*A black text on a white background

Description automatically generated with medium confidence*

**AL2002-Artificial Intelligence - Lab**

**Project Report**

Project Name:

*Handwritten Digit Recognition Model*

Members:

Ahmed Raza 21K-3056

Khush Bakht Aliza 21K-4713

Muhammad Hasnain 21K-4714

**1. Introduction**

Digit recognition is an important task in computer vision and pattern recognition . It involves identifying handwritten digits. Deep learning, specifically neural networks, has been successful in solving this problem. In this project, we built a neural network model to recognize digits. We used a dataset of handwritten digit images and trained the model on part of the data. We evaluated the model's performance on a separate dataset. The model used multiple layers and activation functions to make predictions. The report explains the methodology, including data preparation and training. It also discusses the results, including sample predictions. The project shows the effectiveness of neural networks for digit recognition and suggests improvements for future work.

**2. Methodology**

The dataset used in this project consists of grayscale images of handwritten digits. Each image is 28x28 pixels in size and represents a digit from 0 to 9. The dataset is divided into training and development sets for model training and evaluation.

Before using the data, we applied preprocessing steps. First, we shuffled the dataset to ensure randomness during training. Then, we normalized the pixel values by dividing them by 255, scaling them to a range of 0 to 1. This normalization helps in training the neural network effectively.

The neural network architecture for digit recognition had two hidden layers. The first hidden layer had 10 neurons with the ReLU activation function, while the second hidden layer also had 10 neurons with ReLU activation. The output layer had 10 neurons, representing the probabilities of each digit class, and used the softmax activation function to produce a probability distribution.

During forward propagation, the neural network processed the input image through the layers to generate predicted probabilities for each digit class. The input image pixels were flattened, multiplied by the weight matrix of the first hidden layer, and passed through the ReLU activation function. This process was repeated for the second hidden layer, and finally, the softmax function was applied to obtain class probabilities.

In backward propagation, the model calculated gradients of the loss function with respect to the parameters. These gradients were then used to update the parameters using gradient descent. The weights and biases of the neural network were adjusted iteratively based on the gradients and a learning rate.

The model was trained using the training set, and its performance was evaluated on the development set at regular intervals. The training process continued until reaching a specified number of iterations or achieving satisfactory performance.

The performance of the model was assessed using accuracy, which measures the percentage of correctly predicted digits. Accuracy was calculated by comparing the predicted labels with the true labels in the development set.

This methodology provides an overview of the steps involved in training and evaluating the digit recognition model using a neural network. The following section will present implementation details and discuss the project's results.

**3. Implementation**

3.1 Dataset Split

The dataset was split into two sets: a training set and a development set. The training set was used to train the neural network, while the development set was used to evaluate the model's performance during training. The development set consisted of 1000 samples, and the remaining samples were used for training.

3.2 Parameter Initialization

The weights (W1, W2) and biases (b1, b2) of the neural network were randomly initialized using a Gaussian distribution. The shape of the weight matrices was determined based on the architecture of the network, with the first weight matrix (W1) having dimensions of 10x784, and the second weight matrix (W2) having dimensions of 10x10. The bias terms (b1, b2) were initialized as 10x1 matrices.

3.3 Gradient Descent and Model Training

The neural network was trained using gradient descent. The training process involved iterating over a specified number of iterations and updating the weights and biases based on the gradients computed during backward propagation. The learning rate (alpha) was set to 0.10, and the number of iterations was set to 500.

During each iteration, the forward propagation algorithm was executed to calculate the predicted probabilities for each digit class. Then, the backward propagation algorithm was performed to compute the gradients of the loss function with respect to the parameters. The weights and biases were updated using the update\_params() function, which subtracted the gradients multiplied by the learning rate from the current parameter values.

3.4 Prediction and Accuracy Calculation

To make predictions on new data, the make\_predictions() function was implemented. It took an input digit image, passed it through the trained neural network, and produced the predicted digit label based on the highest probability.

The accuracy of the model was calculated using the get\_accuracy() function. It compared the predicted labels with the true labels in the development set and calculated the percentage of correctly predicted digits.

3.5 Visualization and Testing

To visualize the predictions, the test\_prediction() function was implemented. It took an index from the training set, made predictions on the corresponding digit image, and displayed the predicted label along with the true label and the actual image.

3.6 Training Progress and Iteration Analysis

During the training process, the model's accuracy on the development set was monitored at regular intervals (every 10 iterations). The training progress, including the iteration number, predictions, and accuracy, was displayed for analysis and tracking.

This implementation section provides an overview of the steps involved in training the digit recognition model using a neural network. The next section will present the results obtained from the model and discuss its performance and implications.

**4. Discussion**

4.1 Model Performance

The implemented neural network model achieved a satisfactory accuracy of 0.892 on the development set. This indicates that the model was able to accurately recognize handwritten digit images with a high degree of success. The results demonstrate the effectiveness of the neural network approach for digit recognition tasks.

4.2 Factors Contributing to Performance

The high accuracy achieved by the model can be attributed to several factors. Firstly, the neural network architecture with two hidden layers and ReLU activation functions allowed the model to learn complex representations and capture intricate patterns in the digit images. The use of softmax activation in the output layer provided a probabilistic interpretation of the predictions.

Additionally, the normalization of pixel values in the preprocessing step helped in improving the convergence and stability of the neural network during training. The use of gradient descent optimization with appropriate learning rate (0.10) also contributed to the model's performance.

4.3 Limitations and Challenges

While the implemented model achieved satisfactory results, there are a few limitations and challenges that should be considered. One limitation is that the model's performance heavily relies on the quality and diversity of the training data. If the training set is not representative of the entire digit dataset, the model may struggle to generalize well to unseen examples.

Furthermore, the model's accuracy may be influenced by the choice of hyperparameters, such as the learning rate, number of iterations, and architecture of the neural network. These hyperparameters should be carefully tuned to achieve optimal performance.

4.4 Future Work and Improvements

There are several avenues for future work and improvements in digit recognition using neural networks. Here are a few recommendations:

1. Data Augmentation: To enhance the model's performance, data augmentation techniques could be applied to artificially expand the training set. Techniques such as rotation, scaling, and translation of the digit images can introduce variability and improve the model's ability to generalize.

2. Hyperparameter Optimization: Fine-tuning the hyperparameters, including the learning rate, number of iterations, and neural network architecture, can potentially lead to further improvements in accuracy. Techniques such as grid search or Bayesian optimization can be employed to systematically explore the hyperparameter space.

3. Ensemble Methods: Building an ensemble of multiple neural network models with different architectures or training strategies can help improve accuracy. Combining the predictions of multiple models can provide a more robust and reliable digit recognition system.

4. Transfer Learning: Transfer learning techniques can be explored by leveraging pre-trained models on large-scale datasets like ImageNet. By utilizing the learned features from such models, the digit recognition model may achieve better generalization and accuracy.

5. Deployment and Optimization: Once the model is trained and evaluated, it can be deployed in real-world applications. Optimizing the model for efficient inference on resource-constrained devices can be crucial for practical deployment scenarios.

**5. Conclusion**

In this project, we successfully developed a neural network model for digit recognition. The implemented model achieved a satisfactory accuracy of 0.892 on the development set, demonstrating its effectiveness in accurately recognizing handwritten digit images.

By leveraging techniques such as gradient descent, ReLU activation, softmax output, and parameter initialization, the model was able to learn complex representations and capture intricate patterns in the digit images. The normalization of pixel values and the careful selection of hyperparameters further contributed to the model's performance.

The project also highlighted the importance of data preprocessing, model training, and evaluation. The dataset was appropriately split into training and development sets, and the model's performance was monitored throughout the training process. Sample predictions and accuracy calculations provided insights into the model's behavior and generalization capabilities.

While the implemented model achieved satisfactory results, there are still opportunities for future improvements. Data augmentation, hyperparameter optimization, ensemble methods, transfer learning, and deployment optimizations are potential areas for further exploration and enhancement.

Overall, this project demonstrates the power and potential of neural networks in digit recognition tasks. The ability to accurately recognize handwritten digits has significant implications in various fields, including optical character recognition, automated document processing, and computer vision systems.

By continuing to refine and advance the techniques and methodologies used in this project, we can further improve digit recognition models and open up new possibilities for their practical applications.

**6. References**

1. LeCun, Y., Cortes, C., & Burges, C. J. (2010). MNIST handwritten digit database. Retrieved from http://yann.lecun.com/exdb/mnist/

2. YouTube

3. Chat GPT

4. Kaggle

**A picture containing text, handwriting, font, number

Description automatically generated**

**A paper with writing on it

Description automatically generated with low confidence**

**A paper with writing on it

Description automatically generated with low confidence**