https://doi.org/10.31891/2219-9365-2026-87-50
UDC 004.4:519.85:616.831-073.97

STEFANYSHYN Ivan
Ternopil Ivan Puluj National Technical University
https://orcid.org/0009-0008-6930-528X
e-mail: ivan_stefanyshyn0707@tntu.edu.ua
YATSYSHYN Vasyl
Ternopil Ivan Puluj National Technical University
https://orcid.org/0000-0002-5517-6359
e-mail: yacyshyn@tntu.edu.ua

FORMALIZATION AND OPTIMIZATION OF SOFTWARE COMPONENT PIPELINES FOR ELECTROENCEPHALOGRAPHIC SIGNAL CLASSIFICATION

The article proposes a formalized approach to the construction and optimization of software component pipelines for electroencephalographic (EEG) signal classification. This is caused by the limitations of existing methods, which are usually empirical when defining software component pipeline configurations, as well as by the lack of a formalized procedure for selecting optimal software components that considers not only one specific metric but a set of metrics. The aim of the study is to formalize and optimize software component pipelines as a basis for their automated usage in a software system for EEG signal classification. The article proposes using a tree-like directed graph structure to represent the software component pipeline. The nodes of the structure correspond to software components, while the edges define valid directions of data transfer between processing stages. To ensure acyclicity, a valid sequence of stages, compatibility of software components, and structural completeness of configurations, rules for constructing the structure are defined. The optimization procedure involves formalized multi-metric evaluation. The evaluation metrics include accuracy, F1 score, area under the ROC curve, and normalized processing time per signal sample. All metrics are normalized, brought to a common minimization direction, and combined into an objective function with weight coefficients. The search for the optimal software component pipeline configuration is implemented using mixed-integer linear programming. The experimental verification of the proposed solutions was performed using real EEG data related to the motor activity of human upper limb fingers. Within the constructed tree-like directed graph structure, 8 valid software component pipeline configurations were formed. According to the optimization results, the optimal software component pipeline configuration for the baseline scenario with equal metric weights was "PCA → CNN". The obtained results confirmed the correctness of the proposed formalizations, their consistency with each other, and their practical suitability for automated construction and optimization of software component pipelines for EEG signal classification.

Keywords: electroencephalographic signals, classification, motor activity, brain-computer interface, computation, software system, optimization, graph, algorithm.

СТЕФАНИШИН Іван, ЯЦИШИН Василь
Тернопільський національний політехнічний університет імені Івана Пулюя

ФОРМАЛІЗАЦІЯ ТА ОПТИМІЗАЦІЯ КОНВЕЄРІВ ПРОГРАМНИХ КОМПОНЕНТІВ КЛАСИФІКАЦІЇ ЕЛЕКТРОЕНЦЕФАЛОГРАФІЧНИХ СИГНАЛІВ

У статті запропоновано формалізований підхід до процесу побудови та оптимізації конвеєрів програмних компонентів класифікації ЕЕГ сигналів. Це обумовлено недосконалістю існуючих методів, зазвичай, емпіричних, при визначенні конфігурації конвеєрів, а також відсутністю формалізованої процедури вибору оптимальних програмних компонентів, яка враховує не лише одну конкретну метрику, а множину метрик. Метою роботи є формалізація й оптимізації конвеєрів програмних компонентів як основи їх автоматизованого використання в програмній системі класифікації ЕЕГ-сигналів. У статті для представлення конвеєра запропоновано використати деревовидну орієнтовану графову структуру. Вузли структури відповідають програмним компонентам, а ребра задають допустимі напрями передавання даних між етапами обробки. Для забезпечення ациклічності, допустимої послідовності етапів, сумісності компонентів і структурної завершеності конфігурацій визначено правила побудови структури. Процедура оптимізації передбачає проведення багатометрикового формалізованого оцінювання. До метрик оцінювання входять точність, F1-міра, площа під ROC-кривою та нормалізований час обробки одного зразка. Усі метрики нормуються, приводяться до єдиного напряму мінімізації та об'єднуються у цільову функцію з ваговими коефіцієнтами. Пошук найбільш оптимальної конфігурації конвеєра реалізується методом змішано-цілочислового лінійного програмування. Експериментальна верифікація запропонованих рішень виконана на реальних ЕЕГ-даних моторної активності пальців верхніх кінцівок. У межах побудованої структури сформовано 8 допустимих конфігурацій конвеєра програмних компонентів. За результатами оптимізації конвеєрів для базового сценарію з однаковими вагами метрик є конфігурація PCA → CNN. Одержані результати підтвердили коректність запропонованих формалізацій, їх узгодженість між собою та практичну придатність для автоматизованої побудови й оптимізації конвеєрів програмних компонентів класифікації ЕЕГ-сигналів.

Ключові слова: електроенцефалографічні сигнали, класифікація, моторна активність, мозок-комп'ютерний інтерфейс, обчислення, програмна система, оптимізація, граф, алгоритм.

Стаття надійшла до редакції / Received 24.07.2026
Прийнята до друку / Accepted 17.08.2026
Опубліковано / Published 10.09.2026

STATEMENT OF THE PROBLEM

The quality of a software system for EEG signal classification in brain-computer interfaces (BCI) is determined not by a separate algorithm, but by the consistency of the entire pipeline, which includes the stages of preprocessing, data enhancement, feature extraction, dimensionality reduction, and classification [1]. For motor activity recognition tasks, this is especially important due to the high variability of signals, the presence of artifacts, intersubject differences, and processing time constraints [2-3].

To ensure the quality of a software system for EEG signal classification in BCI, two main problems must be solved: the construction and formalized representation of the software component pipeline for EEG signal classification, and the development of a procedure for its optimization. Solving the first problem makes it possible to formally define the structure of the software component pipeline, determine the set of its valid configurations, define rules for combining software components, and verify the correctness of their construction.

Solving the second problem ensures a justified selection of the optimal configuration of the software component pipeline based on a set of metrics in accordance with the requirements of a specific applied task.

ANALYSIS OF RECENT STUDIES AND PUBLICATIONS

Modern studies on EEG signal classification for upper limb motor activity recognition mainly use a typical processing pipeline that includes preprocessing, data enhancement, feature extraction, dimensionality reduction, classification, and result evaluation. The authors' previous systematic review of this research area showed that this structure is dominant in most studies [1].

Recent publications mainly focus on improving individual stages of the pipeline. In studies [2–3], classification quality is improved by refining separate signal processing methods and classification models. At the same time, these approaches do not present the software component pipeline for EEG signal classification as a formalized construction object for which the structure, rules for combining software components, constraints on their use, and the set of valid configurations can be defined. As a result, pipeline construction remains mostly empirical, while its configuration is formed without a formal definition of valid alternatives.

The issue of optimizing the formed configurations also remains insufficiently studied. The authors' previous systematic review found that most studies select between alternative pipeline variants by directly comparing the best or average values of individual metrics, primarily classification accuracy [1]. This does not sufficiently support pipeline optimization, since it does not make it possible to integrate several metrics at the same time, including accuracy, robustness, and computational costs, and does not ensure that the selected configuration is aligned with the requirements of a specific applied task.

The usage of AutoML systems in BCI tasks also requires separate analysis. The authors' previous systematic review found that existing AutoML solutions simplify the automation of model and hyperparameter selection; however, they do not provide an explicit representation of EEG pipeline stages as controllable components and are mainly focused on single-objective optimization [1]. As a result, they do not allow the formal description of the set of valid configurations, the definition of rules for their construction, or the controlled selection of the pipeline structure according to the requirements of a specific applied task.

PRESENTING THE MAIN MATERIAL

1. Formalization of software component pipeline construction

To reduce dependence on empirical construction of software component pipelines for EEG signal classification, their formalized representation as a tree-like directed graph structure is proposed (see Fig. 1). Within this representation, the nodes correspond to individual software components and signal processing stages, while directed edges define the valid directions of data transfer between them.

This representation makes it possible to describe the pipeline not as a manually formed sequence of algorithms, but as a formally defined object with predefined structural relationships and construction rules. As a result, pipeline construction is shifted from the level of manual combination of individual methods to the level of formal definition of a valid configuration suitable for verification, comparison, and further optimization.

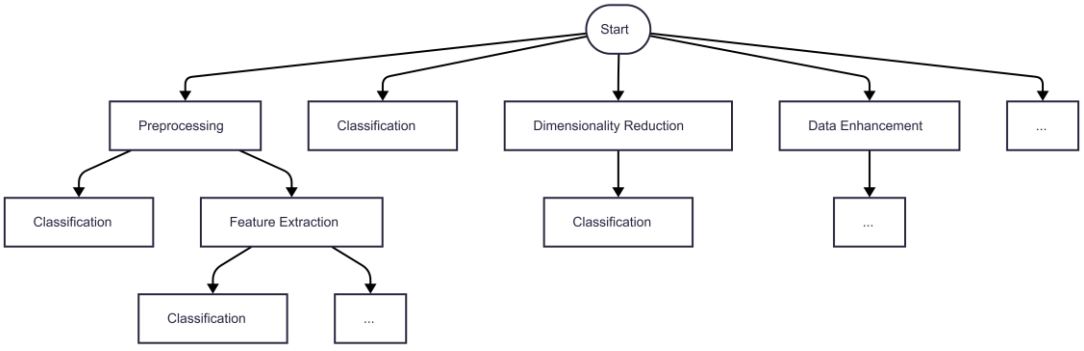

Fig. 1. An example of a tree-like directed graph structure of a software component pipeline

As shown in Fig. 1, the tree-like directed graph structure defines the composition of the software component pipeline, the order of interaction between software components, and the valid transitions between EEG signal processing stages. This makes it possible to consider the software component pipeline not as a manually sequence of algorithms, but as a formally defined object suitable for verification, modification, and further optimization.

1.1 Advantages of using a tree-like directed graph structure in software component pipeline construction

An important advantage is the possibility of verifying the correctness of a software component pipeline configuration before computations are launched. Since the connections between nodes are formed according to predefined rules, the constructed tree-like directed graph structure can be checked for compliance with the logic of processing stages, absence of cycles, structural completeness, and compatibility of software components. This verification makes it possible to exclude invalid configurations before the evaluation stage, which increases the validity of further analysis.

Another advantage is the possibility of formally representing a set of alternative configurations within a single tree-like directed graph structure. In this case, different configurations of the software component pipeline are considered not as separate manually constructed variants, but as elements of a single space of valid solutions. This creates a basis for their systematic study, reproducible comparison, and targeted selection according to the requirements of a specific task.

It is fundamentally important that this structure creates a basis for further optimization. When nodes, connections, and constraints are formally defined, the task of selecting the optimal configuration of the software component pipeline is reduced to a search within the set of valid configurations, rather than an informal comparison of manually formed combinations of algorithms. Due to this, pipeline construction and selection can be included in a single formalized procedure of evaluation and optimization according to the requirements of a specific EEG signal classification task.

1.2 Mathematical representation of the tree-like directed graph structure of a software component pipeline

To formally represent the construction process of an EEG signal classification pipeline, we introduce a tree-like directed graph structure (1):

$$G = (V, E), V = \{v_1, \dots, v_n\}, E \subseteq V \times V, \tag{1}$$

where V – the set of nodes, E – the set of directed edges between nodes, n – the number of nodes.

Each node $v \in V$ corresponds to a separate software component that implements a specific stage of EEG signal processing, while each edge $(u, v) \in E$ defines a valid direction of data transfer from component u to component v.

To represent the assignment of nodes to processing stages, we introduce a node type function (2):

$$\tau: V \to S, \tag{2}$$

where S – the set of pipeline stages: preprocessing, data enhancement, feature extraction, dimensionality reduction, and classification.

A separate software component pipeline is defined as a directed path in the graph (3):

$$\pi = (v_0, v_1, \dots, v_k), \tag{3}$$

where k – the number of transitions between nodes. For each pair of neighboring nodes in the path, the following condition must hold (4):

$$(v_{i-1}, v_i) \in E, i = 1, \dots, k \tag{4}$$

To formalize the validity of construction, we introduce a stage order function (5):

$$\rho: S \to N \tag{5}$$

Equation (5) represents the logical order of data flow within the pipeline. Therefore, for each edge$(u, v) \in E$ the following condition must hold (6); it prevents transitions to previous processing stages and ensures the direction of the pipeline from the initial components to classification:

$$\rho(\tau(u)) \leq \rho(\tau(v)) \tag{6}$$

The tree-like nature of the structure is defined by the existence of a single initial node $r \in V$, which has no incoming edges, while each other node has no more than one direct predecessor. This makes it possible to define alternative branches of pipeline construction without violating the integrity of the structure and without forming cycles.

In this representation, the tree-like directed graph structure G defines the space of possible pipeline configurations, while a separate path π corresponds to one specific software component pipeline.

1.3 Rules for constructing the tree-like directed graph structure of a software component pipeline

The construction of the tree-like directed graph structure of a software component pipeline must follow a set of rules that define the validity of a configuration and ensure its suitability for further optimization. These rules are defined as follows.

Acyclicity of the tree-like directed graph structure. The structure must not contain cycles, loops, recursive transitions, or returns to previous stages.

Fixed direction of data flow. All edges must be directed from input to output, that is, from the signal source to the classification result.

Valid sequence of stages. Software components must be arranged according to the logic of signal processing: preprocessing, data enhancement, feature extraction, dimensionality reduction, and classification [1]. The classification stage is mandatory, while the other stages may be optional depending on the experimental conditions and the type of data.

Uniqueness of the input data source for a node. Each node must have one input, which ensures a clear origin of data.

Possibility of alternative outputs. One node may transfer its results to several subsequent software components, which makes it possible to form parallel or alternative branches for comparing different processing methods.

Compatibility of input and output data. A connection between two nodes is valid only if the output of the previous software component meets the input requirements of the next one.

Structural completeness of the configuration. The constructed structure must form a complete route from the initial node to the final result. Isolated nodes, unconnected fragments, or incomplete branches cannot be considered a correct software component pipeline.

Configuration validation before execution. Configurations that violate the rules above must be rejected at the construction stage.

Compliance with these rules ensures the formalized construction of the tree-like directed graph structure of a software component pipeline, limits the set of valid configurations, and creates a basis for automated optimization and further selection of the optimal configuration according to the requirements of a specific EEG signal classification task.

2 Optimization of software component pipelines

The construction of the tree-like directed graph structure of a software component pipeline defines the set of valid configurations; however, by itself, it does not determine which of them is optimal for a specific EEG signal classification task. When alternative configurations, different combinations of methods, and several valid processing variants are available, the task of optimizing the software component pipeline arises. This task consists in selecting the optimal configuration for solving the given problem.

In this study, the optimization of a software component pipeline is presented as a combination of an evaluation method and a search method. The first method provides a quantitative assessment of the effectiveness of each valid pipeline according to the defined metrics, while the second method selects the configuration with the minimum integral value, which represents the optimal software component pipeline.

2.1 Optimization metrics for software component pipelines

For the optimization of a software component pipeline, a set of metrics is proposed that covers both classification quality and computational efficiency: accuracy, F1 score, area under the ROC curve, and normalized time per data sample.

The accuracy metric [4] represents the share of correctly classified samples relative to the total number of samples (7).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, \tag{7}$$

where TP – the number of true positive decisions, TN – the number of true negative decisions, FP – the number of false positive decisions, FN – the number of false negative decisions.

The F1 score [4] is used to evaluate the balance of the pipeline between precision and recall (8):

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, Precision = \frac{TP}{TP+FP}, Recall = \frac{TP}{TP+FN}, \quad (8)$$

The area under the ROC curve [5] characterizes the ability of the software component pipeline to distinguish between classes when the decision threshold changes (9).

$$ROC\_AUC = \int_0^1 TPR(FPR)d(FPR), TPR = \frac{TP}{TP+FN}, FPR = \frac{FP}{FP+TN}, \quad (9)$$

where TPR – the true positive rate, a FPR – the false positive rate.

The normalized time per data sample evaluates the computational efficiency of the pipeline and is defined as the average normalized processing time of one sample (10):

$$NTPS = \frac{T}{N \cdot k}, \quad (10)$$

where k – the number of cross-validation folds, T – the total execution time of all k cross-validation folds, N – the number of EEG signal samples in the dataset.

The proposed set of metrics makes it possible to consider not only the overall classification quality, but also the balance of recognition, the ability to separate classes, and computational efficiency. This creates a basis for further formalization of the optimization of valid software component pipeline configurations.

2.2 Formalized representation of the optimization problem

After the tree-like directed graph structure is constructed, the set of valid configurations of the software component pipeline is formed (11):

$$Pipeline = \{Pipeline_1, \ldots, Pipeline_m\}, \quad (11)$$

where m – the number of valid configurations of the software component pipeline.

Each valid configuration is computed and then analyzed. For this purpose, cross-validation is used. As a result, a set of metric values is obtained for each configuration at separate cross-validation folds. However, the existence of a set of valid configurations alone does not determine which of them is optimal for a specific EEG signal classification task. Therefore, the optimization problem should be considered as a minimization problem, where the smallest integral value corresponds to the optimal configuration of the software component pipeline.

After cross-validation is performed, a set of metric values is obtained for each pipeline at separate folds (12):

$$metric = (metric_1, \ldots, metric_k), \quad (12)$$

where k – the number of cross-validation folds.

In this study, an integral value is introduced for each metric. For this purpose, the metric results are represented as a vector, after which its length is determined (13):

$$Metric = \sqrt{\sum_{j=1}^{k} (metric_j)^2}, \quad (13)$$

After the integral values are calculated, all metrics are brought to a common scale by normalization in the range from 0 to 1. This is necessary because the used metrics have different numerical ranges and different meanings; therefore, without normalization, they cannot be correctly combined into a single objective function (14):

$$Norm\_Metric = \frac{Metric - Metric_{min}}{Metric_{max} - Metric_{min}} \quad (14)$$

Since the optimization of the software component pipeline is considered as a minimization problem, all metrics must be brought to the same direction. Therefore, maximization metrics are transformed into the minimization form (15):

$$Min\_Norm\_Metric = 1 - Norm\_Metric \quad (15)$$

After this, the objective function is introduced. It combines the final integral values of all metrics. For each metric, a weight coefficient is defined by the researcher according to the priorities of a specific EEG signal classification task (16):

$$F(Accuracy, F1, ROC\_AUC, NTPS) = a_{Accuracy} Min\_Norm\_Accuracy + \\ + a_{F1} Min\_Norm\_F1 + a_{ROC\_AUC} Min\_Norm\_ROC\_AUC + a_{NTPS} Norm\_NTPS \\ \text{under the condition: } a_{Accuracy} + a_{F1} + a_{ROC\_AUC} + a_{NTPS} = 1 \quad (16)$$

In this representation, the objective function combines classification quality and computational efficiency into a single integral value of a valid software component pipeline configuration.

To find the optimal configuration, this study uses mixed-integer linear programming. Its use makes it possible to formalize the procedure for selecting one pipeline from the space of valid configurations for which the value of the objective function is minimal (17):

$$min \sum_{i=1}^{m} F(Accuracy, F1, ROC\_AUC, NTPS)_i x_i; \\ \text{under the condition: } \sum_{i=1}^{m} x_i = 1, x_i \in \{0,1\} \quad (17)$$

In this formulation, the optimization of the software component pipeline combines the procedure for determining the integral value for each valid configuration and the procedure for finding the optimal one. This makes it possible to move from empirical comparison of separate variants to a formalized selection of the software component pipeline configuration based on the proposed optimization procedure.

3 Experimental verification of the formalization of software component pipeline construction and optimization

3.1 Initial conditions of the scientific study

For the experimental verification of the proposed approach, a private dataset of EEG signals related to the motor activity of human upper limb fingers was used. This dataset was described in previous studies [6–8]. The usage of real EEG data is important because it makes it possible to verify whether the proposed formalized approach to the construction and optimization of a software component pipeline for EEG signal classification is suitable for practical conditions, where noise, inter-subject variability, and the difficulty of separating classes are present.

3.2 Construction of the tree-like directed graph structure of the software component pipeline

For the experimental verification of the proposed solutions, software components representing the key stages of EEG signal classification were selected: the ICA method [9] at the feature extraction stage, the PCA method [10] at the dimensionality reduction stage, and the SVM [11] and CNN [12] methods at the classification stage (see Fig. 2). The selection of these software components is explained by the fact that they represent different approaches to signal processing and classification and make it possible to verify how the proposed tree-like directed graph structure forms valid software component pipeline configurations when alternative components are available at separate stages.

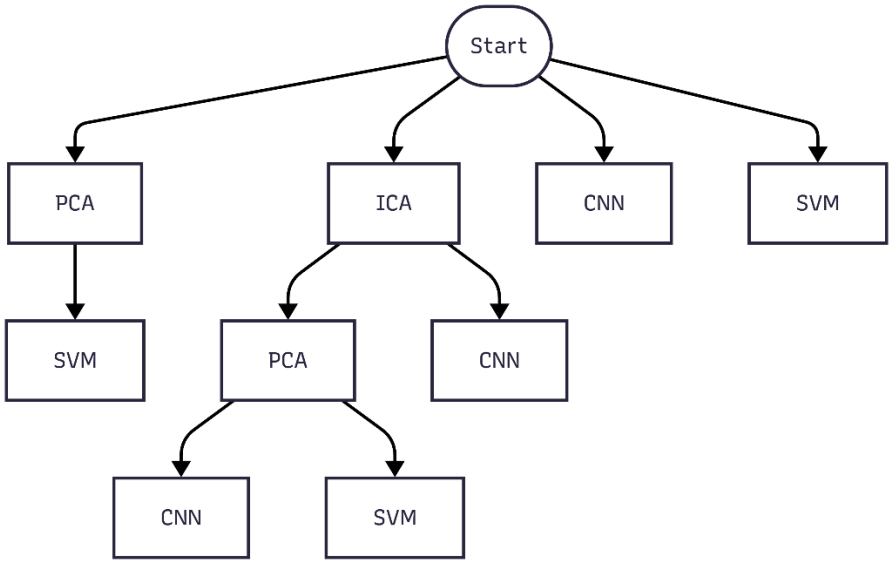

Fig. 2. An example of constructing the tree-like directed graph structure of the software component pipeline for the EEG signal classification task

As shown in Figure 2, the constructed tree-like directed graph structure defines valid transitions between the ICA, PCA, SVM, and CNN software components and thereby forms a set of valid software component pipeline configurations for further optimization. As a result of the construction, 8 valid software component pipeline configurations were formed.

3.3 Computational conditions of the scientific study

The computational verification of the proposed formalization of software component pipeline construction and optimization was performed on a remote personal computer with an AMD Ryzen 7 7800X3D 8-Core Processor, an NVIDIA GeForce RTX 5070 Ti graphics card, and 30.90 GB of RAM. The usage of a remote computing node made it possible to perform computations under real workload conditions, which is important for verifying the practical suitability of the proposed approach when working with a set of valid software component pipeline configurations.

The computation launch was organized as a script that sequentially performed the formation of valid configurations, their computation, cross-validation, and further optimization (see Fig. 3). To implement this process, Scikit-learn [13] was used as the main tool for building machine learning models, Joblib [14] was used to support parallel execution and saving intermediate results, and Dask [14] was used to organize distributed computing.

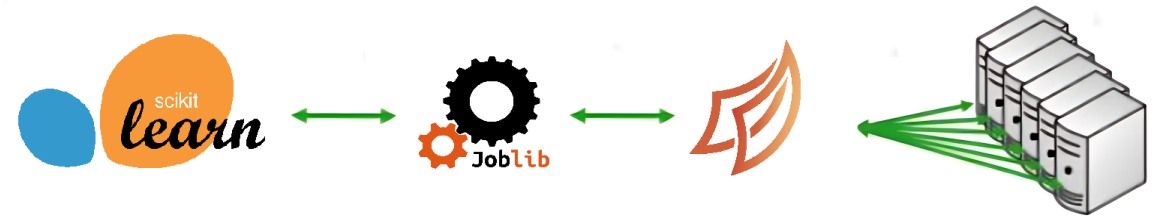

Fig. 3. Scheme for organizing the distributed launch of computations

It is important that this approach to launching computations is not an isolated solution used only within this study, but is based on the computational scheme previously described in the authors' earlier studies [8]. These studies already showed that the combination of Scikit-learn, Joblib, Dask, and distributed execution makes it possible to scale experiments with EEG data, reduce the load on individual system nodes, and ensure efficient computation. In this article, this approach is used as a computational tool for the practical implementation of the formalized construction and optimization of software component pipeline configurations, rather than as an independent object of study.

The computational conditions of the scientific study covered not only the hardware environment, but also the organization of computation and optimization of valid software component pipeline configurations in distributed mode. This made it possible to verify that the proposed solutions can be used for automated optimization and selection of a software component pipeline configuration.

3.4 Optimization results for valid software component pipeline configurations

The optimization was performed within the set of valid software component pipeline configurations formed after constructing the tree-like directed graph structure for the given EEG signal classification task. Within this set, 8 valid configurations were analyzed, and an integral value was computed for each of them.

For the basic verification scenario, the condition of equal importance of all used metrics was adopted. Therefore, the metric weight coefficients in the optimization procedure were set equally. Under this setting, no metric has priority, and the configuration is selected based on their balanced consideration. This makes it possible to verify the operation of the formalized optimization procedure in a neutral scenario, where the decision does not give priority in advance to accuracy, robustness, or computational efficiency.

The optimization results are presented in Table 1. The minimum integral value was obtained for the "PCA → CNN" configuration, which makes it possible to define it as the optimal software component pipeline configuration within the adopted usage scenario. The obtained result shows that, within the constructed set of valid configurations, this configuration provided the best alignment between the evaluation metrics.

Table 1

Optimization results for valid software component pipeline configurations

№	Software component pipeline	Accuracy	F1 score	Area under the ROC curve	Processing time per data sample (s/sample)	Integral value
1	PCA → CNN	0.935993	0.936058	0.998630	0.000584	0.000618
2	ICA → CNN	0.936106	0.936111	0.998613	0.000661	0.001355
3	ICA → PCA → CNN	0.935428	0.935472	0.998592	0.000666	0.004158
4	CNN	0.930889	0.931004	0.998418	0.000538	0.022818
5	ICA → PCA → SVM	0.886896	0.889267	0.996164	0.017326	0.350765
6	ICA → SVM	0.886690	0.889092	0.996162	0.018099	0.357199
7	PCA → SVM	0.774797	0.798156	0.989296	0.033998	0.994036
8	SVM	0.773468	0.796451	0.989264	0.033745	0.998110

The analysis of the results in Table 1 shows that configurations using CNN obtained much lower integral values than configurations using SVM. This means that, within the formed set of valid configurations and under the condition of equal importance of all metrics, CNN-oriented configurations provided a more balanced relation between classification quality and computational efficiency. At the same time, the difference between the "PCA → CNN", "ICA → CNN", and "ICA → PCA → CNN" configurations shows that even with a fixed classifier, the previous stages of the pipeline affect the final efficiency. Therefore, configuration selection cannot be reduced to an isolated comparison of classification methods only.

The obtained results confirm that the proposed optimization procedure makes it possible to move from empirical comparison of separate variants to a formalized selection of the optimal software component pipeline configuration within the set of valid configurations. Thus, the experimental verification shows the practical suitability of the proposed approach for automated selection of the pipeline configuration according to the defined set of metrics.

For additional analysis of the classification behavior of the selected configuration, Figure 4 presents the confusion matrix for the "PCA → CNN" configuration.

	1	12	12345	13	14	2	23	2345	24	3	34	4	5	RETURN
1	98.98%	0.04%	0.10%	0.02%	0.04%	0.28%	0.03%	0.05%	0.03%	0.08%	0.07%	0.09%	0.18%	0.02%
12	0.06%	92.96%	0.10%	3.00%	1.85%	0.02%	1.33%	0.04%	0.41%	0.01%	0.10%	0.03%	0.02%	0.09%
12345	0.06%	0.12%	90.58%	0.08%	0.08%	0.03%	0.13%	3.58%	0.13%	0.06%	0.55%	0.04%	0.02%	4.54%
13	0.03%	3.55%	0.07%	90.77%	3.18%	0.01%	1.49%	0.05%	0.63%	0.02%	0.12%	0.01%	0.01%	0.06%
14	0.03%	2.64%	0.11%	3.91%	83.52%	0.01%	7.53%	0.04%	1.94%	0.02%	0.10%	0.03%	0.01%	0.08%
2	0.17%	0.02%	0.02%	0.02%	0.02%	96.85%	0.01%	0.01%	0.02%	1.59%	0.01%	0.69%	0.56%	0.01%
23	0.05%	1.43%	0.16%	1.22%	4.91%	0.01%	82.47%	0.05%	9.45%	0.02%	0.08%	0.05%	0.00%	0.11%
2345	0.03%	0.06%	4.50%	0.04%	0.03%	0.01%	0.05%	93.47%	0.06%	0.01%	0.63%	0.01%	0.01%	1.09%
24	0.04%	0.55%	0.14%	0.57%	1.07%	0.02%	6.99%	0.04%	90.12%	0.02%	0.20%	0.04%	0.03%	0.18%
3	0.07%	0.03%	0.03%	0.00%	0.01%	1.26%	0.02%	0.01%	0.02%	94.74%	0.02%	3.14%	0.62%	0.01%
34	0.04%	0.05%	0.80%	0.07%	0.03%	0.02%	0.08%	0.63%	0.12%	0.04%	97.58%	0.02%	0.01%	0.51%
4	0.05%	0.02%	0.01%	0.01%	0.01%	0.36%	0.01%	0.03%	0.03%	2.32%	0.02%	95.75%	1.37%	0.01%
5	0.08%	0.05%	0.03%	0.01%	0.02%	0.24%	0.01%	0.02%	0.02%	0.75%	0.01%	2.31%	96.45%	0.01%
RETURN	0.02%	0.13%	8.27%	0.06%	0.07%	0.01%	0.15%	1.41%	0.22%	0.04%	0.74%	0.02%	0.01%	88.85%

Fig. 4. Confusion matrix of the optimal software component pipeline configuration

The confusion matrix for the "PCA → CNN" configuration shows that most classes are classified correctly, since the largest values are concentrated on the main diagonal. The most difficult classes to separate were 23, 14, and RETURN, while the largest errors were observed for the transitions "23 → 2345", "14 → 23", "24 → 23", and "12345 → RETURN". This confirms that even for the optimal configuration, the main difficulties arise between classes with similar motor patterns.

CONCLUSIONS FROM THIS STUDY AND PROSPECTS FOR FURTHER RESEARCH IN THIS DIRECTION

Within this study, the processes of constructing and optimizing software component pipelines for EEG signal classification were formalized. The practical suitability and correctness of the proposed solutions were confirmed through experimental verification using real EEG signals related to the motor activity of human upper limb fingers.

The formalized process of constructing a software component pipeline for EEG signal classification was proposed to be represented as a tree-like directed graph structure. This makes it possible to move from the empirical combination of separate algorithms to the formal definition of valid configurations that can be verified and used in the further optimization procedure. In addition, the tree-like directed graph structure is used to arrange pipeline stages, clearly define connections between software components, and remove uncontrolled combinations of these components. The developed mathematical notation of the tree-like directed graph structure makes it possible to formally describe the set of nodes, directed links, processing stages, and separate software component pipeline configurations. As a result, each pipeline configuration can be considered as a formally defined object suitable for further evaluation and optimization. The rules for constructing the tree-like directed graph structure of the software component pipeline defined in the study make it possible to set requirements for acyclicity, fixed data transfer direction, valid sequence of stages, compatibility of input and output data, and structural completeness of configurations. Due to these construction rules, invalid combinations of software components are rejected before computations are launched.

The study also proposed a formalized procedure for optimizing software component pipelines. The essence of this procedure is that the selection of the optimal software component pipeline configuration for EEG signal classification is performed by solving a minimization problem using multi-metric evaluation, weight coefficients, and

mixed-integer linear programming. This makes it possible to move from a simple comparison of separate metric values to a reproducible and controlled selection of a software component pipeline configuration. The defined set of optimization metrics ensures a comprehensive evaluation of pipeline configurations in terms of classification quality and computational efficiency. The developed mathematical representation made it possible to combine different metrics within a single objective function and ensure a clear selection of the optimal configuration. Flexible control of weight coefficients makes it possible to adapt the optimization result to the requirements of a specific applied task.

The study verified the correctness of the proposed solutions for constructing and optimizing software component pipelines for EEG signal classification, their consistency with each other, and their practical suitability for automated construction and optimization.

The prospects for further research are primarily related to the transition from separate formalization of pipeline construction and optimization to the development of a complete technology and its software implementation as a software system for automated construction and optimization of software component pipelines for EEG signal classification. Further development of this direction should be focused on the implementation of the proposed solutions in real applied and everyday scenarios of BCI. A separate prospect is the improvement of the methods of formalized pipeline construction and optimization, including the extension of construction rules, refinement of constraints, development of mechanisms for taking metrics into account, and adaptation of the configuration selection procedure to different conditions of applied EEG signal classification tasks.

References

[1] Stefanyshyn V. Review of Brain-Computer Interfaces in Motor Imagery: Integrating Machine Learning, Big Data, and High-Performance Computing. CEUR Workshop Proceedings. 2025. URL: https://ceur-ws.org/Vol-4160/paper23.pdf

[2] Farhana U., Ferdous M. Improving Motor Imagery EEG Signals Classification Accuracy with CSP by Available Machine Learning Approach. Journal of Engineering Science. 2021. Vol. 12. P. 67–77. https://doi.org/10.3329/jes.v12i2.54632

[3] Shelishiyah R., Thiyam B., Margaret M. Machine Learning-Based Classification of Hybrid BCI Signals using Mayfly-Optimized Multiclass Weighted Random Forest. International Journal on Recent and Innovation Trends in Computing and Communication. 2023. Vol. 12. P. 29–35. https://doi.org/10.17762/ijritcc.v12i1.7907

[4] Kardam V.S., Taran S., Pandey A. BSPKTM-SIFE-WST: bispectrum based channel selection using set-based-integer-coded fuzzy granular evolutionary algorithm and wavelet scattering transform for motor imagery EEG classification. Signal, Image and Video Processing. 2025. Vol. 19, No. 4. Article 273. https://doi.org/10.1007/s11760-025-03851-z

[5] Zhou L., Tao X., He F., Zhou P., Qi H. Reducing False Triggering Caused by Irrelevant Mental Activities in Brain-Computer Interface Based on Motor Imagery. IEEE Journal of Biomedical and Health Informatics. 2021. Vol. 25, No. 9. P. 3638–3648. https://doi.org/10.1109/JBHI.2021.3066610

[6] Stefanyshyn V., Stefanyshyn I., Pastukh O., Yatsyshyn V., Yakymenko I. Accuracy of software and hardware of computer systems for human-machine interaction. CEUR Workshop Proceedings. 2024. Vol. 3842. P. 178–183.

[7] Stefanyshyn V., Stefanyshyn I., Pastukh O., Kulikov S. Comparison of the accuracy of machine learning algorithms for brain-computer interaction based on high-performance computing technologies. Scientific Journal of the Ternopil National Technical University. 2024. No. 3 (115). P. 82–90. https://doi.org/10.33108/visnyk_tntu2024.03.082

[8] Bryk O., Stefanyshyn I., Stefanyshyn V., Pastukh O. Robustness evaluation of machine learning algorithms for neurocomputer interface software using distributed and parallel computing. Computer Systems and Information Technologies. 2024. No. 2. P. 82–88. https://doi.org/10.31891/csit-2024-2-11

[9] Chen R., Xu G., Jia Y., Zhou C., Wang Z., Pei J., Han C., Wang Y., Zhang S. Enhancement of Time-Frequency Energy for the Classification of Motor Imagery Electroencephalogram Based on an Improved FitzHugh–Nagumo Neuron System. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2023. Vol. 31. P. 282–293. https://doi.org/10.1109/TNSRE.2022.3219450

[10] Wei F., Xu X., Jia T., Zhang D., Wu X. A Multi-Source Transfer Joint Matching Method for Inter-Subject Motor Imagery Decoding. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2023. Vol. 31. P. 1258–1267. https://doi.org/10.1109/TNSRE.2023.3243257

[11] Lu Y., Wang W., Lian B., He C. Feature Extraction and Classification of Motor Imagery EEG Signals in Motor Imagery for Sustainable Brain–Computer Interfaces. Sustainability (Switzerland). 2024. Vol. 16. No 6627. https://doi.org/10.3390/su16156627

[12] Hwaidi, Jamal & Chen, Thomas. Classification of Motor Imagery EEG Signals Based on Deep Autoencoder and Convolutional Neural Network Approach. IEEE Access. 10. 48071-48081. https://doi.org/10.1109/ACCESS.2022.3171906

[13] Scikit-learn. API Reference. URL: https://scikit-learn.org/stable/api/index.html

[14] Dask. Scikit-Learn & Joblib. URL: https://ml.dask.org/joblib.html