**SPL-1 Project Report**

**Hand-written Digit Recognition**

**Using Neural Network**

****

****

**Institute Of Information Technology**

**University Of Dhaka**

**17 - 12 – 2023**

Table of Contents

1. **Introduction3**
2. **Background5**

**2.1 Neural Network5**

**2.2 Mnist Dataset5**

**2.3 Training and Testing6**

**2.4 Forward and Back-propagation6**

**2.5 Loss function7**

**3. Description8**

**3.1 Data Processing8**

**3.2 Neural Network Design9**

**3.3 Training11**

**3.4 Testing and Evaluation12**

**3.5 Custom Input Drawing14**

**4. Implementation and Testing16**

**5. User Interface18**

**6. Challenges19**

**7. Conclusion24**

**8. Reference25**

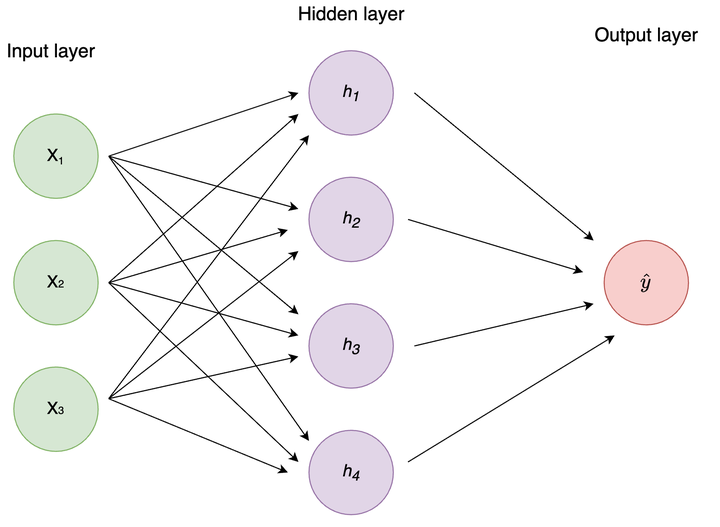
1. **Introduction**

The goal of this project is to develop a basic understanding of neural networks by implementing a simple network from scratch in C++. The network will be trained to recognize handwritten digits using the MNIST dataset, and will then be tested with custom hand-drawn digits. The implementation involves writing all the mathematical calculations of the neural network from scratch, providing a comprehensive understanding of the deep concepts underlying neural networks.

The main mission here is to teach the neural network to recognize handwritten digits, using both the training dataset and the test dataset. The training involves feeding the network tons of examples from the training dataset, letting it learn from the patterns. Then comes the testing part, where I throw new, unseen examples from the test dataset at the network to see how well it learned.

Accuracy is the key metric here, and it's calculated by comparing the network's predictions with the actual correct answers. This involves a series of mathematical steps like forward propagation (where the network makes predictions), backpropagation (where errors are identified and used to adjust the network), and a loss function (which measures how far off the predictions are from the actual values).

This project serves as an educational exploration, emphasizing a thorough understanding of the mathematical foundations of neural networks. The utilization of the MNIST dataset, a standard benchmark for digit recognition, adds practical relevance to the theoretical concepts explored in this project. Through the process of developing the network, training it on real-world data, and assessing its performance, I aspire to acquire a holistic understanding of the nuances involved in building and training neural networks.



Neural Network

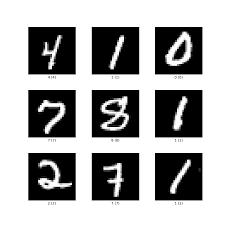
**2 Background:**

**2.1 Neural Network:**

In this project, I'm constructing a simple neural network, layer by layer, entirely from scratch in C++. This includes defining how it processes information, learns from examples, and improves over time.

**2.2 MNIST Dataset:**

The MNIST dataset is like the textbook for our neural network. It consists of thousands of handwritten digits (0 to 9), making it a standard benchmark for training and testing digit recognition models. By using MNIST, I'm putting my neural network through a real-world challenge of recognizing diverse handwritten numbers.



Hand written digits from

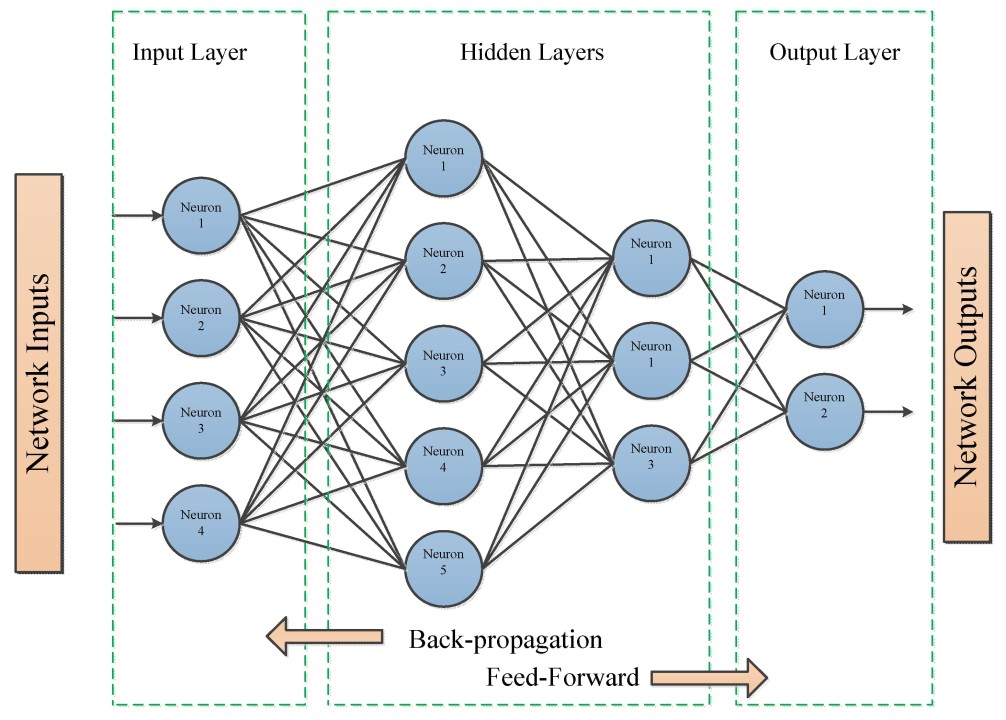
Mnist Dataset

**2.3 Training and Testing:**

Training the neural network involves exposing it to the examples in the training dataset, letting it learn patterns and associations. Then, the real test happens with the test dataset, where the network encounters new, unseen examples. Accuracy, a crucial measure, is calculated by comparing the network's predictions with the actual answers.

**2.4** **Forward Propagation and Backpropagation:**

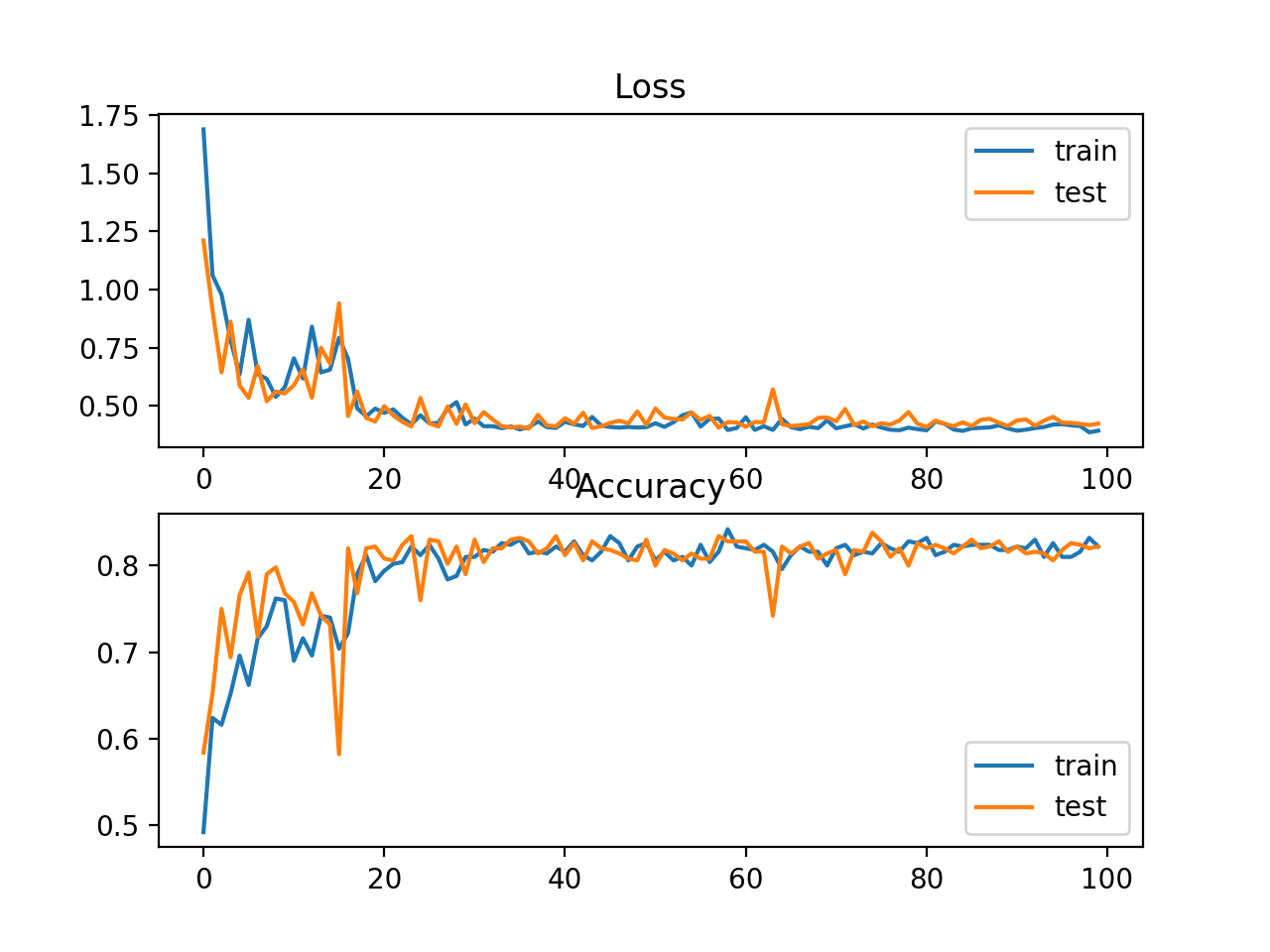
These are the nuts and bolts of how the neural network learns. Forward propagation is like the network's thought process — it makes predictions based on what it has learned. Backpropagation, on the other hand, is the learning process. Errors are identified, and the network adjusts its internal settings to minimize those errors, getting better with each iteration.



Forward Propagation and Backpropagation

**2.5 Loss Function:**

The loss function is like a teacher checking the network's homework. It measures how far off the predictions are from the correct answers. The network's goal during training is to minimize this loss, becoming more accurate in recognizing digits.



Graph of Loss Function

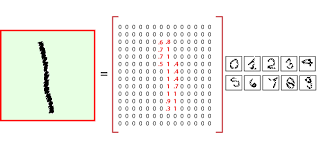
Understanding these background concepts is crucial as they form the foundation for the hands-on exploration in this project. It's not just about creating a program; it's about understanding how and why the program works, from the ground up.

**3 Description:**

**3.1 Data Processing:**

* **MNIST Dataset Exploration:**
  + Commencing with the MNIST dataset, which serves as a collection of handwritten digits.
* **Loading the Dataset:**
  + The first step involves loading the dataset, extracting the raw information contained within.
* **Normalization:**
  + To facilitate effective neural network processing, a crucial step is normalizing the data. This ensures that the input values are adjusted to a scale suitable for the network's operations.
* **Reshaping:**
  + Depending on the intricacies of the neural network's design, there might be a need to reshape the data. This step aligns the dataset structure with the specific architecture of the neural network.

This initial phase sets the foundation for the subsequent stages of the project, transforming raw handwritten digits into a format suitable for consumption by the neural network.

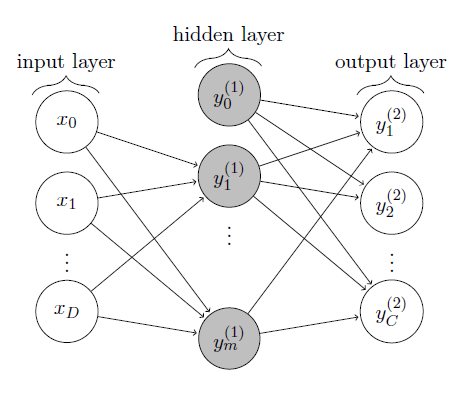


2D Digit Array from Data-set

**3.2** **Neural Network Design:**

* **Architectural Elements:**
  + The neural network is meticulously crafted, featuring distinct architectural elements. It comprises an input layer, one or more hidden layers, and an output layer.
* **Input Layer Configuration:**
  + The input layer is tailored to align with the number of pixel features, akin to the pixels found in an image. This configuration ensures that the network can process the visual information represented by these pixels.
* **Hidden Layers:**
  + One or more hidden layers are strategically integrated into the design. These layers serve as intermediaries, extracting and learning hierarchical representations of features from the input data.
* **Output Layer Significance:**
  + The output layer holds particular significance, consisting of 10 neurons. Each neuron corresponds to one of the ten possible digits (0-9). The network's ultimate task is to activate the appropriate neuron, effectively classifying the input digit.
* **Tailored Craftsmanship:**
  + This neural network is not an off-the-shelf solution; it's a product of careful craftsmanship, aligning its architecture with the specific requirements of recognizing handwritten digits.

This phase represents the core of the project, where the digital mind is not just a tool but a purposefully designed entity, tailored to the intricacies of recognizing and classifying handwritten digits.

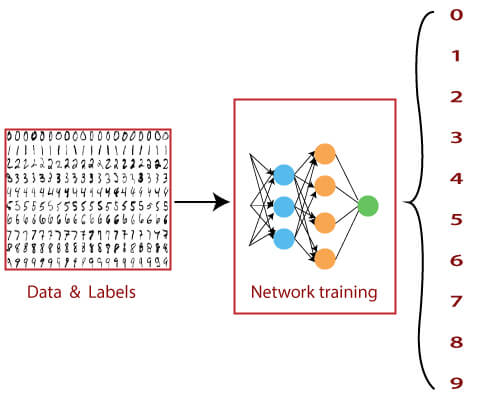


Neural Network Design

**3.3 Training:**

* **Neural Network Architecture Setup:**
  + Before diving into training, the neural network architecture is established. This includes defining the layers, connections, and overall structure.
* **Feedforward Propagation:**
  + The training process kicks off with feedforward propagation. Inputs traverse the neural network, activating neurons along the way, ultimately producing outputs.
* **Loss Calculation:**
  + Simultaneously, the network computes the loss, which represents the difference between its predictions and the actual values. This metric quantifies how far off the network is from the correct answers.
* **Backpropagation:**
  + The heart of the learning odyssey, backpropagation, follows feedforward propagation. It involves an iterative dance:
    - Identifying errors: Recognizing the disparities between predictions and actual values.
    - Weight and Bias Adjustments: Systematically tweaking the network's internal configurations to minimize errors.
* **Iterative Refinement:**
  + The training process is an iterative journey. Each cycle of feedforward and backpropagation refines the network's understanding. It's akin to a learning odyssey, where the neural network hones its ability to discern and classify handwritten digits with each iteration.

This training phase is the soul of the project, where the neural network evolves, learns from examples, and refines its proficiency in recognizing diverse handwritten digits.

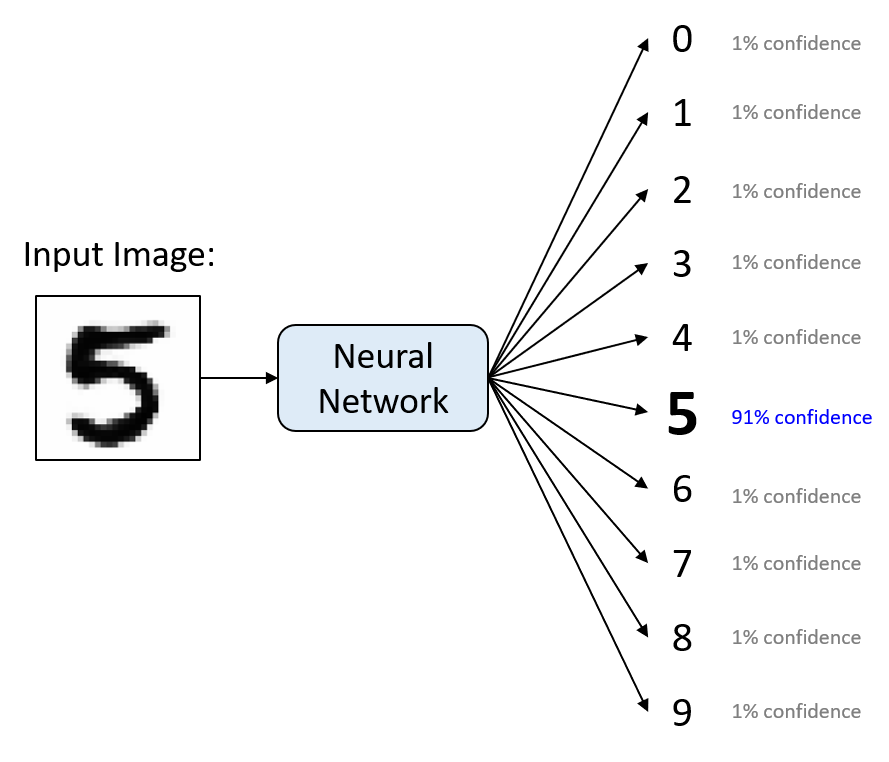


Network Training

**3.4** **Testing and Evaluation:**

* **Real-World Scenario Simulation:**
  + This phase mirrors a real-world scenario where the neural network encounters new, previously unseen examples. The objective is to mimic the challenges the network might face when confronted with novel inputs.
* **Robust Evaluation:**
  + The process provides a robust evaluation of the network's capabilities. It goes beyond its ability to memorize training data and gauges how well it generalizes to new, unfamiliar situations.
* **Performance Metrics:**
  + Various performance metrics, including accuracy, are calculated to quantify how well the neural network performs during this crucial evaluation phase.

This phase is pivotal in determining the real-world proficiency of the neural network. By objectively assessing its performance on unseen data, we gain insights into its adaptability, robustness, and ability to generalize beyond the training set.



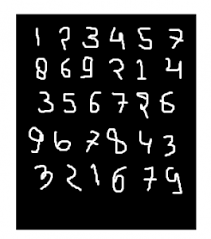
Testing and Evaluation

**3.5** **Custom Input Drawing:**

* **Interface Development:**
  + The project introduces a user-friendly interface designed for drawing custom digit inputs. This serves as a bridge between theoretical concepts and user interaction.
* **Real-Time Testing:**
  + The interactive feature allows users to draw custom digit inputs in real-time, providing immediate feedback on how the neural network interprets and classifies these hand-drawn digits.
* **Adaptability Assessment:**
  + The drawn inputs serve as a diverse set of examples, offering insights into the neural network's adaptability. It goes beyond standardized datasets, gauging its performance on inputs that might vary in style, quality, and representation.
* **User Interaction Dynamics:**
  + Users actively participate in the testing process by providing unique inputs. This not only makes the testing process engaging but also sheds light on how well the neural network can handle inputs generated outside conventional datasets.
* **Insights into Performance:**
  + The real-time testing of hand-drawn digits provides valuable insights into the practical performance of the neural network. It offers a nuanced understanding of its strengths and areas for improvement in handling diverse inputs.

This custom input drawing phase adds an interactive dimension to the project, allowing users to actively contribute to the testing process and providing a more holistic evaluation of the neural network's adaptability and real-world usability.

This comprehensive description outlines the multifaceted components of the project, ranging from data preprocessing to the design, training, testing, and interactive aspects of the neural network. The subsequent sections will delve into the nuts and bolts of implementation and testing, providing a closer look at the inner workings of our digital creation.

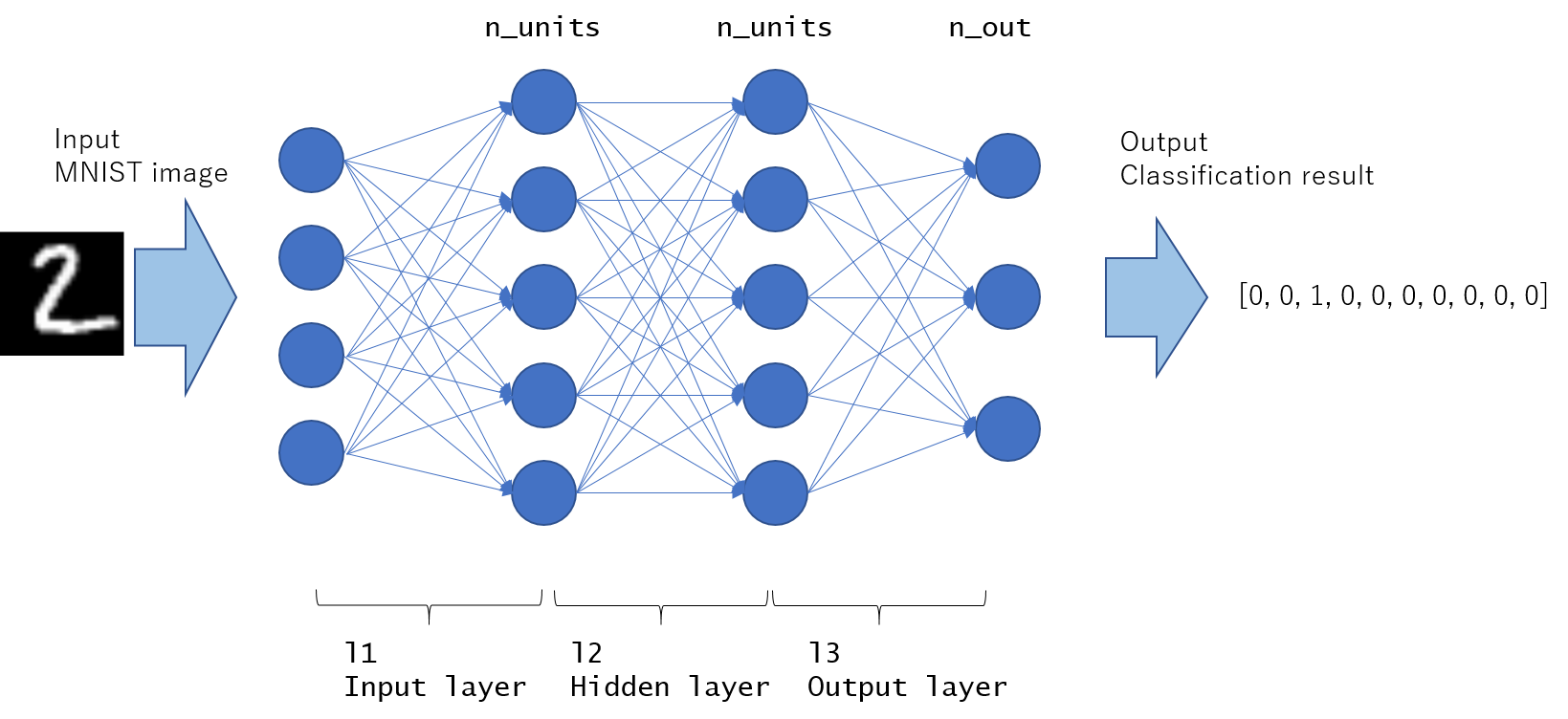


Custom Input Drawing

**4** **Implementation and Testing:**

* **Choice of Language:**
  + The implementation phase begins by choosing C++ as the language for building the neural network. This decision aligns with the project's objective of crafting the network from scratch.
* **Development Environment:**
  + The Integrated Development Environment (IDE), such as CLion, is selected for its compatibility with C++. It provides a conducive environment for coding, debugging, and testing.
* **Coding from Scratch:**
  + The implementation is a hands-on process involving writing code from scratch. This includes defining the neural network architecture, configuring layers, and implementing mathematical operations for forward and backward propagation.
* **Training Process:**
  + The training process is initiated, involving the orchestration of feedforward propagation, backpropagation, and weight/bias adjustments. This phase refines the network's internal parameters through exposure to the MNIST training dataset.
* **Testing Infrastructure:**
  + A testing infrastructure is established to assess the neural network's performance. This includes the deployment of the MNIST test dataset and the introduction of custom-drawn digits for real-time testing.
* **Performance Metrics:**
  + Various performance metrics, such as accuracy, loss, and possibly additional metrics, are calculated during the testing phase. These metrics quantitatively measure how well the neural network generalizes to both standardized and custom inputs.
* **Debugging and Optimization:**
  + Throughout the implementation and testing, the code is debugged, and optimization techniques may be applied to enhance the efficiency and effectiveness of the neural network.
* **Iterative Refinement:**
  + The implementation and testing phases operate iteratively. Insights gained during testing may lead to refinements in the implementation, fostering an ongoing cycle of improvement.

This combined phase encapsulates the hands-on process of translating the conceptual design of the neural network into a functional implementation, followed by rigorous testing to gauge its performance and identify areas for refinement.



Implementation and Testing

**5 User Interface:**

The user interface plays a crucial role in facilitating interaction with the neural network, offering a seamless experience for users to draw custom digit inputs and observe the network's real-time responses.

* **Drawing Interface:**
  + A user-friendly drawing interface is designed, allowing users to sketch custom digit inputs directly. This intuitive feature serves as a bridge between the user and the neural network, enabling hands-on interaction.
* **Real-Time Feedback:**
  + The interface provides real-time feedback, allowing users to witness how the neural network interprets and classifies the hand-drawn digits instantly. This dynamic interaction enhances user engagement and understanding.
* **Interactive Testing:**
  + Users can actively participate in the testing process by providing diverse and custom inputs. This interactive element goes beyond standardized datasets, offering a more realistic assessment of the neural network's adaptability.
* **Output Display:**
  + The interface includes a display area to showcase the neural network's predictions for the drawn digits. This visual representation enhances the user's understanding of the network's decision-making process.
* **Intuitive Controls:**
  + User controls are designed to be intuitive, ensuring that individuals, regardless of technical expertise, can easily draw and test custom digit inputs. This promotes accessibility and inclusivity in interacting with the neural network.
* **Clear Instructions:**
  + The interface includes clear instructions or prompts to guide users on how to interact with the drawing feature and interpret the network's responses. This transparency enhances the user experience and understanding.
* **User-Friendly Design:**
  + The overall design of the interface prioritizes user-friendliness, with a clean layout, easy navigation, and visually appealing elements. This approach aims to make the interaction with the neural network both enjoyable and educational.

The user interface serves as a crucial link between the user and the neural network, providing a platform for interactive testing and fostering a deeper understanding of how the network processes and classifies hand-drawn digit inputs in real-time.

**6 Challenges:**

**6.1 Data Loading and Preprocessing:**

* **Challenge**: Loading and preprocessing the MNIST dataset proved to be a substantial challenge due to its voluminous size and intricate binary format. The sheer magnitude of handwritten digits in the dataset demanded an efficient and robust approach to handle the data**.**
* **Strategy:** To tackle the data loading challenge, a custom loading mechanism was devised. This mechanism efficiently read the binary data, converting it into a format suitable for neural network consumption. Additionally, a meticulous normalization process was implemented to ensure that the data fed into the neural network was appropriately scaled, contributing to effective training and testing.
* **Optimization Efforts:** Further challenges arose during the preprocessing phase, including reshaping the data to align with the network's architecture. Strategies were implemented to optimize the preprocessing pipeline, ensuring seamless integration with the neural network design.
* **Real-time Data Exploration:** To gain insights into the challenges posed by the raw data, real-time data exploration tools were employed. This facilitated a deeper understanding of the dataset's structure and guided the preprocessing strategies.
* **Documentation and Collaboration:** Comprehensive documentation of the preprocessing steps was crucial to ensure transparency and collaboration. This documentation served as a reference point for the project team, aiding in troubleshooting and collaborative problem-solving.

The intricacies of handling the MNIST dataset underscored the significance of a robust data loading and preprocessing pipeline. Overcoming these challenges required a blend of creativity, technical expertise, and adaptability, ensuring that the neural network was provided with high-quality, well-prepared data for effective training and evaluation.

**6.2 Neural Network Optimization:**

* **Challenge:** Optimizing the neural network for efficient training and achieving high accuracy posed multifaceted challenges. Fine-tuning parameters and achieving the desired level of performance demanded a nuanced understanding of the network's dynamics.
* **Strategy:** The optimization process unfolded through an iterative journey of refinement. Key strategies included:
  + **Architectural Adjustments:** Iteratively refining the architecture by experimenting with different configurations of hidden layers, neurons, and activation functions.
  + **Hyperparameter Tuning:** Fine-tuning hyperparameters such as learning rates and batch sizes to strike a balance between training speed and accuracy.
  + **Regularization Techniques:** Implementing regularization techniques, such as dropout, to enhance the network's generalization capabilities and prevent overfitting.
* **Performance Monitoring:** Continuous monitoring of the network's performance metrics during training and testing phases played a pivotal role. This involved scrutinizing accuracy, loss, and other relevant metrics to gauge the effectiveness of optimization efforts.
* **Adaptable Learning Rates:** The challenge of finding an optimal learning rate was addressed by implementing adaptive learning rate techniques. This allowed the neural network to dynamically adjust its learning rate based on the current state of training.
* **Exploration of Activation Functions:** Experimentation with different activation functions, such as ReLU and Sigmoid, contributed to finding the most suitable functions for specific layers of the network.
* **Feedback Loop:** Establishing a feedback loop between optimization strategies and real-time testing provided insights into the impact of adjustments on the network's real-world proficiency.

The process of neural network optimization showcased the intricate dance between architectural design and parameter fine-tuning. The strategies employed aimed not only at overcoming immediate challenges but also at fostering a deeper understanding of the network's behavior and enhancing its overall efficiency and accuracy.

**6.3 Image Compression:**

* **Challenge:** Incorporating image compression into the project, specifically utilizing bicubic interpolation, introduced a distinct challenge. The goal was to effectively downsize console-drawn images from 420x420 to 28x28 while maintaining crucial details for accurate digit recognition.
* **Strategy:** A tailored strategy was developed to address the challenge of image compression. Bicubic interpolation, known for its superior quality compared to bilinear and nearest-neighbor methods, was implemented to ensure a more accurate representation of the original image during downsizing. This involved careful consideration of the interpolation parameters to strike a balance between compression and retaining essential information for neural network training.
* **Console Drawing Considerations:** The challenge extended to handling images drawn in the console, where pixel representation and character density played a role in compression. Fine-tuning the interpolation process and adjusting console drawing techniques became integral parts of the strategy.
* **Integration with Preprocessing:** Seamless integration of image compression into the overall preprocessing pipeline was crucial. This required adapting the preprocessing workflow to handle both raw and compressed image data, ensuring a harmonious transition for the neural network.

The challenge of image compression added a unique dimension to the project, emphasizing the importance of preserving information while reducing image dimensions. The strategies employed aimed at optimizing this process, contributing to a comprehensive understanding of handling diverse data transformations within the neural network framework.

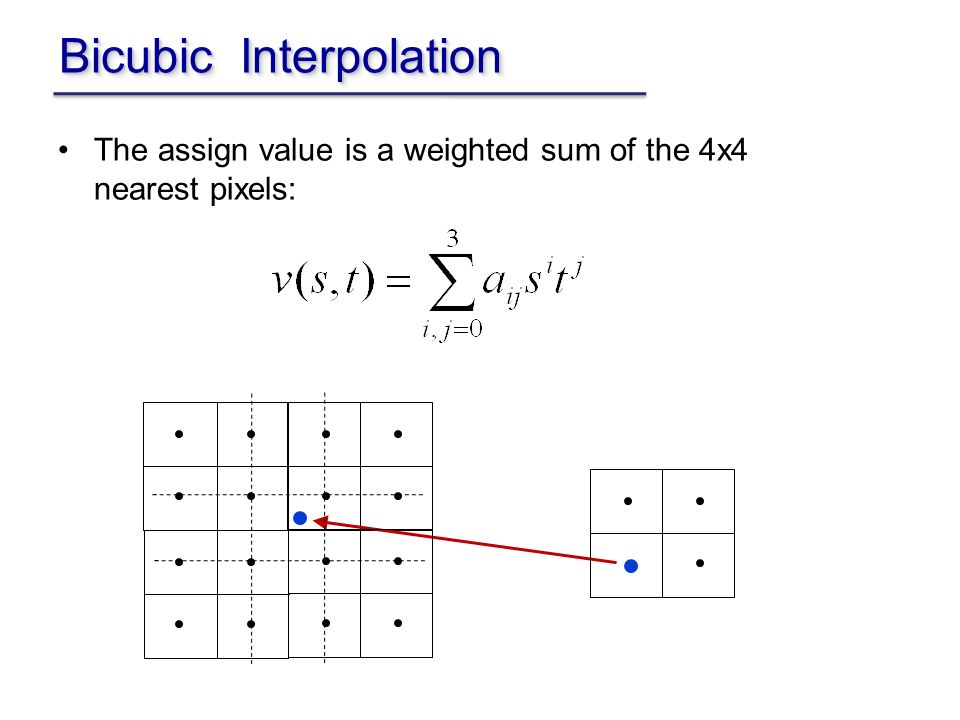


Image Compression

These challenges underscored the intricate nature of implementing a neural network from scratch. Addressing data loading complexities and optimizing network performance required a combination of creative problem-solving and a deep understanding of the underlying principles of neural networks. The strategies employed aimed not only to overcome immediate hurdles but also to enhance the overall robustness and effectiveness of the implemented solution.

**7. Conclusion:**

In traversing the realms of neural network implementation, from crafting the digital mind to overcoming intricate challenges, this project has been a voyage of discovery and learning. The endeavor to develop a profound understanding of neural networks was not merely about creating a functional system but about unraveling the mysteries of their inner workings.

**Neural Network Journey:** The heart of the project lay in the meticulous design and implementation of a neural network. Unlike off-the-shelf solutions, this digital mind was carefully crafted, featuring layers of intricacy that mimicked the human brain's capacity to learn and recognize patterns. The journey extended beyond theory, with the actual coding of forward propagation, backpropagation, and the orchestration of a learning odyssey.

**Challenges Faced and Overcome:** The challenges encountered, particularly in data loading and preprocessing, and the subsequent optimization of the neural network, provided fertile ground for problem-solving. Through custom data loading mechanisms, preprocessing refinements, and an iterative approach to optimization, each challenge was transformed into an opportunity for growth.

**Real-World Proficiency:** The testing and evaluation phase, deploying the network on the MNIST dataset and custom hand-drawn digits, offered a glimpse into its real-world proficiency. The interactive user interface provided a bridge between theoretical concepts and tangible interaction, emphasizing adaptability and performance in a diverse set of scenarios.

**Key Takeaways:** The project reinforced the significance of hands-on learning, delving into the complexities rather than relying on ready-made solutions. From loading raw data to optimizing network parameters, every step illuminated the intricacies of neural networks, offering invaluable insights.

**Future Extensions:** Looking forward, this project lays the foundation for future extensions. Refinements in the neural network architecture, exploration of advanced optimization techniques, and the integration of additional datasets could further enhance the network's capabilities. The custom drawing interface could evolve to simulate more diverse real-world scenarios, pushing the boundaries of the network's adaptability.

In conclusion, this project transcended the boundaries of a conventional coding exercise. It became a personal exploration into the realm of neural networks, an odyssey marked by challenges, triumphs, and a deeper understanding of the digital minds that shape our technological landscape.

**8 Reference:**

1. LeCun, Y., Cortes, C., & Burges, C. J. (1998). "MNIST handwritten digit database." *Yann LeCun's Website*. <http://yann.lecun.com/exdb/mnist/>
2. D. Stutz. (2019). "Understanding MNIST." *Towards Data Science*. <https://towardsdatascience.com/understanding-mnist-8a6e77d00fe2>
3. L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, ... & Y. LeCun. (1994). "Comparison of classifier methods: a case study in handwritten digit recognition." *International Conference on Pattern Recognition (ICPR)*, 1, 77-82.
4. S. Hochreiter, & J. Schmidhuber. (1997). "Long short-term memory." *Neural Computation*, 9(8), 1735-1780. DOI: 10.1162/neco.1997.9.8.1735
5. I. Goodfellow, Y. Bengio, & A. Courville. (2016). "*Deep Learning*." MIT Press.
6. A. Krizhevsky, I. Sutskever, & G. E. Hinton. (2012). "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in Neural Information Processing Systems (NIPS)*, 25, 1097-1105.
7. D. Cireşan, U. Meier, J. Masci, L. M. Gambardella, & J. Schmidhuber. (2011). "Flexible, high-performance convolutional neural networks for image classification." *International Joint Conference on Artificial Intelligence (IJCAI)*, 22(1), 1237-1242.

Top of Form

Top of Form