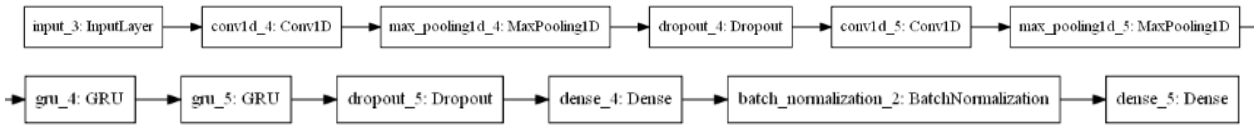
Classification of EEG sleep signals:
Dennis Hokke (2045370), Lisa Warrink (2045339) , Yaohua Liu (2041746) , CodaLab username : u979375

Data preprocessing: For the assignment we used the features from the spectrograms. The data is in the shape (15375, 2, 100, 30). This four dimensional data could not be used as input into the model. To address this, we started off by flattening the data, but we observed that our model performed very poorly when we did this. Therefore, we decided to reshape it into a three dimensional array with size (15755, 200,30) by squeezing the two areas of EEG data into one. The resulting shape was (200,30). Additionally, we tried a variety of feature engineering methods. For example, we performed normalization on the data and adding Gaussian noise on it in order to increase the generalization ability of the model. Unfortunately, these did not improve the performance of the model.

Model architecture: For our final model architecture, we combined 1D convolutional neural networks (CNNs) and Gated Recurrent Units (GRUs). The CNN layers were used to extract different patterns in the features, and two layers of stacked GRUs were used to learn sequential trends in the data (Zhu, Chen & Chen, 2019; Chollet, 2017)). We initially started with a CNN-LSTM model, but we switched from LSTM to GRU, a variant of LSTM, because it was simpler and faster, especially when dealing with large datasets (Chen, Jiang & Zhang, 2019). The standard LSTM uses three gates, forget, input and update, that allows the network state to be updated, reset or progress without any modification (Alhagry, Fahmy & El-Khoribi, 2017). The GRU is more efficient because it synthesizes the forget state and the input gate to a single update gate (Chen, Jiang & Zhang, 2019; Zhu, Chen & Chen, 2019).

Figure 1. Schematic of the 1D CNN-GRU model



We used two layers of 1D CNN (32 filters, kernel size 5 and relu activations). The first convolution layer was followed by a maxpooling layer (size 3) and a dropout function (probability 0.3). The second convolution layer was followed by a maxpooling layer (size 3). The output from this was passed onto two layers of stacked GRUs (size 256). A dropout function (probability 0.3) was added to the output from the GRUs to reduce overfitting. This was then passed to a fully connected dense layer (size 10) to further extract the features of the input. After this, we applied batch normalization to the hidden neurons of this layer and passed it through the output layer (size 6, softmax activation) to do the classification. The summary of the model architecture is shown in Figure 1. The hyperparameters for the model were determined using automatic hyper-parameter tuning using keras tuner recently released by Google (O'Malley, 2020).
The results of the different model architectures we tried are summarized in Table 1.

Table 1. Results of 1D CNN GRU models

Model	filter_ num	dropout_r ate	Units (GRU)	Units (Dense layers)	Layers (Conv1d)	Layers (GRU)	Learning rate	Score (Validation)	Score (Test data)
1D CNN-GRU with (15375, 200, 30) data shape	32	0.3	256	10	2	2	2e-3	81.2%	65.45%
1D CNN-GRU with two concatenated model	64	0.1	128	10	2	1	1e-2	75.1%	63.16%
1D CNN-GRU with normalized data	32	0.3	256	10	2	2	1e-3	70.2%	59.17%

Discussion: The 1D CNN-GRU model with normalized data yielded the lowest accuracy score of 59.2% on the test set. The 1D CNN-GRU model with two concatenated models yielded an accuracy of 63.16% on the test set. The 1D CNN-GRU model with reshaped data yielded the highest accuracy of 65.45%. Overall, for all three models there is approximately a 10%+ difference between the test data accuracy and the validation accuracy, and the higher the validation accuracy the greater the difference with the test accuracy. We suspect the difference between validation and test accuracy is high because the underlying distributions of the train/validation and the test set are different. Another reason might be the model does not fit the data well, this can be observed by the good convergence of loss in the training set but high fluctuation of loss in the validation set as shown in graph 1. Additionally, we applied gaussian noise to the training data. By adding noise to the train data and training the model on the noisy data, we expect the model to be more robust and have better performance on the test data. However, this was not the case, this supports our assumption that the test data has a different underlying distribution than the training/validation data. Furthermore, the biggest limitation of our approach for this assignment is that we do not perform feature engineering on the data. We expect our model to perform better if we included feature engineering.

References

Alhagry, S., Fahmy, A. A., & El-Khoribi, R. A. (2017). Emotion recognition based on EEG using LSTM recurrent neural network. Emotion, 8(10), 355-358.

Chen, J. X., Jiang, D. M., & Zhang, Y. N. (2019). A Hierarchical Bidirectional GRU Model With Attention for EEG-Based Emotion Classification. *IEEE Access*, *7*, 118530-118540.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. (2017). *Deep Learning with Python*. Shelter Island, NY: Manning Publications

O'Malley, T. (2020). Hyperparameter tuning with Keras Tuner. Retrieved 16 March 2020, from https://blog.tensorflow.org/2020/01/hyperparameter-tuning-with-keras-tuner.html

SM, I. N., Zhu, X., Chen, Y., & Chen, W. (2019, October). Sleep Stage Classification Based on EEG, EOG, and CNN-GRU Deep Learning Model. In *2019 IEEE 10th International Conference on Awareness Science and Technology (iCAST)* (pp. 1-7). IEEE.

Appendix

Appendix A
Graph 1: Training and validation accuracy of our model.

Appendix B
Graph 2: Training and Validation loss of our model