# Scalable Deep Learning for Pilot Performance Analysis Using Multimodal Physiological Time Series

Noah Lee[1,⋆], Patrick W. Moore[2,⋆], Laura J. Brattain[1]
[1]Human Health & Performance Systems, MIT Lincoln Laboratory, Lexington, MA, USA
[2]DAF AI Accelerator, Cambridge, MA, USA
{noah.lee, brattainl}@ll.mit.edu
{p_moore}@mit.edu

***Abstract*—Sensors used to collect human physiological data often necessitate the processing and classification of time series data, which can quickly become intractable with very lengthy inputs or many time series features. In this study we compared the performance of two methods of time series feature extraction and dimensionality reduction, Minimally Random Convolutional Kernel Transform (MiniRocket) and statistical feature engineering using TSFresh, to determine the optimal hardware configurations and associated performance-accuracy trade-offs between model speed and complexity. Our results showed that MiniRocket scales extremely well with only linear complexity while the scaling of TSFresh is dependent on the set of features selected for computation. Further, MiniRocket outperformed the TSFresh model accuracy for all configurations except the most comprehensive (but slowest) feature extraction set thereby highlighting MiniRocket as a great all-purpose dimensionality reduction tool for human physiological time series data.**

***Index Terms*—Artifical Intelligence, dimentionality reduction, Multimodality, MiniRocket, Physiology, Scalability, TSFresh**

## I. Introduction

High global demand for experienced pilots has led to a critical deficit of aviators, both within the civilian and government organizations in the United States [1] and is expected to lead to a severe pilot shortage by 2037 [2]. The reasons for the shortage in the United States range from Federal Aviation Authority (FAA) regulations (forced retirement at age 65) to an explosion of commercial use of airlines due to e-commerce expansion in the last two decades [3].

Adequate training of new pilots is of critical importance but has come at the expense of long training pipelines and rigorous practical testing before accession into the piloting career field. Further, according to the 2022 FAA report of U.S. civil airman statistics, the pass rate on FAA practical tests decreased on average for all types of pilots for the first time since 2012 [4]. Further, the RAND corporation identified a critical pilot shortage in the US military as early as 2000 and projected the retention of experienced pilots and the lengthy certification process to be the primary reasons for continued future shortages [5].

While the crisis extends from initial training to experienced pilot retention, the focus of this paper is on improving the efficacy and throughput of the pilot training pipeline so support the burgeoning demand of both commercial and military aviators. While the Air Force leadership has recently emphasized flight simulators to modernize pilot training [6] as a way to reduce the material cost of training flights during the 55 week long course, objective evaluation of pilot performance has not yet been adequately established. While previous research conducted by the U.S. Department of the Air Force and Massachusetts Institute of Technology/Lincoln Laboratory have demonstrated promising gains toward such objective assessments, a comparison of both the methods and the associated performance and computing demands is warranted [7] [8] [9].

Our approach to evaluating these objective measures of performance was to choose two distinct methods and compare both accuracy and computational expense to generate inferences using only human physiological data gathered from the pilot during the simulated flight. The first method, MiniRocket, applies 1-dimensional convolutions to the time series for dimension reduction and feature extraction over very high-dimensional spaces [10]. The second method calculates a large number of time series statistics in order to characterize the observations. Both methods were then passed into various classifier models to predict the difficulty of the flight run and the accuracy was measured against the objective difficulty level [11]. We then compared the relative computational expense of the two methods and discuss the trade-offs between accuracy and performance, as well as practical use-cases for such models.

Providing personalized feedback using Artificial Intelligence (AI) tools can help reduce the need for human instructors . Previous work has been done to equip simulators with personalized AI feedback; Yang et al. [12] described

*Equal contributions from authors

Research was sponsored by the United States Air Force Research Laboratory and the United States Air Force Artificial Intelligence Accelerator and was accomplished under Cooperative Agreement Number FA8750-19-2-1000. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the United States Air Force or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

TABLE I
DESCRIPTION OF DATA ANALYZED

| Modality | Location | Facet | Unit | Sampling Rate (Hz) |
|---|---|---|---|---|
| Accelerometry | Right forearm | x-axis | m/s^2 | 128 |
| | | y-axis | m/s^2 | 128 |
| | | z-axis | m/s^2 | 128 |
| | Torso | x-axis | m/s^2 | 128 |
| | | y-axis | m/s^2 | 128 |
| | | z-axis | m/s^2 | 128 |
| Electrocardiogram | Chest | LA-RA | mV | 512 |
| | | LL-RA | mV | 512 |
| | | Vx-RL | mV | 512 |
| | | Respiration | mv | 512 |
| Electromyography | Right wrist | Extensor | mV | 128 |
| | | Flexor | mV | 128 |
| Electrodermography | Left hand | | kOhms | 1024 |
| Photoplethysmography | Left hand | | mV | 1024 |
| Plethysmography | Diaphragm | | V | 1024 |

a conceptual model to design an ML system for real-time feedback to pilots, and Guevarra et al. [13] implements a reinforcement learning agent to first learn flight manuevers from instructors, then pass this knowledge on to student pilots. For these works, the systems make use of exact plane position data generated by the simulator. However, some of the simulator systems, including those used by DAF Specialized Undergraduate Pilot Training (SUPT) program, are legacy systems, thus integrating AI teachers into and reading position data from the simulators can be overly arduous.

A more flexible system design would avoid needing to integrate with pre-existing flight simulators. Bineas et al. [14] confirmed that it is possible to detect unexpected events via EEG measurements, leading to the possibility of an AI system that provides feedback based directly on physiological readings taken from the pilot. Such a system wouldn't need to interface with the simulator, and could even be deployed inside real trainer jets.

The CogPilot dataset [11] was collected to facilitate the development of physiology-informed machine learning models to monitor flight performance in tasks of varying difficulty. The CogPilot team so far has run two competitions among teams of university researchers to find the best performing models. Two additional works have been published based on the CogPilot dataset by Powell [15] and Cabellero et al. [16].

Our work extends upon prior work done on the CogPilot dataset. We analyze the degradation in performance of models under different sampling frequencies, and detect artifacts present in the different modalities of the CogPilot dataset. We also evaluate two different featurizing transforms: MiniRocket [10], a modern deep-learning inspired approach and TSFresh [17], which features traditional time series feature calculators. Finally, we introduce a scalable pipeline to evaluate hundreds of hyperparameter combinations on any distributed computing environment using Dask [18], Optuna [19], and the Scikit-Learn API [20].

## II. MATERIALS AND METHODS

### A. High performance computing environment

The computation for this paper was performed on the high performance computing (HPC) system provided by MIT Supercloud [21], where we were able to utilize nodes equipped with Intel Xeon Platinum 8260 CPUs (48 total cores) with 192 GB of RAM. The high memory capacity was especially critical to our research, since resampling our dataset to 1024Hz creates a 23.5 GB object in memory. We run jobs through a SLURM workload manager [22] and use 4 nodes for each job, out of a maximum of 8 nodes for a base user.

### B. Dataset

The dataset used by this paper is a subset of the physiological data collected by the DAF-MIT CogPilot team [11]. All subjects volunteered to participate in the study and gave written informed consent under a protocol approved by the MIT Committee on the Use of Humans as Experimental Subjects. The study involved 35 subjects varying between zero hours and 3000+ hours of fixed-wing flight experience who each attempted 12 simulated landings of an airplane under 4 different difficulty levels.

The subset used in this paper excluded the eye tracking and head movement modalities which have previously been explored [8]. Table I describes the six modalities studied, which include electrocardiography, respiration, accelerometry, electromyography, electrodermal activity, and photoplethysmography, from which 14 distinct features were derived.

### C. Preprocessing

As with many multimodal sensor studies, preparation of the dataset was carefully attended to ensure quality and unbiased data was eventually feed to the classifier models. Across the six separate modalities included in the data, three distinct nominal sampling rates were found (128Hz, 512Hz, 1024Hz). The first preprocessing step required aligning all time series observations to a common start time through timestamp conversion and interpolation where necessary.

Once the time series observations had a common starting timestamp, the various sampling rates were rectified through aggregation, where necessary. The decision to resample the time series to the greatest common factor of 128Hz or less was made to allow aggregation of time elements rather than interpolation, which also assisted in filling random missing values due to sensor fault.

Outlier analysis revealed that nearly all time series observations began and ended with significant noise and artifacts. This was most pronounced in the accelerometry data, which suggested that the subjects were likely adjusting their position for comfort early on or were reacting to stimulus outside the simulator (e.g., instruction from the research administrator) both directly before and after the flight run.

Further, the variability across subjects for the time required to complete the exact same runs suggested that some flights ended prematurely compared to the expected time for completion, while others lingered on for nearly twice as long.

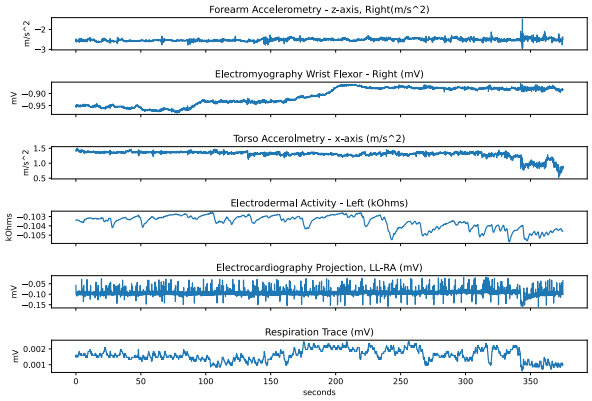


Fig. 1. Example time series data by modality

This was confirmed by administrators to be scenarios where subjects crashed early in the flight or became disoriented and were unable to find their way back to the runway for a successful landing.

Clipping the time series to between 25 seconds and 470 seconds was done to eliminate the "settling" period at the beginning of each run as well as terminate the observation beyond a reasonable time to land the plane. For those subjects who crashed prior to the expected time to land, the series was padded with the last valid value to ensure uniform length of the time series across all observations (a requirement of MiniRocket Multivariate).

### D. MiniRocket

Minimally Random Convolutional Kernel Transform, or MiniRocket, is a transform for time series data that handles both univariate or multivariate data [10]. In the case of univariate data, each time series observation of size $m \times n$ is transformed with a 1-dimensional convolution using a kernel of length 9. Biases are randomly sampled from the output so that all convolutions are of the same relative scale, thereby eliminating the need to normalize the data before application of the MiniRocket method. After each observation is convoluted, it is further distilled as the proportion of positive values in the output. Hence, one value from the interval [0,1] is returned for each input observation and each of $k$ kernels, resulting in a new transformed dataset of size $m \times k$.

For multivariate datasets, the same process is applied except for each observation all time series features are padded to the same length (length $n$) and concatenated together before the convolution. This provides additional benefits as the original data is reduced from $m \times n \times t$ (where $t$ is the number of time series features) to size $m \times k$. Common values for $k$ are typically between 5000 and 30000 kernels, with more complex relationships likely captured for greater kernel samplings. For our experimentation, 5000 kernels were used for each model and each of the 14 unique time series features was clipped between to the interval [25, 470] seconds. Thus, for a given sampling rate $h$, the dataset was transformed from $m \times 445 * h \times 14$ to $m \times 5000$.

### E. TSFresh

TSFresh (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests) [17] is a Python package for time series transformation that we explore as an alternative to MiniRocket. TSFresh automates the feature extraction process with 63 different time series characterization methods, resulting in 794 total computed time series features from a single input feature. The package also automatically performs feature selection using the FRESH [23] algorithm on these generated features, to mitigate the combinatorial explosion of features.

TSFresh implements the Scikit-Learn API [20], and works solely with Pandas DataFrames [24]. This makes the library widely compatible with most of the Python machine learning ecosystem. For production environments, TSFresh is highly parallelizable to enable high throughput. On a single machine, it uses the Python module `multiprocessing` and in a distributed environment, it uses Dask [18].

The reason we choose to use TSFresh as a comparison to MiniRocket is the flexibility it provides for choosing time series characteristics. For example, the summary statistics computed in [16] are all available as feature calculators in TSFresh. We choose to use the three default feature sets in TSFresh: minimal, efficient, and comprehensive. The minimal set of feature calculators has the lowest compute requirement but the lowest expressivity, whereas the comprehensive feature set takes the longest to compute, but is expected to lead to the most accurate classifier

### F. Lazy Dataset Pre-processing Pipeline

One of the problems faced while pre-processing the dataset was running out of RAM on the node. The CogPilot team used the lab streaming layer [25] to synchronize data streams from multiple sensors, yet when all of the data streams were collected into a single DataFrame without merging by timestamp, the RAM consumption of the complete frame ballooned to 140+ GB. An exclusive node on the MIT SuperCloud can load the DataFrame since it has a memory limit of 192 GB, but it is difficult to perform operations on this DataFrame because Pandas has adopted a copy-only programming model. We turned to Dask [18] as a solution to merge all of the data into a single DataFrame and resample it without running out of memory on the node. Dask processes data lazily, and optimizes computations so that the memory limit is never exceeded. As an added bonus, Dask is able to perform the entire process of loading all of the CSV files, merging them, and resampling the DataFrame within 5 minutes. Further optimizations include storing datasets as HDF5 [26] files and compressing them using the LZO [27] compressor for faster read/write from disk.

## III. Experimental Results

### A. Computational Performance

The comparison of computational performance between TSFresh and MiniRocket methods was done both by the relative size of the time series observation and the number of CPU cores available to process the data. During the preprocessing step, each time series observation was clipped to the interval

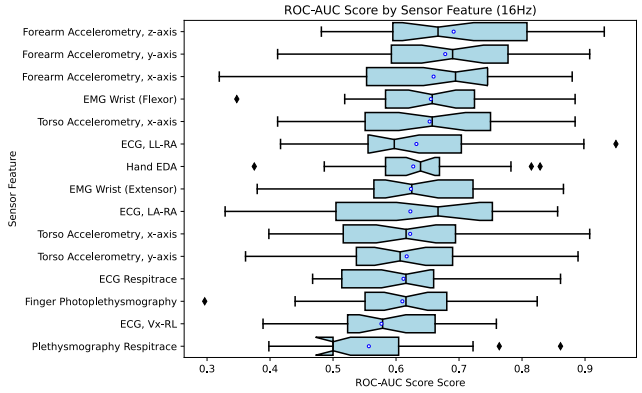


Fig. 2. Receiver Operating Characteristic Area Under the Curve for model built with 16Hz downsample data

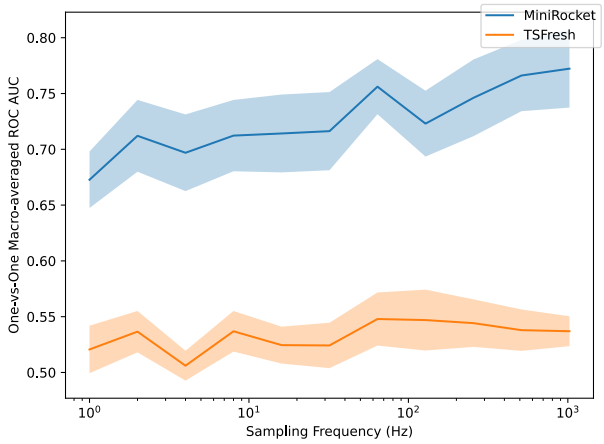


Fig. 3. Comparison between MiniRocket and TSFresh featurizers as sampling frequency changes.

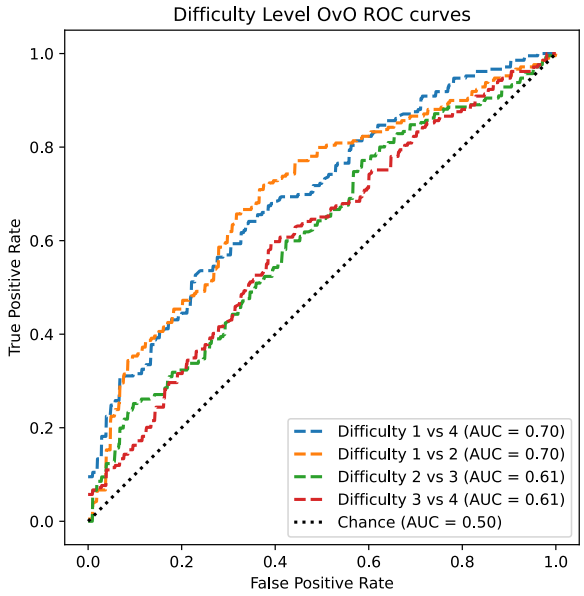


Fig. 4. One-vs-One Receiver Operating Characteristic Area Under the Curve for 16Hz dataset model

[25, 470] seconds, so by adjusting the sampling rate aggregation we were able to precisely scale the data by increasing the density of data rather than extending the length of the time the observation spanned. This allowed for direct comparison of accuracy and computational performance between various sampling rates and across hardware configurations.

For all MiniRocket models, the size of the output was fixed at exactly $m \times 5000$, regardless of the number or length of the input time series. For a fixed number of CPU cores, the time required to fit/transform the MiniRocket model scaled linearly for linear increases in the length of the time series input. The TSFresh model exhibited the same behavior, as shown in figure 6.

When fixing the sampling rate and scaling the number of available CPUs, the MiniRocket method observed a gradual speed-up of performance from 2 to 12 CPU cores before leveling around 46.5 seconds per Hertz to fit/transform the data. This trend was noticed for all sampling rate levels and indicates little benefit to utilizing more than 20 CPU cores for data with sampling rates up to 128 Hz. Potential performance gains for time series lengths beyond 128 Hz are left to future experimentation. We saw no such benefit for TSFresh. The performance with respect to the number of CPU cores remained consistent all the way up to 48 cores.

### *B. Accuracy*

While accuracy was not the explicit focus of this study, it is important to acknowledge that the scalability of the MiniRocket method does carry potential trade-offs with accuracy. As we vary the sampling rate of the dataset from 1Hz to 1024Hz, we note in 3 that MiniRocket very slightly trends downward in performance as the sampling frequency decreases, whereas the TSFresh featurizer experiences almost no change in performance.

A drawback to using the MiniRocket method is the loss of explainability of the model due to the convolutions. Fortunately, the speed and consistency of MiniRocket allows for building quick classifiers for each time series feature to inspect the relative contribution to the overall performance. Figure 2 shows an example of how the same fixed set of MiniRocket transforms provide different levels of predictive power for different features.

It should also be noted that the complexity for MiniRocket is $\mathcal{O}(k * n * l_{input})$ [10], so for a fixed length of time series and number of observations, the complexity scales linearly with respect to an increase in kernels. For this experiment we fixed the number of convolutions applied to the time series input at 5000 kernels, however, we leave the determination of the impact of kernel choice on final accuracy to future work.

## IV. Discussion and Future Work

In this study we compared two different methodologies for preparing time series data for a machine learning classification task. MiniRocket was shown to be a highly scalable method

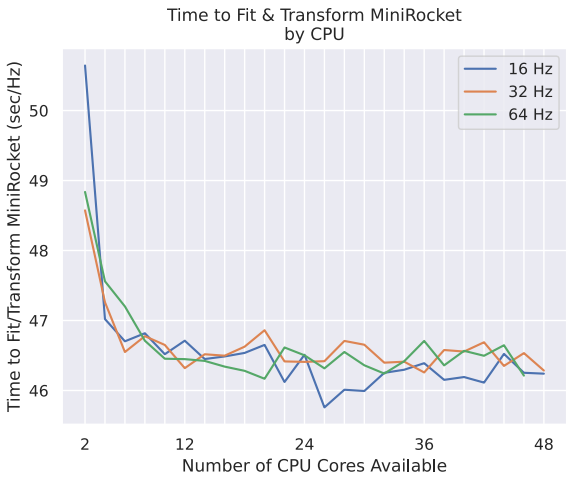


Fig. 5. Time (sec/Hz) to fit and transform time series data by the number of CPU cores available

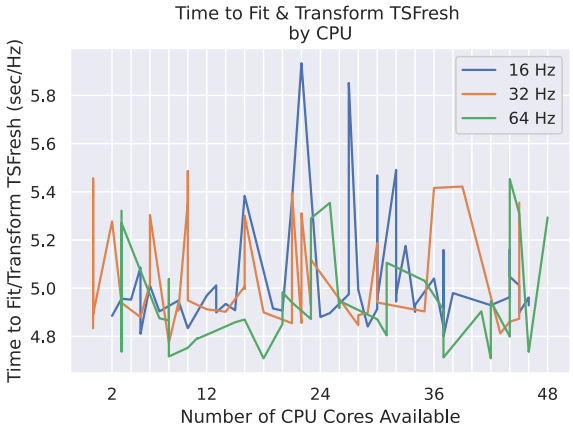


Fig. 6. Time (sec/Hz) to fit and transform the dataset by the number of CPU cores used, for the TSFresh-based model.

capable of marginal accuracy when used with untuned vanilla classifiers while TSFresh showed better accuracy with more comprehensive features, albeit at the expense of greater time to fit and transform the input data. The information from this study can be used to estimate resource requirements when performing time series classification in future experiments or when incorporating in edge technologies.

A significant drawback to using MiniRocket, however, is the loss of explainability in the output after feature selection. This is manageable, to some extent, by performing feature selection prior to pass through the MiniRocket model, but this excludes potentially useful signal from the final classifier model. Another alternative is to apply the MiniRocket method to univariate time series and generate transforms of each sensor feature separately rather than concatenating all together. Although this leads to an increased final output size before passing into a classifier, it is possible to extract through feature selection methods only those kernel transforms that provide significant signal from each of the sensor features. Unfortunately, it is difficult to interpret the significant of the percentage positive value (PPV) even when characterized as an important feature. In this sense, TSFresh provides a much more interpretable model and would likely be a better option in instances where explainability is more important than performance.

Other considerations for choice of methodology is the final application of the model and any associated hardware constraints. While MiniRocket underperforms slightly in accuracy when compared to the most comprehensive TSFresh model, the time to fit and transform MiniRocket is relatively constant up to 128Hz when more than 12 CPUs are available, regardless of the number of the length of the time series input. This is also the case for TSFresh using minimal feature calculators; however, the runtime of TSFresh is heavily dependent on which calculators are used.

Future work could include examining the impact of very high sampling rates on the ability of MiniRocket keep the fit and transform time per Hertz fixed around 46.5 seconds. Additionally, MiniRocket maintains an option to leverage GPUs for performance speed ups, however, TSFresh does not allow for the same ability so this option was excluded from our study. Future work could compare the performance gains of GPU vs CPU models and seek to define the break even point for number of each type of processor for equivalent performance at different fix sample rates.

## V. Conclusion

In this study we compared two different feature extraction methodologies, MiniRocket and TSFresh, for processing and classifying human physiological time series data. Our study found generally that MiniRocket scaled linearly with increases in either time series input length or number of kernels chosen, while TSFresh also scaled in a similar manner. We also found that classifer models with MiniRocket inputs outperformed TSFresh inputs for all sampling rates and configurations except for the most comprehensive (and slowest) TSFresh statistical feature set. Overall, this study characterized the expected performance of each methodology when using human physiological time series data under various hardware constraints.

## Acknowledgment

The authors would like to acknowledge the DAF-MIT CogPilot Team and the MIT Lincoln Laboratory Supercomputing Center (LLSC) for their support of high performance computing tasks.

## References

[1] B. Farrell, L. Atkinson, V. Buquicchio, T. Carr, M. Jones, F. Kerrison, A. Lesser, M. Silver, and N. Williams, "Dod needs to re-evaluate fighter pilot workforce requirements," 2018. [Online]. Available: https://www.gao.gov/products/gao-18-113

[2] C. Caraway, "A looming pilot shortage: It is time to revisit regulations," *International Journal of Aviation, Aeronautics, and Aerospace*, vol. 7, no. 2, 2020.

[3] E. S. Klapper and H.-J. K. Ruff-Stahl, "Effects of the pilot shortage on the regional airline industry: A 2023 forecast," *International Journal of Aviation, Aeronautics, and Aerospace*, vol. 6, no. 3, 2019.

[4] 2023. [Online]. Available: https://www.faa.gov/data_research/aviation_data_statistics/civil_airmen_statistics

[5] W. W. Taylor, C. Moore, and J. Charles Robert Roll, *The Air Force Pilot Shortage: A Crisis for Operational Units?* Santa Monica, CA: RAND Corporation, 2000.

[6] R. S. Cohen, "Can the air force train new pilots without planes?" *The Air Force Times*, 2022. [Online]. Available: https://www.airforcetimes.com/news/your-air-force/2022/11/16/can-the-air-force-train-new-pilots-without-planes/

[7] "Daf-mit ai accelerator." [Online]. Available: https://www.aia.mit.edu/

[8] P. W. Moore, H. M. Rao, C. Beauchene, E. Cowen, S. Yuditskaya, T. Heldt, and L. J. Brattain, "Efficient deep learning on wearable physiological sensor data for pilot flight performance analysis," (forthcoming).

[9] L. Kenworthy, P. Moore, H. M. Rao, L. J. Brattain, K. James, and T. Heldt, "Fatigue assessment from facial videos using deep neural networks and engineered features informed by domain knowledge," 2023, (in press).

[10] A. Dempster, D. Schmidt, and G. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," 08 2021, pp. 248–257.

[11] H. Rao, E. Cowen, S. Yuditskaya, L. Brattain, J. Koerner, G. Ciccarelli, R. Carter, V. Sze, T. Broderick, H. Reynolds, K. McAlpin, and T. Heldt, "Multimodal physiological monitoring during virtual reality piloting tasks: Cogpilot data challenge," *PhysioNet*, 2022. [Online]. Available: https://physionet.org/content/virtual-reality-piloting/1.0.0/

[12] S. Yang, K. Yu, T. Lammers, and F. Chen, "Artificial intelligence in pilot training and education -towards a machine learning aided instructor assistant for flight simulators," 07 2021.

[13] M. Guevarra, S. Das, C. Wayllace, C. D. Epp, M. E. Taylor, and A. Tay, "Augmenting flight training with ai to efficiently train pilots," 2022.

[14] B. Binias, D. Myszor, and K. A. Cyran, "A machine learning approach to the detection of pilot's reaction to unexpected events based on eeg signals," *Computational Intelligence and Neuroscience*, 2018. [Online]. Available: https://doi.org/10.1155/2018/2703513

[15] S. D. Powell, "Bio-signal analysis for personalized pilot training," Ph.D. dissertation, Massachussetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2022.

[16] W. N. Caballero, N. Gaw, P. R. Jenkins, and C. Johnstone, "Toward automated instructor pilots in legacy air force systems: Physiology-based flight difficulty classification via machine learning," *Expert Systems with Applications*, vol. 231, p. 120711, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423012137

[17] M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr, "Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package)," *Neurocomputing*, vol. 307, pp. 72–77, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218304843

[18] Dask Development Team, *Dask: Library for dynamic task scheduling*, 2016. [Online]. Available: https://dask.org

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," *CoRR*, vol. abs/1907.10902, 2019. [Online]. Available: http://arxiv.org/abs/1907.10902

[20] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

[21] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas, "Interactive supercomputing on 40,000 cores for machine learning and data analysis," in *2018 IEEE High Performance extreme Computing Conference (HPEC)*. IEEE, 2018, pp. 1–6.

[22] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 44–60.

[23] M. Christ, A. W. Kempa-Liehr, and M. Feindt, "Distributed and parallel time series feature extraction for industrial big data applications," *arXiv preprint arXiv:1610.07717*, 2016.

[24] Wes McKinney, "Data Structures for Statistical Computing in Python," in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010, pp. 56 – 61.

[25] T. Stenner, C. Boulay, M. Grivich, D. Medine, C. Kothe, tobiasherzke, chausner, G. Grimm, xloem, A. Biancarelli, B. Mansencal, P. Maanen, J. Frey, J. Chen, kyucrane, S. Powell, P. Clisson, and phfix, "sccn/liblsl: v1.16.2," May 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7978343

[26] The HDF Group. (1997-NNNN) Hierarchical Data Format, version 5. Https://www.hdfgroup.org/HDF5/.

[27] M. F. Oberhumer, "The lzo data compression library," 2017. [Online]. Available: http://www.oberhumer.com/opensource/lzo/