Stress Classification from Biosignals using Neural Networks

Justyna Czestochowska, Lara Orlandic, and Maja Stamenkovic *

November 30, 2020

***Abstract*— Noninvasive psychological stress monitoring can help individuals cope with daily stress. Algorithms that detect stress must have minimal computational complexity in order to perform real-time, low-power monitoring on wearable devices. This work proposes two Deep Learning architectures, CNN and GRU, to perform stress classification based on four time-series biosignal inputs. This approach enables a faster prediction time compared to the state-of-the-art classification methods involving feature extraction. Various data preprocessing and augmentation strategies are assessed in terms of added value to the network's classification accuracy. The Deep Learning models performed comparably to the state-of-the-art methods, with the GRU having only a 1.42% lower accuracy than the Random Forest approach. However, all models exhibited significant overfitting and inter-subject variability, which reflects the subject-specific nature of biosignals.**

I. Introduction

Chronic psychological stress can contribute to a variety of health problems, such as depression, suppressed autoimmune function, and coronary artery disease [1]. Developing a reliable stress detection mechanism can enable people to monitor their daily stress levels and develop coping mechanisms to prevent stress-related illnesses [2]. Ideally, such a stress monitoring device should be small, portable, and noninvasive to enable real-time, continuous classification.

Previous studies have used noninvasive physiological monitoring techniques to detect stress [2], [3]. These studies typically extract upwards of 22 features from a set of filtered biosignals, and then use common machine learning algorithms to classify between the stress and control states [3], [4]. However, calculating these features greatly increases the computational burden of the stress detection algorithm, making it cumbersome for implementation on ultra-low power wearable devices. It would therefore be advantageous to develop an algorithm that performs classification using only the biosignals themselves.

The aim of our study is to design a neural network to classify between stress and control states across subjects by analyzing four time-series biosignals. To generate a gold-standard classification accuracy for our data set, we apply traditional machine learning (ML) methods to several features extracted from the signals. We then compare these results to those of our newly-proposed Deep Learning approach, in which we train Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to classify between stress and control states using only the biosignals as inputs. This approach circumvents energy-consuming feature extraction procedures and therefore reduces the prediction time of stress detection algorithms.

*This work was performed in collaboration with the Embedded Systems Laboratory of EPFL. The data was provided by the Laboratory of Behavioral Genetics of EPFL.

II. Data Manipulation

A. Database Description

We use a database of 116 subjects that performed a two-stage stress experiment. First, half of the subjects performed a stress task consisting of mental arithmetic with negative feedback, while the other half performed a control task. In the second stage, the two groups switched tasks. Each stage lasted for 10 minutes.

Throughout the experiment, four biosignals were unobtrusively measured from each subject: electrocardiogram (ECG), photoplethysmogram (PPG), respiration (RSP), and electrodermal activity (EDA). ECG enables heart rate and heart rate variability monitoring. RSP monitors the subject's breathing rate, which typically increases in response to stress. PPG is an optical blood volume pulse measure that also contains stress-related biomarkers. Finally, EDA measures sweat gland activity, which increases due to stress. Fig. 1 provides a visualization of each signal under typical control and stress conditions. Each signal was collected with a sampling rate of 1 kHz. The data was labeled as "stress" or "control" based on the task that the subject performed during its acquisition.

B. Data augmentation

Deep neural networks require a large training dataset to achieve a reasonable generalization capability. Since the study only contains 116 subjects, we tested different data augmentation methods to provide sufficient inputs to the neural networks.

First, each subject's biosignals are divided into a series of fixed-length time segments with a 50% overlap, which are treated as independent data samples. The overlap factor was chosen based on optimizations

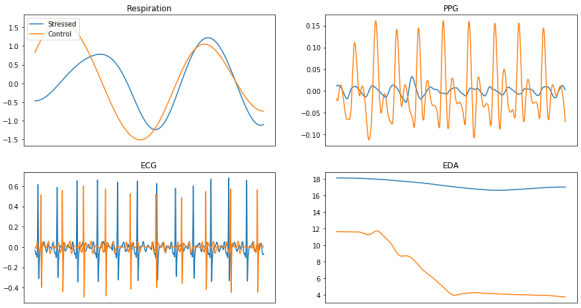

Fig. 1. 10s windows of a subject's biosignals during stress and control tasks. Typically during stress, respiration rate, heart rate, and EDA all increase.

performed in [3]. We then chose the optimal window length to maximize the classification accuracy. Finally, we implement the amplitude scaling augmentation method proposed in [5].

C. Data preprocessing

Various signal processing tools can be applied to the raw biosignals to potentially increase the classification accuracy of the neural networks. First, the signals can be downsampled from 1 kHz to 125 Hz, as previous studies have shown that ECG sampling frequencies of 120 Hz and higher provide adequate signal information [6]. This decreases the width of the networks and thereby reduces their computational burden and potential for overfitting. Secondly, the biosignals can be normalized using min-max scaling to maintain similar signal amplitudes between samples. Finally, the biosignals can be filtered to reduce noise. The filters are implemented using the BioSPPy toolbox [7] and described in Table I.

TABLE I
FILTER PARAMETERS

Signal	Filter Type	Cutoff Frequencies (Hz)
ECG	FIR bandpass	3-45
EDA	Butterworth lowpass	5
RSP	Butterworth bandpass	0.1-0.35
PPG	Butterworth bandpass	1-8

D. Testing and Validation

When working with physiological data, the training, validation, and testing data sets must consist of samples from strictly different subjects. This is because biosignals vary greatly between subjects, and a robust stress detection algorithm must classify correctly for any subject. Therefore, we split the feature data such that the segments belonging to 33% of subjects are designated as testing data. Furthermore, we implement a modified 5-fold cross-validation in which the remaining training subjects are randomly divided into groups of 5, and the corresponding samples are used for cross-validation.

For the raw biosignal data, the segments belonging to 15% of the subjects are designated as testing data. Then, three different training/validation configurations are generated such that 15% of subjects are randomly selected to be in the validation group, but the training/validation subjects are shuffled in each configuration. This ensures that we do not bias our model selection based on highly subject-specific signals in one particular validation group.

III. CLASSIFICATION USING RNN AND CNN

Recurrent neural networks (RNNs) are extensively used to model sequential data and can be applied to biosignal processing [8]. The two most popular RNN architectures are Long Short-Term Memory (LSTM) [9] cells and Gated Recurrent Units (GRU) [10]. Our approach uses two GRU layers for feature extraction from raw biosignals followed by a fully-connected layer and a sigmoid activation function. The choice of GRU over LSTM was imposed by a larger number of parameters in the latter which leads to a longer training and prediction time. The function of each GRU layer is described by the following equations

$$\begin{aligned} r_t &= \sigma\left(W_{ir}x_t + b_{ir} + W_{hr}h_{(t-1)} + b_{hr}\right) \\ z_t &= \sigma\left(W_{iz}x_t + b_{iz} + W_{hz}h_{(t-1)} + b_{hz}\right) \\ n_t &= \tanh\left(W_{in}x_t + b_{in} + r_t * \left(W_{hn}h_{(t-1)} + b_{hn}\right)\right) \\ h_t &= (1 - z_t) * n_t + z_t * h_{(t-1)} \end{aligned} \tag{1}$$

where h_t is the hidden state at time t, x_t is the input at time t, $h_{(t-1)}$ is the hidden state of the layer at time t-1 or the initial hidden state, and r_t, z_t, n_t are the reset, update, and new gates, respectively. σ is the sigmoid function, and $*$ is the Hadamard product [11]. To avoid overfitting, we apply a dropout layer after first GRU cell.

The full architecture of the proposed GRU model is presented in Fig. 2. The network is trained using the Adam optimization algorithm [12]. We used a batch size of 32, learning rate of 0.001, hidden size of 64, and dropout probability of 0.2.

The second deep learning algorithm we implement is a Convolutional Neural Network (CNN). We utilize

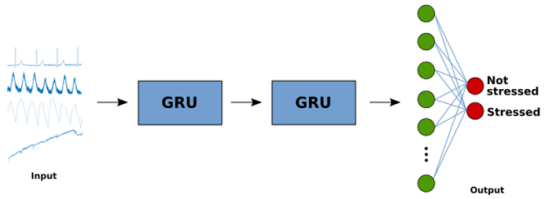

Fig. 2. Architecture of the proposed RNN

1-dimensional (1-D) convolutions to perform time-series analysis. Furthermore, due to the simple and compact configuration of 1-D CNNs, which perform only scalar multiplications and additions, a real-time and low-cost hardware implementation is feasible [13].

The proposed CNN architecture is displayed in Fig. 3. We use a simple CNN with two convolutional layers, which are used to extract features from the raw siginals, and one fully-connected layer, which performs classification. In the first convolutional layer, a filter of dimension 10x1 is applied to the input with a stride of 2, while the next convolutional layer has a kernel size of 20 and also a stride of 2. Three kernel size pairs were tested and these dimensions produced the highest validation accuracy. Both convolutional layers are followed by batch normalization, rectified linear units (ReLU), and a max pooling layer with kernel sizes 2 and 3, respectively, both with a stride of 2. To avoid overfitting, we employ dropout with a ratio of 0.5 before the fully-connected layer.

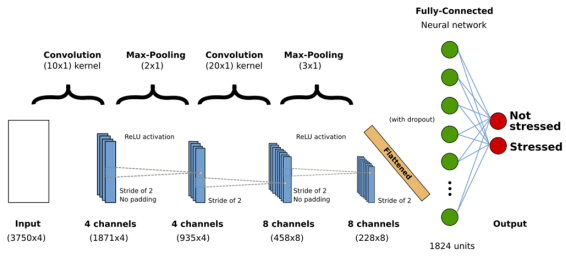

Fig. 3. Architecture of the proposed CNN

IV. Classification using features

The state-of-the-art approach to classifying biosignals is to extract features and then apply ML algorithms. The four biosignals are segmented into 1 min. windows and 66 features are extracted from each window. Samples containing NaN values for any feature are removed. Moreover, outliers are removed from the training set by eliminating samples that have more than one feature whose value is three standard deviations away from the training mean. Then, we train 7 different classification models whose hyperparameters are tuned based on the modified 5-fold cross-validation described in Section II-D. Furthermore, we developed a simple Feed-Forward Neural Network with one hidden layer and 7 hidden layer nodes as an additional classifier.

The results of the state-of-the-art classification methods are presented in Fig. 4. The Random Forest (RF) algorithm exhibited the highest testing accuracy: 60.7%. Regardless of the algorithm, the results show high variability in accuracies for samples belonging to different subjects. The accuracy for each test subject ranged from 25-84.2% with a standard deviation of 11.9, which is reflective of the pronounced differences in individuals' physiological signals.

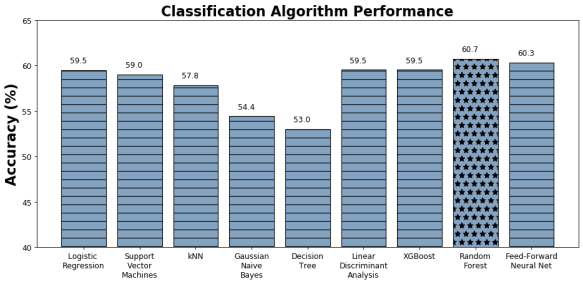

Fig. 4. Classification accuracies of different algorithms on the data set of extracted features. RF produces the highest accuracy.

V. Results of Deep Learning Methods

A. Impact of Preprocessing

The first preprocessing step we implemented was downsampling the data by a factor of 8. When training the CNN on unfiltered, unnormalized data with 30s windows for 300 epochs, downsampling increased its validation accuracy from 52.0% to 58.41%. This optimization also decreased the training time of the algorithm and reduced the amount of overfitting.

Next, we studied the effects of additional preprocessing on the classification accuracy of the CNN, which are reported in Table II. Surprisingly, the highest accuracy was achieved without any data manipulation. The validation accuracy of the CNN dropped by 2.4% when the data was normalized and by 5% when the data was filtered. This could mean that the filters described in Section II-C removed valuable information that the CNN requires to perform classification.

TABLE II

SIGNAL PREPREPROCESSING PIPELINE COMPARISON

Pipeline	Validation Accuracy	Train Accuracy
No preprocessing	**58.41%**	68.52%
Normalized	56.01%	69.54%
Filtered and normalized	53.37%	66.36%

B. Impact of Augmentation

The first data augmentation technique we tested was varying the window length of the data samples. The shorter the window length, the more the signals are divided into independent samples. This leads to more inputs for the neural networks and potentially less overfitting. Contrarily, having segments that are too short destroys important physiological information. For example, ECG segments shorter than one heartbeat cannot provide information about heart rate.

The results of the window length variation are displayed in III. Training was run for 300 epochs of the CNN. The 30s window produced the highest validation accuracy.

TABLE III

WINDOW LENGTH VARIATION RESULTS

Window Length (s)	Validation Accuracy	Train Accuracy
30	**58.41%**	68.52%
20	54.85%	70.46%
10	56.05%	74.23%

An additional augmentation method of amplitude scaling was tested. Each sample from the train data set, using 30s windows and no preprocessing, was multiplied by a random constant $\alpha \in [0.9, 1.1]$. This method decreased the CNN validation accuracy to 55.18% and increased the training accuracy to 71.82%, which means that it increased the amount of overfitting. The RNN exhibited similar results: 55.18% validation and 80.03% training accuracies. Therefore, this method was not used in the final model.

C. CNN vs RNN

The final testing accuracies were **58.07%** for the CNN and **59.28%** for the RNN. The RNN's accuracy was only 1.42% lower than that of the state-of-the-art methods. The Deep Learning methods also exhibited strong variability in classification accuracy for samples belonging to different subjects. For example, in the RNN, the classification accuracies per subject ranged from 14.1-97.1% with a standard deviation of 21.4%.

Fig. 5 displays the accuracy and loss of the training and validation sets during RNN training. One can clearly see that in both cases, the validation loss curve almost instantly diverges from the training one, which implies excessive overfitting. Even though regularization techniques such as dropout and batch normalization were applied, they did not manage to improve the model's generalization. The accuracy plot shows that the validation accuracy does not increase but rather oscillates around 55%. This could be due to the fact that stress biomarkers differ strongly across subjects.

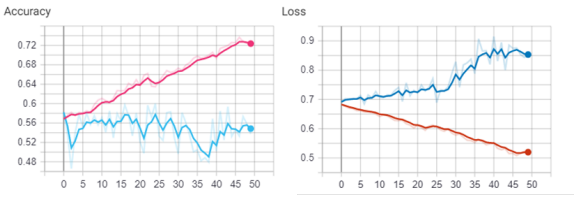

Fig. 5. RNN accuracy and loss values during training for train (red) and validation (blue) data sets

VI. DISCUSSION AND CONCLUSIONS

The goal of this work was to classify between stress and control states based on time-series biosignal data using deep learning models and compare their performance to that of traditional ML methods. We developed two neural networks, a CNN and an RNN, and tested different preprocessing and data augmentation approaches to determine their efficacy. In terms of signal preprocessing, neither filtering nor normalization increased the network's classification accuracy. Moreover, amplitude scaling augmentation decreased the accuracy and led to further overfitting. Finally, the window length of 30s and downsampling to 125 Hz produced the highest validation accuracy.

The final testing accuracy of the RNN model was only 1.42% below that of the state-of-the-art methods. However, both the CNN, RNN, and classical methods suffered significant overfitting and high inter-subject variability. These phenomena may reflect the subject-specific nature of biosignals, or the limited size of the data set. Furthermore, it is possible that the data set contained mislabeled data, as subjects were not asked to rank their stress levels, but it was assumed that they were stressed throughout the stress task. In future work, we propose using larger available biosignal datasets, and perhaps exploiting tools like transfer learning to aid in classification.

REFERENCES

[1] S. Cohen, D. Janicki-Deverts, and G. E. Miller, "Psychological stress and disease," pp. 1685–1687, 10 2007.

[2] A. Arza, J. M. Garzón-Rey, J. Lázaro, E. Gil, R. Lopez-Anton, C. de la Camara, P. Laguna, R. Bailon, and J. Aguiló, "Measuring acute stress response through physiological signals: towards a quantitative assessment of stress," *Medical and Biological Engineering and Computing*, vol. 57, no. 1, pp. 271–287, 1 2019.

[3] N. Momeni, F. Dell'agnola, A. Arza, and D. Atienza, "Real-Time Cognitive Workload Monitoring Based on Machine Learning Using Physiological Signals in Rescue Missions*," Tech. Rep.

[4] J. A. Healey and R. W. Picard, "Detecting stress during real-world driving tasks using physiological sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 2, pp. 156–166, 6 2005.

[5] A. Sakai, Y. Minoda, and K. Morikawa, "Data augmentation methods for machine-learning-based classification of bio-signals," in *BMEiCON 2017 - 10th Biomedical Engineering International Conference*, vol. 2017-January. Institute of Electrical and Electronics Engineers Inc., 12 2017, pp. 1–4.

[6] E. Ajdaraga and M. Gusev, "Analysis of sampling frequency and resolution in ECG signals," in *2017 25th Telecommunications Forum, TELFOR 2017 - Proceedings*, vol. 2017-January. Institute of Electrical and Electronics Engineers Inc., 1 2018, pp. 1–4.

[7] "API Reference — BioSPPy 0.6.1 documentation." [Online]. Available: https://biosppy.readthedocs.io/en/stable/biosppy.html

[8] N. Ganapathy, R. Swaminathan, and T. Deserno, "Deep Learning on 1-D Biosignals: a Taxonomy-based Survey," *Yearbook of Medical Informatics*, vol. 27, no. 01, pp. 098–109, 8 2018. [Online]. Available: http://www.thieme-connect.de/DOI/DOI?10.1055/s-0038-1667083

[9] S. Hochreiter and J. J. Urgen Schmidhuber, "LSTMOriginal," Tech. Rep. 8, 1997. [Online]. Available: http://www7.informatik.tu-muenchen.de/ hochreithttp://www.idsia.ch/ juergen

[10] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches," 9 2014. [Online]. Available: http://arxiv.org/abs/1409.1259

[11] "torch.nn — PyTorch master documentation." [Online]. Available: https://pytorch.org/docs/stable/nn.html?highlight=grutorch.nn.GRU

[12] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 2015.

[13] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-D Convolutional Neural Networks for Signal Processing Applications," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., 5 2019, pp. 8360–8364.