applications[1,2]
Raja Rajeswari College of Engineering, Bengaluru, Karnataka, India
divya.k.gowda115@gmail.com and annumca01@gmail.com

Abstract: *The NN Visualizer is an interactive web-based application designed to demystify the workings of artificial NNs by offering an intuitive platform to explore how trained models process and classify handwritten digits from the MNIST dataset. Utilizing a fully connected NN built with TensorFlow and Keras, the visualization component, created with Streamlit, allows users to observe real-time activation patterns across network layers. A Flask-based server ensures efficient data handling and model predictions. Key features include layer-by-layer activation visualization, real-time predictions, and a user-friendly interface, making it a valuable educational tool. This project enhances transparency in AI, supporting trends in responsible AI development by providing insights into the internal representations learned by NNs.*

Keywords: NN Visualization, MNIST Dataset, Deep Learning, TensorFlow, Keras

I. INTRODUCTION

The NN Visualizer project is an innovative educational tool designed to demystify the inner workings of artificial NNs. As machine learning and artificial intelligence become increasingly significant in various fields, understanding the fundamental principles behind these technologies is crucial. This project focuses on visualizing the decision-making process of a NN trained on the MNIST dataset, a canonical machine learning problem involving the classification of handwritten digits.

At its core, the NN Visualizer combines a trained NN model with an interactive web-based interface, allowing users to explore how the network processes and classifies handwritten digits. Users can select random images from the MNIST test set and observe in real-time how the NN's various layers activate in response to the input. This visualization provides invaluable insights into the hierarchical feature extraction and decision-making processes within the NN.

It is built using the latest technologies such as TensorFlow and Keras for model development, Flask for backend services, and Streamlit for the frontend, the project ensures efficient model training and seamless integration of components. One of its key features is the ability to display activation patterns for each layer of the NN, showing how input images are transformed through successive layers. Beyond its educational value, the visualizer serves as a platform for exploring interpretability in machine learning models, contributing to the development of more transparent and interpretable AI systems in line with trends in responsible AI development.

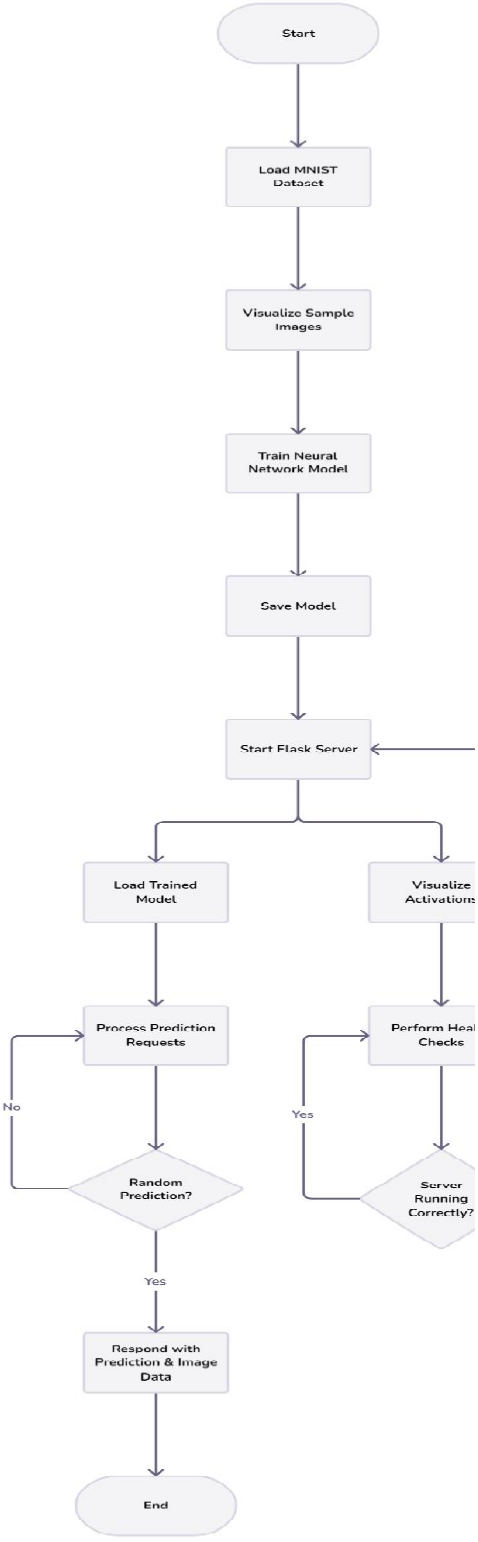
II. LITERATURE SURVEY

The concept of visualizing NNs to enhance interpretability was first introduced by Zeiler and Fergus [1]. Their method used deconvolutional networks to project feature activations back to the input pixel space, providing insights into the representations learned by CNNs.

Olah et al. [2] proposed feature visualization techniques to create human-interpretable representations of neurons in NNs. Their work demonstrated how optimization techniques could be used to generate images that maximally activate specific neurons, providing a visual understanding of what different parts of the network are detecting.

TensorFlow Playground, developed by Smilkov et al. [3], introduced an interactive web-based tool for visualizing simple NNs in real-time. This project allowed users to construct small networks and observe their training process, making it an invaluable educational resource.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, July 2024



DOI: 10.48175/IJARSCT-19127

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, July 2024

In [4], Kahng et al. presented ActiVis, a visual analytics system for interpreting large-scale deep learning models and results. Their system combined a novel neuron activation visualization design with interactive node-link diagrams, enabling users to explore complex NN structures.

The "Building Blocks" approach proposed by Olah et al. [5] provided a unified framework for visualizing and understanding NNs. This work introduced techniques for visualizing individual neurons, interpolating between representations, and identifying abstract concepts learned by the network.

Carter et al. [6] developed the Activation Atlas, a technique for visualizing and interpreting the internal representations of convolutional NNs at scale. Their method aggregates millions of activation vectors to create a map of conceptual features learned by the network.

The NN Visualizer project follows a systematic approach to create an interactive tool for visualizing the inner workings of a NN. Initially, the MNIST dataset is loaded and pre-processed using Keras, with images reshaped into 2D arrays and normalized. A sequential model with multiple dense layers is defined and optimized using the Adam optimizer with sparse categorical cross-entropy loss function, with the trained model saved for later use. The backend is developed using Flask, which loads the pre-trained model and defines API endpoints for generating random predictions and performing health checks. The frontend, created with Streamlit, includes interactive features such as buttons to trigger random predictions, sections to display images, model predictions, true labels, and visualizations of layer activations. For visualization, activation extract values by layer and displayed as heatmaps using matplotlib.

Integration and testing ensure accurate model loading, correct visualization generation, proper user interaction handling, and system performance. Optimization techniques such as batch processing and caching improve performance, while the user interface is refined based on feedback. Comprehensive documentation provides setup instructions, usage guidelines, and visualization explanations. Finally, the system is deployed to a web server for public access, resulting in an effective educational tool that combines technical implementation and user experience considerations to demystify NN processes.

III. EXPERIMENTAL RESULTS

These images collectively demonstrate:
The final user interface for visualizing layer activations
The input data (MNIST digits) the network is trained on and classifying
The underlying architecture of the NN

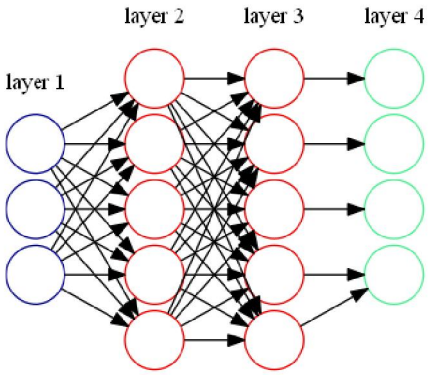


Fig. 1. This image depicts a simple NN architecture with four layers.
It shows:
Layer 1 (input layer) with multiple nodes
Layer 2 and 3 (hidden layers) with multiple nodes each
Layer 4 (output layer) with multiple nodes

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, July 2024

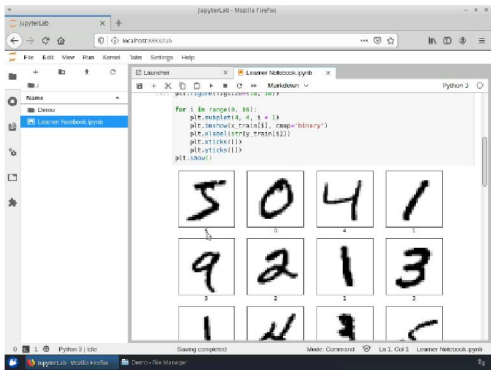


Fig. 2. The JupyterLab interface displays a Python notebook with a grid of labeled handwritten digit images from the MNIST dataset. This visualization helps users understand the input data used for training the NN. It provides a clear view of the types of images the network classifies

Fig. 3. The NN Visualizer interface displays a grid of grayscale heatmaps showing activation patterns for different network layers. Each row represents a layer, and each cell shows the corresponding activation strength. This allows users to observe how the input is transformed through the network layers.

IV. CONCLUSION

The NN Visualizer project effectively demonstrates the power of interactive visualization in clarifying the complex operations of NNs. By offering a user-friendly interface to explore the inner workings of a model trained on the MNIST dataset, this tool bridges the gap between theoretical understanding and practical implementation of machine learning concepts. The success of the project lies in its ability to transform the abstract processes of NNs into tangible, create visual representations that are clear and understandable for students. educators, and researchers.

Throughout the development process, challenges such as optimizing real-time visualizations and ensuring scalability for concurrent users were addressed. Utilizing TensorFlow for training models, Flask for backend services, and Streamlit for the frontend interface provided a seamless user experience while managing complex computational tasks. The project's modular architecture facilitated easier development and testing, positioning the system well for future enhancements. Key achievements include interactive visualization of NN activations, significant educational value by demystifying NNs, and successful integration of various technologies.

REFERENCES

[1] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media.
[2] Chollet, F., & others. (2018). Keras: The Python Deep Learning library. Retrieved from https://keras.io/
[3] TensorFlow Developers. (2019). TensorFlow: Large-scale machine learning on heterogeneous systems. Retrieved from https://www.tensorflow.org/
[4] McKinney, W. (2018). Data Structures for Statistical Computing in Python. Proceedings of the 9th Python in Science Conference, 56-61.
[5] Harris, C. R., Millman, K. J., van der Walt, S. J., et al. (2020). Array programming with NumPy. Nature, 585(7825), 357-362.
[6] Hunter, J. D. (2018). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95.
[7] Waskom, M. L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021.
[8] Streamlit Inc. (2023). Streamlit: Turn data scripts into shareable web apps. Retrieved from https://docs.streamlit.io/
[9] LeCun, Y., Cortes, C., & Burges, C. J. C. (2018). The MNIST database of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist/
[10] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2018). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12(Oct), 2825-2830.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.53
Volume 4, Issue 1, July 2024

[11] Dosilovic, F. K., Brcic, M., & Hlupic, N. (2018). Explainable artificial intelligence: A survey. In 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 0210-0215). IEEE.
[12] Zhang, Z., Lipton, Z. C., Li, M., & Smola, A. J. (2020). Dive into deep learning. arXiv preprint arXiv:2106.11342.