Teymourlouei, A., Gentili, R. J., & Reggia, J. (2023, March). Decoding EEG signals with visibility graphs to predict varying levels of mental workload. In 2023 57th Annual Conference on Information Sciences and Systems (CISS) (pp. 1-6). IEEE. Link: https://ieeexplore.ieee.org/document/10089662.

Decoding EEG Signals with Visibility Graphs to Predict Varying Levels of Mental Workload

Arya Teymourlouei[1]
Department of Computer Science
University of Maryland
College Park, USA
ateymo2@terpmail.umd.edu

Rodolphe J. Gentili
Department of Kinesiology
University of Maryland
College Park, USA
rodolphe@umd.edu

James Reggia
Department of Computer Science
University of Maryland
College Park, USA
reggia@umd.edu

***Abstract*—Recent work in predicting mental workload through EEG analysis has centered around features in the frequency domain. However, these features alone may not be enough to accurately predict mental workload. We propose a graph-based approach that filters EEG channels into five frequency bands. The time series data for each band is transformed into two types of visibility graphs. The natural visibility graph and horizontal visibility graph algorithms are used. Six graph-based features are then calculated which seek to distinguish between EEGs of low and high mental workload. Feature selection is evaluated with statistical tests. The features are fed as input data to two machine learning algorithms which are random forest and neural network. The accuracy of the random forest method is 90%, and the neural network has 86% accuracy. The graphical analysis showed that higher frequency ranges (alpha, beta, gamma) had a stronger ability to classify levels of mental workload. Unexpectedly, the natural visibility graph algorithm had better overall performance. Using the method presented here, accurate classification of MWL using EEG signals can enable the development of robust BCI.**

Keywords— Mental workload (MWL), electroencephalography (EEG), machine learning, classification, visibility graph (VG), horizontal visibility graph (HVG)

I. Introduction

Mental workload (MWL) is defined as the neural resources engaged to face task demand during performance [1, 2]. The study of MWL has progressed along with the goal of developing brain-computer interfaces (BCI). Accurate classification of MWL using EEG signals can power the machine learning (ML) component of a BCI. Previous work has investigated the classification of MWL levels using Fourier transforms, wavelet transforms, and power spectral density [3, 4]. There has been some investigation of features from the graphical domain, such as phase lag index and partial directed coherence [5, 6]. However, the use of the visibility graph (VG) algorithm in MWL classification has been very limited. This research investigated the effectiveness of VG-based features for classification of varying MWL levels.

Recently [7] proposed a novel approach that uses three features from VGs. The clustering coefficient, mean degree, and degree distribution are extracted from graphs of EEG channel signals. However, the methodology in [7] has several limitations. The data preprocessing does not apply any filtering methods to the EEG signals. Consequently, graphs are created from raw signals, which contain artifacts. Also, the paper does not investigate the relationship between graphs of different frequency ranges. Another limitation of the previous work is the use of only one VG algorithm, and the use of only three graph-based features.

We expand on the method proposed in [7] by addressing the limitations above. A Butterworth filter is applied (1-40 Hz) on signals to remove larger artifacts. Then, five band-pass filters are applied to extract signals into separate frequency bands. The bands are δ (1-4 Hz), θ (4-8 Hz), α (8-13 Hz), β (13-30 Hz), and γ (30-40 Hz). This approach increases the total number of features and expands the feature analysis. The proposed method utilizes features from both the VG algorithm and the horizontal visibility graph (HVG) algorithm for feature extraction. Six graph-based features are extracted from both algorithms. The features are transitivity, average clustering coefficient, average degree connectivity, assortativity coefficient, average degree centrality, and average shortest path length. The optimal features are then fed to random forest and neural network ML models. These are two powerful ML algorithms which have been proven successful in many applications. Random forest is a direct improvement over the decision tree used in [7] because of the use of ensemble learning.

II. Methodology

A. Dataset

The open-access Simultaneous Task EEG Workload (STEW) dataset was used. The STEW dataset was collected under a low and high workload where participants did not execute any task and performed the Simultaneous Capacity Multi-Tasking task (SIMKAP), respectively [8]. SIMKAP is a cognitive task which involves completing several tasks at once. The tasks are checking for matching numbers, checking a telephone book, and checking a calendar. The MWL of the participants was evaluated using EEG to determine cortical activity. There were 14 electrodes used in data collection, which

[1]Acknowledgment: Arya Teymourlouei is the recipient of a 2023 Maryland Space Grant Consortium research internship.

are AF3, AF4, F3, F4, F7, F8, FC5, FC6, P7, P8, T7, T8, O1, and O2. To reduce computational complexity, features were only calculated for 10 of the 14 channels. AF3, AF4, FC5, and FC6 were omitted. Raw EEG data was preprocessed with a Butterworth filter (1-40 Hz) and linear trends were removed. Then, the five typical frequency bands (δ, θ, α, β, γ) were extracted by means of band-pass filtering. Subject data was collected as three minutes of at-rest and in-task channel data for each subject. After data collection, 30 seconds was removed from both conditions for each subject [8]. The total remaining time was 150 seconds. The sampling frequency was 128 Hz. Therefore, there were 19,200 samples per channel.

B. Visibility Graphs

The VG algorithm was introduced in [9] as a method of converting a time series to a graph. It concerns a set of data points, which can be illustrated in the Cartesian plane. Fig. 1 presents a toy example of how a set of time series samples is transformed into a VG. There are ten bars in the figure, each one representing one collected sample. A node in the VG corresponds to a sample in the time series. The bar graph shows the height of data point, where all the space underneath each bar creates an obstacle. Edges are formed when two points can reach each other in a straight line, without touching another obstacle. The illustration presented in Fig. 1 is inspired from [9], and created in part with [11].

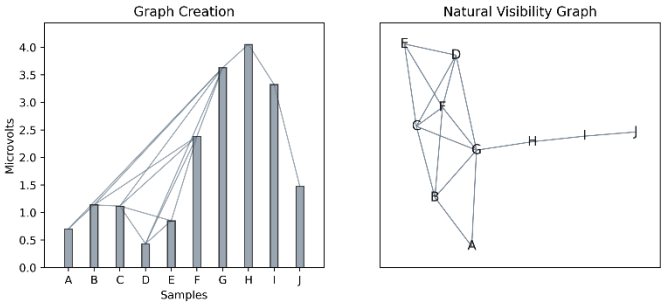

Fig. 1. A visual illustration of the visiblity graph algorithm.

C. Horizontal Visibility Graphs

The HVG algorithm was introduced shortly after the VG algorithm [10]. The HVG was developed as a simpler and easier to calculate version of the original algorithm. It was shown to better handle noise and chaos in a series [10]. This is significant because real-world applications of MWL are often noisy and less robust. Fig. 2 presents a toy example of the HVG with the same data used in Fig. 1. Nodes and edges are understood the same way as before. However, two nodes only form an edge if the values of their associated data points are larger than all other values between them. This requirement results in fewer edges being formed. Over thousands of samples, the differences between VG's and HVG's become significant due to this difference. Notice how some edges are present in Fig. 1. that are not present in Fig. 2.

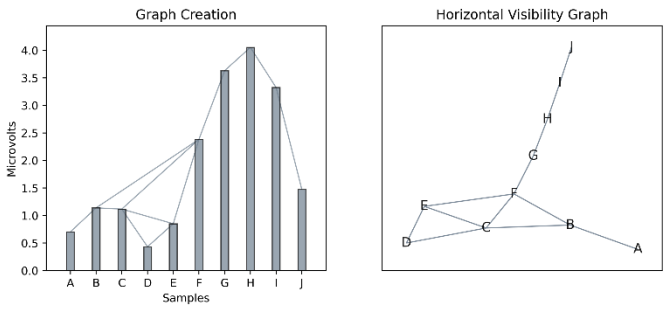

Fig. 2. A visual illustration of the horizontal visiblity graph algorithm.

The ts2vg package converts the time series for each of the frequency bands to VGs and HVGs [11]. The resulting graphs are then converted to NetworkX graph objects for further analysis [12]. All graphs are directed. The direction of an edge is determined by the y-values of the associated data points, where the starting node always has the larger y-value. The graphs are unweighted. This improves the efficiency of the network reconstruction but limits the analysis.

D. Feature Extraction

Six features are extracted from the VGs and HVGs. Transitivity and average clustering coefficient (ACC) measure the tendency of nodes to form triangles, or cluster. The clustering coefficient is defined in [13]. The transitivity was originally thought to be an equivalent algorithm; however, it has been shown to calculate different results [14].

The degree of a node is the number of edges it forms with other nodes. Many features make use of the degree of a node. The assortativity coefficient measures the tendency of nodes with a similar degree to be connected [15]. Degree centrality is computed as the degree of a node, normalized dividing by the maximum possible degree. The average degree centrality (ADC) is the sum of the centrality of all nodes in the graph. This feature is relatively simple compared to others but is useful in measuring the average connectivity of nodes in a graph. The average degree connectivity is a measure of the connectivity of the individual degrees in a graph [16]. It calculates the tendency of nodes of degree k to be connected with other nodes of degree k. The formula is defined in [12]. The average shortest path length (ASPL) is the average number of steps it takes to make the shortest path for all possible combinations of nodes. It is calculated with Dijkstra's shortest path algorithm [17].

E. Feature Selection

Data was evaluated for normality using a Shapiro-Wilk test. Features which followed a normal distribution were then tested for significance using a paired t-test. Otherwise, significance was assessed with a Wilcoxon test. Features which showed a statistically significant difference ($p < 0.05$) between low and high MWL were selected as input to the ML algorithm. To reduce the Type I error, the false discovery rate (FDR; *Benjamini–Hochberg* procedure) was employed to correct for multiple comparisons. *Cohen's d* effect sizes were also provided, when appropriate. Features were split into a 75%-25% training-testing ratio, then fed to two ML algorithms. A diagram of the full methodology, including the ML, is presented in Fig. 3. The ML models are explained in the sections below.

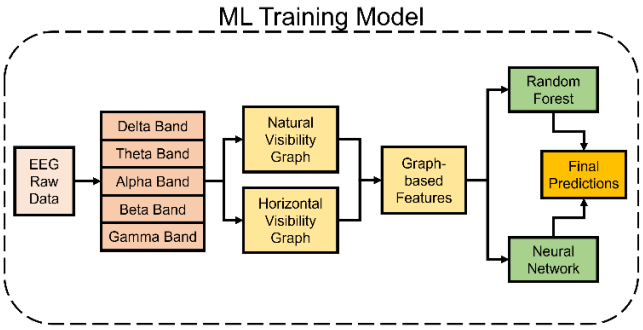

Fig. 3. A summary of the methodology. Filtered EEG data is reconstructed as VGs and HVGs. The graph-based features are then extracted, which are fed to ML models to predict the level of MWL.

F. Random Forest

The chosen ML models have shown high accuracy in many other classification problems. The first model is random forest (RFC), an ensemble learning classifier. The classifier starts by feeding the graph-based features as inputs to 100 decision trees. Trees are formed based on the Gini impurity criterion. Each tree evaluates the feature based on several binary decision nodes. The end of the tree has the final prediction result. All the results are averaged, and a final prediction is made. The trees are pruned to a maximum length of five branches. Multiple decision trees help improve the model's accuracy and reduce over-fitting.

G. Neural Network

The second classifier is a supervised neural network. It is also known as multi-layer perceptron (MLP). The algorithm starts with the input layer of features. Each input has an associated weight. Inputs are passed through five hidden layers which each contain twenty hidden units. In the hidden layer, inputs are multiplied by weights and bias is added. Then, the rectified linear unit activation function is applied to nodes. The activation function applies a non-linear transformation to the previously linear equation. The result is passed to the output layer, where the final prediction is made. If the predictions have high error, the model goes back and re-trains to reduce the error. This is the process of error back-propagation, where the weights are re-assigned using a stochastic gradient-based optimizer.

III. RESULTS

All data processing, graphs, and ML were implemented in Python. ML was done with Scikit-learn [18]. All plots were generated with Matplotlib [19]. Computations were done on a Dell PC (64-bit) with Intel Core i7 CPU and 24 GB RAM.

A. Classification Performance

The ML classifiers' performance were evaluated using three metrics. These are accuracy, F1 score, and logarithmic loss. Accuracy is the score of correct predictions over total predictions. F1 is the harmonic mean of precision and recall of the model. Log loss measures how close the predicted probability of a value is to the actual values. Table 1 displays the evaluation results of the testing dataset for both classifiers.

TABLE I. CLASSIFICATION RESULTS

Classifier	**Metrics**		
	Test Accuracy	***Loss***	***F1***
Random Forest	0.90	0.43	0.88
Neural Network	0.86	0.63	0.84

Using data from all the subjects, the RFC model achieved a 90% classification accuracy, and the MLP classifier had an 86% accuracy. These results provide evidence that the methodology was successful in predicting MWL with high accuracy. The graph-based approach did not show a significant decrease in accuracy when combining multiple subject's data. The accuracy learning curve for both ML models is presented in Fig. 4. This is a plot that shows the cross-validated accuracy scores for training and testing data of different sizes and is created with [18]. The end of the curve reports the final accuracy score with the full data set size. These are the scores which are displayed in the table. Fig. 4. is not a performance curve for epochs of training. Such a plot was unobtainable due to limitations with the plotting software.

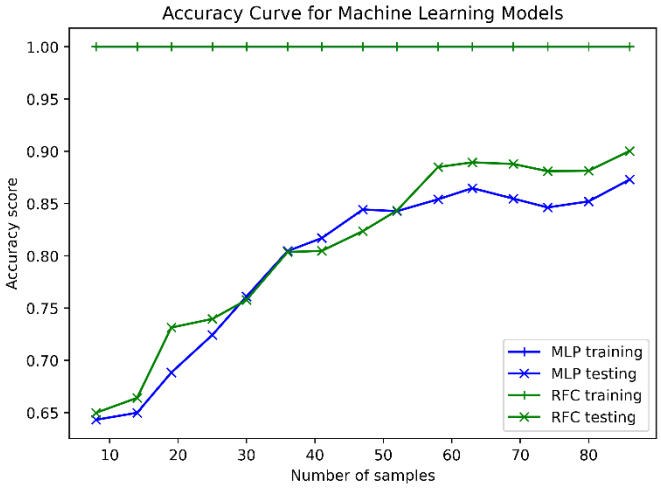

Fig. 4. The accuracy learning curve using the graph-based features, for the RFC and MLP models. The MLP and RFC training data curves are both constant at a maximum accuracy level, so the MLP training curve is not visible.

The loss learning curve for both ML models is presented in Fig. 5. This plot is created in the same fashion as Fig 4., except reporting loss scores as opposed to accuracy scores. In Fig 5., we see the loss for MLP training data was very close to zero (~0.001) and slightly larger for RFC. Despite having perfect accuracy (as seen in Fig. 4.), the training data loss for both models were nonzero. This is likely due to repeated cross-validation. The testing data rapidly decreased then converged for the MLP, and slowly decreased for the RFC. For the testing sets, we see RFC had a better loss than the MLP model.

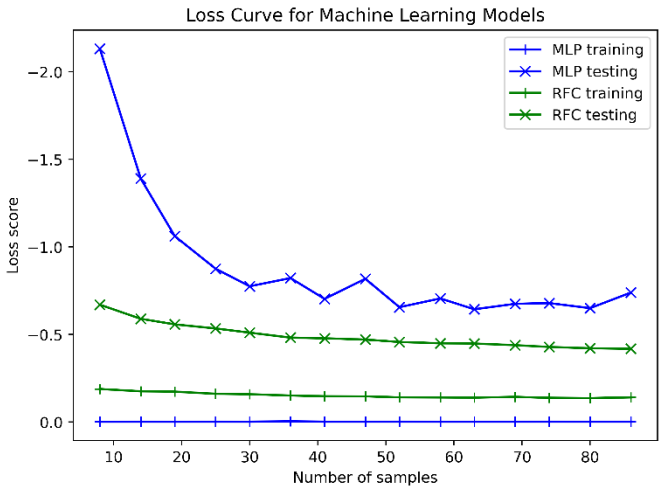

Fig. 5. The loss learning curve using the graph-based features, for the RFC and MLP models. Unlike the accuracy scores, the loss scores for the training data of both models are not perfect (zero), but very close to it. This is likely due to the cross-validation procedure, where a small loss can emerge.

In an ML model, performance is not limited to the scoring metrics. Performance also considers the scalability of the model. The overhead, measured in seconds, for the learning curves of both ML models is presented in Fig. 6. This figure shows the time taken to fit the models for different sizes of input data. This plot was generated in the same way as Fig. 4. and Fig. 5.

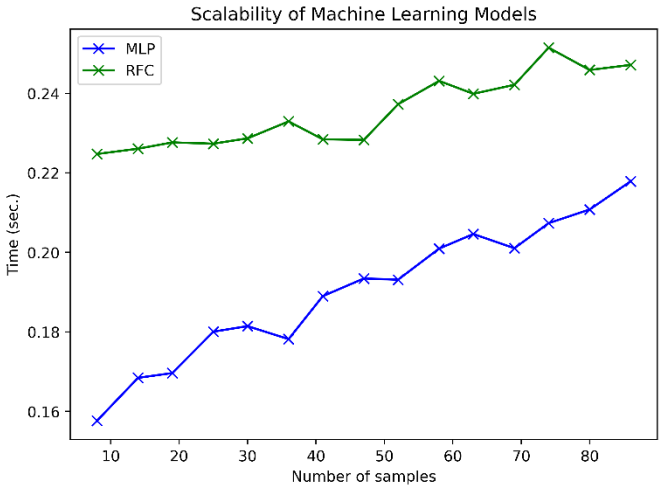

Fig. 6. The model overhead for the two ML models. Overhead is calculated as the seconds taken to fit the model for a given sample size.

In Fig. 6., we see in the largest sample size, the overhead of the MLP model was better than RFC. However, the MLP increased at a faster rate than the RFC. The RFC model was significantly affected by the number of trees. In fact, doubling the number of trees doubled the running time for RFC. However, in this case, 100 trees were enough to reach the maximum performance. The results of Fig. 6. suggest ML training with a small sample size may be better suited to an MLP model.

B. Graph Analysis

In this section, we present an analysis of the best feature sets used by the ML algorithms. The figures in this section illustrate some of the feature sets that were discriminatory between high and low MWL. Each feature was calculated for the ten EEG channels. The features were calculated for both types of VG's and for each frequency band. There were 100 possible values, for each feature, that could be statistically significant when comparing low to high cognitive workload.

Only the δ band did not have many statistically significant results from feature extraction. In the θ band, the ACC and ADC were discriminatory in several channels when using the VG algorithm. As workload increased, the ACC decreased, and was seen in the frontal ($p = 0.001$, $d = 0.601$) and parietal ($p = 0.027$, $d = 0.340$) regions. The ADC decreased in the frontal (p=0.003, d = -0.566) and temporal ($p = 0.038$, d = -0.403) regions. For the HVG algorithm, a decrease in transitivity was observed in the frontal region ($p < 0.0001$, $d = 0.889$).

In the α band, when workload increased, a decrease in the ACC was observed when using the VG algorithm. This was observed in the frontal ($p < 0.0001$, $d = 0.972$), parietal (p=0.016, $d = 0.400$), temporal ($p = 0.008$, $d = 0.462$), and occipital ($p = 0.008$, $d = 0.446$) regions. The boxplot of ACC, for the α band, is shown in Fig. 7. With the HVG algorithm, a decrease in transitivity was observed in the frontal ($p = 0.0002$, d=0.639), parietal ($p = 0.016$, $d = 0.381$), temporal (p = 0.018, d= 0.368), and occipital ($p = 0.0002$, $d = 0.634$) regions.

In the β band, as workload increased, there was an increase in ACC using the HVG algorithm for the frontal ($p = 0.008$, d = -0.436), parietal ($p < 0.0001$, $d = 0.888$), temporal (p = 0.019, d= -0.379) and occipital (p<0.0001, d = -0.917) regions. For the VG algorithm, there was in increase in the average degree connectivity for the frontal (p=0. 018, d = -0.406) and temporal (p=0.018, d= -0.477), and occipital (p=0.037, d=-0.341) regions.

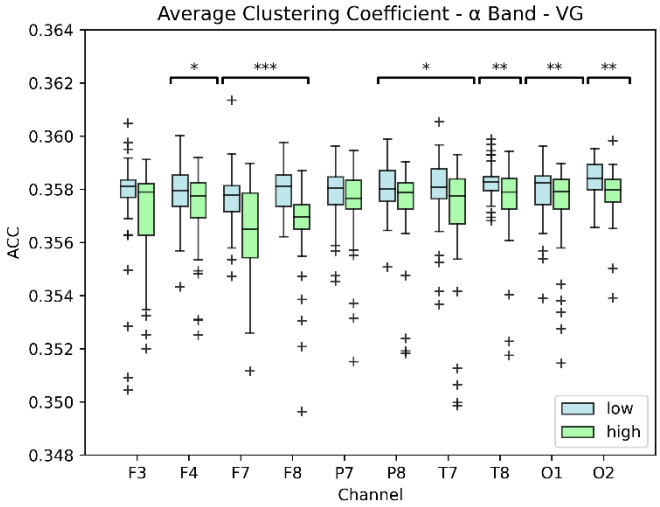

Fig. 7. The boxplot of ACC in the alpha band, using the VG algorithm. Blue denotes low workload, and green denotes high workload. Statistical significance is denoted as follows: *p<0.05, **p<0.01, ***p<0.001

The γ band recorded 72 total discriminatory features, the most of any frequency range. For both graph algorithms in the γ band, the average shortest path length was discriminatory for every channel. The boxplot of ASPL, for the γ band, calculated with HVG, is plotted in Fig. 8. For the VG algorithm, the

increase in ASPL was observed for the frontal (p=0.0003, d = -0.611), parietal (p = 0.0003, d = -0.653), and temporal (p = 0.0003, d = -0.644) regions.

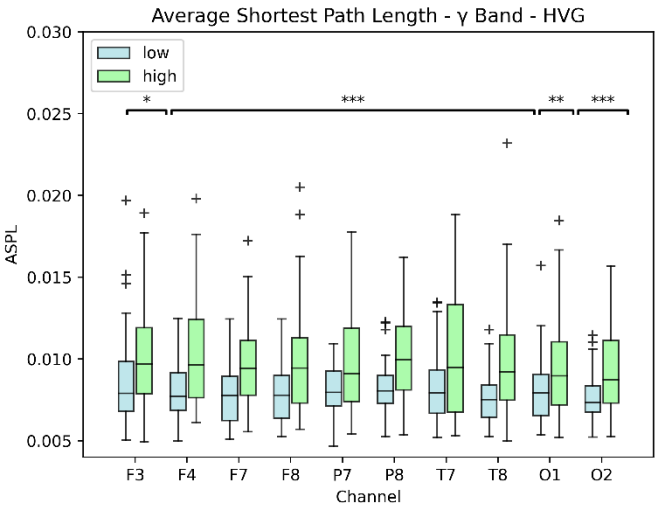

Fig. 8. The boxplot of ASPL in the gamma band, using the HVG algorithm. Blue denotes low workload, and green denotes high workload. Statistical significance is denoted as follows: *p<0.05, **p<0.01, ***p<0.001

The graph from the γ band using VG algorithm achieved the most discriminatory features of any graph. Each of the six features had at least 50% of channels being discriminatory, with three features having 100% of channels being discriminatory. For the VG algorithm (γ band), as workload increased, an increase in ADC was observed in the frontal (p = 0.0001, d = -0.662), parietal (p = 0.0001, d = -0.551), temporal (p = 0.0003, d = -0.589), and occipital (p=0.006, d= -0.446) regions.

C. Graph Algorithms

In the performance section, we focused on utilizing features from both graph algorithms to maximize performance. In this section, we compare the performance of features from VG and HVG algorithms separately. We compare the number of discriminatory features, and the ML performance when separating the feature sets. Table 2 shows the counts of discriminatory features for the two algorithms.

TABLE II. DISCRIMINATORY FEATURES

Features	Graph Algorithm	
	VG	***HVG***
Average clustering coeffiicent	**24**	17
Transitivity	8	**18**
Assortativity coeffiicent	**14**	1
Average degree connectivity	**22**	6
Average degree centrality	**25**	10
Average shortest path length	14	**17**
Total	**107**	69

We see that the VG algorithm outperformed the HVG algorithm in four out of the six graph-based measurements. In total, the VG algorithm had a 55% increase in the number of features. These results suggest that despite being less time-efficient, the VG algorithm reaches a level of depth that the HVG algorithm cannot do. To further this claim, we present the results of comparing the number of features by the frequency range in Table 3. The results of the delta band are omitted in Table 3 because of the lack of discriminatory features (for either graph algorithm).

TABLE III. DISCRIMINATORY FEATURES, BY FREQUENCY RANGE

Frequency Range	Graph Algorithm	
	VG	***HVG***
θ (4 – 8 Hz)	**10**	9
α (8 – 13 Hz)	**24**	17
β (13 – 30 Hz)	**25**	19
γ (30 – 40 Hz)	**48**	24

In Table 3, we see that every frequency range had more discriminatory features in the VG algorithm compared to HVG. The gamma band had double the number of features with the VG algorithm. These results are important for the purposes of feature engineering in ML. However, they also suggest that neural analyses involving VGs are better suited to the original VG algorithm. The larger number of features indicates the VG was better at discriminating between high and low MWL. This is an important observation with respect to understanding the neural basis for MWL.

Finally, we present the results of the ML algorithms using only feature sets containing VG and HVG features. While this reduces the total number of features in the ML models, it allows us to compare the two algorithms standing alone. In Table 4, we present the performance of the random forest model, comparing VG vs HVG features as inputs. All scoring metrics concern the testing set, not the training set.

TABLE IV. RANDOM FOREST PERFORMANCE, VG VS HVG ALGORITHM

Metrics	Graph Algorithm	
	VG	***HVG***
Accuracy	**0.93**	0.48
Loss	**0.39**	0.74
F1 Score	**0.91**	0.48

In Table 4, we see that the VG feature set outperformed the HVG feature set for the random forest model in every scoring metric. The accuracy was significantly higher. In fact, the accuracy scored here was higher than with both feature sets combined. This suggests the HVG feature set was not beneficial to the overall ML performance when using the RFC model. The loss score was lower for the VG set compared to HVG, and the F1 score was significantly higher. In Table 5, we present the performance of the neural network model, comparing VG vs HVG features as inputs.

TABLE V. NEURAL NET PERFORMANCE, VG VS HVG ALGORITHM

Metrics	Graph Algorithm	
	VG	***HVG***
Accuracy	**0.85**	0.48
Loss	**0.63**	0.74

Metrics	Graph Algorithm	
	VG	***HVG***
F1 Score	**0.82**	0.42

In Table 5, we see similar results to that of the RFC model. The VG feature set outperformed the HVG set in every metric. In this case, the accuracy of the VG set was 85%, which is lower than the accuracy when both feature sets are combined. This suggests that, for the neural network model, combining VG and HVG features can be slightly beneficial to overall performance. It is important to note, due to the small size of the input data, the scoring metrics are subject to slight variability. Comparing small differences between metrics should be done with caution. The loss and F1 score were significantly better in the VG feature set.

The overhead of the two algorithms is of particular interest. BCIs that utilize ML must be able to make predictions quickly. Therefore, we present the overhead of the ML learning curves for both graph algorithms and both ML models in Fig 9. In the figure, we see that the neural network model increased at a faster rate for both algorithms, which is consistent with our previous overhead plot. The MLP model with only HVG features had the largest overhead and increased at a fast rate. For both algorithms, the growth rate of the random forest was slow, again, consistent with our previous findings. Based on these results, ML training with a larger sample size (i.e., over 1000) may be better suited to the RFC model.

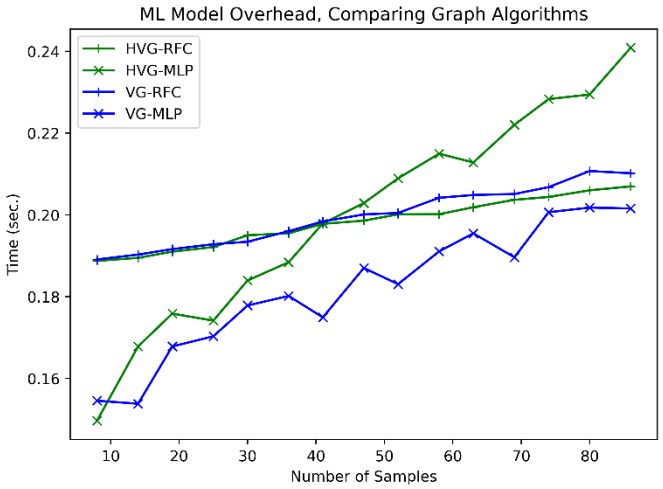

Fig. 9. The model overhead for the ML models, comparing the results of VG vs HVG algorithm feature sets. The green curves represent ML models trained only with features from the HVG algorithm. The blue curves represent ML models trained only with VG algorithm features.

IV. Discussion

This research investigated a novel method to classify MWL using a graph-based and ML approach. The proposed method has an 88-90% classification accuracy from two ML models. Higher frequency ranges (α, β, γ) had a strong ability to classify levels of MWL. The γ band had the strongest ability to distinguish between low and high MWL, across multiple features. When comparing the VG and HVG algorithms, the VG algorithm had a significantly better performance in both ML models. This finding was unexpected because the HVG algorithm has previously been found to handle noise better [10]. The advantages that we observed with the VG algorithm may be due to factors specific to the dataset, the use of filtering, or other unknown factors.

The performance of our ML models is like that of [7]. The RFC model with only VG features achieved 92% accuracy, which was an improvement over the highest accuracy identified in [7], 89.6%. The strong discrimination identified in the frontal region for several features is consistent with findings from [7], who specifically identified F8 and AF4 (but, in our method, we omitted AF4). The identification of the α, β, and to a lesser extent, θ bands is consistent with previous work [3, 4, 5]. The significance of the γ band was also found by [20], albeit in the context of spectral power.

The methodology had several limitations. In our approach, we only considered supervised learning algorithms. The method also was limited to a binary classification task. In more practical applications, the fluctuation of MWL can have more than two states. Another limitation was the lack of publicly available data. The STEW dataset is one of few MWL datasets that use EEG. Other datasets were explored, but either were not suitable for a classification task or the EEG data was not properly collected.

We suggest future work focus on validating the graph-based features by using more MWL datasets. Further, it is worth exploring other VG algorithms and graph measurements to optimize the ML performance. BCIs that use ML need to be efficient and accurate using only minimal training data. We suggest variations to our approach that use fewer EEG samples to construct VGs, while attaining a high level of performance.

References

[1] Wickens, C. (2008). Multiple resources and m. *Human Factors*, *50*(3), 449-455.

[2] Young, M., Brookhuis, K., Wickens, C., & Hancock, P. (2015). State of science: Mental workload in ergonomics. *Ergonomics*, *58*(1), 1-17.

[3] Murata, A. (2005). An attempt to evaluate mental workload using wavelet transform of EEG. *Human Factors*, *47*(3), 498-508.

[4] Shaw, E., Rietschel, J., Shuggi, I., Xu, Y., Chen, S., Miller, M., Hatfield, B. & Gentili, R.. (2019). Cerebral cortical networking for mental workload assessment under various demands during dual-task walking. *Experimental Brain Research*, *237*(9), 2279- 2295.

[5] Mazher, M., Qayyum, A., Ahmad, I., & Alassafi, M. O. (2020). Beyond traditional approaches: A partial directed coherence with graph theory-based mental load assessment using EEG modality. *Neural Computing and Applications*, 1-16.

[6] Shimomura, Y., Yoda, T., Sugiura, K., Horiguchi, A., Iwanaga, K., & Katsuura, T. (2008). Use of frequency domain analysis of skin conductance for evaluation of mental workload. *Journal of Physiological Anthropology*, *27*(4), 173-177.

[7] Zhu, G., Zong, F., Zhang, H., Wei, B., & Liu, F. (2021). Cognitive load during multitasking can be accurately assessed based on single channel electroencephalography using graph methods. *IEEE Xplore*, *9*, 33102-33109.

[8] Lim, W., Sourina, O., & Wang, L. (2018). STEW: Simultaneous task EEG workload data set. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *26*(11), 2106-2114.

[9] Lacasa, L., Luque, B., Ballesteros, F., Luque, J., & Nuno, J. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, *105*(13), 4972-4975.

[10] Luque, B., Lacasa, L., Ballesteros, F., & Luque, J. (2009). Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, *80*(4), 046103.

[11] Bergillos, C. (2020). A study of visibility graphs for time series representations. Universitat Politècnica de Catalunya.

[12] Hagberg, A., Swart, P., Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX. Los Alamos National Lab. Los Alamos, NM.

[13] Watts, D., & Strogatz, S. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*(6684), 440-442.

[14] Schank, T., & Wagner, D. (2005). Approximating clustering coefficient and transitivity. *Journal of Graph Algorithms and Applications*, *9*(2), 265-275.

[15] Newman, M. (2002). Assortative mixing in networks. *Physical Review Letters*, *89*(20), 208701.

[16] Barrat, A., Barthelemy, M., Pastor-Satorras, R., & Vespignani, A. (2004). The architecture of complex weighted networks. *Proceedings of the National Academy of Sciences*, *101*(11), 3747-3752.

[17] Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269-271.

[18] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, *12*, 2825-2830.

[19] Hunter, J. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90-95.

[20] Howard, M., Rizzuto, D., Caplan, J., Madsen, J., Lisman, J., Aschenbrenner-Scheibe, R., ... & Kahana, M. (2003). Gamma oscillations correlate with working memory load in humans. *Cerebral Cortex*, *13*(12), 1369-1374.