RNN-BASED HUMAN ACTIVITY RECOGNITION

A PREPRINT

Baran Can Gül
Deep Learning Lab 20/21
Institute of Signal processing and System Theory
University of Stuttgart
st168861@stud.uni-stuttgart.de

Lydia Schönpflug
Deep Learning Lab 20/21
Institute of Signal processing and System Theory
University of Stuttgart
st169955@stud.uni-stuttgart.de

February 16, 2021

ABSTRACT

Deep Learning approaches to Human Activity Recognition (HAR) are an active field of research with applications such as elderly and youth care, daily life monitoring or assisting Industry Manufacturing [1]. Based on the Human Activities and Postural Transitions (HAPT) Dataset we developed a RNN-based classifier for both sequence-to-sequence (S2S) and sequence-to-label (S2L) classification. Through hyperparameter optimization we optimized our model architecture to two LSTM layers followed by temporal attention layers for S2L-classification, then Dropout and two Dense layers. The models achieved 94.9% train, 78.81% validation and 94.0% test balanced accuracy for S2S- and 99.0% train, 84.24% validation and 88.43% test balanced accuracy for S2L-classification. As the HAPT dataset shows a great imbalance between basic activities and transition activties (91.38% to 8.62%) we proposed loss weighting and focal loss as ways to overcome this challenge. The ideal configuration was using categorical cross entropy loss with loss weighting, resulting in the overall best performance of 94.16% train, 79.35% validation and 80.94% test balanced accuracy for S2S- and 97.2% train, 90.3% validation and 92.3% test balanced accuracy for S2L-classification. Moreover we evaluated our model on a self-recorded dataset, the best results on this dataset was achieved by the S2S-model using focal loss with 30.56%.

1 Introduction (Baran)

Human Activity Recognition (HAR) using various wearable body sensors has become one of the most demanded studies for healthcare, wellbeing, and sports [2]. The sensor data used for this purpose can be categorized into two main classes: vision-based and non-vision-based. Vision-based sensors observe human activities with cameras mainly for security purposes, whereas non-vision-based sensors such as external and body-worn sensors focus on analyzing activity patterns. In this study, our focus is HAR, based on body-worn sensors.

Wearable sensors such as smartphones containing an accelerometer or gyroscope sensors are carried by people for a while to record different activities. The success of recognition, particularly in a practical case, depends on the position where it is placed on the body of the wearable device providing the sensor data [3]. Even though the conventional approaches to HAR schemes are generally based on machine learning methods [4], Recurrent Neural Networks (RNNs) have become the benchmark for these recognition tasks due to the ability to exploit the temporal dependencies from the sensor data. To do this, RNN based on Long Short Term Memory (LSTM) cells are implemented. The studies [5] and [4] used LSTM and Bidirectional LSTM (BiLSTM) with Principal Component Analysis and their accuracies are $89.8\% \pm 8.2\%$ and 97.32% respectively. All these RNN based studies prove that high accuracy is achievable for this recognition task. To make a comparison in different RNN models, [6] shows that performances of those models are similar but the LSTM's is better compared to other simplified versions of it such as Gated Recurrent Unit (GRU). Thus, we decided to use LSTM.

Our contributions to this task are as follows: First, we investigated the effect of increasing the LSTM layer in RNN. Second, we used temporal attention to observe how well it helps to recognize the activity. Thirdly we adjusted the loss function to account for class imbalance. Finally, we recorded our own data with a smartphone and tested our model on

it. In the following sections, we describe our approach to this by first, we give the model architecture, then we give the experimental setup, and finally, we share the results and make conclusions.

2 Model architecture (Baran)

We used different model architectures for the HAR task. Firstly, we started with a simple RNN model containing a single LSTM layer. Then, we used RNN containing 2 LSTM layers. In addition, we implemented temporal attention to those structures. The reason behind this decision is to ignore the irrelevant parts in the signal. This leads to removing the noisy part in the signals and paying more attention to the important parts. It is also known that temporal attention is very effective and improves performance if there are long-range dependencies. They are used with an LSTM to learn the temporal dependencies of time series data. If the length of the input sequences is long enough, then the LSTM itself might not perform as expected and attention helps to learn those long-range dependencies. Apart from temporal attention, there is also so-called sensor attention, however, we did not deal with it because different sensors play differently on different activity classes [7]. Therefore, it is necessary to learn for each activity the right combinations of sensor modalities. For this reason, we only implemented temporal attention.

There are two approaches with regards to the output of a HAR model: Sequence-to-label (S2L) classification outputs one label for a sequence of input data, while sequence-to-sequence classification classifies each time step of the input data separately. Sequence-to-label (S2S) classification comes with a certain amount of information loss, as its maximum time resolution is limited to the size of the sliding window. In case of an activity transition during the data sequence, the label only accounts for the most prominent activity. In contrast to NLP, where the dimension of the sequence-to-sequence model might change from input to output [8], input and output dimension are identical for HAR, as it provides an activity classification for each given timestep. For a visualization of the output for S2S- and S2L classification see Figure A.1 and A.2.

Figure 1 shows the optimal RNN architectures we found based on our search with two LSTM layers containing 256 units each with temporal attention in sequence-to-label and without attention in sequence-to-sequence classification. The temporal attention block contains a dense layer with one neuron, flatten layer, and an activation layer. Then, the repeat vector is used to add another dimension and lastly, the second and third dimensions are permuted. The attention block output is multiplied with the LSTM layer output and in this way, the effect of each state from the LSTM layer can be weighted at the output. Then, there is a dropout layer to make the model more robust followed by 2 dense layers with 256 dense units, and finally, the output layer containing 13 dense units.

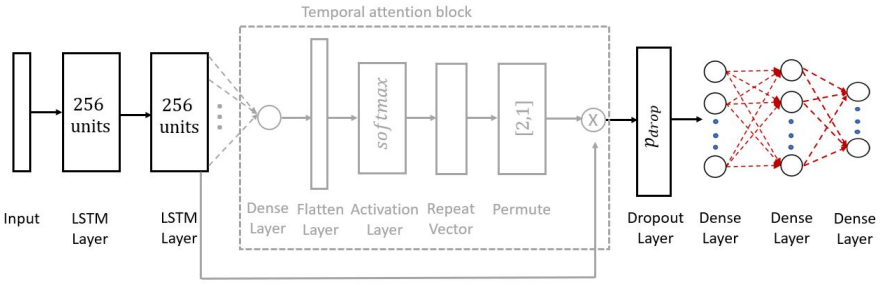

Figure 1: The RNN model with the LSTM and attention

3 Experimental Setup (Lydia)

3.1 Datasets

The model was trained and evaluated on the Human Activities and Postural Transitions (HAPT) Dataset [9]. To test the generalization of the model we additionally recorded our own sensor data as a custom evaluation dataset.

3.1.1 HAPT Dataset

The HAPT dataset contains triaxial accelerometer and gyroscope sensor data recorded with a smartphone (Samsung Galaxy S II). The smartphone was fixed around the waist and 30 volunteers between the ages of 19-48 performed the following basic activities: standing, sitting, lying (static postures), walking, walking downstairs, walking upstairs (dynamic activites). Moreover transitions between the basic static activities were recorded, including: stand-to-sit, sit-

to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. Labeling was done manually by evaluating video recordings of the participants. We used the raw version of the dataset and generated the following three splits:

Split:	Train	Test	Validation
Num of samples:	1.403.000	441.750	223.500
User:	1 ··· 21	22 ··· 27	28 ··· 30

Figure 2: HAPT dataset splits

When looking at the class distribution of the HAPT dataset in Table 1 there is a strong imbalance between the amount of samples for basic and transition activities. Looking at each sample (S2S classification) basic activities make up 91.38% of the dataset while transition activities are strongly underrepresented with only 8.62% of the dataset. When using only the most dominant label in a 250-window (S2L) the differences get even slightly larger with 91,83% for basic and 8.17% for transition actitivities.

Table 1: HAPT Dataset class distribution (for S2S classification)

Label	Train samples	Percentage	Label	Train samples	Percentage
Standing	170.200	15.35%	stand-to-sit	14.382	1.3%
Sitting	160.313	14.46%	sit-to-stand	11.136	1.0%
Laying	147.044	13.27%	sit-to-lie	17.204	1.55%
Walking	167.596	15.12%	lie-to-sit	15.930	1.44%
Walking downstairs	185.992	16.78%	stand-to-lie	21.746	1.96%
Walking upstairs	181.782	16.4%	lie-to-stand	15.124	1.36%
Basic Activities	1.012.927	91.38%	Dynamic Activities	95.522	8.62%

3.1.2 Self-recorded Dataset

In addition to the HAPT dataset we recorded our own sensor data. To recreate the conditions of the HAPT dataset setting, we used a smartphone (Samsung Galaxy A2) as the recording device and placed it on the waist (see figure 3).

Figure 3: Device Placement

Label	Samples	Percentage
Standing	9.750	7.99%
Sitting	16.500	13.52%
Laying	15.500	12.7%
Walking	30.250	24.8%
Walking downstairs	30.750	25.2%
Walking upstairs	19.250	15.89
Total	122.000	100%

Table 2: Self-recorded dataset distribution

Recordings were conducted by a single user (age: 24) with the Sensor Data Collector App. It enables labeling the data while recording, thus a manual labeling of the data was unnecessary. To avoid data of bad quality, that might have required manual relabeling or deletion, only the six basic activities (standing, sitting, lying, walking, walking downstairs, walking upstairs) are contained in the dataset (see Table 2).

3.2 Preprocessing

Unlabeled data in the HAPT datasets was mapped to '0'-labels and ignored for training and metrics, but remained in the dataset. Both datasets were Z-score-normalized for each user and sensor:

$$x_i' = \frac{x_i - \mu_{x,i}}{std(x_i)}, \quad \mu_{x,i} = \frac{1}{N}\sum_{n=1}^{N} x_i(n), \quad std(x_i) = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(x_i(n) - \mu_{x,i})^2}$$

x_i': z-score-normalized sensor data, $x \epsilon \{acc_x, acc_y, acc_z, gyro_x, gyro_y, gyro_z\}$: sensor data, $i \epsilon \{1, 2, ...30\}$: user

This way user-dependent biases or variations, through slightly different placement of the phone, can be removed. In the next step a sliding window operation was applied. The data was split into windows of 250 samples with an overlap of 125 samples for each split.

4 Results (Lydia)

As the HAPT dataset shows a major imbalance between the six basic and six dynamic activites (see Table 1) we evaluated the models performance using balanced accuracy:

$$Balanced\ Accuracy = \frac{1}{C}\sum_{c=1}^{C}\frac{N_{\hat{c}=c}}{N_c}$$

num of classes: C, correctly classified samples: $N_{\hat{c}=c}$, amount of samples of class c: N_c

To show the difference of results for balanced accuracy to unbalanced accuracy, we have added the unbalanced accuracy values for the results in hyperparameter optimization (see Table 4).

4.1 Hyperparameter optimization (Baran and Lydia)

Starting out with a single-LSTM-layer model we could improve our model by adjusting its architecture based on hyperparameter optimization and adding temporal attention to the model. We implemented hyperparameter optimization using HyperOptSearch search algorithm [10]. The optimum combination for sequence-to-sequence (S2S) and sequence-to-label (S2L) classification with and without attention are given in Table 3.

Table 3: Hyperparameters with searched and chosen values

Hyperparameter	Searched Values	S2S	S2L	S2L with attention
Number of LSTM units	64, 128, 256	256	256	256
Number of LSTM layers	1, 2	2	2	2
Number of dense units	64, 128, 256	256	256	256
Dropout rate	0.0-0.5	0.3515	0.01855	0.1018
Batch size	16, 32 , 64	32	32	16

The results for hyperparameter optimization can be found in Table 4. After adding temporal attention to our S2L-model we achieved an increase of $\sim 3\%$ in train and validation accuracy. Regarding test and validation accuracies the S2L- performs significantly better than the S2S-models, however this is most likely related to the smoothing done in S2L-preprocessing, by only using the most prominent label in the 250 sample window. The S2S-model strives to classify even short transitions and therefore takes the label 7-12 more into account (as these tend to be shorter), while the S2L-model is less sensitive to such short period activities.

Table 4: Results before and after hyperparameter optimization

S2S	balanced train accuracy	train accuracy	balanced val accuracy	val accuracy	balanced test accuracy	test accuracy	custom dataset
Before	92.20%	93.548%	76.04%	89.84%	81.27%	94.20%	21.28%
After	94.92%	95.49%	78.81%	92.13%	79.74%	93.99%	26.74%
S2L							
Before	93.81%	90.15%	83.90%	90.35%	87.75%	94.70%	27.99%
After	96.00%	87.16%	84.37%	91.21%	87.67%	93.19%	29.60
+ attention	98.96%	96.33%	84.24%	93.23%	88.43%	94.12%	24.61%

Overall the model is overfitting, as train accuracy compared to validation and test accuracy differs by over 10% for almost all configurations. Regarding the performance on the self-recorded custom dataset all models perform equally bad. When looking at the confusion matrix (see Figure B.5 and Figure B.6) one can see that the model mainly confuses walking with walking upstairs and downstairs and sitting with standing (and reverse). The confusion between sitting and standing was already visible in the HAPT dataset, however the effect in the self-recorded dataset is even stronger.

4.2 Countering Class Imbalance (Lydia)

Looking at the test confusion matrix after hyperparameter optimization (exemplary for S2S: FigureB.5) it is noticeable that the model performs well for the predominant classes 1-6, while it performs poorly on the less represented classes 7-12. To counter the effect of the class imbalance in training and to further improve our accuracy, we propose loss weighting [11] and focal loss [12], instead of categorical cross entropy loss (CCE Loss), to provide a stronger representation for the minor classes during optimization [13].

4.2.1 Focal loss

Focal loss weights the loss depending on the confidence (predicted probability p_i for the true class) of the object classifier [12]:

$$Focal\ Loss(\hat{p}_y, y) = -(1-\hat{p}_y)^{\gamma} log(\hat{p}_y), \quad \hat{p}_y = softmax(\mathbf{z}) = \frac{exp(z_y)}{\sum_{j=1}^{C} exp(z_j)}$$

$y \epsilon \{0, ..., C-1\}$: class label, $\hat{p}_i \epsilon [0,1]$: predicted probability for true class, $\mathbf{z}$: output of last layer, γ: weighting factor

This way the loss of well learned samples with high confidence levels are down-weighted, while minority class samples with low confidence levels gain larger weights and thus importance in the optimization process.

4.2.2 Loss weighting

We used loss weighting based on the effective number of samples in each class [11]. The weighting factor for each class is denoted by:

$$w_i = \frac{1-\beta}{1-\beta^{N_i}}, \quad Weighted\ Loss = \frac{1-\beta}{1-\beta^{N_i}} L(\hat{p}_i, y)$$

$i \epsilon \{0, ..., C-1\}$: class label, N_i: number of samples in class i, $L(\hat{p}_i, y)$: Loss function

For β we chose $0,9999$ after trying out multiple values ([0,999, 0,9999, 0,99999]) hyperparameter optimization. As the weighting is loss-independent, it can be combined with both CCE and focal loss (used in sparse categorical form).

4.2.3 Results for different loss configurations

To check which loss configuration is most favorable, we considered the following options: CCE loss or focal loss (with $\gamma = 0.5$) and with or without weighting. For the runs with focal loss we found that the loss was oscillating more strongly and the runs did not always converge. Reason for this might be, that forcing the model to focus on the minor classes from the beginning, it was not able to learn the patterns for the majority classes. Thus we adapted the code to allow for a switch from CCE loss to focal loss after a certain epoch number. We found that switching after 50 epochs lead to the best results. The results for focal loss in Table 5 are all from using this switching concept.

Table 5: Results for different loss configurations

S2S	balanced train accuracy	balanced val accuracy	balanced test accuracy	custom dataset
SCCE	94.92%	78.81%	79.74%	26.74%
SCCE+ weighting	94.16%	79.35%	80.94%	22.82%
Focal loss	97.17%	76.74%	78.13%	30.56%
Focal loss + weighting	94.99%	77.35%	80.51%	27.05%
S2L with attention				
SCCE	98.96%	84.24%	88.43%	24.61%
SCCE+ weighting	97.20 %	90.29%	92.30%	21.05%
Focal loss	97.49%	89.50%	91.43%	21.99%
Focal loss + weighting	98.63%	88.97%	89.47%	22.86%

Overvall using CCE loss with weighting showed the best results regarding the generalization of the S2L- and S2S-model. Focal loss (with and without weighting) showed a positive impact for S2L classification as well, while for S2S classification it did not majorly impact the models performance. But it can be noted that using focal loss lead to better results for train accuracy for the S2S model. However at the same time overfitting became even more severe (for comparison see Figure B.7 and B.8).

5 Conclusion (Lydia)

In this work we have identified class imbalance, due to longer duration of certain activities, as well as overfitting as a main obstacle to HAR based on the HAPT dataset. Through hyperparameter optimization and temporal attention we could slightly improve our models performance . To overcome the effect of class imbalance we proposed using weighted, focal loss which lead to a overall improvement for S2L- and a slight improvement in train loss for S2S-classification. However overfitting remains as a problem and would be the next challenge to tackle, f.e. by applying recurrent dropout for the LSTM layers.

References

[1] Jubil T Sunny, Sonia Mary George, Jubilant J Kizhakkethottam, Jubil T Sunny, Sonia Mary George, Jubil T Kizhakkethottam, Jubilant JSunny, Sonia Mary George, Jubilant J Kizhakkethottam, Jubil T Sunny, Sonia Mary George, and Jubilant J Kizhakkethottam. Applications and challenges of human activity recognition using sensors in a smart environment. *IJIRST Int. J. Innov. Res. Sci. Technol*, 2:50–57, 2015.

[2] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga. Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *23th International Conference on Architecture of Computing Systems 2010*, pages 1–10, 2010.

[3] L. Atallah, B. Lo, R. King, and G. Yang. Sensor positioning for activity recognition using wearable accelerometers. *IEEE Transactions on Biomedical Circuits and Systems*, 5(4):320–329, 2011.

[4] Masaya Inoue, Sozo Inoue, and Takeshi Nishida. Deep recurrent neural network for mobile human activity recognition with high throughput, 2016.

[5] Deepika Singh, Erinc Merdivan, Ismini Psychoula, Johannes Kropf, Sten Hanke, Matthieu Geist, and Andreas Holzinger. Human activity recognition using recurrent neural networks. *Machine Learning and Knowledge Extraction*, page 267–274, 2017.

[6] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, Oct 2017.

[7] M. Zeng, X. Wang, L. T. Nguyen, P. Wu, O. J. Mengshoel, and J. Zhang. Adaptive activity recognition with dynamic heterogeneous sensor fusion. In *6th International Conference on Mobile Computing, Applications and Services*, pages 189–196, 2014.

[8] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[9] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Samà, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomputing*, 171:754–767, jan 2016.

[10] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[11] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. pages 9260–9269, Long Beach, CA, USA, 2019. IEEE.

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. August 2017.

[13] Trong Huy Phan and Kazuma Yamamoto. Resolving class imbalance in object detection with weighted cross entropy losses. June 2020.

A Model prediction visualization

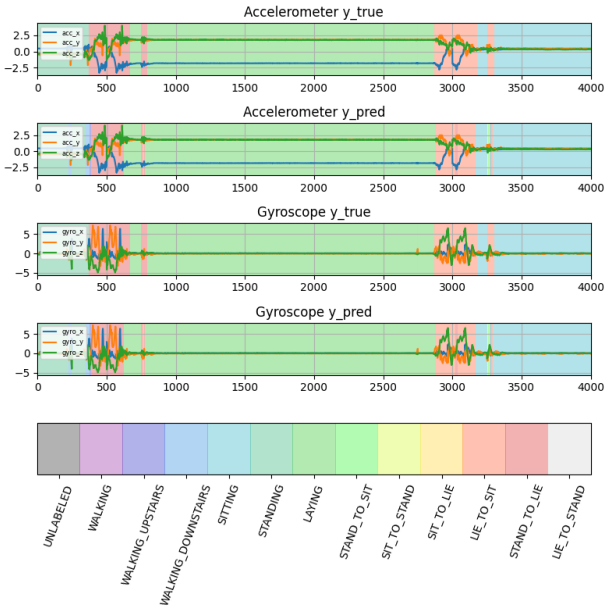

Figure A.1: Prediction vs true labels for S2S

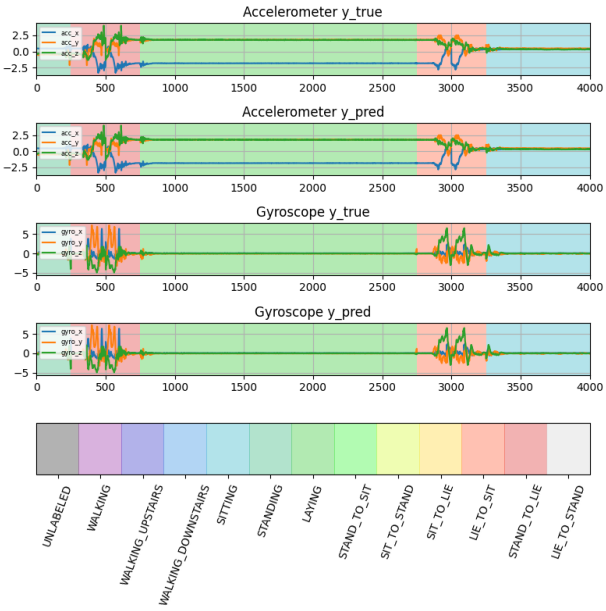

Figure A.2: Prediction vs true labels for S2L

B Confusion Matrices

B.1 Before hyperparameter optimization

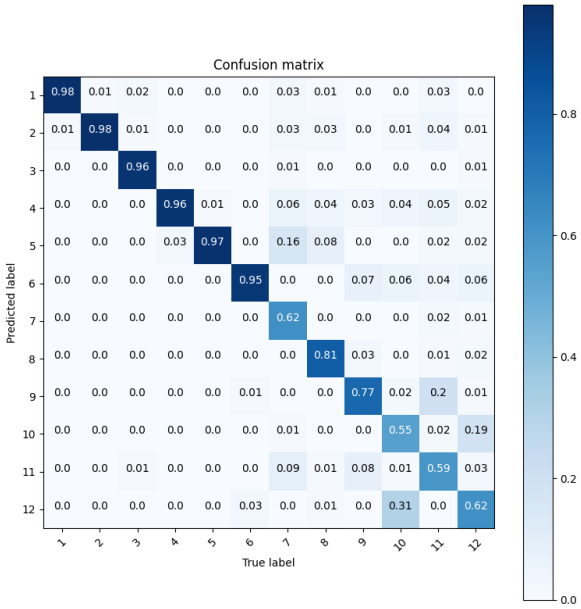

Figure B.3: S2S on HAPT test dataset

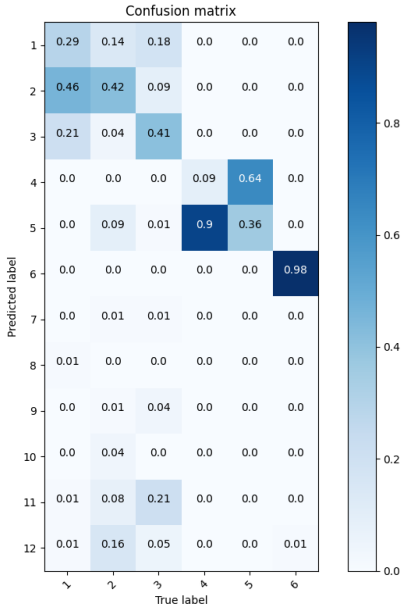

Figure B.4: S2S on custom dataset

B.2 After hyperparameter optimization

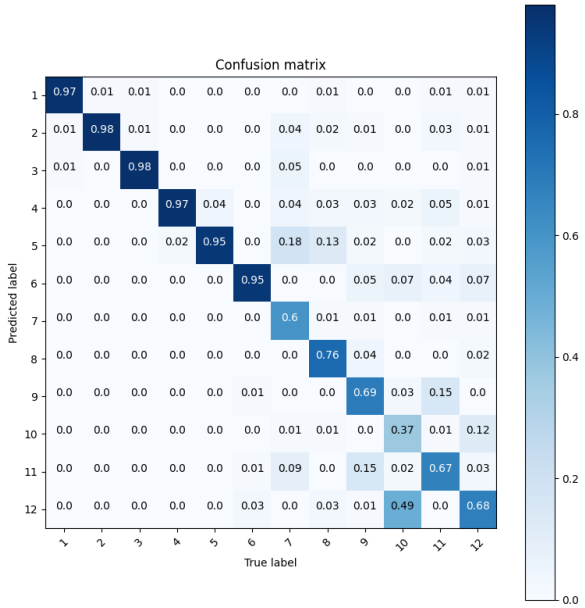

Figure B.5: S2S on HAPT test dataset

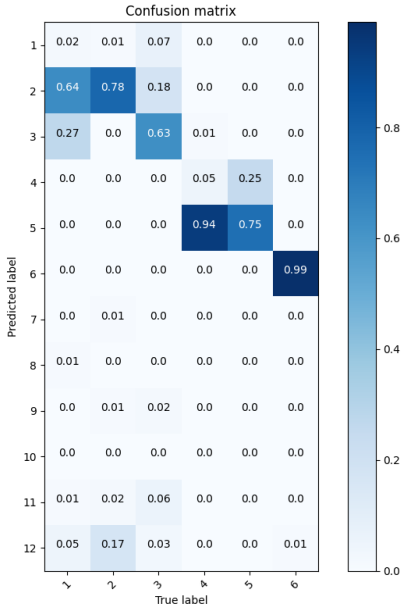

Figure B.6: S2S on custom dataset

B.3 Applying focal loss

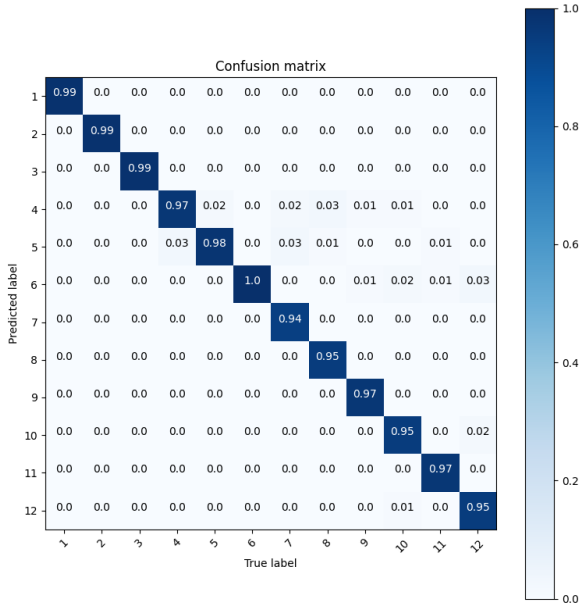

Figure B.7: S2S on HAPT train dataset

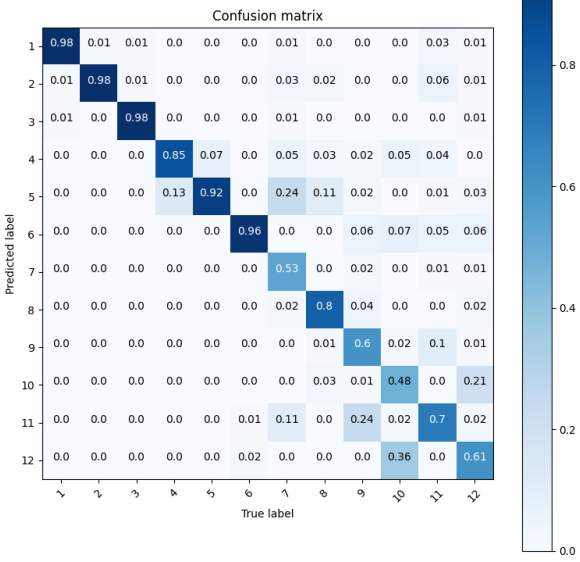

Figure B.8: S2S on HAPT test dataset