Représentations Parcimonieuses: Challenge Dreem

Master MVA (2020 - 2021)

Dorian Desblancs
École Normale Supérieure Paris-Saclay
4 Avenue des Sciences, 91190 Gif-sur-Yvette, France
dorian.desblancs@mail.mcgill.ca

Liam Gonzalez
École des Ponts ParisTech
6-8 Avenue Blaise Pascal, 77420 Champs-sur-Marne
fragloth35@gmail.com

Abstract

In this project, we tackled the problem of automatic sleep apnea detection proposed by Dreem, a French neuro-technology start-up. We were given a set of signals that are commonly used for manual sleep apnea detection, and a set of masks indicating where and when a apneic event was occurring. The goal of this project: to automatically derive this mask from the set of signals provided, using an algorithm of our choice. Our best solution merged a multi-layer convolutional neural network (CNN) with a Gated Recurrent Unit (GRU) network. We used a Tversky loss function during training. Our network used all eight 1D signals as input, and achieved a final F1-score of 0.5593 on the hold-out set. These results place us near the top of the challenge ranking (currently sixth). This report explores our best method in depth. It also explores some of the other approaches we implemented, and discusses whether these should be further inspected.

1 Introduction

Sleep is a natural task carried out by all mammals. It is characterized by an altered consciousness, and allows our brain and body to rest and process information from the past day. Unfortunately, repeated disrupted sleep can have dire consequences on our health. Chronic sleep deprivation can lead to diabetes and heart disease, and in the worst cases premature death. These days, one of the most common disruptions to the general population's sleep is a condition called sleep apnea.

Sleep apnea is a sleeping disorder during which a person's breathing repeatedly starts and stops. It is characterized by repeated episodes of apnea, during which breathing stops, and hypnopea, during which breathing is insufficient. In both cases, blood oxygen levels drop. This leads to fragmented sleep, and in most cases chronic sleep deprivation. Approximately 25% of males and 10% of females are believed to have this disorder. Even more seriously, according to the American Academy of Sleep Medicine (AASM), more than 80% of sleep apnea cases are left undetected.

In this project, we tackled the sleep apnea detection problem proposed by Dreem, a French neurotechnology start-up. We were given polysomnography (PSG) data recordings from 44 nights (one for each patient). Note that PSGs record physiological data, and are used to manually detect sleep apnea in people. We were given 8 signals for each night: an abdominal contraction measure, a respiratory airflow measure, a photoplethysmogram (PPG), a thoracic contraction measure, a snoring indicator, SpO2, and two EEG derivations (CA-A1 and O2-A1).

These 44 nights were then split into 200 windows of 90 seconds. For each window, we were given a binary mask of length 90, which indicated where and when sleep apnea was occurring. The goal of this project was simple: from the physiological data given, derive the binary mask. Sample signals and their associated mask can be found in the Appendix of this report. We explored numerous ways to solve this problem. We most notably experimented with various signal representations (raw signals, spectrograms, and scaleograms), multiple loss functions, and a variety of neural network architectures. Our best model achieved an F1-score of 0.5593 on the hold-out set (sixth on the public leader board), and surpassed the benchmark score by more than 0.03 points. This report will explore our best method and its results. We will then cover some of the other experiments we conducted, and discuss what worked

and what did not. Finally, we will explore some of our ideas for the future and other methods we would have liked to try.

2 Related Work

2.1 Neural Networks

Time series classification and generation has been a popular problem for many decades. Whether it be for stock predictions or ECG classification, the ability to process and learn from time series has led to numerous discoveries and technologies.

Over the past few years, neural networks have taken over this class of problems [5]. Popular architectures include Convolutional Neural Networks (CNNs) [7], Long Short-Term Memory Networks [6] and its variants, and more recently attention-based networks [18]. Most of the above are behind the state-of-the-art results for time series problems, and were used as inspiration to tackle our challenge.

In the realm of segmentation, the research regarding time series is a little bit more limited, especially regarding the use of neural networks for such a task. Image segmentation using masks is however extremely popular in computer vision. More precisely, in medical imaging, these are widely used to isolate tumours or other anomalies that can lead to disease [16]. Most state-of-the-art methods rely on CNNs, and especially the famous U-Net [11] architecture. Research in this domain was thus used as inspiration for our solution.

2.2 Sleep Apnea

Past research for automatic sleep apnea detection has been focused on the problem of sleep apnea classification. Usually, the task goes as follows: from a set of signals, derive whether a patient has sleep apnea or not. The results regarding this task are quite stellar. In 2012, Xie and Minn [20] achieved an accuracy of 82% using classic machine learning algorithms such as KNN and AdaBoost and signals similar to ours. Very recently, Zhao et al. [21] achieved more than 88% using EEG sub-band signal characteristics using SVM, KNN, and random forests.

Deep learning methods are also popular in the sleep apnea domain. They are especially useful for segmentation tasks, the main one being sleep stage segmentation. This task has been very helpful for detecting the severity of sleep apnea in patients. Please consult [9] for more information.

Finally, the paper most similar to our task was published in 2019 by members of the Dreem team [3]. In their paper, Chambon et al. propose a novel method for identifying sleep apnea events in short signals of 30-second duration. They use a deep learning architecture called Dreem One Shot Event Detector (DOSED) in order to do so, and achieve state-of-the-art results on a variety of data sets (F1 scores in the range of 0.5 to 0.8 depending on the task and data set). The architecture used includes convolutional and max-pooling layers. We drew some of our inspiration for this challenge from this paper and encourage the reader to consult it for a more in-depth look into sleep apnea micro-event detection.

3 Methodology

3.1 Data Processing

The signal pre-processing we used was rather simple. We took the signals as they were, and tried to either normalize or standardize them between 0 and 1. We found that normalizing the signals using the following formula:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

(x is the signal values, μ the signal's mean value, and σ the signal's standard deviation) helped the most with performance. Note that the mean and standard deviation were computed independently for each signal. We also tried to normalize the signals on a per patient, i.e to have a separate $\mu^{(i)}$ and $\sigma^{(i)}$ for each of the 44 patients. This did not yield to any performance gain, however.

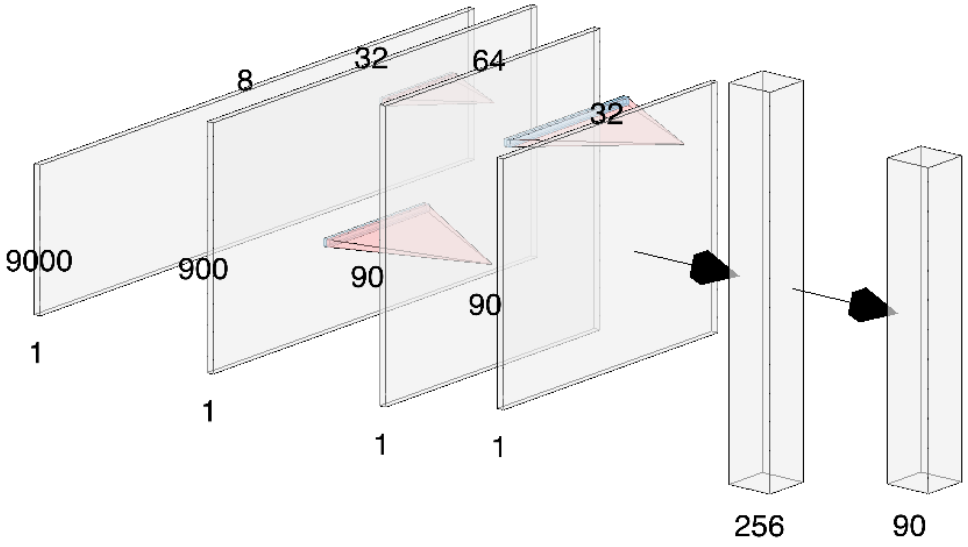


Figure 1: Gru-CNN Network Architecture

3.2 Network architecture

Our best model's architecture did not end up being very complex. It takes all 8 signals of size 9000 as input. A one-dimensional convolution with kernel size 15 is then applied to these. This convolution is followed by a max-pooling step, which downsizes the signals to a length of 900, a ReLU activation, and a dropout step ($p = 0.2$). The second layer is exactly the same, except that the kernel size used is 21. The output signals are of size 90. Finally, a third convolution of kernel size 25 is applied to the output of the second layer, followed by an ReLU activation and dropout. Figure 1 illustrates our CNN's architecture (note that the signal's length and the number of channels is inverted). Surprisingly, we found that increasing the kernel size as the signals were downsized to work better than the opposite. We also found that keeping our model's width small led to better performance, hence the reason why our model never exceeds 64 channels.

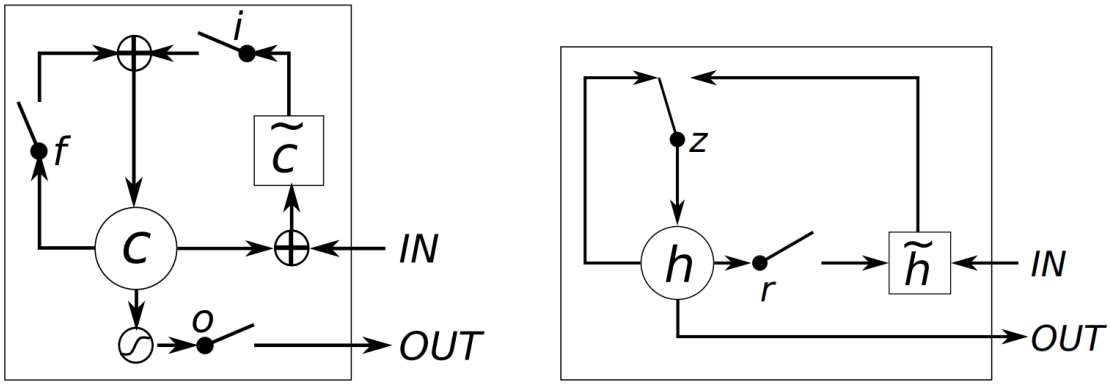
The output of our convolutional layers (size 8×90) was then fed into a Gated Recurrent Unit (GRU) network [4] with 32 hidden layers. GRUs are recurrent neural networks similar to LSTM networks. However, they lack an output gate. They therefore contain less parameters to train, and have been shown to perform better than LSTMs and RNNs on a number of tasks involving sequential data. Figure 2 displays a Gated Recurrent Unit and its differences with an LSTM unit. We found that including the GRU after having downsized the signals led to faster training and better performance. More generally, our convolutional layers served as feature extractors for our GRU, whose architecture is optimized for sequential data. We limited our GRU to one layer and one forward direction (no bidirectionality).

Finally, the output of our GRU (of size 32×90) was flattened and fed into a fully-connected layer with 256 feature outputs. This layer was followed by and ReLU activation, a 90-feature output fully-connected layer, and a sigmoid activation to squash the model's outputs between 0 and 1. Note that our model's outputs were converted to 0s and 1s using a threshold of 0.5 to calculate F1-scores. This was not done when computing the loss (for convergence purposes).

3.3 Loss Functions

We experimented with numerous loss functions throughout this project. The first one we used successfully was binary cross-entropy, with a mean reduction, defined as:

$$l(x, y) = mean(L) \tag{2}$$



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 2: Illustration of (a) LSTM and (b) gated recurrent unit. (a) i, f and o are the input, forget and output gates, respectively. c and $\widetilde{c}$ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and $\widetilde{h}$ are the activation and the candidate activation. Taken from [4].

where:

$$L = (l_1, ..., l_n)^T \tag{3}$$

and:

$$l_n = -y_n log(x_n) - (1 - y_n) log(1 - x_n) \tag{4}$$

where y_n denotes the target values and x_n our model's output. Note that $n \in [0, 90[$ for us (size of the predicted and target masks). This loss allowed our model to converge towards a working solution. Note that we also experimented with a weighted binary cross-entropy loss, which penalized false negatives. This rarely helped with our models' performances, however.

We then experimented with a few loss functions that are popular in medical image segmentation. These included Focal loss [12], Dice Loss [17], and Tversky loss [1]. We define Tversky loss as:

$$TL(p, p') = 1 - \frac{1 + pp'}{1 + pp' + \alpha(1 - p)p' + \beta p(1 - p')} \tag{5}$$

where p corresponds to the target values and p' the model outputs. Notice that the loss function can be re-written as:

$$TL(p, p') = 1 - \frac{1 + TP}{1 + TP + \alpha FP + \beta FN} \tag{6}$$

in our case. We used $\alpha = 0.4$ and $\beta = 0.6$ in order to decrease our false negative rate. This loss function was used to generate our best results, and allowed our models to detect substantially more sleep apnea events than we had with binary cross-entropy.

4 Results

In order to evaluate our model's performance, we split the training set into five folds of size 880. We then trained five models on a different subset of four folds. The model that achieved the highest F1-score on each validation set was then saved for future use.

We used a batch size of 100 and the Adam [8] optimizer for training. Our learning rate was set to 0.0001. Note that our Gru-CNN only contains 858,842 total trainable parameters. We trained each model for 150 epochs. Results for each validation set were generated in less than 12 minutes. Table 1 displays our model's best performance on the validation set using either Tversky or BCE Losses. Notice how much better our model performs using Tversky loss.

Table 1: Model Performance using Binary Cross-Entropy and Tversky Losses

Best F1 Score		
Test Set	Tversky Loss	BCE Loss
Validation Set 1	0.514	0.457
Validation Set 2	0.492	0.428
Validation Set 3	0.580	0.449
Validation Set 4	0.472	0.435
Validation Set 5	0.529	0.463
Hold-out Set	0.559	NA

Once a model was generated for each set of folds, we used a voting procedure to generate results for the hold-out set. For each mask, our five models predicted an output with 0s and 1s. At each time step of the mask, the value that was predicted the most was selected as our final, hold-out set output. That is the reason why our final test set performance is actually better than the models' performances on each fold. Our voting mechanism allowed us to combat over-fitting on the training set.

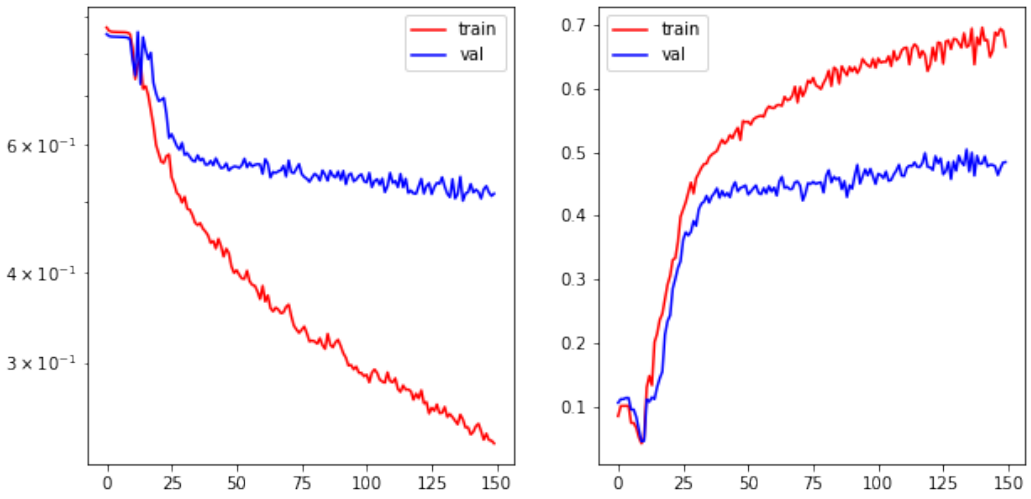
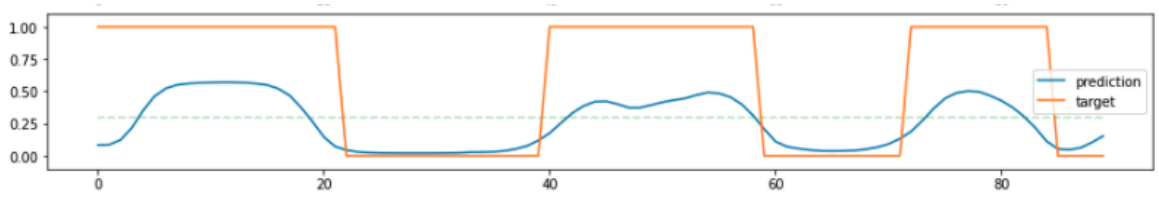


Figure 3: Training and Validation Set 1 Loss and F1-Scores

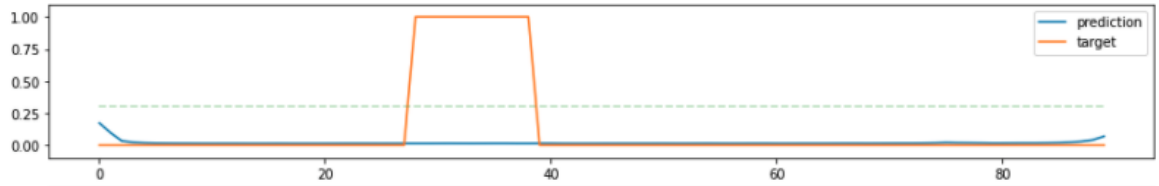
Figure 3 outlines the evolution of the Tversky loss and F1-score of the training and validation sets as the model's weights are adjusted. One can clearly notice that the validation loss plateaus much more quickly than than the training loss. We however opted to continue training, and in some sense, keep over-fitting on the training set. That is because we noticed that the F1-score on the validation set slowly increased as the model over-fit on the training set. More importantly, our voting mechanism allowed us to combat drastic overfitting. In some way, having five overfit models led to better performance than five models trained too lightly. After all, most of the error we obtained was due to missing out on sleep apnea events totally. Training our models for longer allowed us to avoid this problem a little. Figure 4 displays the most common ways in which our model accurately or inaccurately classified a sleep apnea event. Notice how the model totally misses the second event.

5 Discussion

We just presented the algorithm and inputs that worked the best for us during this challenge. We also experimented with a variety of other input representations and CNN architectures. In this section, we will delve into the techniques we implemented, and discuss whether they worked or not.



(a)



(b)

Figure 4: Accurately (a) and Inaccurately (b) Classified Masks

We first experimented with two common signal transformations: the Fourier transform and the Wavelet transform. Illustrations of each transform can be found in the appendix.

For the Fourier transform, we found that transforming our signals into mel-spectrograms of size 8×900 helped us achieve F1 scores in the range of 0.35 to 0.45. The CNN architecture used was similar to our Gru-CNN, except that the convolutions were two-dimensional. We also did not have a GRU layer in this CNN. We used mel-spectrograms because we found that most of the power in our regular spectrograms was concentrated in the frequencies under 10 Hz (note that all signals were sampled at 100 Hz). Mel-spectrograms allowed us to increase the resolution across the lower frequencies (see Appendix for an example of this). We also found that increasing the time resolution helped our model learn. After all, a max-pooling layer was very beneficial for extracting features. We do not report precise resuluts in this paper because we quickly abandoned this method, but bear in mind that the results were decent. We encourage the reader to further explore the Fourier domain for this challenge.

We also explored the Wavelet transform. For each signal, we generated scaleograms with 16 scales. We used the Morlet wavelet [2], and resized the time dimension from 9000 to 720 (input size of 16×720). Our CNN was composed of 2D convolutional layers, ReLU activations, max-pooling, two GRU layers, a 1D convolutional layer, and a final sigmoid activation. Our network achieved validation F1-scores between 0.45 and 0.5, suggesting potential for very good results. The wavelet transform was however extremely demanding computationally. As such, we abandoned this approach for this challenge. More research is however necessary in the realm of wavelets and CNNs, and their potential for time-series learning tasks. Examples of our signal transformations are represented on 5,6,7.

Finally, we also trained our Gru-CNN on a single type of signal (only Snoring for example), and we surprisingly obtained decent results for most of them (F1-scores of 0.2 at least), and we even reached 0.5 F1-score when only using ThorBelt or AbdoBelt. This suggests that some signals were more useful than others, and sometimes even complementary. This suggests that a global model could only be learning on the ThorBelt or AbdoBelt signals, and considering the other signals as mere noise, and hence totally missing apnea events related to those ignored parts of the data.

Lastly we also attempted to train our models using soft masks instead of the hard, binary masks. This led to faster training at the expense of worse performance.

6 Future Work

In the future, we hope to expand upon the work already completed during this challenge. First, we would like to further research CNN architectures that are adapted to the Fourier and Wavelet domain. The time-frequency domain clearly highlights where and when sleep apnea occurs in some cases. Perhaps combining raw and transformed signals as inputs to one model would help our mask generation process.

We would also like to explore more complex architectures for our 1D signals. We did not successfully implement a U-Net architecture [15], but truly believe that these could serve as perfect feature extractors in our task. More

precisely, inserting a U-Net feature extractor before our Gru-CNN layers might drastically help our model's performance.

Since each channel of the signal should probabl be processed separately, we suggest that multi-modal approaches that separately optimize the featurization of each single signal and the combination of those features could lead to interesting results. We tried to implement a multi-modal fusion network based on the Central-Net approach [19], but did not succeed.

Finally, we definitely need to address the problem of feature selection and outlier processing. Some of the signals used for classification are clearly not very useful for our task. Perhaps getting rid of these would help our model's performance. Outlier removal is also very common in time series tasks, and should probably be explored in our pre-processing step.

7 Conclusion

Although we believe there is still more work to be done, we are quite satisfied with our solution. The Gru-CNN attained an F1-score of 0.5593 on the hold-out set. It surpassed both the provided benchmark and a number of other proposed solutions. Exploring other loss functions also proved to be highly beneficial. More particularly, using a Tversky loss function allowed us to generate our best results, with a gain of almost 0.1 compared to standard binary cross-entropy loss. On a separate note, this project allowed us to gain experience in the field of time series applied to the field of medicine. We hope to further explore this field in the future, and thank Dreem and the Collège de France for providing us with such a fun challenge.

Acknowledgements

We used the following libraries extensively during our project (for citation purposes): PyTorch [14], Librosa [13], and PyWavelets [10].

Note that our code will be available to the public once the challenge ends (GitHub username: d-dawg78, repository: MVA_RP).

Updates

After slightly changing our training methodology and model (small alterations to number of channels and training set shuffling), we obtained a performance of 0.591 on the public hold-out set and 0.568 on the entire hold-out set. We are 7th out of 33 for the challenge!

References

[1] Nabila Abraham and Naimul Mefraz Khan. "A novel focal tversky loss function with improved attention u-net for lesion segmentation". In: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE. 2019, pp. 683–687.

[2] Alexandre Bernardino and José Santos-Victor. "A real-time gabor primal sketch for visual attention". In: *Iberian Conference on Pattern Recognition and Image Analysis*. Springer. 2005, pp. 335–342.

[3] Stanislas Chambon et al. "DOSED: a deep learning approach to detect multiple sleep micro-events in EEG signal". In: *Journal of neuroscience methods* 321 (2019), pp. 64–78.

[4] Junyoung Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint arXiv:1412.3555* (2014).

[5] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33.4 (2019), pp. 917–963.

[6] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. "A convolutional neural network for modelling sentences". In: *arXiv preprint arXiv:1404.2188* (2014).

[8] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[9] H. Korkalainen et al. "Accurate Deep Learning-Based Sleep Staging in a Clinical Population With Suspected Obstructive Sleep Apnea". In: *IEEE Journal of Biomedical and Health Informatics* 24.7 (2020), pp. 2073–2081. DOI: 10.1109/JBHI.2019.2951346.

[10] Gregory Lee et al. "PyWavelets: A Python package for wavelet analysis". In: *Journal of Open Source Software* 4.36 (2019), p. 1237.

[11] Xiaomeng Li et al. "H-DenseUNet: hybrid densely connected UNet for liver and tumor segmentation from CT volumes". In: *IEEE transactions on medical imaging* 37.12 (2018), pp. 2663–2674.

[12] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[13] Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. Citeseer. 2015, pp. 18–25.

[14] Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *arXiv preprint arXiv:1912.01703* (2019).

[15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[16] Dinggang Shen, Guorong Wu, and Heung-Il Suk. "Deep learning in medical image analysis". In: *Annual review of biomedical engineering* 19 (2017), pp. 221–248.

[17] Carole H Sudre et al. "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations". In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2017, pp. 240–248.

[18] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[19] Valentin Vielzeuf et al. "CentralNet: a Multilayer Approach for Multimodal Fusion". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. Sept. 2018.

[20] Baile Xie and Hlaing Minn. "Real-time sleep apnea detection by classifier combination". In: *IEEE Transactions on information technology in biomedicine* 16.3 (2012), pp. 469–477.

[21] Xiaoyun Zhao et al. "Classification of sleep apnea based on EEG sub-band signal characteristics". In: *Scientific Reports* 11.1 (2021), pp. 1–11.

Appendix

Signal Transformations Tested

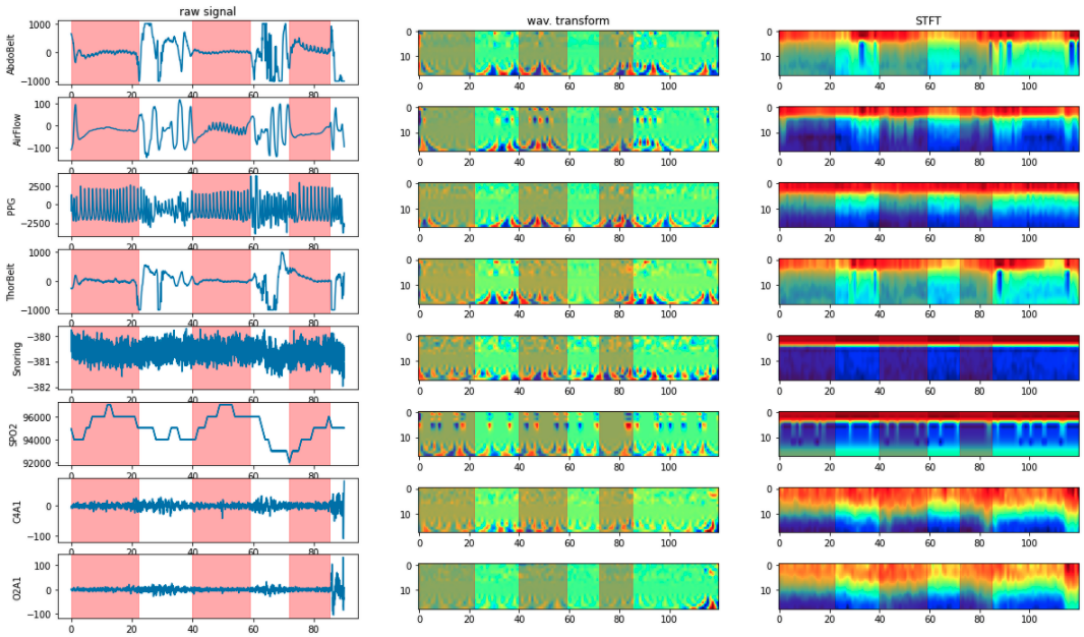


Figure 5: An example of signal transforms on a successful event detection

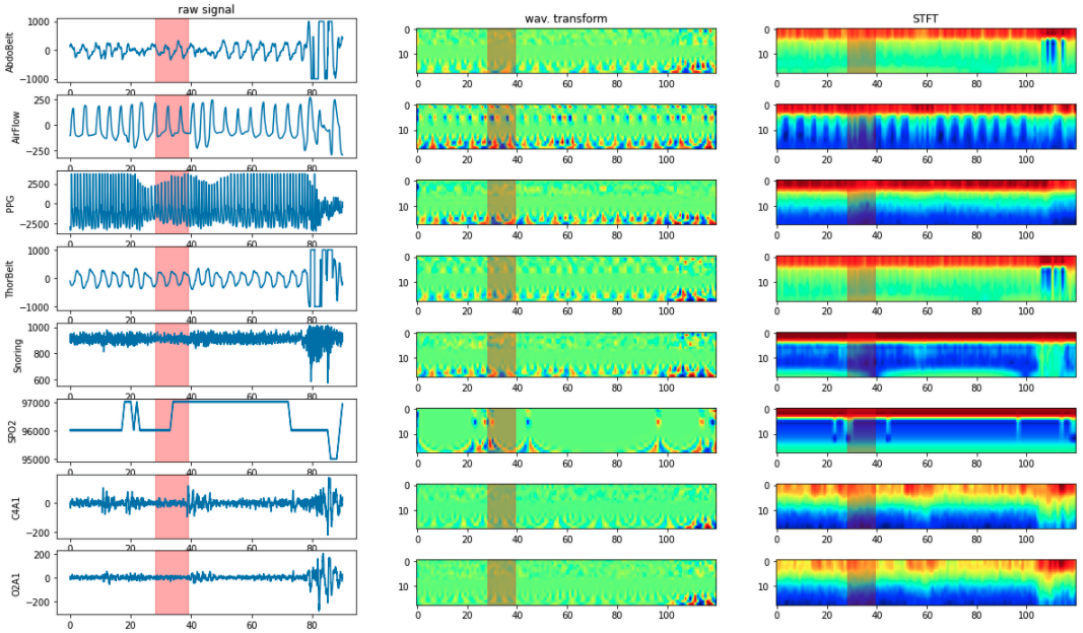


Figure 6: An example of signal transforms on an undetected event
f

Spectrogram Representations

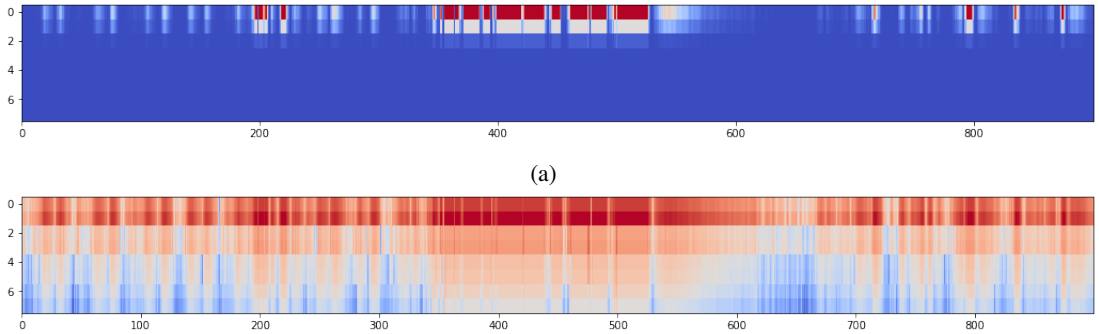


Figure 7: Spectrogram (a) and Mel-Spectrogram (b) Representation of an Abdominal Belt signal

Training Loss and F1-Scores

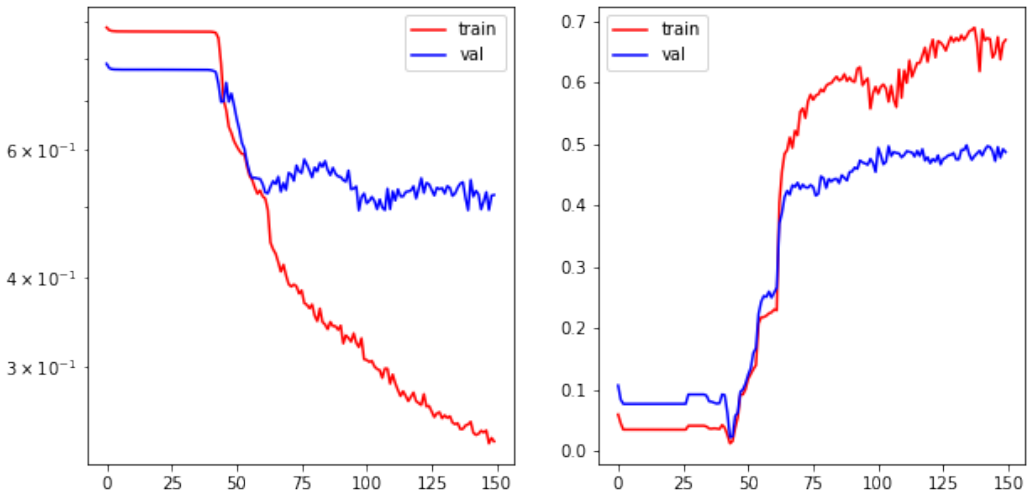


Figure 8: Training and Validation Set 2 Loss and F1-Scores

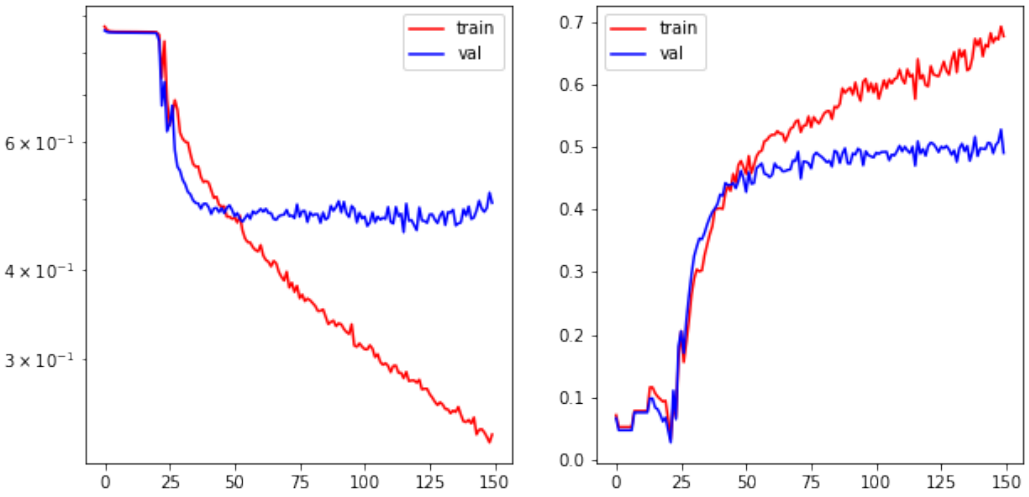


Figure 9: Training and Validation Set 3 Loss and F1-Scores

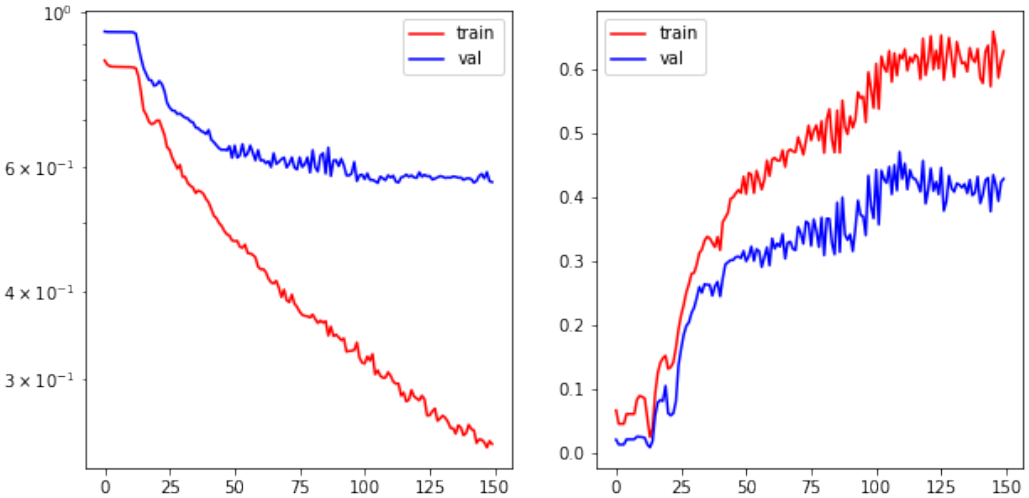


Figure 10: Training and Validation Set 4 Loss and F1-Scores

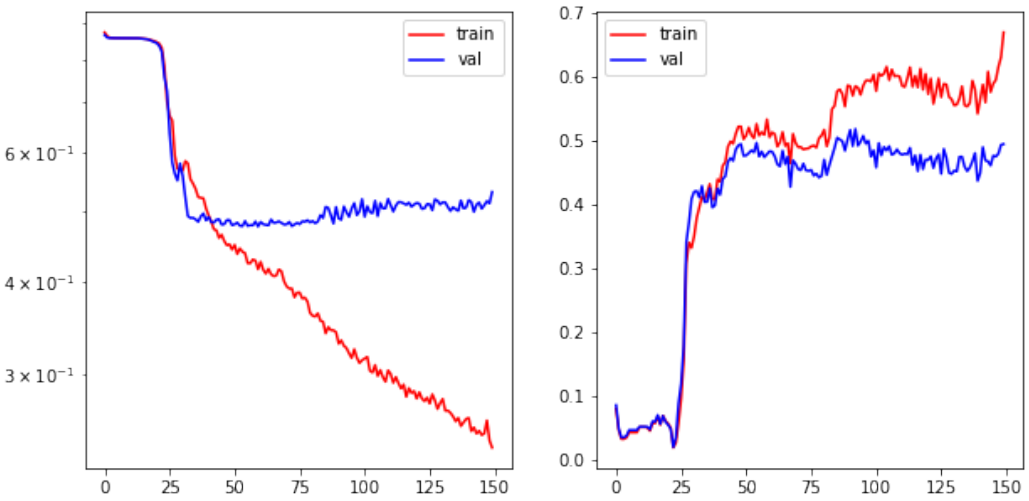


Figure 11: Training and Validation Set 5 Loss and F1-Scores