GATED RECURRENT UNIT NETWORK FOR HUMAN ACTIVITY RECOGNITION

A PREPRINT

ZHONG Liangyu
Electromobility
University Stuttgart
st160455@stud.uni-stuttgart.de

CAO Shijia
Electromobility
University Stuttgart
st160464@stud.uni-stuttgart.de

February 11, 2020

ABSTRACT

Gated Recurrent Unit (GRU) networks are effective to capture long-term dependencies of the time-series signal. Based on Human Activities and Postural Transitions(HAPT) dataset, we propose a model referring to GRU for the sequence-to-sequence classification task. In addition, we apply Bayesian Optimization to tune hyperparameters. Finally, the proposed model achieves accuracy of of 92.4% on test dataset of HAPT .

1 Introduction (by CAO Shijia)

Human Activity Recognition (HAR) based on sensory data is an active research field of classifying data recorded by sensors into known well-defined movements. A challenging problem is dealing with the large number of time-series data and finding a clear way to match data to corresponding movements. We propose a generic framework to predict sequences of human activities from sequences of inertial sensor data. Our approach provides good performance on HAPT.

2 Model (by ZHONG Liangyu)

For our time series signal, we need a model which could dual with time series signal. Typically, vanilla Recurrent Neural Network(RNN) and its variants are used in this scenario.
A typical vanilla RNN could have not bad performance for a normal application. However, when there is a large lag between input sequence signal and the corresponding teacher signal, RNN is reluctant to capture such kind of relation. The reason behind is that, using conventional Back-Propagation Through Time(BPTT), the error signal in RNN's training is easy to explode or vanish[1]. Remedy could be using other models like Long Short-term Memory(LSTM) network and Gated Recurrent Unit(GRU) network.

2.1 LSTM and GRU

LSTM and GRU are capable of capturing the long-term dependencies of time series signal. Because they could selectively forget and remember some information regarding the new input. This mechanism is called gating. There are three different gates in LSTM which helps to manipulate the information states.

- Forget gate layer: decides which part of information is unnecessary in cell state and abandon it.
- Input gate layer: control what kind of new information should be added in cell state.
- Output gate: filter and output the cell state.

GRU is a variant of LSTM. There are just two gates in GRU, update gate and reset gate. That means, GRU will pass all content of its cell state to the next neuron. In general, GRU and LSTM have similar performance. But since GRU has fewer parameters than LSTM, which will make it converge faster and make it less sensitive to overfitting[2].

Table 1: Proposed model's architecture

GRU(unit=32)
GRU(unit=16)
Dense(unit=16), ReLU
Dense(unit=12), Softmax

2.2 Proposed Model

Since the dataset we use is a small dataset, we would like to use GRU layers in our model to reduce overfitting and have fast convergence. Our proposed model is shown as in Table 1.

3 Experiment

3.1 Dataset (by CAO Shijia)

HAPT dataset contains data from tri-axial accelerometer and gyroscope of a smartphone, both are captured at a frequency of 50 Hz. The dataset consists of six basic activities (static and dynamic) and six postural transitions between the static activities. The static activities include standing, sitting and lying. The dynamic activities include walking, walking downstairs and walking upstairs. stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand are the classes for postural transitions between the static activities. We split the dataset into train/validation/test datasets in proportion 0.7/0.1/0.2.

3.2 Input pipeline (by CAO Shijia)

Input Normalization We use Z-Score normalization to transform input data to zero mean and unit variance. Normalized 6-channels time-series data could give us better performance because it could eliminate user dependent biases and offset due to device placement errors.

TFRecords To load data efficiently, we serialize the 6-channel input data and corresponding labels and store them in in TFRecord format.

Sliding Window In order to have fixed-length input and label sequences, we apply sliding window technique. In training we set window length 250 and window shift 125, that means 50% overlap, which serves as a kind of data augmentation. In validation and test set, the window length is set to 250 without overlap.

3.3 Implementation Details (by ZHONG Liangyu)

Optimizer We select Adam[3] as our optimizer. In most scenarios, Adam could have a faster convergence and even better final performance than its competitor AdaGrad and RMSprop.

Metrics Confusion matrix and accuracy are selected as our metrics. Since our dataset is highly imbalanced, confusion matrix could show us the exact accuracy for each class. The unlabeled data will not be accounted in any of our metrics.

Loss We select cross-entropy loss as our loss function, which is defined as follows:

$$CE = -\sum_{i}^{C} t_i \log(c_i) \tag{1}$$

where t_i is the label and c_i is the logit output of our model of specific class i.

Training Procedure Since there is some unlabeled data in our dataset, we manually annotate the labels of these data as all zero vectors. And we use ont-hot coding of labeled data in our training procedure. As shown in equation 1, the cross-entropy loss of zero vector will be zero. So unlabeled data won't have generate any loss.
To improve the performance of our network, we use drop out in GRU layers and fully-connected layers. And we set an unified dropout rate as a hyperparameter. We will try our network for 5 epochs and test our network after 5 epochs of training.

3.4 Hyperparameter tuning (by ZHONG Liangyu)

Manually selecting hyperparameters strongly depends on human's experience. Grid search is time-consuming and not feasible. After consideration, we take Bayesian Optimization as our method of hyperparameter tuning. We have the following hyperparameters and their range:

- Unit of GRU layers from 16 to 64
- Unit of the first fully-connected layer from 16 to 64
- Drop out rate of GRU layers and fully-connected layers from 0.0 to 0.4
- Learning rate of optimizer from 10_{-2} to 10_{-5}

For convenience, the first GRU layer always have two times so many units as the second GRU layer, and first dense layer have same number of neurons as the second GRU layer. And we select 5 exploits and 5 explores (in total 10 iterations) as the hyperparameter of our Bayesian Optimization.

4 Result (by CAO Shijia)

The results on test dataset are shown in table 2. By means of Bayesian Optimization, our best test accuracy is 92.4%. The corresponding confusion matrix is shown in Figure1. We randomly choose a segment of sequence in test dataset to visualize the prediction in Figure 2.

Table 2: Results on test dataset

Iteration	Dropout	Learning rate	Unit	Test accuracy
GRU-1	0.1668	0.002804	30	91.10%
GRU-2	0.0587	0.009078	32	54.5%
GRU-3	0.1587	0.004617	48	69.26%
GRU-4	0.08178	0.001228	48	**92.40**%
GRU-5	0.1669	0.004419	25	88.64%
GRU-6	0.1669	1e-05	64	80.79%
GRU-7	0.03255	1e-05	16	55.6%
GRU-8	0.3989	1e-05	41	71.3%
GRU-9	0.000896	1e-05	58	78.82%
GRU-10	0.4	1e-05	20	51.98%
LSTM-1	0.08178	0.001228	48	92.07%

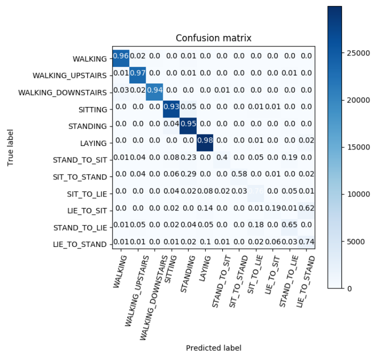

Figure 1: Confusion Matrix of test dataset

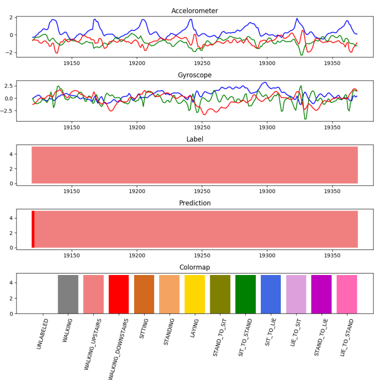

Figure 2: Visualization of the result

The result shows that the our network is effective to predict the static activities but not to postural transitions between the static activities. In future we could work on that using e.g. upsampling of postural transitions. With same hyperparameters of our GRU network, we try replacing GRU layer in our network with LSTM layer. The result in table 2 shows that LSTM network does have similar performance as GRU network. But we did observe slower convergence during the training, which will be more serious when we train network on a large data set.

References

[1] Hochreiter, Sepp and Schmidhuber, Jürgen. Long Short-term Memory. In *Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.*, 1997

[2] Chung, Junyoung and Gulcehre, Caglar and Cho, KyungHyun and Bengio, Y.. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *http://arxiv.org/abs/1412.3555*, 2014

[3] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization *http://arxiv.org/abs/1412.6980*, 2014