Enhancing CNN and RNN Performance: A Comparative Study with and without Autoencoder

Introduction

This report evaluates four different approaches for a classification problem using the MNIST dataset. The accuracy results of each approach have been analyzed and compared. The approaches used are as follows:

- CNN
- CNN after compression and resizing with Autoencoder
- RNN
- RNN after compression and resizing with Autoencoder

Simulation Steps

The simulation was conducted in the MATLAB environment. Four different methods were examined:

Use of CNN Only:

- Handwritten digits from the MNIST dataset were processed directly using a CNN model.
- The CNN model was trained on the training dataset for 50 epochs, and classification accuracy was measured on the test dataset.
- Data augmentation was applied to enhance the model's generalization capability.
- The training process was repeated 10 times, and the test accuracy for each simulation was calculated.

Using Autoencoder for Data Compression and CNN:

- The training dataset was first processed and compressed using an Autoencoder.
- Compression Ratio: The Autoencoder compressed the 28x28 input images to a 7x7 size (using max pooling), reducing the data size by approximately 16 times. The data was then restored to its original dimensions via the decoder.
- Subsequently, this compressed data was used as input for the CNN. The epoch size for the Autoencoder was set to 10.
- The Autoencoder aimed to minimize data loss during compression.
- The CNN was trained on the compressed data for 50 epochs, and classification accuracy was measured on the test dataset.
- The training process was repeated 10 times, and test accuracy was calculated for each simulation.

Use of RNN Only:

- Handwritten digits from the MNIST dataset were processed directly using an RNN model.
- Since the RNN model operates based on time steps, the 28x28 image files were transformed into 28 sequential time steps, each containing 28 data points.
- The RNN model was trained on the training dataset for 50 epochs, and classification accuracy was measured on the test dataset.
- Data augmentation was applied to enhance the model's generalization capability.
- The training process was repeated 10 times, and test accuracy was calculated for each

Using Autoencoder for Data Compression and RNN:

- The training dataset was first processed and compressed using an Autoencoder.
- Compression Ratio: The Autoencoder compressed the 28x28 input images into 7x7 dimensions (using max pooling), reducing the data size by approximately 16 times. The data was then restored to its original dimensions via the decoder.
- Subsequently, this compressed data was used as input for the RNN. The epoch size for the Autoencoder was set to 10.
- Since the RNN model operates based on time steps, the 28x28 image files were transformed into 28 sequential time steps, each containing 28 data points.
- The Autoencoder aimed to minimize data loss during compression.
- The RNN model was trained on the compressed data for 50 epochs, and classification accuracy was measured on the test dataset.
- The training process was repeated 10 times, and test accuracy was calculated for each simulation.

Simulation Results

CNN

```
Simülasyon Sonuçları:
Her bir simülasyon doğruluğu:
    0.9654
    0.9512
    0.9564
    0.9568
    0.9344
    0.9476
    0.9146
    0.9838
    0.9428
    0.9476

Ortalama Doğruluk: 95.01%
```

Autoencoder + CNN

```
Simülasyon Sonuçları:
Her bir simülasyon doğruluğu:
    0.9102
    0.9092
    0.8910
    0.8790
    0.8844
    0.9266
    0.9048
    0.9054
    0.9304
    0.9148

Ortalama Doğruluk: 90.56%
```

RNN

```
Simülasyon Sonuçları:
Her bir simülasyon doğruluğu:
    0.9910
    0.9934
    0.9916
    0.9928
    0.9904
    0.9896
    0.9908
    0.9892
    0.9888
    0.9926

Ortalama Doğruluk: 99.10%
```

Autoencoder + RNN

```
Simülasyon Sonuçları:
Her bir simülasyon doğruluğu:
    0.9856
    0.9878
    0.9762
    0.9822
    0.9860
    0.9836
    0.9812
    0.9846
    0.9848
    0.9862

Ortalama Doğruluk: 98.38%
```

CNN and RNN Comparison

Accuracy Performance:

- **CNN**: The CNN model achieved an average accuracy of 95.01%, demonstrating its effectiveness in classifying static images.
- **RNN**: The RNN model achieved a higher accuracy of 99.10%, surpassing the CNN. This highlights the RNN's ability to process sequential data effectively, even in tasks involving compressed images.
- **Conclusion**: The RNN showed a better understanding of the data characteristics, resulting in higher classification accuracy. However, RNNs generally come with higher computational costs.

Application Scenarios:

- **Use of CNN**: Preferred for scenarios requiring static image classification or when a faster and cost-efficient solution is needed.
- **Use of RNN**: Suitable when higher accuracy is required, and the model needs to adapt to sequential data.

Comparison of Using and Not Using Autoencoder

Accuracy Performance:

- Autoencoder + CNN: The CNN model used after data compression with an Autoencoder achieved an average accuracy of 90.56%, reflecting slightly lower performance. This suggests that some information may have been lost during the compression process.
- Autoencoder + RNN: The RNN model used after data compression with an Autoencoder achieved an average accuracy of 98.38%, demonstrating that RNN performs better even with compressed data.

Conclusion

In this study, four different classification models—CNN, RNN, Autoencoder + CNN, and Autoencoder + RNN—were compared. The results indicate that using RNN alone provides the highest accuracy rate. Autoencoders, on the other hand, stand out as a useful tool in scenarios requiring data compression. They are particularly effective for reducing data size when working with large datasets or limited hardware resources. However, a slight decrease in classification accuracy was observed during the compression process. This loss was largely mitigated when the Autoencoder was combined with RNN, demonstrating its effectiveness in maintaining performance even with compressed data.