Project Progress Report: Human Activity Classification

Gurkan Kilicaslan
kilia064@umn

I'm attempting to create a network that uses input from three different sensors across T = 150 time samples to classify human activities. There are 3000 samples in the training set and 600 samples in the test set. Wearable sensors that are attached to loose clothing are used to measure static postures. Despite the fact that the data I'm utilizing is a modified version of the data in [1]'s supplementary materials, it still provides a very clear picture of the project.

To do this, I considered applying three distinct types of methods—RNN, LSTM, and GRU—and contrasting them. Although I am already having difficulty with RNN implementation, and the other two are also challenging techniques, it might not be feasible to implement all of these strategies. Nevertheless, based on your comments, I may use a lot of functions in the code to implement LSTM and GRU if the RNN algorithm is accurate and producing decent results. Feedback is therefore very valued.

Peer reviews have recommended using k-nearest neighbors, logistic regression, etc. to compare the outcomes to RNN. These ideas, in my opinion, are excellent and may help you save time. The worst-case scenario is if I choose to use one of the listed techniques rather than GRU and LSTM. I appreciate the reviews.

Application of RNN

Following is a small description of the algorithm:

To classify human activity, I put into practice a single-layer RNN design with 128 hidden neuron numbers and hyperbolic tangent activation. With a mini-batch size of 32 samples, a learning rate of 0.1, a momentum rate of 0.85, and a maximum of 50 epochs, I utilized a stochastic gradient descent technique. I used the Xavier Uniform distribution for weight initialization. The formula for it is as follows:

$$w_o = \sqrt{\frac{6}{L_{pre} + L_{post}}}, W = [-w_o, w_o]$$

The first layer is used as the recurrent layer during network training:

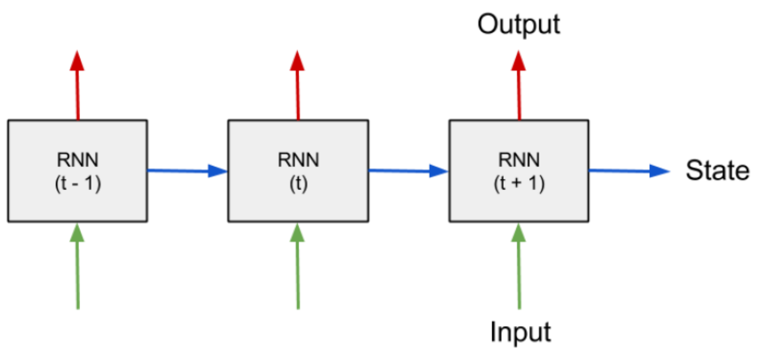

Figure 1: Recurrent Layer

For the issue, the RNN's mathematical meaning at each time step t is as follows:

$$h(t) = \tanh(x(t)W_{ih} + h(t-1)W_{hh} + bias)$$

where h is the hidden layer neuron number and i is the input neuron number.

The results are shown below:

We can see from the loss plot (Fig. 2) that there is fluctuation, which indicates that the network is not stable—something we don't want to see. The loss tends to increase extremely easily when learning rate rises. This is due to the cumulative nature of backpropagation through time (BPTT). Additionally, as shown in Fig. 4, the network only learns a part of the data and is unable to correctly predict the majority of classes (see **Appendix**).

The accuracy and loss plots show some instability after RNN training. The network is not learning if, in the long term, it does not always approach a reduced loss as training progresses because of the vanishing and ballooning gradients problem of RNN. In this situation, gradients become extremely high or extremely low, which causes loss to shift quickly. Rapid gradient changes for RNN are also produced with 150 time samples and the BPTT cumulative features. The instability may be influenced by the parameters. For instance, lowering the learning rate can help reduce overshooting, but if it's done too much, it might also cause training to go slowly or even stop altogether. The goals of GRU and LSTM are to solve these issues. I'll make an effort to use at least one GRU and LSTM. Prior to that, I want to use the suggested methods in reviews to check if it performs as the peer reviews have shown.

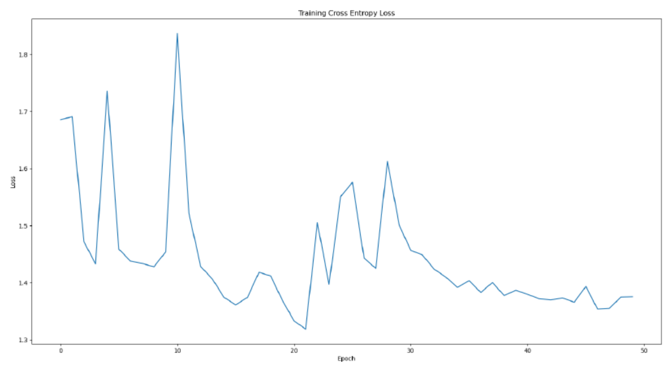

Figure 2: RNN Training Cross Entropy Loss

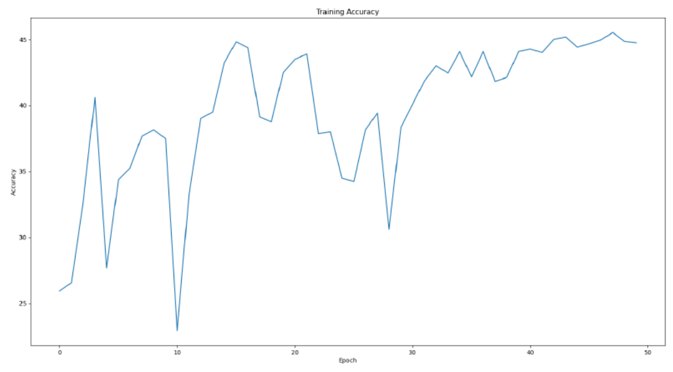

Figure 3: RNN Training Accuracy

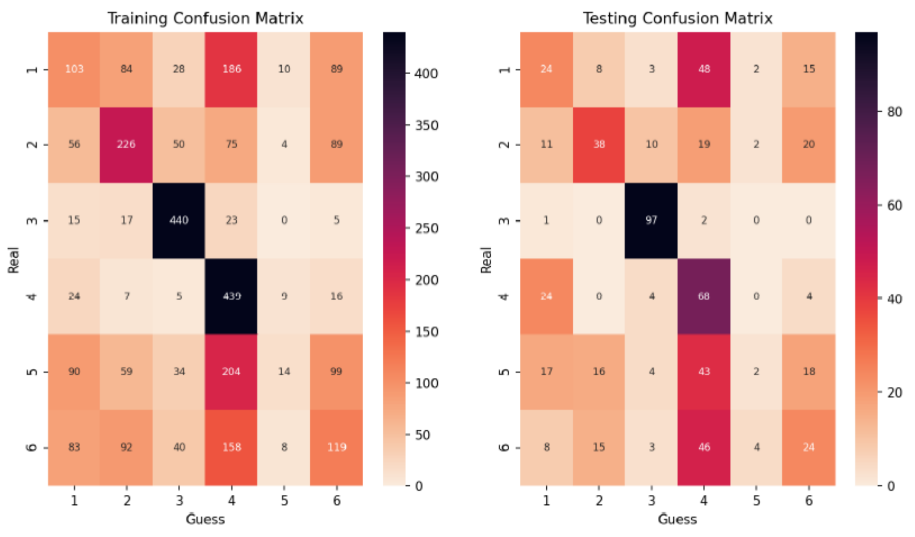

Figure 4: RNN Confusion Matrix

References

[1] Jayasinghe, U., Janko, B., Hwang, F., & Harwin, W. S. (2023). Classification of static postures with wearable sensors mounted on loose clothing. *Scientific Reports, 13 (1).* https://doi.org/10.1038/s41598-022-27306-4

Appendix

```
Epoch: 1 === Training Loss: 1.791,Validation Loss: 1.791,Training Accuracy: 16.519,Validation Accuracy: 17.333
Epoch: 2 === Training Loss: 1.773,Validation Loss: 1.777,Training Accuracy: 17.037,Validation Accuracy: 17.667
Epoch: 3 === Training Loss: 1.748,Validation Loss: 1.753,Training Accuracy: 17.926,Validation Accuracy: 19.667
Epoch: 4 === Training Loss: 1.743,Validation Loss: 1.753,Training Accuracy: 17.407,Validation Accuracy: 16.333
Epoch: 5 === Training Loss: 1.745,Validation Loss: 1.755,Training Accuracy: 18.926,Validation Accuracy: 17.667
Epoch: 6 === Training Loss: 1.728,Validation Loss: 1.745,Training Accuracy: 31.667,Validation Accuracy: 28.333
Epoch: 7 === Training Loss: 1.722,Validation Loss: 1.741,Training Accuracy: 31.926,Validation Accuracy: 28.667
Epoch: 8 === Training Loss: 1.786,Validation Loss: 1.787,Training Accuracy: 28.185,Validation Accuracy: 24.000
Epoch: 9 === Training Loss: 1.774,Validation Loss: 1.778,Training Accuracy: 31.667,Validation Accuracy: 27.333
Epoch: 10 === Training Loss: 1.734,Validation Loss: 1.746,Training Accuracy: 23.556,Validation Accuracy: 21.333
Epoch: 11 === Training Loss: 1.704,Validation Loss: 1.716,Training Accuracy: 21.815,Validation Accuracy: 20.333
Epoch: 12 === Training Loss: 1.717,Validation Loss: 1.732,Training Accuracy: 31.778,Validation Accuracy: 30.333
Epoch: 13 === Training Loss: 1.704,Validation Loss: 1.719,Training Accuracy: 30.778,Validation Accuracy: 28.000
Epoch: 14 === Training Loss: 1.735,Validation Loss: 1.731,Training Accuracy: 31.481,Validation Accuracy: 33.333
Epoch: 15 === Training Loss: 1.700,Validation Loss: 1.716,Training Accuracy: 31.963,Validation Accuracy: 29.667
Epoch: 16 === Training Loss: 1.786,Validation Loss: 1.799,Training Accuracy: 16.074,Validation Accuracy: 14.667
Epoch: 17 === Training Loss: 1.751,Validation Loss: 1.765,Training Accuracy: 16.667,Validation Accuracy: 15.000
Epoch: 18 === Training Loss: 1.753,Validation Loss: 1.771,Training Accuracy: 16.630,Validation Accuracy: 15.000
Epoch: 19 === Training Loss: 1.751,Validation Loss: 1.768,Training Accuracy: 16.630,Validation Accuracy: 15.000
Epoch: 20 === Training Loss: 1.760,Validation Loss: 1.779,Training Accuracy: 16.630,Validation Accuracy: 15.000
Epoch: 21 === Training Loss: 1.745,Validation Loss: 1.760,Training Accuracy: 16.630,Validation Accuracy: 15.000
Epoch: 22 === Training Loss: 1.742,Validation Loss: 1.754,Training Accuracy: 16.630,Validation Accuracy: 15.000
Epoch: 23 === Training Loss: 1.743,Validation Loss: 1.758,Training Accuracy: 16.630,Validation Accuracy: 15.000
Epoch: 24 === Training Loss: 1.797,Validation Loss: 1.801,Training Accuracy: 16.926,Validation Accuracy: 14.667
Epoch: 25 === Training Loss: 1.797,Validation Loss: 1.801,Training Accuracy: 16.926,Validation Accuracy: 14.667
Epoch: 26 === Training Loss: 1.797,Validation Loss: 1.801,Training Accuracy: 16.963,Validation Accuracy: 14.667
Epoch: 27 === Training Loss: 1.795,Validation Loss: 1.799,Training Accuracy: 17.259,Validation Accuracy: 15.333
Epoch: 28 === Training Loss: 1.785,Validation Loss: 1.789,Training Accuracy: 16.630,Validation Accuracy: 15.333
Epoch: 29 === Training Loss: 1.784,Validation Loss: 1.788,Training Accuracy: 16.704,Validation Accuracy: 15.333
Epoch: 30 === Training Loss: 1.784,Validation Loss: 1.788,Training Accuracy: 16.704,Validation Accuracy: 15.000
Epoch: 31 === Training Loss: 1.783,Validation Loss: 1.788,Training Accuracy: 16.704,Validation Accuracy: 15.333
Epoch: 32 === Training Loss: 1.783,Validation Loss: 1.787,Training Accuracy: 16.741,Validation Accuracy: 15.000
Epoch: 33 === Training Loss: 1.783,Validation Loss: 1.787,Training Accuracy: 16.741,Validation Accuracy: 15.000
Epoch: 34 === Training Loss: 1.798,Validation Loss: 1.801,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 35 === Training Loss: 1.798,Validation Loss: 1.800,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 36 === Training Loss: 1.797,Validation Loss: 1.799,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 37 === Training Loss: 1.797,Validation Loss: 1.798,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 38 === Training Loss: 1.797,Validation Loss: 1.799,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 39 === Training Loss: 1.796,Validation Loss: 1.799,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 40 === Training Loss: 1.796,Validation Loss: 1.799,Training Accuracy: 16.593,Validation Accuracy: 15.000
Epoch: 41 === Training Loss: 1.795,Validation Loss: 1.798,Training Accuracy: 16.444,Validation Accuracy: 15.000
Epoch: 42 === Training Loss: 1.794,Validation Loss: 1.798,Training Accuracy: 16.037,Validation Accuracy: 14.000
Epoch: 43 === Training Loss: 1.794,Validation Loss: 1.798,Training Accuracy: 15.852,Validation Accuracy: 14.000
Epoch: 44 === Training Loss: 1.794,Validation Loss: 1.798,Training Accuracy: 16.370,Validation Accuracy: 14.667
Epoch: 45 === Training Loss: 1.794,Validation Loss: 1.797,Training Accuracy: 16.185,Validation Accuracy: 14.667
Epoch: 46 === Training Loss: 1.793,Validation Loss: 1.796,Training Accuracy: 16.000,Validation Accuracy: 14.333
Epoch: 47 === Training Loss: 1.793,Validation Loss: 1.797,Training Accuracy: 16.259,Validation Accuracy: 14.667
Epoch: 48 === Training Loss: 1.793,Validation Loss: 1.796,Training Accuracy: 16.370,Validation Accuracy: 15.000
Epoch: 49 === Training Loss: 1.792,Validation Loss: 1.795,Training Accuracy: 16.407,Validation Accuracy: 15.000
Epoch: 50 === Training Loss: 1.791,Validation Loss: 1.794,Training Accuracy: 16.481,Validation Accuracy: 15.000
```