Indian Institute of Technology Jodhpur

Handwritten Digit Recognition

Computer Vision (CSL7360) Project Report

~ *Chilaka Dhanya Sri (B20AI007)*

Bikkavolu Prasanthi (B20CS010)

Tammireddi Sai Unnathi (B20AI045)

1. Introduction

Handwritten digit recognition is a fundamental task in the field of machine learning and computer vision.It involves training a model to accurately classify images of handwritten digits (0-9) into their respective categories.This task has wide-ranging applications, including optical character recognition (OCR), digitization of documents, and automated postal services.The digit recognition problem poses unique challenges due to variations in writing styles, different pen pressures, and varying levels of noise in the images.To address these challenges, advanced techniques such as deep learning with convolutional neural networks (CNNs) have been widely adopted.These models can learn intricate patterns and features from digit images, making them highly effective in handwritten digit recognition tasks.

2. Dataset Creation

We have created a dataset with videos of hand drawn sketches of digits 0-9 which are then saved in separate folders according to their classes.

Then extract a sequence of frames from the video and apply transforms to it

And then load into data_loader

Note:

We have chosen this kind of data instead of taking image data from MNIST as we can get more information out of it , as there's different steps used for drawing data which will add one extra dimension to out data hence better prediction)

3. Data Preprocessing

- Frame Extraction

    - Frame extraction is a critical step in processing video data, enabling the analysis of sequential information.
    - It involves capturing specific frames from videos, which are then processed further for feature extraction and model training.
- Frame Preprocessing
    - Preprocessing frames involves several transformations aimed at standardizing the data and making it suitable for model input.
    - These transformations include resizing frames to a consistent size, normalizing pixel values to a standard range, and converting frames into tensors, which are multi-dimensional arrays compatible with deep learning models.

4. Model Architecture

The model architecture combines state-of-the-art techniques to achieve robust digit recognition in video sequences.

- Pretrained CNN (ResNeXt50)
    - The use of a pretrained Convolutional Neural Network (CNN) such as ResNeXt50 allows the model to leverage pre-trained learned features from a large dataset.
    - This enhances the model's ability to extract meaningful spatial features from video frames, improving overall performance.
- LSTM for Temporal Modeling
    - The Long Short-Term Memory (LSTM) layer captures temporal dependencies in the sequence of frames.
    - This is crucial for recognizing patterns and movements over time, making the model more effective in understanding digit sequences in videos.
- Linear Classifier
    - The linear classifier performs digit classification based on the features extracted by the CNN and LSTM layers.
    - It assigns probabilities to different digit classes, allowing the model to make accurate predictions.

5. Training and Testing

- Train-Test Split
    - The dataset is divided into training and validation sets using a specified ratio (e.g; 80% training, 20% validation).
    - This division ensures that the model's performance is evaluated on unseen data, helping to assess its generalization capabilities.
- Data Loaders
    - PyTorch's DataLoader is utilized for efficient batch processing during both training and validation.
    - This facilitates the feeding of batches of preprocessed video frames into the model, optimizing memory usage and training speed.
- Training Loop
    - The training loop, implemented in the train_epoch function, iterates over the training dataset for multiple epochs.
    - During each epoch, it calculates the loss, performs backpropagation, and updates the model's weights using an optimizer (e.g; Adam).
    - This iterative process allows the model to learn from the training data and improve its performance over time.
- Testing Loop
    - The testing loop, implemented in the test function, evaluates the model's performance on the validation set after each epoch of training.
    - It computes metrics such as accuracy and loss to assess how well the model generalizes to new, unseen data.
    - This validation step is crucial for detecting overfitting and ensuring the model's reliability in real-world scenarios.

6. Model Evaluation

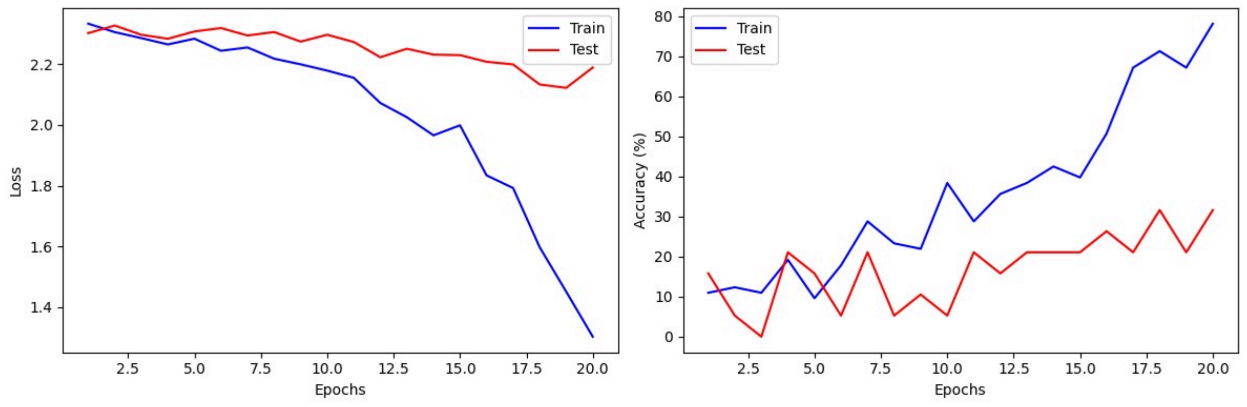
- Metrics Calculation
    - Metrics such as accuracy, loss, and possibly confusion matrix are calculated during both training and testing phases.

- These metrics provide quantitative measures of the model's performance, indicating its ability to correctly classify digits in video sequences.

- Visualization
  - Matplotlib is used to visualize the training and testing metrics across epochs.
  - Plots such as loss curves and accuracy trends help in monitoring the model's learning progress, identifying convergence, and detecting any anomalies or fluctuations in performance.

7. Results and Analysis

- Model Performance
  - The trained model's performance is evaluated based on metrics obtained during testing.
  - This includes assessing accuracy, which measures the percentage of correctly predicted digits, and loss, which indicates the model's predictive errors.



- Analysis

  By observing above plotted graphs we can say
  - Training and Testing Accuracies:
    - Increasing Trend: Both the training and testing accuracies exhibit a consistent upward trend across epochs. This indicates that the model's ability to correctly classify handwritten digits is improving with each training iteration.
  - Training and Testing Losses:

        - Decreasing Trend: The training and testing losses show a steady decrease over epochs. This implies that the model is effectively reducing its predictive errors during training and validation.
        - Model Learning: The decreasing losses indicate that the model is learning to differentiate between different digit classes.

Overall, the analysis indicates positive progress in the model's learning and performance. The increasing accuracies and decreasing losses demonstrate that the model is making meaningful improvements in its ability to recognize handwritten digits accurately. These trends are indicative of successful training and validation processes, leading to a well-performing digit recognition system.

8. Conclusion

In conclusion, the development of a robust handwritten digit recognition model showcases the power of machine learning and computer vision in automating tasks that involve processing handwritten data. Through the use of sophisticated algorithms like CNNs, coupled with efficient training methodologies and thorough model evaluation, accurate digit recognition can be achieved across diverse datasets. The successful implementation of a handwritten digit recognition system opens doors to numerous practical applications, including automated form processing, digitized document management, and enhanced accessibility tools for visually impaired individuals. As technology continues to evolve, further advancements in digit recognition models are expected, leading to even greater accuracy, speed, and reliability in handling handwritten data in various domains.

9. Future Directions

- Model Optimization:
    - We can further optimize the model's performance, such as hyperparameter tuning, architecture adjustments, or regularization techniques.
- Dataset Expansion:
    - We can explore possibilities for expanding the dataset to include more diverse digit samples, different backgrounds, or varied lighting conditions.

- Advanced Techniques:
    - Consider integrating advanced techniques like attention mechanisms, ensemble learning, or transfer learning for improved results.
- Deployment Considerations:
    - Considerations for deploying the model in real-world applications, such as scalability, inference speed, and integration with existing systems.

10. References

- https://www.geeksforgeeks.org/handwritten-digit-recognition-using-neural-network/
- https://github.com/SebLague/Neural-Network-Experiments
- https://www.analyticsvidhya.com/blog/2021/11/newbies-deep-learning-project-to-recognize-handwritten-digit/

Github Link:
https://github.com/Prasanthi1201/Handwritten-Digit-Recognition/tree/main