Filter Evolution Using Cartesian Genetic Programming for Time Series Anomaly Detection

Andreas Margraf[1] [a], Henning Cui[2], Stefan Baumann[2] and Jörg Hähner[2]

[1]*Fraunhofer IGCV, Am Technologiezentrum 2, 86159 Augsburg, Germany*

[2]*University of Augsburg, Universitätsstraße 2, 86159 Augsburg, Germany*

Keywords: CGP, Evolutionary Learning, Signal Processing, Condition Monitoring, Non-Destructive Testing.

Abstract: Industrial monitoring relies on reliable and resilient systems to cope with unforeseen and changing environmental factors. The identification of critical conditions calls for efficient feature selection and algorithm configuration for accurate classification. Since the design process depends on experts who define parameters and develop classification models, it remains a time-consuming and error-prone task. In this paper, we suggest an evolutionary learning approach to create filter pipelines for machine health and condition monitoring. We apply a method called Cartesian Genetic Programming (CGP) to explore the search space and tune parameters for time series segmentation problems. CGP is a nature-inspired algorithm where nodes are aligned in a two-dimensional grid. Since programs generated by CGP are compact and short, we deem this concept efficient for filter evolution and parameter tuning to create performant classifiers. For better use of resources, we introduce a dependency graph to allow only valid combinations of filter operators during training. Furthermore, this novel concept is critically discussed from a efficiency and quality point of view as well as its effect on classifier accuracy on scarce data. Experimental results show promising results which - in combination with the novel concept - competes with state-of-the-art classifiers for problems of low and medium complexity. Finally, this paper poses research questions for future investigations and experimentation.

1 INTRODUCTION

Production environments depend on reliability and performance of machinery throughout the workflow to ensure a high level of efficiency, quality, and output, to meet customers' needs. Over the course of their operational lifespan, machine elements experience degradation resulting from physical impacts and varying external conditions. Without countermeasures, these effects lead to unexpected failures with possibly catastrophic consequences for the process built around these elements. As an intermediary, *condition monitoring (CM)* allows production machines to run within a defined tolerance window. CM enables engineers to closely monitor key parameters along the processing chain and obtain an up-to-date overall picture of the state of the entire production cycle. For the manufacturing industry, detecting oncoming machine faults helps to reduce unplanned downtime, minimize repair effort and even decrease replacement expenses. Furthermore, companies can minimize costs of equip-

ment failures by planning ahead and timing procurement. Hence, CM becomes an essential factor in optimizing operational efficiency. Parameters that can be measured to monitor the production environment comprise engine power, valve states and pressure of steering fluids, to name a few.

In this paper we will consider the following parts in particular: valve states of a hydraulic system and eddy current data of aluminum surface structures.

As part of CM, *Digital Signal Processing (DSP)* employs signal filters to detect critical states in real-time, enabling failure prediction and maintenance scheduling. However, designing CM systems remains complex due to the need for individual parameterized filters. For this purpose, we propose an *Organic Computing (OC)* approach, employing evolutionary learning for automated digital filter design. OC (Müller-Schloer and Tomforde, 2017) represents a design paradigm using nature-inspired concepts. The evolution of filters is set out to allow for predicting critical states in machine parts, using *Cartesian Genetic Programming (CGP)* for feature selection and OC principles, denoted *self-x properties*. More precisely,

[a] https://orcid.org/0000-0002-2144-0262

the approach is deemed to be self-*configuring*, self-*learning* and self-*optimizing*, thereby employing core principles of OC on signal processing.

As part of CM, we also consider eddy current testing of carbon fiber reinforced plastics (Matvieieva et al., 2018; Bardl et al., 2018) and the adaption of signal filters for depth inspection. In this regard, CM represents a preliminary step to anticipate production downtime and unscheduled replacement of parts by employing *reactive*, *preventive* and *predictive maintenance*.

1.1 Related Work

Various other works have concentrated on automating the CM procedure. However, to the best of our knowledge, the authors have not encountered any relevant publications on filter evolution beyond those already mentioned. Yang et al. presented a CM approach for rolling-element bearings using envelopes (Yang, 2015). In the machine learning related fields of research, Miller et al. introduced Cartesian Genetic Programming (CGP) as a derivative of genetic programming, initially to evolve electronic circuits (Miller and Thomson, 2000). Subsequent work identified great potential of CGP for learning Boolean functions (Miller, 1999) or designing executable FPGA circuits (Sekanina et al., 2011). Since then, several variants of CGP have been proposed and employed in different areas of application (Miller, 2019).

Glette et al. (Glette et al., 2008) presented an overview on evolutionary hardware (EHW) for classification tasks discussing the potential for digital signal processing. Following studies were dedicated to the classification problem of hand prostheses movements with evolutionary algorithms by Boschmann et al. (Boschmann et al., 2009). Kaufmann et al. (Kaufmann et al., 2012) compared different classifiers for electromyography signals using EHW. Also, Jack et al. (Jack and Nandi, 2000) proposed a method for feature selection with GA in condition monitoring. In a DSP related but theory based work, Hammami et al. (Hammami et al., 2018) proposed a filter-wrapper for feature construction using evolutionary learning.

1.2 Structure

The remainder of this paper is structured as follows: We start by introducing the applications in Section 1 and continue by defining the adaptation of CGP for evolutionary learning of signal filters in Section 2. In the following Section 3 we describe the experimental setup in which we elaborate on the datasets we used for testing our hypotheses. We then discuss the re-

sults in Section 4 and conclude with an overview of future applications, their potential, and research goals in Section 5.

2 APPROACH

CM is set to monitor parameters of a machine's condition, e.g., temperature, vibration, or position. The underlying data is usually derived from analogue sensors and converted to the digital domain. In order to evaluate a machine's state, several parts of the same device have to be monitored at once and analyzed. Data processing relies on either statistical approaches (e.g., tolerance, pre-defined parameters, probabilistic methods) or machine learning techniques. In this study, CM is employed to eddy current and valve sensors to demonstrate how reliable classifiers can be designed to identify critical states or make predictions for emerging operational faults.

2.1 Design of Digital Filters

The design workflow for DSP pipelines can be divided into six stages: *(1) signal acquisition, (2) AD conversion, (3) digital filtering, (4) feature selection, (5) dimensionality reduction and (6) classification*. For stages 1 and 2, the system components, i.e. sensors and converters as well as interfaces vary with the technical environment. In the digital domain, though, signal data can be represented in the time and frequency domain and analyzed in different dimensions. This paper takes only stages 3 to 6 into account. Signal and filter requirements share common characteristics: *1) discrete vs. continuous nature, 2) multidimensionality (channels, locality, intensity), 3) high-volume data, and 4) their reproducibility*. When dealing with continuous signals, isolating the core frequency is vital due to signal overlays from neighboring parts, resonances, and noise. Adding filter nodes, such as low-pass or wave rectifiers, to the pipeline allows cleaning and preparation for further processing. Digital filters can be categorized as finite impulse response (FIR) or infinite impulse response (IIR). For signal filter applications that meet requirements such as noise suppression, cutoff frequency, and pass band matching, as well as anomaly detection, we propose an approach utilizing CGP for evolutionary filter design. The methodology renders algorithm development self-configuring and self-adaptive, as elaborated in Section 2.2.

2.2 Adaptation of Cartesian GP

2In general, CGP is an evolutionary algorithm that represents programs as a *directed acyclic graph (DAG)* with nodes arranged in rows and columns on a Cartesian grid. In this section, we outline an approach for grouping DSP filters to avoid non-executable connections but allow all remaining valid combinations. Regular CGP employs a two-dimensional grid to represent the genotype. Three general node types are used for training: *input nodes*, *function nodes*, and *output nodes*. All function nodes F are positioned on the grid and can be connected using vertices to form the DAG. However, not every node needs to be linked to an output node, making them non-coding genes. During training, a high number of nodes remain inactive, which has been found to increase the probability of evading local optima (Turner and Miller, 2015; Cui et al., 2023), partly because it allows more *neutral genetic drift* and favors genetic diversity. Typically, the genotype exhibits a significant number of non-coding genes that have no impact on the phenotype. This property allows for greater variation in the underlying encoding during training. The program P is defined as follows:

$$P := \{G, i, o, n, Ff, r, c, l\} \quad (1)$$

In the set, G represents the genotype, while the number of inputs and outputs of each pipeline are denoted by i and o. The grid is limited by indices r and c which indicate the number of rows and columns, adding up to n nodes. This constraint helps mitigate genetic bloat. The encoding is represented as a sequence of operator nodes as can be seen in Figure 1.

Except for the levels back parameter l, regular CGP configurations do not impose constraints on the order of successive operators, generating a theoretically large number of invalid solutions in early generations. In some previous studies, the possible combinations between operators were considerably limited, as presented by (Margraf et al., 2017). For the sake of efficient search space exploration, we suggest to apply limitations only to prevent invalid solutions during evolution. However, the configuration should allow any executable filter combination to preserve the probability of identifying an optimum. Therefore, any data that has been processed by one operator and passed to the next must carry enough information for reasonable filtering. F consists of f functions forming the search space, while l denotes the levels back parameter. Similar to the modifications proposed by (Margraf et al., ress) for image processing, we introduce a *Dependency Graph $DG(T, E_{DG})$* with operator type set T and connections E_{DG} with operator types $T_i \in T$ with $i = 1, \ldots, n$. The dependency graph

is applied to the columns and thereby defines which columns can be associated with a column under consideration. Each node in the dependency graph represents a column in the grid. To pool the node types, we introduce a program P', derived from P which is defined as follows:

$$P' = \{F, T, G, r\} \quad (2)$$

The number of layers contained in the graph G implicitly define the number of columns c, therefore P' does directly depend on c. The height of G equals its number of columns c and its size, thereby: $c = |G| = height(G)$. Unlike earlier implementations (Leitner et al., 2013)), the arity a of each function varies according to its precondition, allowing to mutate each parameter of a function, therefore:

$$a = max(\#Inputs(F_i)) \quad (3)$$

For the evaluation of DSP filter pipelines, the fitness will be defined as a candidate solution's accuracy on the training data. More precisely, for the set of labeled training feature vectors $X = (x, l)_j$ the fitness of an evolved filter pipeline p is defined by Matthew's correlation coefficient (*MCC*). *MCC* represents the intersection between ground truth and prediction for each series in the series-label pair and is established as:

$$\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$
$$(4)$$

This metric allows to accurately reflect the quality of binary classifications, particularly when dealing with imbalanced datasets. Unlike related metrics, it effectively considers the ratio between positive and negative elements, resulting in a more conclusive evaluation. In this case, false classification is not directly penalized.

2.3 Processing Pipeline Specifications

The training yields executable chains of operators denoted 'processing pipelines'. They consist of operators and their directed connections and represent explicit components in the data flow, equivalent to phenotypes in classical CGP. Figure 2 depicts a sample pipeline represented as a graph with a *Threshold* operator that outputs predictions. The left branch applies the square root of each value on input time series, while the right branch transforms the time series with *FFT* and computes a *Lowpass* to attenuate frequencies before inversing to the time domain.

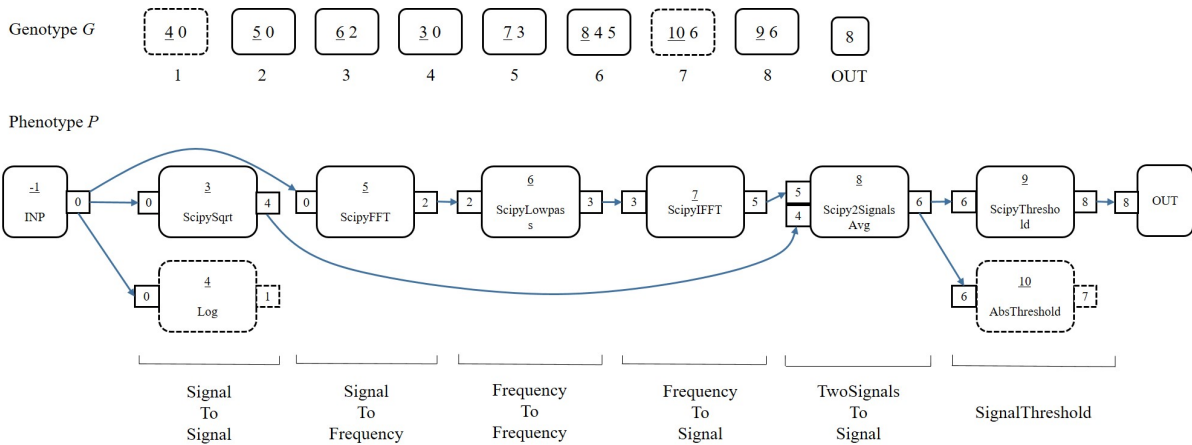*DG* is designed to accommodate repetitions of the core graph. This flexibility is achieved through the

Figure 1: Illustration of a CGP encoding for genotype *G* and phenotype *P*, visualizing the flow of information from input nodes through non-coding (dashed border) and active nodes to the output nodes; in the bottom row, the operator type is depicted which follows the definition given by dependency graph *DG*.
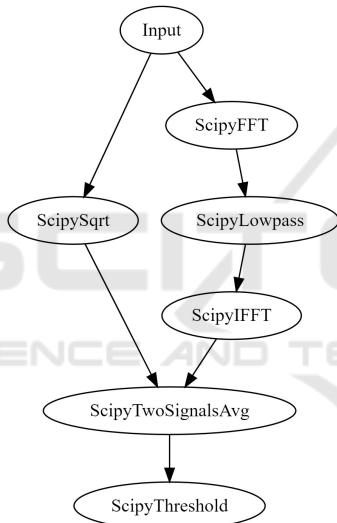

Figure 2: Sample pipeline with filter operators.

configuration of a hyperparameter called *cycled_N*, where *N* represents the number of repetitions of the predefined initial graph. By setting the *cycled_N* value, the number of columns *c* in the graph can be derived in a forward manner, directly from the graph configuration. This way, *DG* becomes scalable by setting one parameter. The extension is introduced to steer transformation between the time and frequency domain, as they only yield valid solutions, if a transformation in one direction is reversed before connecting to the output operator. The simplest solution to enforce this rule would have been to only allow one filter operation between the transformations. The same issue occurs for forked operator pipelines that run parallel to each other, as it is the case for *ScipySqrt* in Figure 2. For each forked pipeline, a join node

TwoSignalsToSignal has to be defined. In order to allow more complex operator combinations while at the same time setting a limitation to a potentially increasing pipeline length, we introduced the parameter *cylced_N* that prevents genetic bloat in the original sense of CGP.
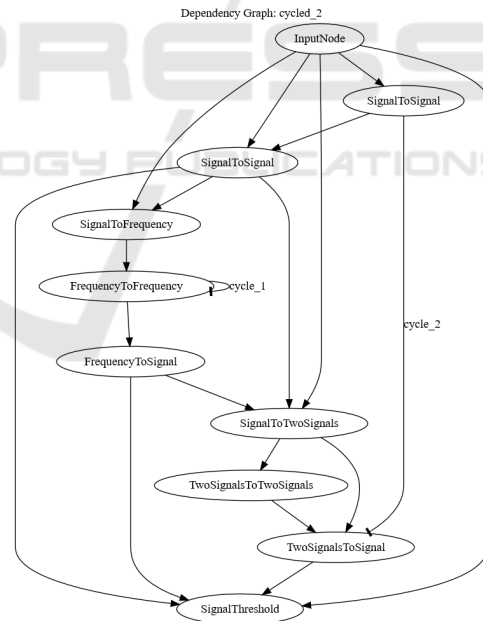

Figure 3: Dependency Graph configuration with 2 cycles.

The proposed adaptation of CGP enables the generation of filter pipelines to accurately segment sections in time series data exhibiting anomalies or defects. However, for multi-class differentiation, the pre-segmented time series data needs to be further processed using subsequent classifiers. For this pur-

pose, common state-of-the-art classification models, e.g. *Artificial Neural Networks (ANN), k-Nearest Neighbors (k-NN), Support Vector Machine (SVM), Decision Trees (DT), Bayesian classifiers, Gaussian mixtures or Hidden Markov Models* can be applied. This is, however, not part of this study, but we encourage the community to pursue future research in this direction.

3 VALIDATION

In order to demonstrate the viability of signal filter evolution by means of CGP, we tested the adapted algorithm on real-world data from *a) a lab-based eddy current testing sensor* and *b) a hydraulics test stand dataset* proposed by (Helwig et al., 2015). The first dataset was created in our lab using an aluminum rod and eddy current sensor. The latter presents a proof of concept to demonstrate automated training of a CM application with several fault scenarios for both fixed and random hydraulic load cycles.

3.1 Experimental Setup

An eddy current sensor operates based on the principle of electromagnetic induction, where a changing magnetic field induces eddy currents in a conductive material. These eddy currents generate a secondary magnetic field that interacts with the sensor, allowing for the detection of conductivity, thickness or surface defects. The sensor consists of a closed coil through which the measured object is passed. For our experiment, an aluminum rod with nuts and 8 successively arranged grooves of different thickness was moved through the sensor while recording the data as shown in Figure 4. This figure plots distance (x-axis in mm) against normalized impedance output (y-axis), showing how eddy currents induced in a conductive object. The surface variations serve as *known anomalies* to trigger sharp deflections in the signal data. With the aid of the defect positions marked with dashed lines on the calibration tube. The eddy current signal deflections can be mapped to specific test defects.

In Helwig et al.'s study, four typical faults of a hydraulic system were simulated and recorded as measurement data. The resulting dataset contains time series data and corresponding labels, and is therefore well-suited to benchmark classification models. The test setup comprises a primary working circuit and a secondary cooling and filtering circuit connected via an oil tank. In repetitive 60-second load cycles, measurement data such as pressure, flow rate, and temperatures are recorded at various locations in the cir-
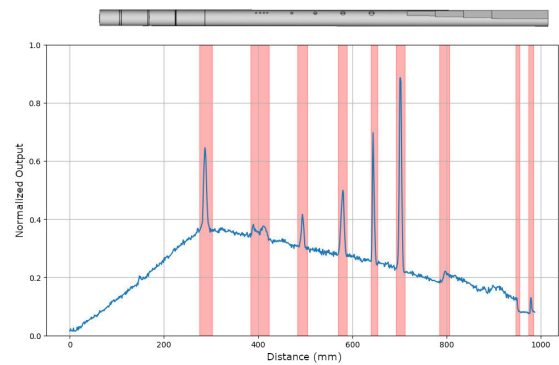

Figure 4: Rendering of the aluminum calibration rod and eddy current signal allocation of the surface with deflections labelled as anomalies with a red overlay.

cuit while the state of four components (pump, cooler, valve, and accumulator) is quantitatively modified. Our proposed configuration was tested on valve values because the anomalies appearing in the readings were of the most explicit among the sensor data.

The algorithm was entirely implemented using the *signal* sub-package of the *SciPy* python library[1]. This is reflected in the names of some filter operators. For the conducted experiments, we introduced variations in key parameters, namely the *scipy series length ssl*, the number of rows *r*, and the cycle configuration of the dependency graph *DG* denoted *cycled_N*, which affects the number of columns *c*. The parameter *ssl* defines the length of the series batch utilized for filtering in one operation that is sequentially performed.

Subsequently, we conducted the experiments using 6 distinct hyperparameter settings for *N* and *ssl* and 4 variations of *r*, resulting in the generation of 144 data tuples for the hydraulic valve data. For the eddy current dataset, only 5 variations of *ssl* were applied, which returned 120 tuples. All experiment data were collected with particular emphasis on the *MCC* fitness score, which ranges between 0 and 1. We minimize the fitness value, which means a fitness of 1 indicates a perfect solution. The evolved operator pipelines were compared against state-of-the-art filters and evaluated through time series segmentation tasks.

3.2 Comparison of Hyperparameters

The top heatmap plot in Figure 5 illustrates the *MCC* fitness values corresponding to various configurations of *cycled_N*, the number of rows *r*, and the series length *ssl*. Similarly, the bottom plot presents the rows *r* associated with each dependency graph con-
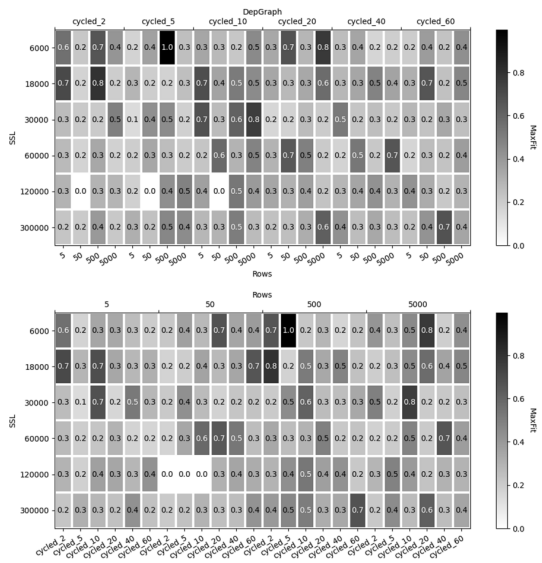
---

[1] see https://scipy.org/

Figure 5: Fitness values achieved on valve data of the hydraulic system for parameters *ssl*, *rows* and *cycled_N*.

figuration *DG*, *ssl*, and their respective mean fitness values *MCC*. In contrast, Figure 6 depicts the results for the eddy current measurement data. They provide visualizations of dependencies among the same four parameters. A range of edge cases was tested for *cycled_N*, including values of $N \in \{2, 5, 10, 20, 40, 60\}$, as well as varying numbers of rows $r \in \{0, 50, 500, 5000\}$, and series lengths $ssl \in \{6k, 18k, 30k, 60k, 120k, 300k\}$. The intervals were chosen to reveal potential trends while balancing computational time and effort. Despite an initial examination not revealing a clear pattern, the obtained results display significant heterogeneity. In the following section, we will critically analyze the conducted experiments.
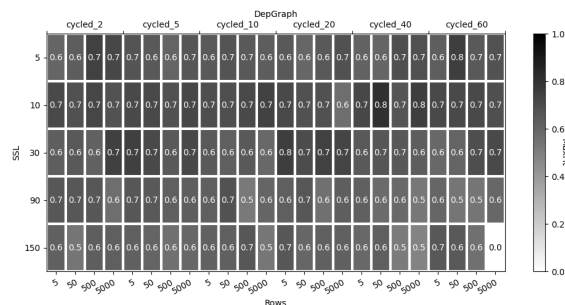

Figure 6: Fitness values achieved on calibration rod scan data for parameters for *ssl*, *rows* and *cols*.

As illustrated in Figure 6, the fitness f_{MCC} of the eddy current filter pipelines achieves values for $f_{MCC} > 0.5$. However, despite the overall high fitness results observed across various hyperparameter

settings, the range still varies from $0.5 < f_{MCC} < 0.8$. It is worth noting that one particular experiment encountered failure due to an assignment error for $ssl = 150, r = 5000, N = 60$. On the other hand, the second experiment conducted on the hydraulic data test demonstrates a more diverse range of fitness values. This wider range spans between $0.2 < f_{MCC} < 1.0$ for the given set of parameters. Additionally, it is important to mention that in this experimental setup, too, three experiments failed due to assignment errors for $ssl = 12k, r = 50, N = \{2, 5, 10\}$. Nevertheless, the results allow for a thorough assessment of the parameters for signal filtering using CGP.

4 RESULTS AND DISCUSSION

This section presents a detailed account of the findings and offers critical reflections on the results obtained through this approach.

4.1 Evaluation of Experiments

The test rod data exhibits clear anomalies in the signal, which can be effectively analyzed using concise and compact filter programs, as can be seen in our experiments. Through the evolutionary process of CGP, favorable fitness values are consistently achieved, generating filter pipelines that contain at most 8 operators for both datasets. For the experiments, the number of generations was set to 150 with 10 iterations. Although this experimental setup represents a straightforward and low-complexity problem, it demonstrates the successful utilization of CGP to evolve compact pipelines with varying hyperparameters.

Upon visually examining the parameter plots, it appears that smaller values of *ssl* tend to yield better results. An analysis of the specific Pearson *r* correlation allows for more objective insights. The correlation between the mean fitness and the *ssl* shows a weak but measurable positive correlation, with a Pearson *r* of $\rho(ssl, \mu_{fit}) > 0.3$. This indicates that, as the series length decreases, the potential fitness increases. There are several potential reasons for this effect. One possibility is that longer series introduce higher complexity, requiring more powerful operators. However, CGP is specifically designed to handle problems with lower complexity. This observation suggests that trainings should be performed with $ssl < 6000$. It should be mentioned that the valve dataset labels were transformed to binary classification (OK / NOK) although the original labels distinguish between 4 fault types. The class definitions

were merged to one class to train for one pipeline. Since several fault types appear in the label data that substantially differ from each other potentially increases the segmentation complexity.

4.2 Analysis and Interpretation

In general, the *MCC* fitness achieved on the eddy current dataset reached favorable *MCC* values independent of the hyperparameter settings. Although the dataset contains anomalies with a comparably large deflection that is supposedly easy to filter, the experiments suggest that the CGP configuration allows evolving compact and well-suited filter pipelines. This is noteworthy because an obvious use case for filter pipeline evolution is the preprocessing of raw signal data before passing it to a more complex classification model.
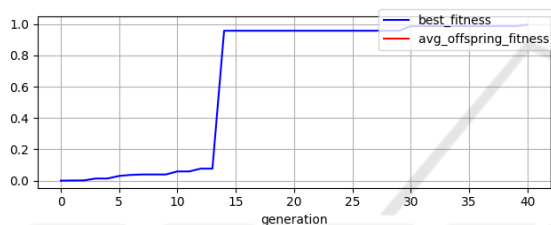

Figure 7: Fitness evolution of the best run on the hydraulic valve dataset.

For one run on the hydraulic valve data, we observed a remarkable top scorer with an $f_{MCC} = 0.99$, which is already considered high. As Figure 7 shows, a significant leap appears in the early stages of the evolution appeared for experimental runs that yield high fitness. This pattern suggests that it would be more efficient to evaluate the evolving pipelines on a small subset of training data initially, as improvements can be achieved more efficiently with fewer data points. As the generations progress, the evaluation set can be gradually expanded to encompass the full-scale dataset for comprehensive assessment. This approach would save valuable time during the early stages. Additionally, an alternative strategy involves computing similarities and analogies between datasets, training on multiple representative datasets, and logging the results. Once the dataset reaches a certain size, efforts can be made to find a suitable match and initiate the evolution with 'pretrained' pipelines to attain a higher fitness state.

5 CONCLUSION

CM and PdM to hold significant potential for enhancing efficiency, due to its high impact in continuously running production systems. In environments with tightly linked schedules that leave little room for unforeseen incidents, maintenance shutdowns set free a chain of complications. The risk for economical damage increases at the same rate as the probability of failures of wear parts while it becomes even less predictable. Therefore, CM will help to keep track of machine conditions and helps to evaluate possible risks more precisely.

5.1 Summary

We presented an evolutionary learning approach for filter evolution to process signal and time series data and detect anomalies or remove noise. The proposed CGP adaptation employed on time series data from different industrial applications show promising results for filter pipeline evolution, specifically for embedded applications or preprocessing in combination with subsequent classifier models. Since data streams from both valve and eddy current sensors were converted to time series, we use a widespread data representation that can be found throughout CM applications. Despite the popularity of ANNs, evolutionary algorithms still exhibit a substantial advantage: the dependency from large training sets is less severe. It should be emphasized that other physical data will likely enhance the success of predictive maintenance, as suggested in this paper. Ultimately, we plan to examine the detection accuracy for different samples of condition sensors to evaluate a wider product range.

The adaption of CGP for evolving signal filters is regarded as an implementation encompassing both the principles of *Automated Design of Processing Pipelines (ADPP)* and *Life Long Learning* (Stein et al., 2018). Since these concepts coincide with Automated Algorithm Design (AAD) and Automated Algorithm Configuration (AAC), there will be future effort to integrate evolutionary learning for DSP into ADPP.

5.2 Outlook

By evolving compact filter pipelines for industrial signal data, the efficiency of monitoring systems is increased. Research work should be continued in this spirit by conducting experiments with smaller search spaces. This would allow for a deeper understanding of how fitness evolves. Also, single operators should be improved to target specific filter problems.

However, the use of further classification models, especially more complex machine learning algorithms, was not examined. Employing further machine learning models to classify the preprocessed signal data can increase the accuracy of anomaly detection and classification. We encourage the community to explore the potential of CGP when interacting with related ML techniques, e.g. by evolving filter programs for data preparation and cleansing before the actual classification. In industrial applications, this approach constitutes a promising concept for future OC efforts with high potential. The adaptation of CGP allows integrating self-optimizing signal filtering in existing condition monitoring systems. This approach can help advance the digitalization of production lines in the manufacturing industry to enhance efficiency, reduce scrap and optimize processes.

REFERENCES

Bardl, G., Kupke, R., Heuer, H., and Cherif, C. (2018). Eddy current testing in cfrp production. *Lightweight Design worldwide*, 11:48–53.

Boschmann, A., Kaufmann, P., Platzner, M., and Winkler, M. (2009). Towards multi-movement hand prostheses: Combining adaptive classification with high precision sockets. In *Proceedings of the 2nd European Conference Technically Assisted Rehabilitation*. Citeseer.

Cui, H., Pätzel, D., Margraf, A., and Hähner, J. (2023). Weighted mutation of connections to mitigate search space limitations in cartesian genetic programming. In *Proceedings of the 17th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*, pages 50–60.

Glette, K., Torresen, J., Kaufmann, P., and Platzner, M. (2008). A comparison of evolvable hardware architectures for classification tasks. In *International Conference on Evolvable Systems*, pages 22–33. Springer.

Hammami, M., Bechikh, S., Hung, C.-C., and Said, L. B. (2018). A multi-objective hybrid filter-wrapper evolutionary approach for feature construction on high-dimensional data. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Helwig, N., Pignanelli, E., and Schutze, A. (2015). Condition monitoring of a complex hydraulic system using multivariate statistics. In *2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*. IEEE.

Jack, L. and Nandi, A. (2000). Genetic algorithms for feature selection in machine condition monitoring with vibration signals. *IEE Proceedings-Vision, Image and Signal Processing*, 147(3):205–212.

Kaufmann, P., Glette, K., Gruber, T., Platzner, M., Torresen, J., and Sick, B. (2012). Classification of electromyographic signals: Comparing evolvable hardware to conventional classifiers. *IEEE Transactions on Evolutionary Computation*, 17(1):46–63.

Leitner, J., Harding, S. L., and Förster, A. (2013). Humanoid learns to detect its own hands.

Margraf, A., Cui, C., Stein, A., and Hähner, J. (2023 in press). Evolving processing pipelines for industrial imaging with cartesian genetic programming. In *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*.

Margraf, A., Stein, A., Engstler, L., Geinitz, S., and Hähner, J. (2017). An Evolutionary Learning Approach to Self-configuring Image Pipelines in the Context of Carbon Fiber Fault Detection. In *Proceedings of the 16th IEEE International Conference for Machine Learning and Applications*.

Matvieieva, N., Schulze, M., Mizukami, K., Kharabet, I., and Heuer, H. (2018). Determination of indication depths at high frequency eddy current testing of cfrp. In *proceedings of the 10th International Symposium NDT in Aerospace*.

Miller, J. F. (1999). An Empirical Study of the Efficiency of Learning Boolean Functions Using a Cartesian Genetic Programming Approach. In *Proc. of 1st Annual Conference on Genetic and Evolutionary Computation*, GECCO'99, pages 1135–1142, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Miller, J. F. (2019). Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines*, pages 1–40.

Miller, J. F. and Thomson, P. (2000). *Cartesian Genetic Programming*, pages 121–132. Springer Berlin Heidelberg.

Müller-Schloer, C. and Tomforde, S. (2017). *Organic Computing: Technical Systems for Survival in the Real World*. Autonomic Systems. Birkhäuser, Cham, 1st edition.

Sekanina, L., Walker, J. A., Kaufmann, P., and Platzner, M. (2011). *Evolution of Electronic Circuits*, pages 125–179. Springer Berlin Heidelberg.

Stein, A., Margraf, A., Moroskow, J., Geinitz, S., and Haehner, J. (2018). Toward an organic computing approach to automated design of processing pipelines. In *ARCS Workshop 2018; 31st International Conference on Architecture of Computing Systems*.

Turner, A. J. and Miller, J. F. (2015). *Genetic programming and evolvable machines*, volume 16, chapter Neutral genetic drift: an investigation using Cartesian Genetic Programming, pages 531–558. Springer Science and Business Media.

Yang, Z. (2015). Automatic condition monitoring of industrial rolling-element bearings using motor's vibration and current analysis. *Shock and Vibration*, 2015.