**Modular Deep Residual Networks for Driver Stress Detection   
from PPG signal**

**Abstract**

**The vast majority of methods for driver stress detection rely on complex multimodal signals that require intrusive and expensive acquisition. In the presented approach we hypothesize that the PPG signal provides enough physiological information to be used as independent reliable stress indicators. To achieve this, we propose a modular end-to-end deep learning architecture that utilizes the residual neural blocks empowered by multi-branch mechanisms. By introducing input network layers which prioritize reducing spatial resolution and extracting key features, the proposed model eliminates manual feature engineering over the input PPG signal and contributes significantly to the overall efficiency of the model. We provide a comprehensive evaluation of the model’s generalization capability, network structure, and classification accuracy. Results demonstrate that our approach is able to achieve superior performance compared to other state-of-the-art methods, while offering options for balancing complexity and accuracy. Therefore, it provides high potential for integration into real-world driving experience.**

**Introduction**

Detection of driver stress is a critical aspect of automotive safety and human-machine interaction research. While several sensor modalities such as Electrodermal Activity (EDA), skin temperature (TEMP), or electrocardiogram (ECG) might be employed to monitor driver stress [1]–[3], this paper focuses on the utilization of Photoplethysmography (PPG) as the sole modality for stress detection.

The PPG signal serves as a valuable indicator of dynamic blood volume pulse (BVP) in the body’s extremities, providing insight into various physiological and cardiovascular parameters, including heart rate (HR), heart rate variability (HRV), oxygen saturation (SpO2) and respiratory rate [4]. Notably, the connection between the heart and the brain through the autonomic nervous system (ANS) means that changes in emotional states can influence heart rate and blood ejection rate. During stressful situations, the sympathetic part of the ANS is activated, leading to a rapid increase in heart activity. This accelerates blood pumping, ensuring a rapid oxygen delivery to the body’s organs in order to enable the individual to react quickly. Increased respiratory rate causes changes to the shape of PPG signal which makes it a valuable parameter for stress assessment. In stressful situations, HR and HRV fluctuations in ANS activity lead to alterations in HR and HRV patterns, rendering them effective indicators of a driver’s stress level [5]. Both of these features can be extracted from the PPG signal.

While multi-modal sensor setups could be valuable in certain applications, there are compelling motivations to rely solely on PPG sensors for driver stress detection in our research. They are listed below:

* Non-intrusiveness: PPG is a non-invasive optical technique that measures changes in blood volume through skin using a relatively simple sensor. It does not require direct contact with the skin or electrodes, making it less intrusive and avoiding potential irritation.
* Comfort and user acceptance: PPG can be seamlessly integrated into common wearable devices like smartwatches or fitness trackers. This makes it more acceptable and comfortable for drivers, as they can continue their regular activities without additional discomfort, unlike other cumbersome sensors.
* Unobtrusive continuous monitoring: PPG allows for continuous monitoring of physiological parameters without the need for frequent recalibration or sensor readjustment. In contrast, EDA sensors may require periodic adjustment to maintain proper skin contact, which can be impractical while driving.
* Rich physiological information: PPG provides valuable physiological data correlated to heart rate, heart rate variability, and arterial stiffness, which can be indicative of stress levels. These parameters are often used as reliable stress indicators, eliminating the need for additional sensors like skin temperature sensors.
* Wearability and integration: PPG can be seamlessly integrated into various wearable devices that are increasingly popular among drivers. This integration allows for continuous, unobtrusive data collection, facilitating real-time stress detection without the need for additional sensors.

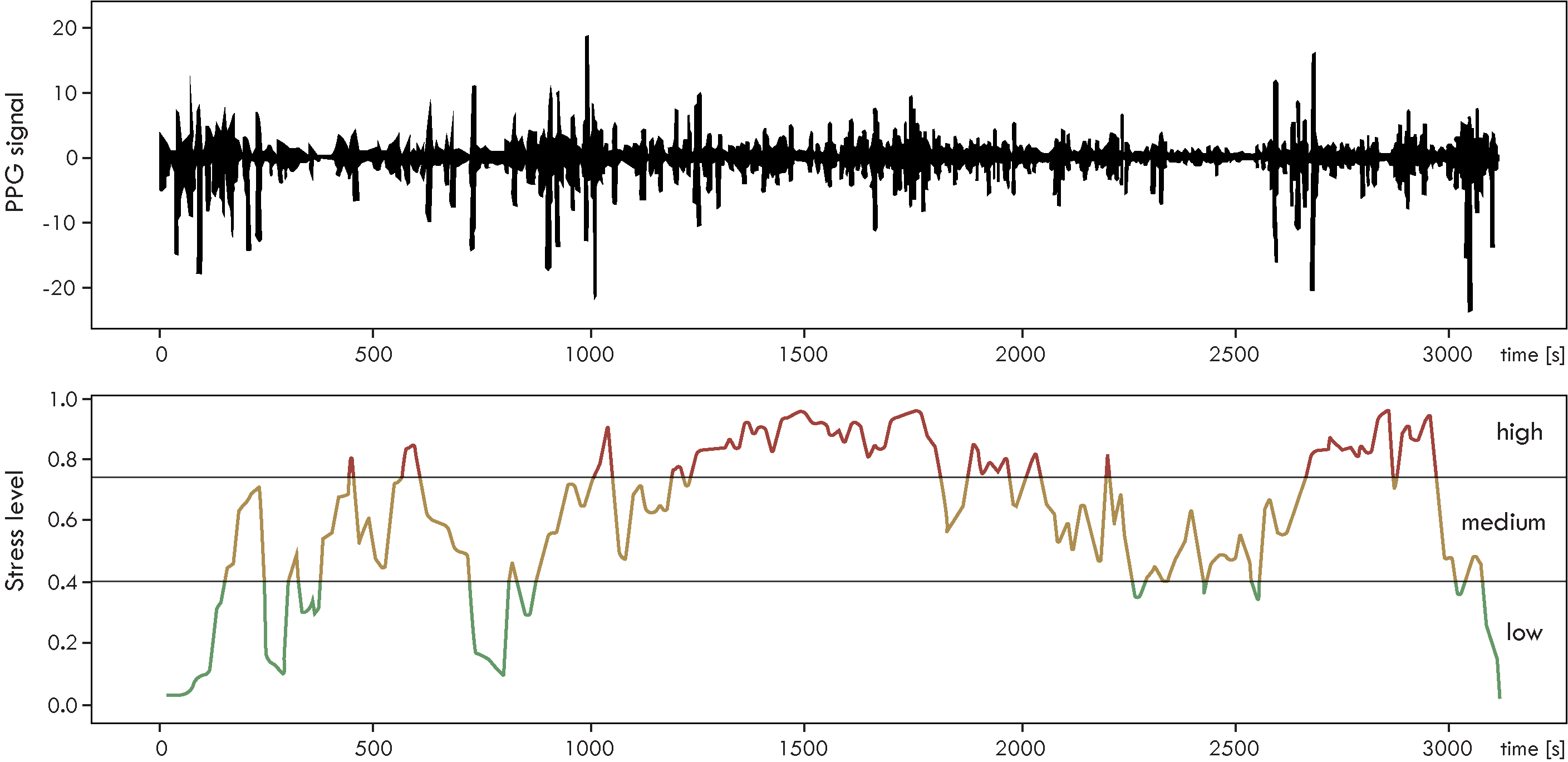


Figure 1. Example of PPG signal (top) and estimated stress level (bottom) during a drive

* Cost-efficiency: By relying solely on PPG, the cost of sensor deployment and maintenance can be significantly reduced compared to multi-modal sensor setups, making it a cost-efficient option for stress detection in automotive applications.

Previous advantages make PPG sensors a very promising choice for developing efficient and user-friendly driver stress detection systems, contributing to enhanced road safety and driver well-being. In the context of driver stress detection from PPG signals, after splitting the input data into sliding window segments, the aim is to classify them into different categories of stress level. Figure 1 illustrates the temporal dynamics of a PPG signal (top graph) and Stress level (bottom graph). Examples of sliding windows belonging to three categories of stress (low, medium, and high) are shown in Figure 2.

From the previous figures it is evident that the PPG signal exhibits a significant level of noise that might be an insurmountable challenge for traditional machine learning techniques, where manual feature extraction has to be performed. However, recent advances in Deep Learning which has made significant progress in Computer Vision [6] and Natural Language Processing [7], has led researchers to explore its application for solving complex time-series classification problems. One of the strengths of deep learning methods is its capacity to automatically extract novel discriminative features from the low-level noisy input signal.

By empowering the PPG physiological signal with modern Deep Learning algorithms for time-series data analyses, we have developed a stress detection method that could be seamlessly integrated into the driving experience to provide accurate monitoring of driver stress levels while minimizing system complexity and cost. To achieve this, we propose a modular end-to-end deep learning architecture that utilizes the residual neural blocks empowered by multi-branch mechanisms. The proposed model eliminates manual feature engineering, by introducing carefully engineered input network layers which prioritize reducing spatial resolution and extracting key features. Central layers consisting of residual blocks allows us to extend to different number and structure of blocks without specialized redesign. A comprehensive evaluation of the model’s generalization capability, network structure, and classification accuracy, demonstrate that our approach achieves superior performance compared to other state-of-the-art methods, while minimizing system complexity and cost. Our paper makes two main contributions: 1) Presenting a highly effective deep learning model for driver stress detection relying solely on PPG signal, 2) Demonstrating that the relatively simple deep learning architecture, based on ResNet blocks combined with multi-path approach, could achieve superior performance, thus having potential to be integrated into real-world automotive applications.

The rest of the paper is organized in XXX main sections. Section II introduces...Section III describes PPG-based methods... In section IV, we discuss emerging... Finally, the conclusion summarizes the paper proposal and briefly..

**Related work**

Detecting stress levels from a PPG signal involves addressing a time series problem with the objective of recognizing meaningful patterns, trends, and fluctuations within the collection of PPG data points recorded at regular time intervals. We approach driver stress identification as a classification problem over PPG time series data. Therefore, it demands robust Machine Learning (ML) models capable of capturing intricate temporal dependencies and patterns within PPG physiological data. Earlier time-series classification approaches relied on traditional ML methods, such as Decision Trees, Linear Regression, and Support Vector Machines (SVM) [Reference]. These methods typically necessitated manual feature engineering, which involves creating new data attributes from existing ones to provide more relevant information to the ML model. Feature extraction can occur across various domains using linear or non-linear methods. This is done by transforming, combining, or aggregating existing features [3], with the primary goal to reduce the number of features while retaining important information.

Deep Learning presents an alternative approach in which models can autonomously learn features from raw time series data, eliminating the need for manual feature engineering. Convolutional and recurrent layers are commonly employed for this purpose. While traditional ML models generally tend to be easier to interpret and explain, with fewer model parameters, this simplicity can hinder their ability to generalize effectively when dealing with noisy or complex data, thus leading to reduced accuracy of results. Modern Deep Learning models can be relatively complex, featuring numerous layers and parameters. This complexity enables them to capture intricate data patterns but also renders them susceptible to overfitting if not properly regularized. When compared to traditional machine learning approaches, deep neural networks offer the advantage of automatic feature extraction, eliminating the need for manually extracting and choosing features. A notable limitation of deep learning methods is its relatively high computational cost, particularly with deep architectures and large input datasets. This was a strong motivation for our research, to investigate optimal design of a deep learning model with limited complexity and high accuracy for time-series classification. Two prominent neural network architectures have been mostly investigated for time-series analyses: Long Short-Term Memory (LSTM) and Residual Networks (ResNet) [REF]. Both architectures represent formidable candidates for effective PPG time series analysis in the context of driver stress detection.

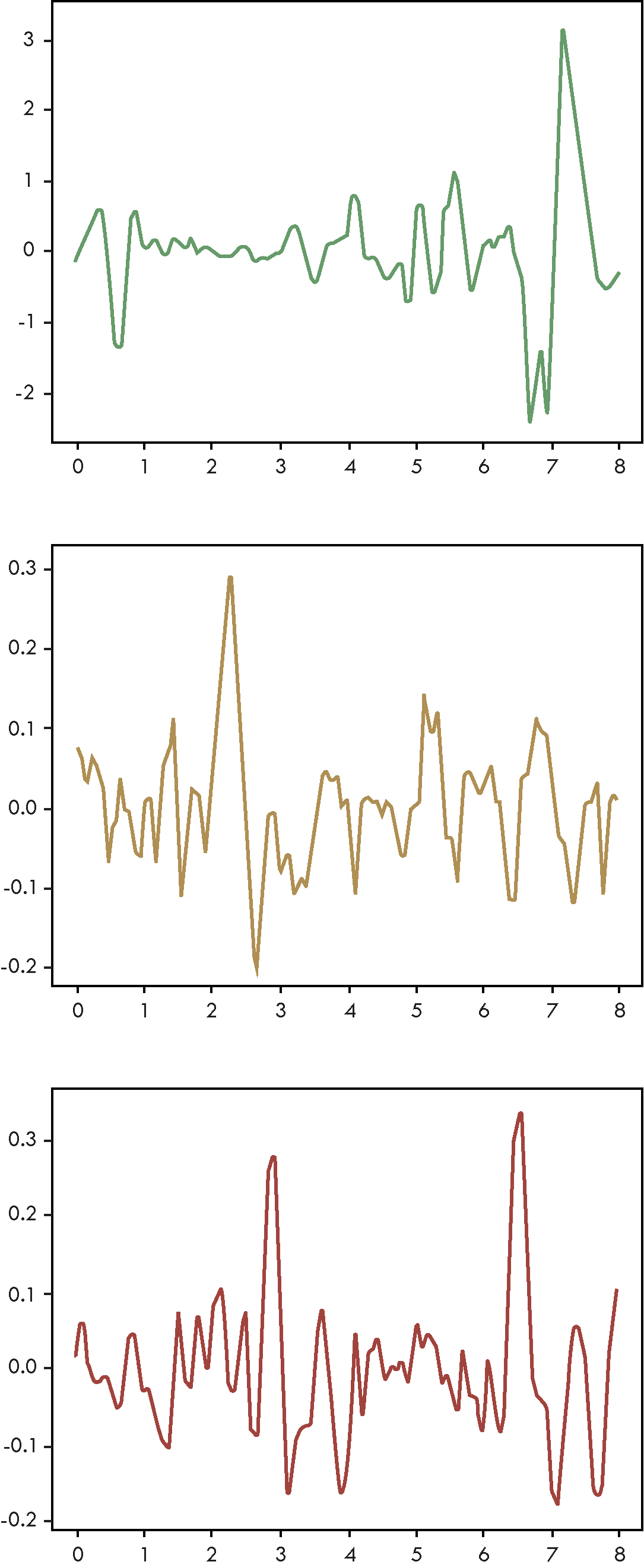


Figure 2. Examples of sliding windows during low (a), medium (b) and high (c) stress intervals

Recurrent Neural Networks (RNNs) are one of the preferred options since they allow parameter sharing, rendering them suitable for handling sequences of varying lengths [8], [9]. A closely related concept involves the application of temporal convolution, as this operation also enables the sharing of parameters across different time points within the data. The main concept behind RNNs is in the presence of a hidden state within each unit, which can be regarded as the memory of the unit. As a result, each recurrent layer receives two inputs: a vector of values from the preceding layer and the vector of states from the same layer at the previous time step. For these networks, computing the gradient of the loss function with respect to the parameters can be an expensive operation which is sequential in nature because each time step may only be computed after the previous one.

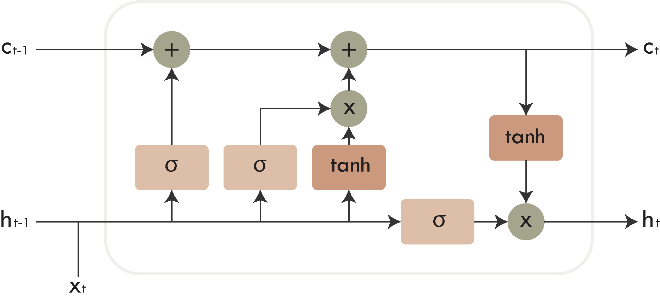


Figure 3. Basic LSTM cell [REF]

Learning long-term dependencies with RNNs can be challenging because gradients propagated across many stages tend to vanish or, less frequently, explode [10]. Even if the recurrent network would remain stable, the challenge of handling long-term dependencies persists due to exponentially smaller weights (characterized by the multiplication of numerous Jacobians) of long-term interactions compared to short-term ones. This implies that features from the beginning of sequences tend to be “forgotten”. To address these challenges, gated RNNs are employed [9]. They are based on the idea of establishing paths through time with derivatives that neither vanish nor explode. One of the most commonly used types of gated RNNs are LSTM networks [7], [11]. The LSTM layers within neural networks consist of a set of gates which regulate the flow of information. Each module comprises a forget gate, an input gate, an output gate, and a cell state (Figure 3). The role of the forget gate is to determine the extent to which information from the previous cell should be preserved. This is achieved using the sigmoid function, whose output ranges from 0 (indicating the decision to forget everything) to 1 (indicating the decision to remember everything). Likewise, the input gate serves to regulate the amount of new information to be incorporated into the cell state and the output gate determines what information should be presented as the LSTM cell’s output. Both of these gates utilize the sigmoid function in their operations.

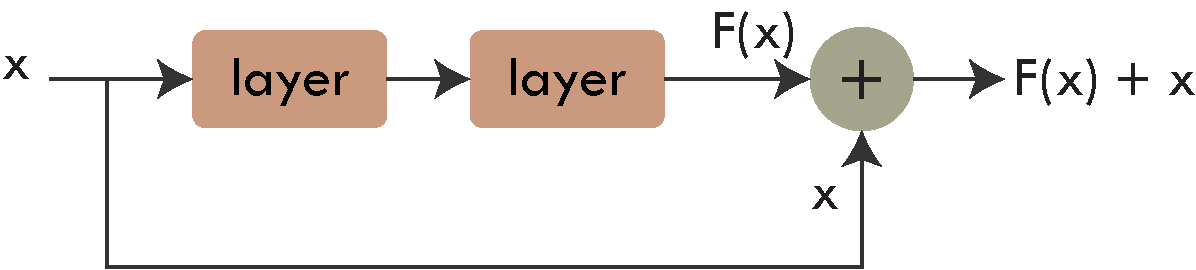


Figure 4. Basic residual block [REF]

Residual Neural Networks (ResNet) were initially introduced in a 2015 paper [12]. While it is well-recognized that deeper networks possess the capability to learn more complex features, a significant challenge arises in the form of vanishing and exploding gradients. This challenge can be partially mitigated through the incorporation of initial normalization and intermediate normalization layers. ResNets were designed to address this challenge by enabling increased network depth without compromising performance. This is achieved by having the neural network learn a residual mapping denoted as F(x) = H(x) – x, rather than the desired mapping H(x). This transformation reframes the original mapping into F(x) + x, which is implemented through the use of “shortcut connections” in a feedforward neural network, as illustrated in Figure 4. These connections are able to bypass one or more layers, introducing neither additional parameters nor computational complexity. The underlying hypothesis suggests that learning such a mapping is more manageable. Even if the identity mapping were considered optimal, it would be more feasible to drive the residual component toward zero than to precisely identify an identity mapping.

The concrete choice between LSTM and ResNet depends on the specific characteristics of the signal, the computational resources available, and the desired trade-off between modeling temporal dependencies and feature extraction. While LSTMs excel at capturing long-term sequential dependencies, they may require substantial computational resources for training and inference that may pose challenges in real-time applications. Additionaly, LSTMs are prone to overfitting, especially when dealing with small datasets. ResNets deep architecture and their efficient training offer powerful feature extraction capabilities for modeling complex noisy patterns, allowing them to automatically learn and represent important hierarchical features within the time series data. This can be highly beneficial when dealing with complex physiological PPG signals as it reduces the need for hand-crafted feature engineering. To facilitate the stable and efficient training even in very deep networks, ResNet architectures introduce residual connections which mitigates the vanishing gradient problem. This is advantageous when dealing with intricate patterns in time series PPG data. While ResNet can capture local temporal dependencies, it may not be effective enough in modeling long-range sequential relationships. However, when dealing with the PPG signal in the context of detecting driver stress, it's essential to note that these temporal dependencies generally do not extend over long intervals. This is because the PPG signal typically exhibits a repetitive, quasi-periodic pattern associated with each heartbeat. The complete waveform for a single heartbeat impulse is relatively brief, typically lasting only around a second [13]. Additionally, PPG sensors in wearable devices often have lower sampling frequencies, usually within the range of 50 to 200 Hz. By adopting multi-branch Residual Blocks, we can process multiple temporal resolutions in parallel branches, that con be utilized for capturing both short-term and long-term patterns within the same model [14].

**Modular Deep Residual Network Architecture**

We design a modular deep network architecture, specifically adapted for PPG time-series data classification. Multiple input layers are introduced to prioritize reducing spatial resolution and selecting the most important features. They are followed by a series of carefully engineered Multi-branch Residual Blocks, designed to allow extension to different number and structure of blocks without specialized redesigns. Final network layers are added to improve the regularization process and to learn complex, non-local patterns. Overall architecture of the proposed modular residual network is presented on Figure 5.

**Input Network Layers**

The proposed architecture starts with a Max Pooling layer, inspired by the original ResNet [12]. It serves a dual purpose. Firstly, it contributes to the reduction of dimensionality within the input data, and secondly, it improves the feature selection process. By selecting the maximum value from each of the pooling regions, the network effectively retains most of the prominent features, filtering out less relevant and noisy information. Then we introduce a strided convolution layer, a component typically used for downsampling, akin to pooling layers [15]. As an additional layer in our design, we also incorporate an Average Pooling layer, which utilizes a filter size of 4 and a stride of 4, leading to notable simplification of the input vector. Generally, our input layer structure prioritizes two key aspects: reducing spatial resolution and selecting key features. This contributes significantly to enhancing the overall efficiency of the model.

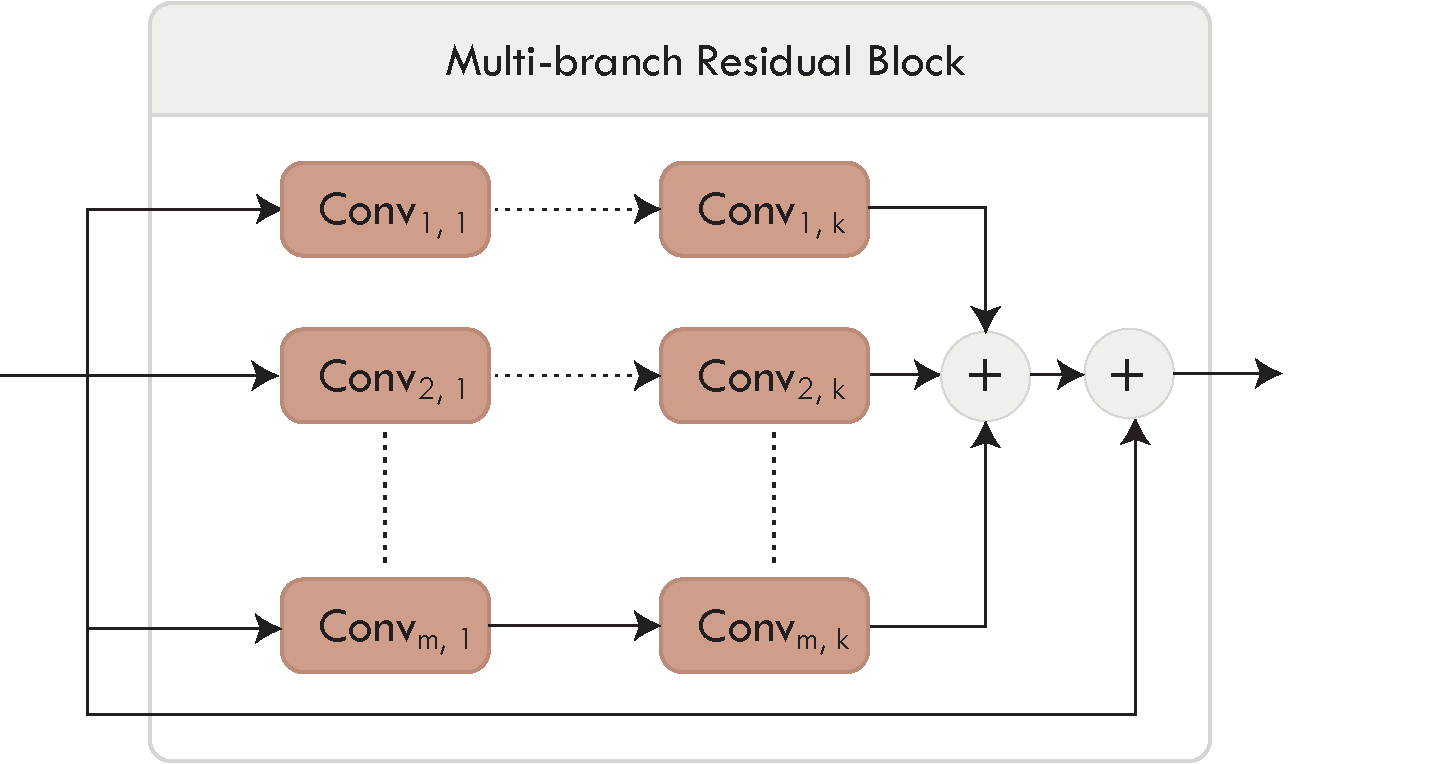


Figure 6. Multi-branch Residual Block

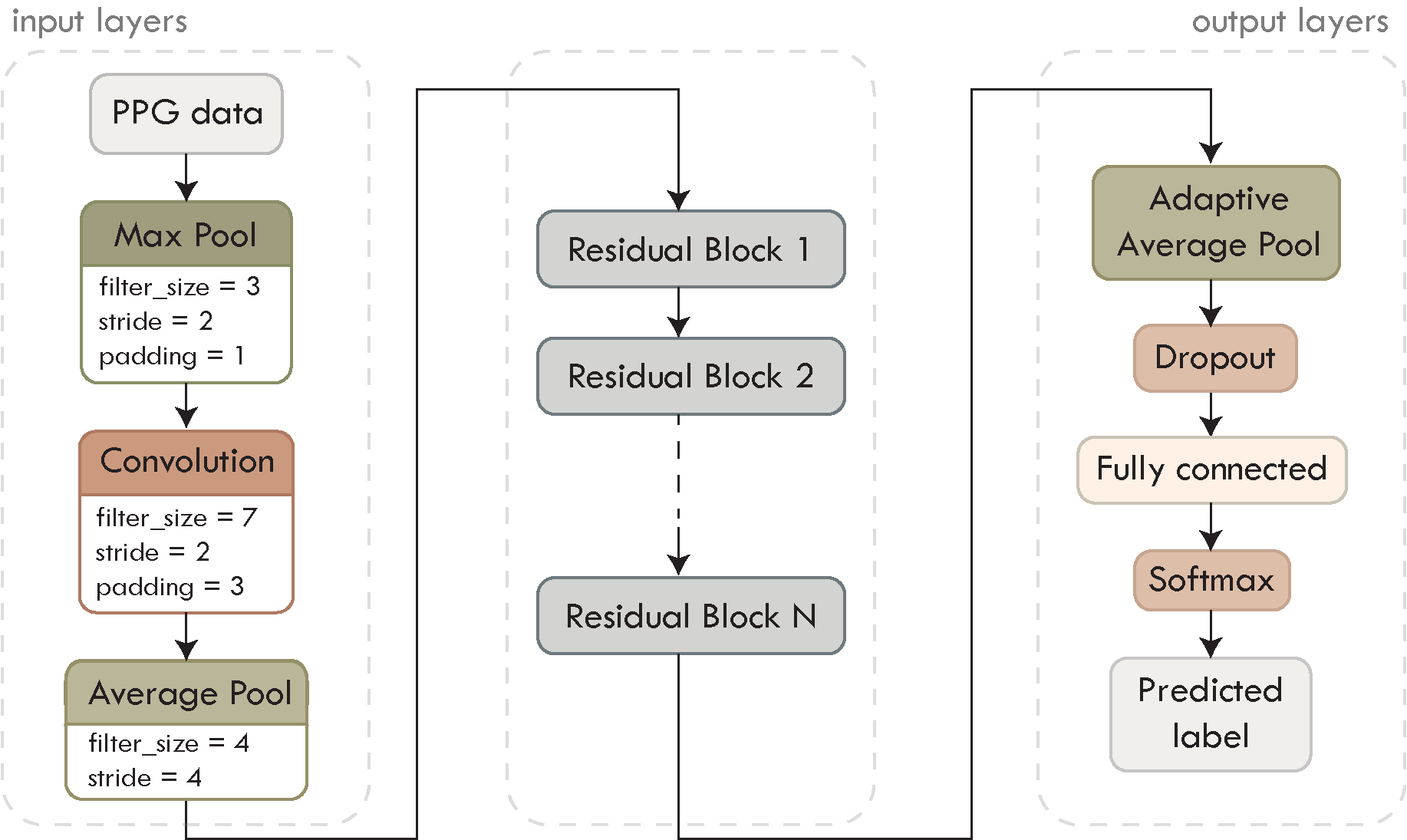


Figure 5. Modular Residual Network architecture

The input shape of the data can be denoted as , where *b* corresponds to the mini-batch size, *s* signifies the number of samples contained within each sliding window (sampling rate multiplied by the time duration of the window), and *c* represents the number of data channels. Given our exclusive utilization of BVP data derived from PPG signal, the channel count can either be or depending on whether the data source comprises a single or data from both wrists.

**Multi-branch Residual Blocks**

Following the initial input data processing, we introduced a series of *N* stacked Multi-branch Residual Blocks of the same structure. Each block has the same generic structure presented on Figure 6. During the evaluation we tested performance for different values of .

The generic residual block configuration involves the use of multiple branches with stacked convolutions of different filter sizes, enabling each branch to capture distinct information. The outputs from these branches are aggregated, and the result is finally combined with the shortcut connection (i.e. identity mapping). Having multiple branches enables the increase of network complexity without increasing the depth. The efficiency of learning temporal dependencies in neural networks is significantly influenced by the length of the pathways traversed by signals moving forward and backward within the network. In essence, the shorter the paths between the input and output sequences, the more effectively the network can capture and learn long-range dependencies in the data [15].

The output of a Generic Multi-branch Residual Block is computed in the following way. Initially, the input *x* passes through several convolution layers to generate outputs for each of the branches (Equation 1). These outputs are summed together as depicted in Equation 2, the range of depends on the number of branches. Finally, the sum of these outputs from the convolution layers is combined with the shortcut connection.

For concrete implementation of specific residual blocks, we considered several configurations, described in following.

1. **Basic Residual Block:** The basic configuration implements a residual block adopted from the paper [12]. It consists of 2 convolutional layers in a single branch and a shortcut connection (Figure 7). In selecting the filter size for the convolutional layers, we've settled on a size of 3. While larger filter sizes have the potential to capture more global features from the input data, they also increase the number of parameters. This can slow down training and render the model susceptible to overfitting. To ensure compatibility for the addition operation between the output and the shortcut, the padding is set to 1.

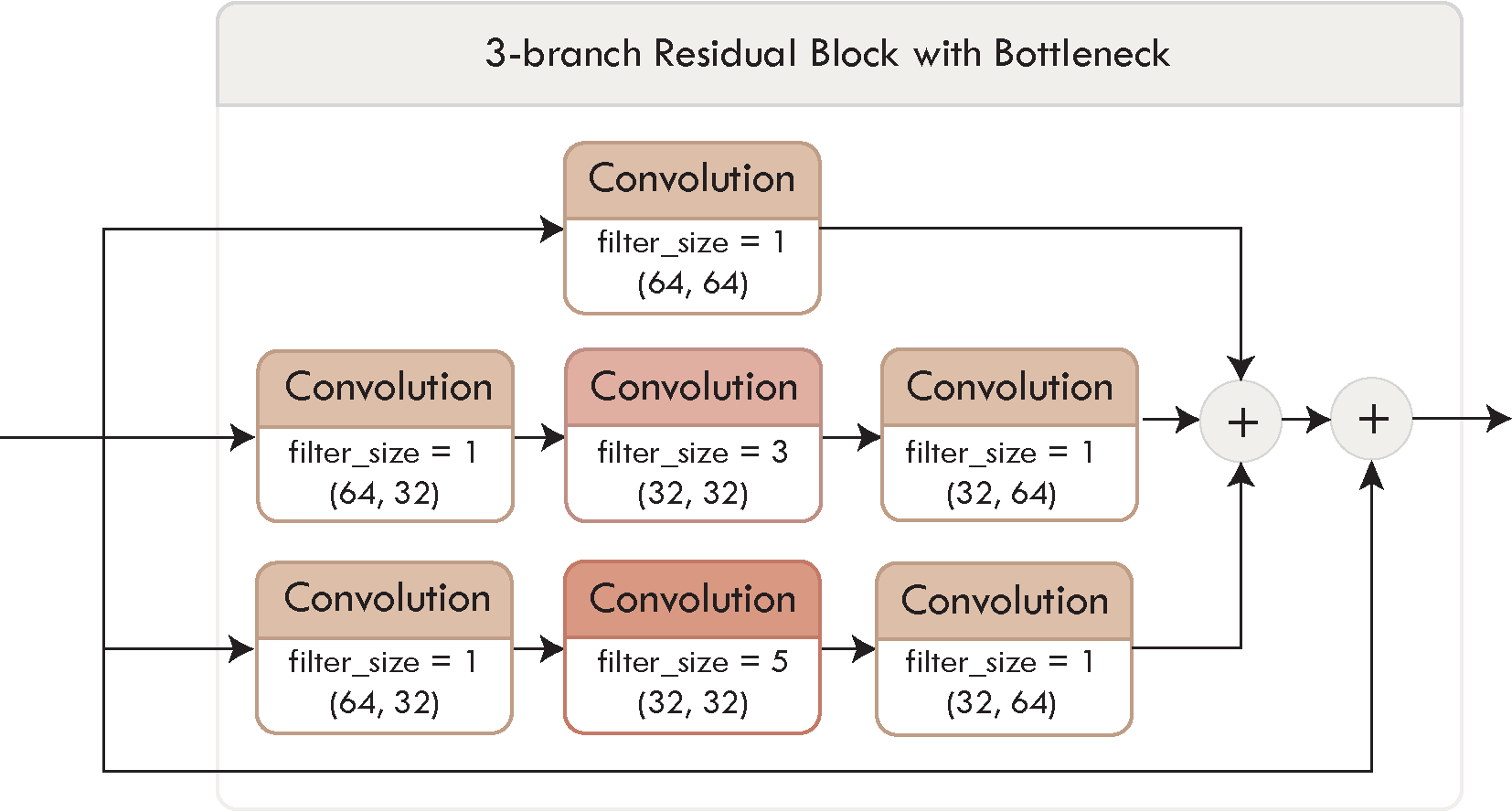


Figure 9. 3-branch Residual Block with Bottleneck

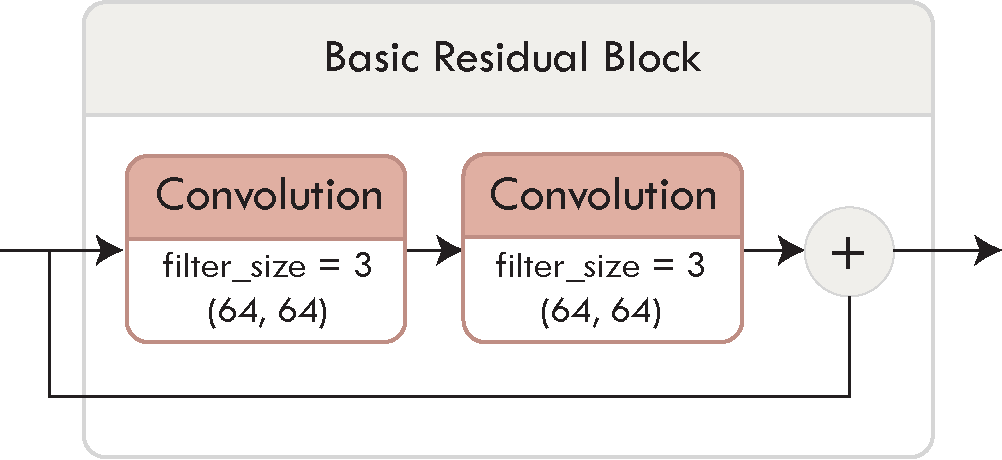


Figure 7. Basic Residual Block

1. **3-branch Residual Block:** *This configuration involves the use of multi-branch residual blocks with different filter size, enabling each branch to capture distinct information (Figure 8). The outputs from these paths are aggregated, and the result is finally combined with the shortcut. Having multiple branches enables the increase of network complexity without increasing the depth. The efficiency of learning temporal dependencies in neural networks is significantly influenced by the length of the pathways traversed by signals moving forward and backward within the network. In essence, the shorter the paths between the input and output sequences, the more effectively the network can capture and learn long-range dependencies in the data* [16].

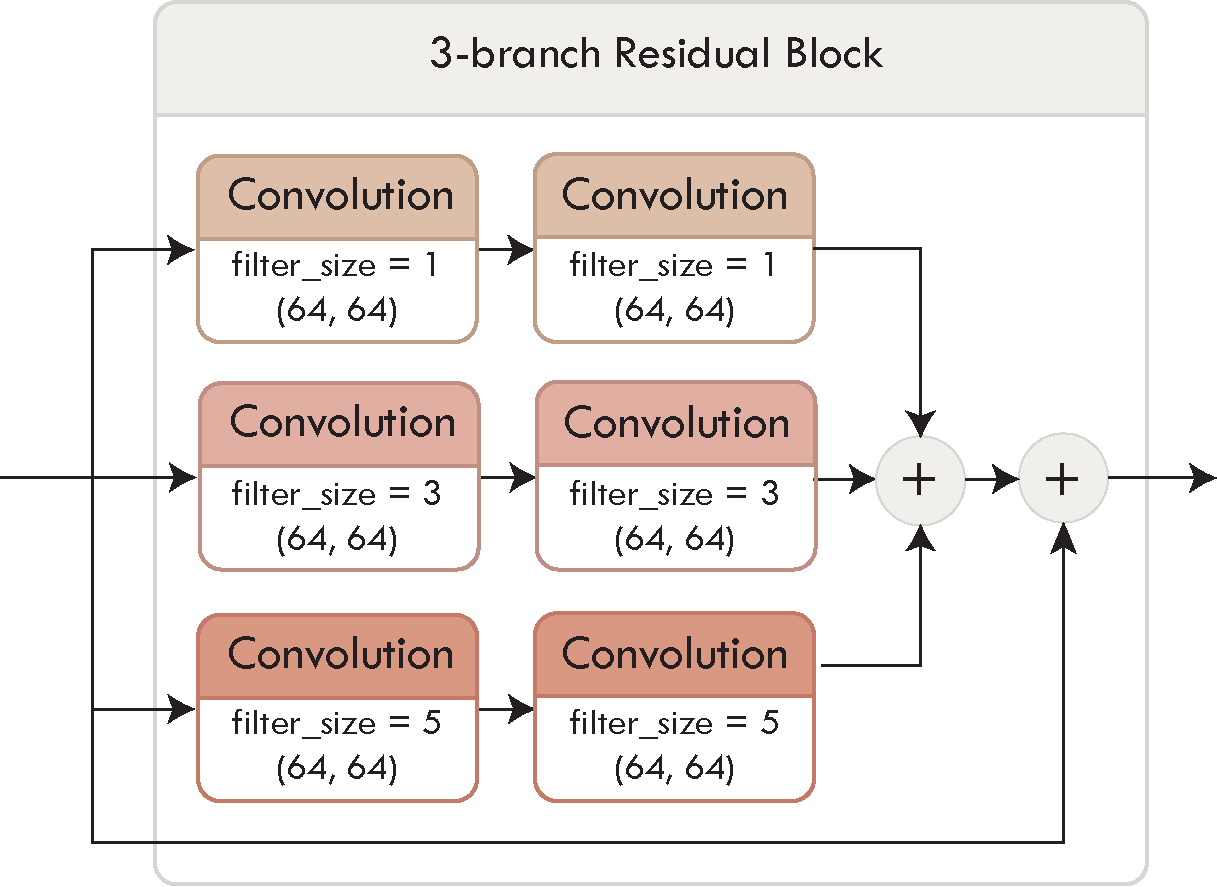


Figure 8. 3-branch Residual Block

1. **3-branch Residual Block with Bottleneck:** The third approach utilizes bottleneck-shaped architectures [14]. While having multiple branches within a Residual Block increases accuracy, this comes at the expense of increased network complexity. A strategy to condense information involves employing convolution layers prior to convolution layers with larger filter sizes [17]. Alternatively, these layers can functions as bottleneck layers, where they are placed both before and after layers with larger filter sizes [12], [18], as shown in Figure 9.

**Output Network Layers**

Before reaching the output layers, we incorporate an Adaptive Average Pooling layer that serves to reshape and streamline the data vector. Unlike traditional average pooling, which operates with fixed kernel sizes, the adaptive variant dynamically adjusts its kernel size based on the dimensions of the input data. This adaptability ensures that the output of the pooling layer is of a consistent size, regardless of the input's spatial dimensions.

The output network layers consist of: Dropout layer, Fully Connected layer and Softmax layer. For the Dropout layer, we employed a dropout probability of 0.2. This layer, along with weight decay, plays a vital role in the regularization process. Such techniques hold significance because neural networks sometimes tend to capture statistical noise present in the dataset, which can potentially result in overfitting. The Fully Connected layer is used for processing and combining the features extracted in earlier stages of the network. These layers allow the network to learn complex, non-local patterns and relationships within the data. In the context of classification tasks, we've incorporated a Softmax layer, which is commonly used in neural networks to transform the model's output into probability distributions over multiple classes. This enables us to make predictions with confidence scores for each class.

**Alternative LSTM architecture**

In addition to the presented Residual Blocks, we designed a LSTM-based network architecture following a similar pattern for input and output layers of the previous model. The design is rooted in the DeepConvLSTM framework [19], which is characterized by a sequence of four convolutional layers followed by two LSTM layers, as presented in Figure 10. The LSTM layers are arranged in a stacked configuration, where the output of one LSTM layer serves as the input to the next.   
Within each LSTM layer, there are M hidden units where we assessed the model's performance for M values of 128 and 256. These hidden units are responsible for learning and representing complex patterns and temporal dependencies within the input data. This stacking of LSTM layers facilitates the modeling of sequential data by gradually building a deeper understanding of both short-term and long-term dependencies.

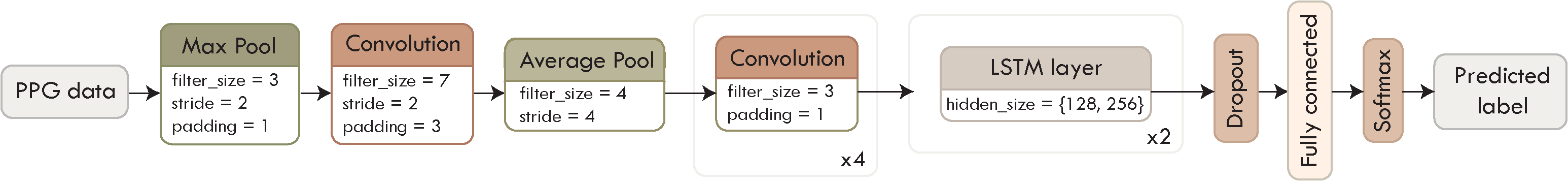


Figure 10. LSTM architecture

**Evaluation**

**Datasets**

For assessing the drivers’ stress level, we utilized two publicly available datasets.

**AffectiveROAD** dataset [20] contains 13 driving sessions performed by 10 distinct drivers. Each drive spanned approximately 85 minutes, incorporating a 30-minute rest interval. These routes include a variety of road types and environmental conditions, leading to varying degrees of stress. A human experimenter, situated in the back seat, subjectively estimated the stress levels using a laptop-based slider scale that ranged from 0 (no stress) to 1 (extremely high stress). After the drives, the drivers were asked to review and, if needed, change the stress scores based on their perception. Throughout the course of the experiment, participants wore two physiological monitoring devices: the Empatica E4 sensor on both wrists and the Zephyr Bioharness 3 on the chest. Using these sensors enabled the capture of many physiological signals, including Blood Volume Pulse (BVP), Electrodermal Activity (EDA), skin temperature (TEMP), and motion (ACC).

Additional evaluation was conducted using the multimodal dataset for Wearable Stress and Affect Detection **-** **WESAD** dataset [21]. The data was collected by using two recording devices: the RespiBAN Professional on the subject’s chest and the Empatica E4 on the wrist of the non-dominant hand. A total of 17 subjects participated in the study, but the data from two participants was excluded due to sensor malfunction. To begin, the participants’ baseline condition was recorded for about 20 minutes. During this initial period, they were sitting or standing while reading materials such as magazines. Subsequently, they viewed 11 video clips, lasting a total of 392 seconds, with the intention of inducing amusement. Following this, the subjects underwent the Trier Social Stress Test (TSST), which included delivering a speech in front of a three-person panel and performing a mental arithmetic task involving counting backward from 2023 to 0, in increments of 17. This phase, on average, spanned approximately 10 minutes. Following both the amusement and stress conditions, participants followed a guided meditation designed to help them return to the baseline state.

It is evident that both datasets utilize the Empatica E4 sensor which records PPG/BVP at a rate of 64 Hz. For the subsequent analyses, only the BVP data derived from the PPG signal, recorded by the Empatica E4 sensor was utilized. To ensure alignment between the BVP data and the label frequencies, we performed downsampling from 700 Hz (for the WESAD dataset) and upsampling from 4 Hz (for the AffectiveROAD dataset).

**Imbalanced data preprocessing**

After importing, the features of the data are standardized by removing the mean and scaling to unit variance, according to the following formula:

Here, *Xstd* represents the standardized data, *X* is the original data while *μ* and *σ* are the mean and standard deviation of the original data, respectively.

Given the substantial amount of data captured by wearable sensors, a segmentation process involving sliding windows is employed before feeding the data into the model. This segmentation involves dividing the data into fixed-length windows, where subsequent windows will have a certain degree of overlap. As demonstrated later in this paper, the evaluation outcomes are influenced by two key factors: the length of the sequence and the overlap ratio. Depending on the dataset in use, each sliding window is assigned a suitable label. The WESAD dataset already has labels for 4 different affective states (baseline, stress, amusement, meditation) for each time step. If all the labels inside a sliding window match, that is the resulting label, if they don’t that window will be discarded. On the other hand, AffectiveROAD has relative stress scores, as previously discussed, which is why a different label assignment approach is used. For each window, we calculate the average of the stress values contained within it and categorize it into one of three levels: low (0-0.4), medium (0.4 – 0.75), or high (0.75 – 1). This was done in the same manner as in [22]. It is assumed that the stress levels will be high during city driving, medium during highway driving and low during rest periods.

As depicted in Figure 11 the distribution of labels across classes exhibits some degree of imbalance, potentially leading to model bias favoring the majority class. The values depicted in the graphs correspond to the scenario where 8-second windows with a 75% overlap were used. To mitigate this issue, various methods can be employed, such as oversampling and undersampling. However, given that the utilized datasets do not exhibit extreme imbalances, our primary strategy was to apply Class Weights [REF], which can be calculated in the following manner:

where *N* represents the total number of samples, *Ni* is the number of samples in class *i* and *k* is the total number of classes.

This is a widely used technique that involves assigning distinct weights to classes based on their prevalence within the dataset. The objective is to enhance the significance of the minority class while diminishing the impact of the majority class during the model's training process. These class weights are seamlessly integrated into the model's loss function during training. Each class's weight is inversely proportional to its frequency in the dataset, resulting in higher weights being assigned to the minority classes.

**Implementation details**

We used the Adam optimizer with 128 samples in a mini-batch. The learning rate was initially set to 0.001 and was dynamically adjusted using a scheduler, decreasing it to 85% of its value every 3 epochs. The training spanned a total of 30 epochs.

Each of the convolutional layers comes with Batch Normalization to enhance training stability. These layers play a crucial role in mitigating challenges such as exploding and vanishing gradients, as well as reducing the potential of the network to get trapped in unfavorable local minima [18]. This is especially important when training very deep networks, as the variables in intermediate layers may take values with substantially varying magnitudes. Another benefit of utilizing Batch Normalization is that it reduces the training time. Moreover, it has a minor influence on regularization, which contributes to reducing the risk of overfitting [8]. To introduce non-linearity and enable the network to learn complex patterns, we employ the Rectified Linear Unit (ReLU) activation function.

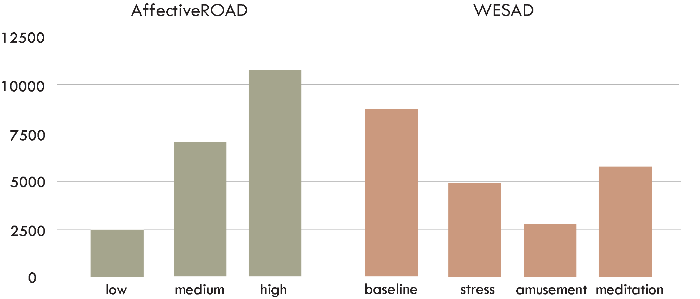


Figure 11. Class distribution for AffectiveROAD and WESAD datasets

For the experiments, we mostly employed 5-fold cross-validation which splits the data into five subsets and iteratively trains and validates the model using different combinations of these subsets, leading to more reliable results. To maintain result consistency across multiple runs and to account for potential sources of variability, we fixed a constant seed value, thus mitigating the effects of dataset split randomness and other factors.

**Experimental Results**

**Sliding window size and overlap**

In our initial evaluation process, to find the optimal sliding window size, we measured performance across a range of window sizes in range from 4 to 16 seconds. We set the overlap between consecutive sliding windows to 75%. These results are summarized in Table 1 for the AffectiveROAD dataset, and Table 2 for the WESAD dataset.

Based on the results, we selected a window size of 8 seconds and analyzed the impact of changing the overlap ratio on the results. Specifically, we examined overlap ratios of 50%, 75% and 87.5%. We chose these values because all the data within 50% overlap windows is also contained within 75% overlap windows, and similarly, 75% overlap windows are a subset of 87.5% overlap windows. As can be seen in Tables *3* and 4, for both datasets and across all utilized architectures, it is evident that a higher overlap ratio consistently yields better results because of a greater number of training samples.

Another notable observation is that in certain cases, the ResNet with 7 Residual Blocks performed less effectively than the one with 5 blocks. This highlights a common challenge encountered when increasing network depth: as neural networks become deeper, they become increasingly susceptible to overfitting. Overfitting implies that while the models excel in fitting the training data, they may struggle to generalize to new, unseen data [8]. This occurs because deeper networks have more parameters that can potentially capture noise within the training dataset.

When considering the alternative LSTM architecture, in most cases, increasing the number of hidden units within LSTM layers leads to improved performance, but it also increases computational demands and time requirements. This performance analysis highlights a notable superiority of ResNets over LSTM networks, with ResNets consistently delivering better results. It’s worth noting that training recurrent networks, such as LSTMs, tends to be considerably slower compared to CNNs. This is because RNNs maintain hidden states that need to be updated at each time step, and this operation is more computationally expensive than the simple convolutional operations performed by CNNs.

For the AffectiveROAD dataset, we also investigated how performance metrics change when using data from only one of the wrists for the scenario of 8-second windows with 75% overlap. This comparison is presented in Table 5. As evident, the models demonstrate decreased performance when provided with less information. Interestingly, the utilization of data from the left wrist provides better results compared to the right wrist.

Table 1. AffectiveROAD dataset – comparison of different window sizes when the overlap ratios is fixed at 75%

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Affective**  **ROAD** | 4-second | | 8-second | | 12-second | | 16-second | |
| Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 |
| ResNet (3 blocks) | 76.45 | 75.15 | 86.9 | 86.16 | 86.09 | 83.96 | 87.88 | 87.26 |
| ResNet (5 blocks) | 81.07 | 79.3 | 91.66 | 91 | 93.04 | 92.43 | 92.25 | 92.16 |
| ResNet (7 blocks) | 82.99 | 81.2 | 91.61 | 90.75 | 91.46 | 90.73 | 93.82 | 93.93 |
| LSTM (128 units) | 70.01 | 68.78 | 78.73 | 77.33 | 66.14 | 66.12 | 75.32 | 73.05 |
| LSTM (256 units) | 69.59 | 67.21 | 77.38 | 75.48 | 74.2 | 71.1 | 62.61 | 60.03 |

Table 2. WESAD dataset - comparison of different window sizes when the overlap ratios is fixed at 75%

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WESAD** | 4-second | | 8-second | | 12-second | | 16-second | |
| Acc. | F1 | Acc. | F1 | Acc. | F1 | Acc. | F1 |
| ResNet (3 blocks) | 86.75 | 85.97 | 90.7 | 90.31 | 93.09 | 92.9 | 96.55 | 93.16 |
| ResNet (5 blocks) | 90.07 | 89.5 | 95.82 | 95.31 | 97 | 96.63 | 96.08 | 95.64 |
| ResNet (7 blocks) | 89.16 | 88.78 | 96.04 | 95.84 | 97.46 | 97.45 | 97.64 | 97.51 |
| LSTM (128 units) | 85.91 | 84.93 | 88.9 | 88.02 | 88.76 | 87.36 | 73.8 | 77.22 |
| LSTM (256 units) | 84.76 | 83.87 | 89.76 | 88.56 | 89.66 | 88.62 | 86.92 | 85.82 |

Table 3. AffectiveROAD dataset - effects of modifying window size when windows size is fixed at 8 seconds

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **AffectiveROAD** | 50% overlap | | 75% overlap | | 87.5% overlap | |
| Acc. | F1 | Acc. | F1 | Acc. | F1 |
| ResNet (3 blocks) | 70.61 | 66.72 | 86.9 | 86.16 | 95.03 | 94.81 |
| ResNet (5 blocks) | 74.68 | 70.85 | 91.66 | 91 | 96.86 | 96.55 |
| ResNet (7 blocks) | 74.63 | 69.67 | 91.61 | 90.75 | 97.47 | 97.33 |
| LSTM (128 units) | 55.89 | 51.21 | 78.73 | 77.33 | 90.85 | 90.39 |
| LSTM (256 units) | 58.39 | 53.92 | 77.38 | 75.48 | 91.36 | 90.87 |

Table 4. WESAD dataset - effects of modifying window size when windows size is fixed at 8 seconds

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **WESAD** | 50% overlap | | 75% overlap | | 87.5% overlap | |
| Acc. | F1 | Acc. | F1 | Acc. | F1 |
| ResNet (3 blocks) | 77.49 | 75.79 | 90.7 | 90.31 | 98.24 | 98.15 |
| ResNet (5 blocks) | 85.54 | 84.32 | 95.82 | 95.31 | 99.18 | 99.2 |
| ResNet (7 blocks) | 84.43 | 83.42 | 96.04 | 95.84 | 99.24 | 99.19 |
| LSTM (128 units) | 76.66 | 74.6 | 88.9 | 88.02 | 97.02 | 96.8 |
| LSTM (256 units) | 77.71 | 75.44 | 89.76 | 88.56 | 97.71 | 97.5 |

Table 5. AffectiveROAD dataset - results when using BVP data from one wrist only (8 second windows with 75% overlap)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **AffectiveROAD** | Left wrist | | Right wrist | |
| Acc. | F1 | Acc. | F1 |
| ResNet (3 blocks) | 81.26 | 79.68 | 79.29 | 77.55 |
| ResNet (5 blocks) | 88.13 | 86.72 | 83.29 | 81.79 |
| ResNet (7 blocks) | 88.37 | 87.48 | 86.19 | 85.2 |
| LSTM (128 units) | 62.22 | 61.07 | 62.88 | 60.54 |
| LSTM (256 units) | 68.08 | 65.25 | 63 | 59.53 |

**Performance of different Multi-branch Residual Blocks**

Based on the previously presented results, we concluded that the ResNet architecture with 5 Residual Blocks delivers the optimal performance, surpassing the model with 3 blocks and often achieving results on par with the model with 7 blocks. In our subsequent evaluation, we explored variations of this network, which involved altering the Residual Blocks. The ideas behind the modifications were detailed earlier, we explored different options with multiple paths employing various filter sizes: (1, 3, 5), (3, 5, 7) and (3, 5).

Our primary focus centered on the 8-second windows with 87.5% overlap. However, we also conducted an evaluation for 75% overlap, as it tends to reveal more pronounced differences in the results. As presented in Tables 6 and 7, these variations generally outperform standard ResNet.

Table 6. AffectiveROAD dataset – achieved results with ResNet variations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Affective**  **ROAD** | 75% overlap | | | | 87.5% overlap | | | |
| Acc. | F1 | Prec. | Rec. | Acc. | F1 | Prec. | Rec. |
| Basic  ResNet | 89.11 | 87.95 | 88.81 | 87.53 | 97.37 | 97.19 | 97.08 | 97.31 |
| 1-3-5  ResNet | 92.53 | 91.86 | 92.26 | 91.66 | 98.28 | 98.17 | 98.11 | 98.24 |
| 3-5-7  ResNet | 90.2 | 88.98 | 89.38 | 88.73 | 97.54 | 97.37 | 97.31 | 97.45 |
| 3-5  ResNet | 90.69 | 89.86 | 90.59 | 89.4 | 97.46 | 97.22 | 97.09 | 97.36 |
| Bottleneck | 92.31 | 91.52 | 91.41 | 91.74 | 97.61 | 97.48 | 97.41 | 97.56 |

Table 7. WESAD dataset – achieved results with ResNet variations

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **WESAD** | 75% overlap | | | | 87.5% overlap | | | |
| Acc. | F1 | Prec. | Rec. | Acc. | F1 | Prec. | Rec. |
| Basic  ResNet | 96.2 | 95.92 | 95.76 | 96.11 | 99.4 | 99.32 | 99.3 | 99.33 |
| 1-3-5  ResNet | 97.14 | 96.96 | 96.69 | 97.27 | 99.45 | 99.4 | 99.36 | 99.45 |
| 3-5-7  ResNet | 95.62 | 95.23 | 95.16 | 95.34 | 99.41 | 99.36 | 99.35 | 99.38 |
| 3-5  ResNet | 96.58 | 96.23 | 96.08 | 96.42 | 99.17 | 99.08 | 99.06 | 99.17 |
| Bottleneck | 96.31 | 96.03 | 95.81 | 96.29 | 99.17 | 99.13 | 99.04 | 99.22 |

**Comparisons with state-of-the-art results**

To assess the performance of our models, we conducted comparisons with several prominent methods, specifically considering the achieved accuracies. Although various approaches have been utilized on both of the test datasets, our primary focus was on the ones utilizing Deep Neural Networks, or exclusively relying on the BVP signal. Table 8 and 9 provide a summary of the results attained through the previously mentioned methods, alongside our own results.

Table 8. Comparison of our model with prominent works – AffectiveROAD dataset

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Utilized signals |
| SVM (rbf kernel) [23] | 70 | EDA + HR |
| RBF (L2 regularization) [23] | 83 |
| RF [24] | 70.4 | BVP |
| DNN [25] | 87.95 | All E4 + BioHarness data |
| StRessNet (our) | 98.28 | BVP |

Table 9. Comparison of our model with prominent works - WESAD dataset

|  |  |  |
| --- | --- | --- |
| Method | Accuracy | Utilized signals |
| Self-supervised encoder network [26] | 78.13 | ACC + BVP + EDA + TEMP |
| DNN with 3 branches [27] | 89.94 | BVP |
| RF [28] | 81.4 | BVP |
| DNN [29] | 93.64 | ECG + EDA + EMG + RESP + TEMP |
| StRessNet (our) | 99.45 | BVP |

Starting with the AffectiveROAD dataset, the Deep Learning architecture employed in [25] consist of four convolution layers with varying numbers of filters (8, 32, 64, and 128) followed by a Global Average Pooling layer and a FC layer. The results were further enhanced by adding an LSTM layer after the FC layer. Here, all the E4 and BioHarness data was used.

Evaluation for both regression and classification was conducted in [24]. For the classification task, the data was divided into two categories: baseline and driving state. Feature extraction was performed, and the Random Forest algorithm was utilized for the classification task.

A non-neural network approach was used in [23] with the goal of differentiating between low (L) and high (H) stress classes. The drives were grouped into different profiles based on the driver’s physiological responses. Drive descriptors were created by calculating the mean of the features for all instances labeled as H. For clustering into different profiles, normalized spectral clustering was used. The signals from which the features were extracted were EDA and HR.

Moving on to the WESAD dataset, one of the methods employed a Self-Supervised Learning approach [26]. Here, an encoder was trained using distinct representations for the following signals: ACC, BVP, EDA and TEMP. The architecture was inspired by Google’s Inception [17], featuring blocks with multiple parallel layers whose outputs are stacked to form the final output.

Another neural network-based method [27] involved extracting features from BPV, EDA and ST signals, resulting in a total of 72 features. The architecture incorporated separate branches for each signal and a concatenating layer to learn shared encoded features. The overall loss was defined as the sum of losses from all branches. It’s worth noting that this approach used binary classification distinguishing between the stress state and non-stress states, which included baseline and amusement.

A relatively high accuracy was achieved in [29]. This was done by utilizing the following signals: EDA, EMG, RESP and TEMP. Each of these signals was individually processed through a dedicated convolution block. The outputs from these blocks were then concatenated and passed through Fully Connected (FC) layers. Each of these blocks featured a convolution layer followed by a Max Pooling layer, and this sequence was repeated three times.

Lastly, in [28], three distinct classifiers were employed. When focusing solely on the BVP signal, the Random Forest classifier delivered the highest performance. Furthermore, this study delved into the impact of altering window sizes. The findings indicate that longer window sizes (up to 120 seconds) result in improved performance. Our choice for shorter window sizes was primarily motivated by the rapid changes that can occur in driving conditions, demanding a more responsive approach to stress detection.

**Complexity analyses**

While the altered ResNet models generally achieve better accuracy than the Basic ResNet, this comes at the cost of computational complexity. Table 10 shows that these variations exhibit slower computational speeds. Notably, the Bottleneck structure displays a compelling balance between speed and accuracy. For further insight into the model’s complexity, Table 11 provides the parameter count for each model and a comparison against the Basic ResNet.

Table 10. Relative training and inference times for ResNet variations

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Training | Inference |
| Basic ResNet |  | 1.00 | 1.00 |
| 1-3-5 ResNet |  | 1.82 | 1.73 |
| 3-5-7 ResNet |  | 2.68 | 2.50 |
| 3-5 ResNet |  | 1.61 | 1.48 |
| Bottleneck |  | 1.13 | 1.21 |

Table 11. Number of parameters of ResNet variations compared with the Basic ResNet model

|  |  |  |
| --- | --- | --- |
|  | Number of parameters | Ratio |
| Basic ResNet | 126083 | 1.00 |
| 1-3-5 ResNet | 293763 | 2.33 |
| 3-5-7 ResNet | 539523 | 4.28 |
| 3-5 ResNet | 250883 | 1.99 |
| Bottleneck | 108483 | 0.86 |

**Conclusion and discussion**

The study presented in this paper...

**Acknowledgement**

This research was supported by the Science Fund of the Republic of Serbia, project XAI4HEAT, grant no. XXXXXX.

**References**

[1] G. Vos, K. Trinh, Z. Sarnyai, and M. Rahimi Azghadi, “Generalizable machine learning for stress monitoring from wearable devices: A systematic literature review,” *Int. J. Med. Inf.*, vol. 173, p. 105026, May 2023, doi: 10.1016/j.ijmedinf.2023.105026.

[2] K. Avramidis, T. Feng, D. Bose, and S. Narayanan, “Multimodal Estimation of Change Points of Physiological Arousal in Drivers.” arXiv, Oct. 27, 2022. Accessed: Nov. 25, 2023. [Online]. Available: http://arxiv.org/abs/2210.15826

[3] A. Nemcova *et al.*, “Multimodal Features for Detection of Driver Stress and Fatigue: Review,” *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3214–3233, Jun. 2021, doi: 10.1109/TITS.2020.2977762.

[4] J. E. Sinex, “Pulse oximetry: Principles and limitations,” *Am. J. Emerg. Med.*, vol. 17, no. 1, pp. 59–66, Jan. 1999, doi: 10.1016/S0735-6757(99)90019-0.

[5] M. N. Rastgoo, B. Nakisa, A. Rakotonirainy, V. Chandran, and D. Tjondronegoro, “A Critical Review of Proactive Detection of Driver Stress Levels Based on Multimodal Measurements,” *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–35, Sep. 2019, doi: 10.1145/3186585.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[7] D. W. Otter, J. R. Medina, and J. K. Kalita, “A Survey of the Usages of Deep Learning in Natural Language Processing.” arXiv, Dec. 21, 2019. Accessed: Nov. 18, 2023. [Online]. Available: http://arxiv.org/abs/1807.10854

[8] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J. Big Data*, vol. 8, no. 1, p. 53, Mar. 2021, doi: 10.1186/s40537-021-00444-8.

[9] N. M. Foumani, L. Miller, C. W. Tan, G. I. Webb, G. Forestier, and M. Salehi, “Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey.” arXiv, Feb. 05, 2023. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/2302.02515

[10] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks.” arXiv, Feb. 15, 2013. Accessed: Nov. 18, 2023. [Online]. Available: http://arxiv.org/abs/1211.5063

[11] H. Hewamalage, C. Bergmeir, and K. Bandara, “Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions,” *Int. J. Forecast.*, vol. 37, no. 1, pp. 388–427, Jan. 2021, doi: 10.1016/j.ijforecast.2020.06.008.

[12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778. doi: 10.1109/CVPR.2016.90.

[13] E. Mejía-Mejía, J. Allen, K. Budidha, C. El-Hajj, P. A. Kyriacou, and P. H. Charlton, “Photoplethysmography signal processing and synthesis,” in *Photoplethysmography*, Elsevier, 2022, pp. 69–146. doi: 10.1016/B978-0-12-823374-0.00015-3.

[14] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, “Aggregated Residual Transformations for Deep Neural Networks.” arXiv, Apr. 10, 2017. Accessed: Dec. 06, 2023. [Online]. Available: http://arxiv.org/abs/1611.05431

[15] F. Bieder, R. Sandkühler, and P. C. Cattin, “Comparison of Methods Generalizing Max- and Average-Pooling,” 2021, doi: 10.48550/ARXIV.2103.01746.

[16] A. Vaswani *et al.*, “Attention Is All You Need.” arXiv, Aug. 01, 2023. Accessed: Nov. 05, 2023. [Online]. Available: http://arxiv.org/abs/1706.03762

[17] C. Szegedy *et al.*, “Going Deeper with Convolutions.” arXiv, Sep. 16, 2014. Accessed: Oct. 26, 2023. [Online]. Available: http://arxiv.org/abs/1409.4842

[18] H. I. Fawaz *et al.*, “InceptionTime: Finding AlexNet for Time Series Classification,” *Data Min. Knowl. Discov.*, vol. 34, no. 6, pp. 1936–1962, Nov. 2020, doi: 10.1007/s10618-020-00710-y.

[19] F. Ordóñez and D. Roggen, “Deep Convolutional and LSTM Recurrent Neural Networks for Multimodal Wearable Activity Recognition,” *Sensors*, vol. 16, no. 1, p. 115, Jan. 2016, doi: 10.3390/s16010115.

[20] N. E. Haouij, J.-M. Poggi, S. Sevestre-Ghalila, R. Ghozi, and M. Jaïdane, “AffectiveROAD system and database to assess driver’s attention,” in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, Pau France: ACM, Apr. 2018, pp. 800–803. doi: 10.1145/3167132.3167395.

[21] P. Schmidt, A. Reiss, R. Duerichen, C. Marberger, and K. Van Laerhoven, “Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection,” in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, Boulder CO USA: ACM, Oct. 2018, pp. 400–408. doi: 10.1145/3242969.3242985.

[22] C. Bustos, N. Elhaouij, A. Sole-Ribalta, J. Borge-Holthoefer, A. Lapedriza, and R. Picard, “Predicting Driver Self-Reported Stress by Analyzing the Road Scene,” in *2021 9th International Conference on Affective Computing and Intelligent Interaction (ACII)*, Nara, Japan: IEEE, Sep. 2021, pp. 1–8. doi: 10.1109/ACII52823.2021.9597438.

[23] D. Lopez-Martinez, N. El-Haouij, and R. Picard, “Detection of Real-world Driving-induced Affective State Using Physiological Signals and Multi-view Multi-task Machine Learning.” arXiv, Jul. 19, 2019. Accessed: Oct. 28, 2023. [Online]. Available: http://arxiv.org/abs/1907.09929

[24] P. Siirtola and J. Röning, “Comparison of Regression and Classification Models for User-Independent and Personal Stress Detection,” *Sensors*, vol. 20, no. 16, p. 4402, Aug. 2020, doi: 10.3390/s20164402.

[25] M. Amin, K. Ullah, M. Asif, H. Shah, A. Mehmood, and M. A. Khan, “Real-World Driver Stress Recognition and Diagnosis Based on Multimodal Deep Learning and Fuzzy EDAS Approaches,” *Diagnostics*, vol. 13, no. 11, p. 1897, May 2023, doi: 10.3390/diagnostics13111897.

[26] V. Dissanayake, S. Seneviratne, R. Rana, E. Wen, T. Kaluarachchi, and S. Nanayakkara, “SigRep: Toward Robust Wearable Emotion Recognition With Contrastive Representation Learning,” *IEEE Access*, vol. 10, pp. 18105–18120, 2022, doi: 10.1109/ACCESS.2022.3149509.

[27] V.-T. Ninh *et al.*, “An Improved Subject-Independent Stress Detection Model Applied to Consumer-grade Wearable Devices.” arXiv, Mar. 17, 2022. Accessed: Oct. 26, 2023. [Online]. Available: http://arxiv.org/abs/2203.09663

[28] P. Siirtola, “Continuous stress detection using the sensors of commercial smartwatch,” in *Adjunct Proceedings of the 2019 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2019 ACM International Symposium on Wearable Computers*, London United Kingdom: ACM, Sep. 2019, pp. 1198–1201. doi: 10.1145/3341162.3344831.

[29] R. Li and Z. Liu, “Stress detection using deep neural networks,” *BMC Med. Inform. Decis. Mak.*, vol. 20, no. S11, p. 285, Dec. 2020, doi: 10.1186/s12911-020-01299-4.