Towards a New, Semi-Explainable Machine Learning Architecture Using Dimensionality Reduction and Neural-Inspired Grammatical Evolution

Anonymized for review

Abstract

The work aims to present a novel, original approach to solving ML problems that could be handled with neural networks being involved in a direct manner. The main idea of the approach is to use algorithms like PCA to extract features from more complex data, such as images, and then, by using Grammatical Evolution and Neuro-inspired Context-free Grammars, construct binary classifiers for each class. Classifiers may then be subjected to a similar process to produce a rules set that will play the role of the final multi-class classifier. The paper demonstrates proof-of-concept using the MNIST dataset, which was reduced to 154 features using the PCA algorithm.

1 Introduction

The paper is aimed at introducing a new method of image processing without using neural networks. The framework of this method is structured in the following steps. The first step is to effectively decompose the spatial data of an image using the PCA method while also limiting the dimensionality. The second step is to use the Grammatical Evolution method with neuro-inspired grammars [Skrzyński and Ligęza, 2026] to create binary classifiers for each of the classes. Classifiers are trained using the PonyGE2 tool [Fenton *et al.*, 2017], using the size of the area under the ROC curve as a fitness function. The third step is standardization of the inferred values for each classifier. As a last step, based on the inferred standardized values, we determine the maximal score in order to provide the final classification. The method is not yet perfect, achieving an accuracy of about 70% in a 10 class scenario on the MNIST dataset; yet it demonstrates a new approach to image classification that uses a different technique, having the potential to decrease the computational complexity of the inference process.

1.1 Inspiration, motivation and similarity to the CNNs

As it is commonly known, convolutional neural networks (CNN) used nowadays in most image processing problems are built from two main components. The first is the Convolutional Base, which serves as a feature extractor and a dimensionality reduction tool at the same time. The convolutional base passes its flattened output to the MLP (Multi Layered Perceptron), which makes the final decision. So, conceptually, the image understanding process in state of the art architectures is divided into two main conceptual parts: extraction of features and then the interpretation of the presence of those features.

Despite great performance and popularity, most CNN architectures have been reported to have huge redundancy. That is, according to multiple research studies [Li *et al.*, 2017] [Skrzyński and Horzyk, 2025], it appears that after the training process, it is possible to prune approximately 86% of FLOPs (Floating Point Operations) while maintaining or improving accuracy. This means that architectures can be significantly compacted after the model is trained. The only problem that arises is that it is impossible to train a smaller network in the first place as it will show signs of underfitting. This leads to the question: is it possible to construct the final model after pruning, omitting the training, pruning, and retraining procedure?

This paper explores that question by proposing how to take advantage of the abilities of Grammatical Evolution [O'Neill and Ryan, 2001] improved by the use of grammar that creates similar expressions to those that normally form in neural networks [Skrzyński and Ligęza, 2026] to facilitate image processing tasks.

1.2 GE and neuro-inspired CFGs

Grammatical Evolution [Ryan *et al.*, 1998] is a well known method of using genetic programming with Context-Free Grammars (grammar that has rules structured that they always transform single non-terminal symbol to something else, not requiring any other symbol to be present near the given non-terminal) in order to create models. The genetic part is responsible for evolution and improving the fitness score and the grammar is used to translate the genotype to phenotype by simple derivation.

Starting from the root node, we use codons to determine which of the available production rules to use. First, given some non-terminal all productions having it on the left side are selected. Let us assume that there are n of them in the grammar and the value of codon will be denoted as c and then the index of selected rule is $c \mod n$. This process of mapping genotype to phenotype allows one to derive a sentence

that belongs to language of that CFG.

This characteristic makes GE a great tool for generating valid computer programs by giving the right grammar. It's potential can be utilized in multiple ways, including for example generation of mathematical formulas that are ready to evaluate in python.

As neural networks are best in image processing, it is potentially a good direction to build mathematical structures resembling the neural networks, so the neural-inspired CFG is used. This grammar describes a language of mathematical expressions in a form possible to quickly evaluate in python using numpy, that are similar to those encountered in neural networks. That is, the expressions derived do contain activations of the neuron that are computed based on weighted sums of subexpressions that are other neurons. Additionally, selected grammars allow for some mathematical transformations on the data.

1.3 Importance of the method

The method itself does make a sufficient step towards defining high performance explainable models. Here, using GE-created standardized binary classifiers, we can see which of the components exactly contributed to the final result. This has further potential to be developed into an explainable by design model that will be able to distinguish pictures based on automatically created filters and their relationships with classes described by GE evolved expressions. This addresses a commonly debated issue of lack of build in explainability [Rudin, 2019]. Additionally, this method shows clear signs of reduction in terms of computational complexity, reducing complex operations of neural network to much simpler and shorter computations, giving potential to implement such models on edge devices.

2 Current state of knowledge

Grammatical Evolution is already well known to be able to solve classification problems as well as curve fitting [El-Kafrawy, 2012]. GE also comes with explainability that is increasingly desired in AI models, and researchers are trying to facilitate that demand. A constructive proposal to put into practice the ideas of [Rudin, 2019] was presented in [Sepioło and Ligęza, 2024]. Covers at least three elements, reduction of dimensionality through process structuration, separation data processing to components of restricted dimensionality, and knowledge discovery with GE. It was also pointed out by [Sepioło *et al.*, 2025] that grammars play a crucial role in terms of the proper utilization of GE possibilities. Currently, there are multiple attempts to utilize that potential in advancing research around neural networks, for example, by evolving architectures of neural networks such as in [Lima *et al.*, 2020] and [Couchet *et al.*, 2007].

In terms of feature extraction, PCA is mentioned several times as a very powerful tool [Aslam and Rabie, 2023] [Rahmat *et al.*, 2024], which supports the idea of using it as the feature extractor in the presented algorithm.

It is also worth mentioning that the field of multi-class problems solved using GE was also explored by [Fitzgerald *et al.*, 2015], however, the authors of that paper proposed a different approach incorporating concepts from decision tree learning and clustering.

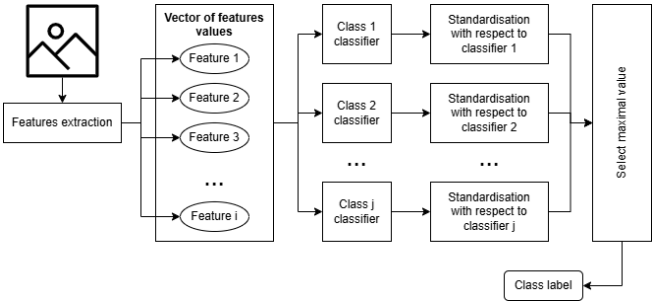

Figure 1: Scheme of the inference process

The proposed idea of using an ensemble of several classifiers was also explored in [Parra *et al.*, 2022], but there GE was responsible for combining the output of multiple tree classifiers, not for producing the classifiers themselves, as is done in this work.

3 Detailed description of the method

The proposed method can be described as multi step process involving decomposition of image to features; then evolving a binary classifier, for each of the classes, and gathering results with standardization and selecting the strongest signal among all to determine the final judgment. In this paper, the implementation follows using PCA with the constant that determines the variance set to 0.95 to convert the images of the MNIST data set into a tabular numerical data set. It is still to be determined what is the best feature extraction method for this approach, as it was not yet tested properly. The next step is evolution of the binary classifier that takes all the evolved features and outputs the value that indicates the element's belonging to the class. For each of the classes present in the dataset, a separate binary classifier is created. The general scheme of the process is shown in Figure 1

The paper contributes by proposing an architecture of classifier composed of multiple binary classifiers evolved in the GE process. This specific design allows the best utilization of GE capabilities. Gathering all classes in a single fitness functions may lead to huge overfitting and according to conducted experiments favoring some classes. Each classifier is evaluated using the AUC metric that balances specificity and sensitivity. This approach allows to work with an imbalanced dataset.

3.1 Training process

Before starting the training, it is crucial to perform the splitting of the dataset into training and test set and stick to this division throughout the training, not to cause any data leakage at any step. As a first part, the feature extraction part must be done, the feature extraction model is fitted only to the training set, but after fitting it, it is necessary to produce also values for the test set for next steps. The tabular dataset can be saved as coma separated value files (CSV) and passed to PonyGE2 as the dataset to evolve the classifier.

In order to conduct this experiment small changes to the PonyGE2 were made. changes allow easier handling of multi-class dataset in creation binary classifiers for each class. It is done by adding a transformation of the dataset in initialization of the fitness function instance that is controlled by parameters. This simply transforms the class column into a binary feature to be equal to or not equal to the desired class.

The next step is to use the neural-inspired CFG in order to evolve classifiers. It was determined experimentally that the depth limit of the derivation tree for our problem should be set to around 17 to 20. For most cases, 250 generations with around 800 individuals were sufficient to achieve good results. For each class, a separate evolutionary process is used targeted at creating classifier if the record belongs to the given class. As was mentioned, the AUC is used as the fitness metric.

The evolved phenotypes are possible to be evaluated in python as numpy vector operations achieving great computation speed due to vectorization. For all samples from training and test classifiers values are inferred and saved. For each classifier based on the Youden's J statistic, the optimal threshold for each classifier is computed. For convenience, we will transform values by subtracting this optimal threshold. To make it possible to compare between multiple classifiers giving values from different populations, the z-score is computed for each value.

3.2 Inference

Based on computed z-score, it is possible to identify for each record which classifier pointed to the highest value of z-score among all of them. The predicted label is defined as the one associated with the classifier producing the highest z-score.

4 The empirical experiment

In order to test if the stated hypothesis that the proposed solution will be capable of multi-class classification is met, an experiment was conducted. The experiment aims at performing image classification on a simple dataset that contains multiple classes of images. To demonstrate the proof-of-concept, the MNIST dataset was selected. The visual scheme of the conducted experiment is presented in Figure 2.

4.1 Data preparation

The Mnist images were subjected to the following preprocessing. First, the images were normalized to scale $[0; 1]$. As the next step, the images were subjected to PCA using the sklearn implementation with parameters $n_components = 0.95$ and $random_state = 42$ in order to enhance reproducibility. Data prepared in this way were saved in csv files. The images had been divided into training and test with a proportion of $80 : 20$.

4.2 Training of binary classifiers

Each binary classifier was trained using Liberal Grammar from [Skrzyński and Ligęza, 2026], maximizing the AUC metric. Multiple values for the evolutionary parameter settings were used and based on plots of the values of the fitness function it was determined that about 250 generations

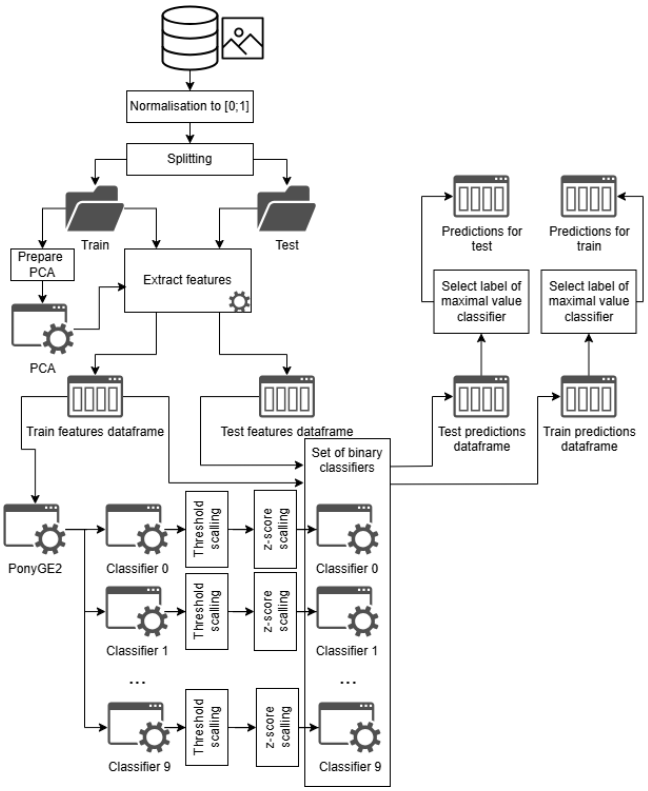

Figure 2: Scheme of the experimental setup with exact paths of data. First pictures are normalised to $[0; 1]$ then split to train and test set. Then using the train the PCA model is created. It is used to transform all pictures into tabular numeric data. Train part of that data is then used in PonyGE2 to create binary classifiers. Each classifier's threshold is adjusted to be exactly at 0. Based on normalised by threshold output the standardisation procedure is conducted for each classifier. Final results are produced by selecting highest z-score of each record

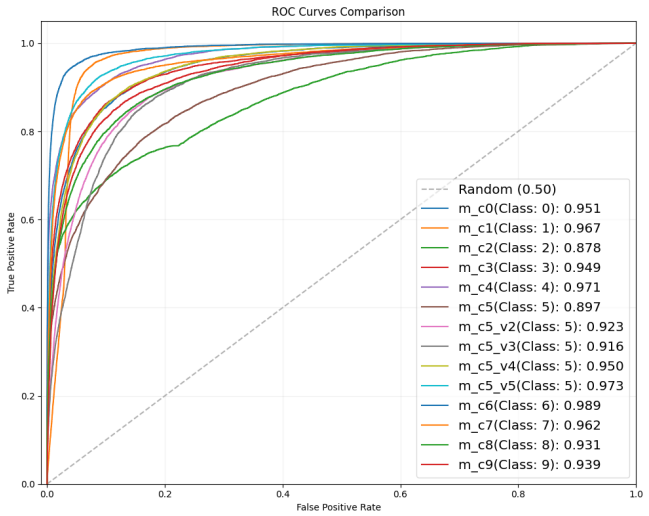

Figure 3: ROC curves for selected classifiers. There are multiple classifiers presented for class number 5 as it was particularly hard to train, each classifier is denoted with another version number. Class is given in bracket alongside each of classes.

are sufficient to produce a good classifier. The population size was determined in the same way and should be set in a range $[750 : 800]$. The crossover probability used was 0.75. Two of ten classifiers posed the most difficulty - classifiers for digits 5 and 3, for them values needed to be adjusted. In order to evaluate the classifier, ROC curves were used that are depicted in Figure 3. Interestingly, the classifiers themselves are not sufficiently powerful to make the prediction and only as an ensemble give valuable results. Example confusion matrices are presented in Figures 5, 4 and 6. Observing them it is clear that it is not possible to derive a final conclusion just based on single classifier output, however, comparing how sure other classifiers it will be possible.

As each of classifiers do produce output that should be perceived as having origin in different populations, they must be adjusted for comparison. First, the threshold value is used to normalize the classifier output by subtracting the value determined by Youden's J statistic. This places the threshold at exactly 0. Then, to perform the comparison, the z-score scaling is performed on each classifier.

4.3 Final inference

The final inference process takes only the z-score values of each classifier output to produce the final label. This step mimics the last step in convolutional networks, giving the index of classifier that was the most sure about the sample. Thanks to the chosen procedure, this process takes less time even using just CPU which would not be the case for CNNs. During the experiment computation of classifier values, scaling then and standardizing takes just a few seconds.

4.4 Results and discussion

The confusion matrices resulting from the computations are depicted in Figure 7 for the training data, in Figure 8 for the test data and in Figure 9 for the entire data set. It is clear that

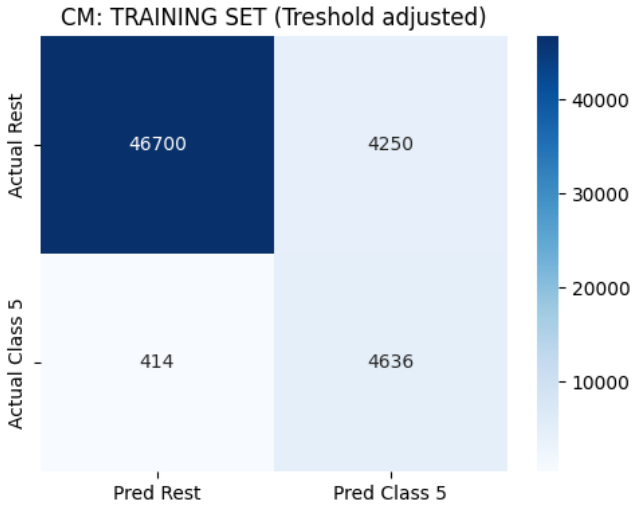

Figure 4: Confusion matrix for 5th version of class 5 classifier achieving $AUC = 0.973$ produced for the training set

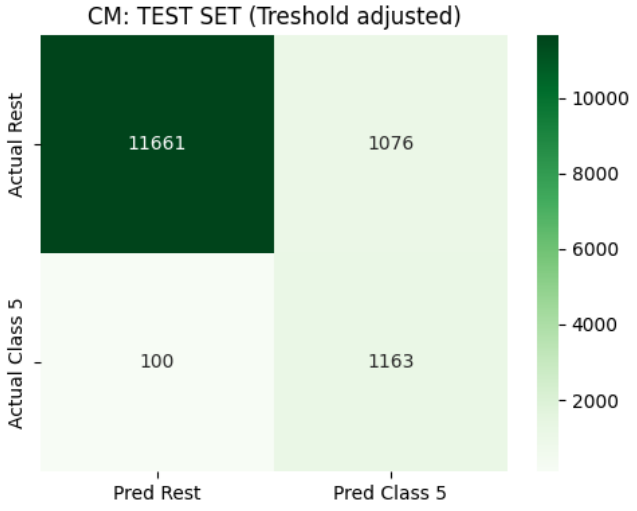

Figure 5: Confusion matrix for 5th version of class 5 classifier achieving $AUC = 0.973$ produced for the test set

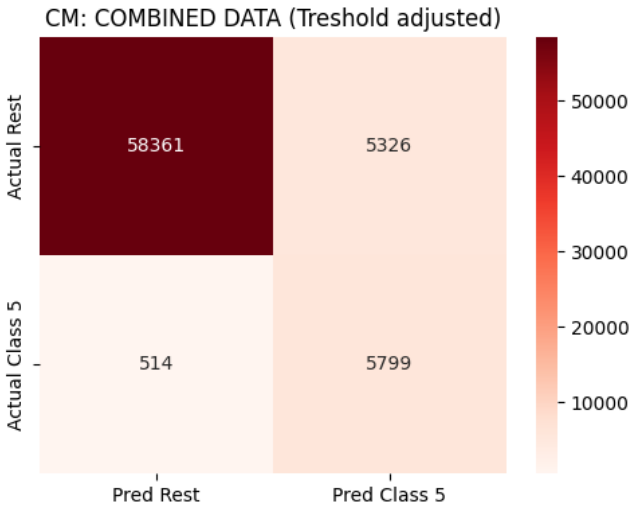

Figure 6: Confusion matrix for 5th version of class 5 classifier achieving $AUC = 0.973$ produced for the combined set

classifier did work properly with small mistakes that, however, can be explained by the visual similarity of the digits that are confused.

5 The final model

The final exportable model is presented conceptually in Figure 10. It is made up of three main layers, the first being some kind of feature extraction model that should produce numerical values. In the experiment presented in the paper, we have used the PCA algorithm here. The next stage is composed of GE evolved classifiers that are in the form of numpy expressions wrapped in function. The function takes features as a vector and computes an output value of the expression returning it. In the next stage, the threshold is centered at zero by subtracting a predetermined optimal threshold value for each classifier from its calculated value. As was described in previous paragraphs, the value is determined by use of Youden's J statistic. The final stage is the z-scalling according to the values established during the training procedure. This gives ready to interpret values of how much each classifier got triggered by the data. In order to give the final answer, the highest value is selected.

The list below specifies all necessary elements of the model that need to be saved in order to allow the model to work properly.

- Feature extraction model used in training.
- Formulas of all classifiers wrapped in a function taking the vector of features and returning the numerical values.
- Optimal threshold value for each classifier determined by Youden's J statistic.
- Mean value for each classifier after centering the threshold to zero.
- Variance value for each classifier after centering the threshold to zero.

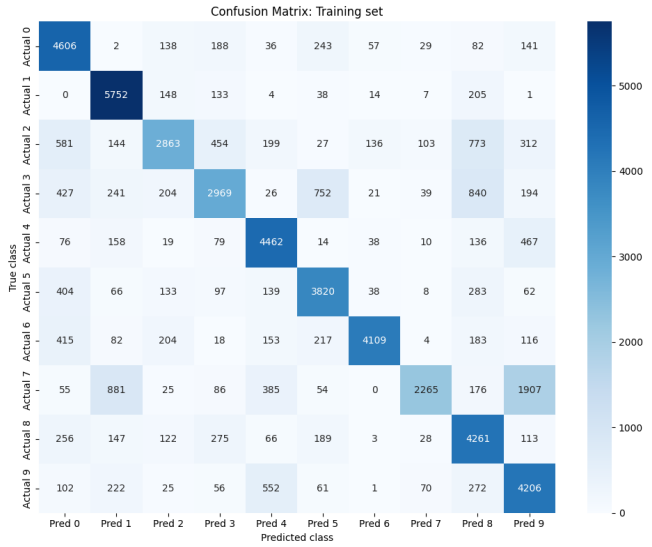

Figure 7: Confusion matrix for final classifier on the training part of dataset. Vertical axis shows the true class and horizontal shows the predicted class. It is clear from the confusion matrix that the classifier produces good results on the training set.

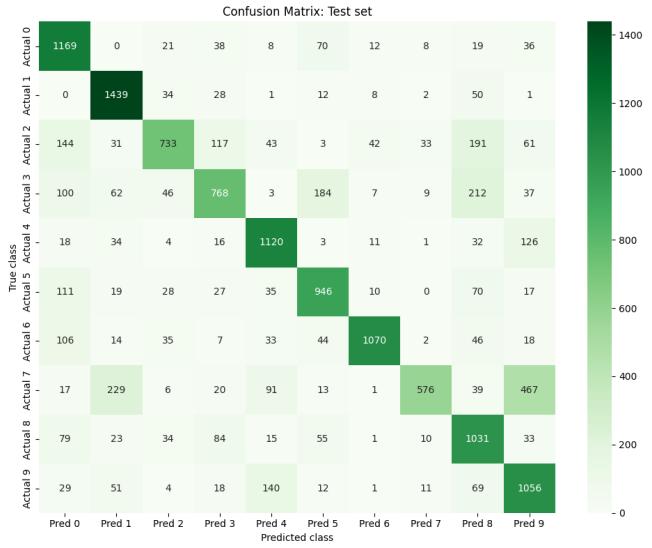

Figure 8: Confusion matrix for final classifier on the test part of dataset. Vertical axis shows the true class and horizontal shows the predicted class. This figure provides a clear evidence that created model is capable of generalisation of knowledge.

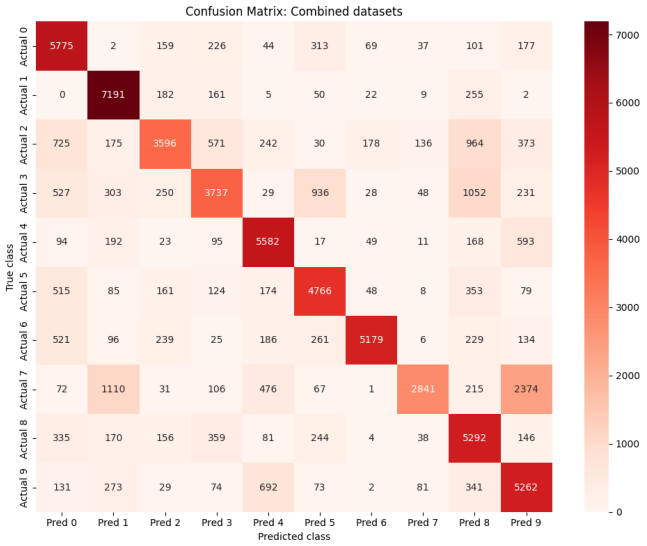

Figure 9: Confusion matrix for final classifier on the training part of dataset. Vertical axis shows the true class and horizontal shows the predicted class. Combined confusion matrix is used to check if the model has some tendencies that should be further examined. From the plot it is clear that case of class 7 requires further improvements but it may be easily explained by visual similarity of digits 7 and 1 as well as in a written font 7 may be similar to 9. The case 7 being predicted as 9 would require further improvement

- Labels of classifiers.

The generalized summarized outline of the model is given by the list of steps below. It constitutes a framework behind this model that might be implemented in future in different ways in order to maximize the performance.

1. Extract features
2. Run each classifier in order to get a numerical score
3. Center threshold to zero by subtracting the saved value of threshold from classifier output
4. Perform z-score scaling for each classifier independently
5. Compare and select highest z-score

6 Contribution to explainability

The conceptual model presented in Figure 10 has a great advantage of simplicity compared to conventional methods. We do have classifier dedicated for each separate type of feature; they usually will be evolved to use only subset of features, allowing us to get insights on which feature's presence or absence gives the highest contribution to the overall score.

Classifiers are compared with each other using simple statistical methods that are easier to compute than dense layers in the final classifier in convolutional neural models.

The overall contribution in the field of explainability of the described method can be formulated as simplifying the classification part of the process and making it more transparent and easy to inspect. Of course, this is given only under conditions in which the features are extracted in a meaningful and

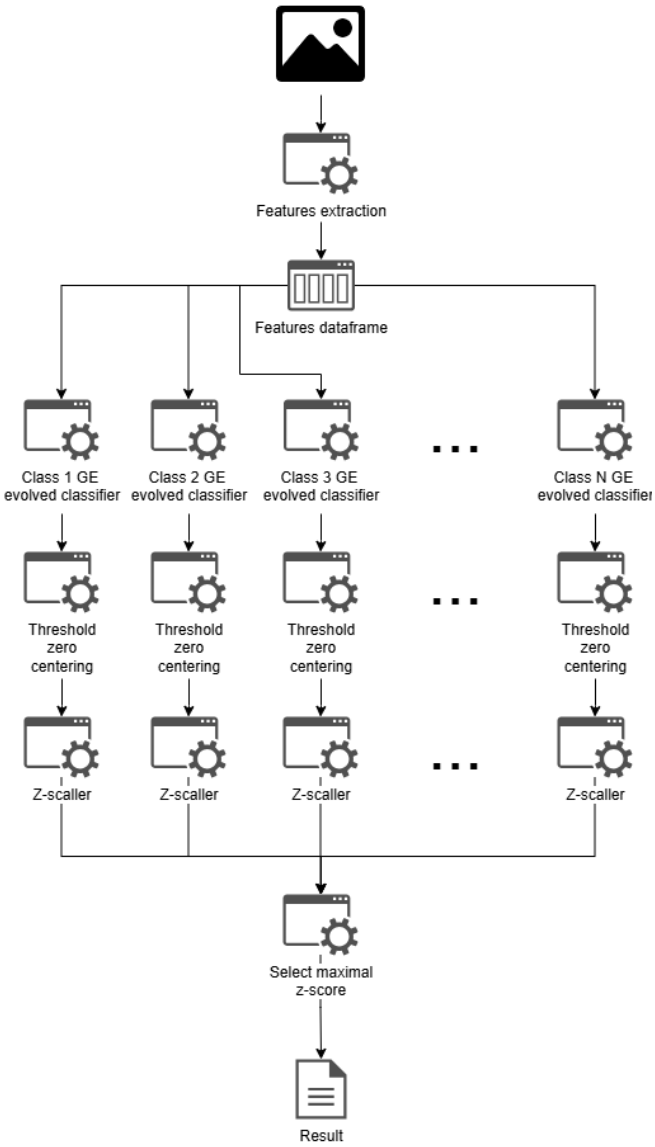

Figure 10: Scheme of the final model and its elements for processing images. The model is composed three main sections. First one being the feature extractor, second being the set of binary classifiers and third one composed of statistics allowing for comparison between model outputs. The last operation is selecting the maximal value among all produced by classifiers.

interpretable way. It can be argued that the PCA used here is not the best in terms of explaining the operations. However, the PCA was used here just as a proof of concept, giving meaningful numerical features from the pictures. Any other meaningful extraction giving well distinguishing features should allow the model to work.

7 Conclusions

This experiment shows that creating smaller models may lead to similar results in terms of accuracy of inference with a much smaller size than state-of-the-art neural solutions. This step confirms that it is possible to build models for complex tasks with build-in explainability and can be treated as a foundation for further research in this area, potentially leading to the creation of much more competitive model architecture with better transparency than neural networks.

It should also be noted that the method is in its early stage of development and in order to understand all limitations of the method and possible issues, further tests have to be carried out. The paper presents a theoretical concept and empirical proof that it yields valid and meaningful results. It was not yet determined how to properly tune the model or what its real limit of performance is. Several aspects, like features extraction, or minimization of lengthy expressions in classifiers pose a field for future work in order to build full knowledge about proposed method.

8 Reproducibility

All experiments conducted in order to achieve the results presented in this paper were properly recorded and described. In order to maintain the highest possible level of transparency, the intermediate results of each phase are also properly documented. The changes to PonyGE2 as well as the results of the experiments are available upon request and also via the Github repository of the author.

9 Future work

Having performed the experiment, the results exceed the expectations as without special tuning the achieved outcome shows signs of great performance. The obvious direction of further studies will be to improve the accuracy of this setup as well as minimize the size of each class classifier - length of the expression.

Further analysis should also include the possibility of changing the feature extraction methods or preparing a custom one with explainability-by-design in mind. This will allow us to actually utilize the potential of this method in the field of explainability.

Additionally, the performance of the method on more complicated datasets should be verified in accordance with the propper feature extraction method.

References

[Aslam and Rabie, 2023] Sidra Aslam and Tamer Farouk Rabie. Principal component analysis in image classification: A review. In *2023 Advances in Science and Engineering Technology International Conferences (ASET)*, pages 1–7, 2023.

[Couchet *et al.*, 2007] Jorge Couchet, Daniel Manrique, and Luis Porras. Grammar-guided neural architecture evolution. In José Mira and José R. Álvarez, editors, *Bio-inspired Modeling of Cognitive Tasks*, pages 437–446, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[El-Kafrawy, 2012] Passent M. El-Kafrawy. Solving classification and curve fitting problems using grammatical evolution. In Vinu V. Das, Ezendu Ariwa, and Syarifah Bahiyah Rahayu, editors, *Signal Processing and Information Technology*, pages 29–33, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[Fenton *et al.*, 2017] Michael Fenton, James McDermott, David Fagan, Stefan Forstenlechner, Erik Hemberg, and Michael O'Neill. Ponyge2: grammatical evolution in python. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, page 1194–1201. ACM, 2017.

[Fitzgerald *et al.*, 2015] Jeannie M Fitzgerald, R Muhammad Atif Azad, and Conor Ryan. Geml: A grammatical evolution, machine learning approach to multi-class classification. In *International Joint Conference on Computational Intelligence*, pages 113–134. Springer, 2015.

[Li *et al.*, 2017] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets, 2017.

[Lima *et al.*, 2020] Ricardo H. R. Lima, Aurora Pozo, Alexander Mendiburu, and Roberto Santana. A symmetric grammar approach for designing segmentation models. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2020.

[O'Neill and Ryan, 2001] M. O'Neill and C. Ryan. Grammatical evolution. *IEEE Transactions on Evolutionary Computation*, 5(4):349–358, 2001.

[Parra *et al.*, 2022] Daniel Parra, Alberto Gutiérrez, Jose-Manuel Velasco, Oscar Garnica, and J. Ignacio Hidalgo. Combining the properties of random forest with grammatical evolution to construct ensemble models. In Juan Luis Jiménez Laredo, J. Ignacio Hidalgo, and Kehinde Oluwatoyin Babaagba, editors, *Applications of Evolutionary Computation*, pages 61–76, Cham, 2022. Springer International Publishing.

[Rahmat *et al.*, 2024] Fariq Rahmat, Zed Zulkafli, Asnor Juraiza Ishak, Ribhan Zafira Abdul Rahman, Simon De Stercke, Wouter Buytaert, Wardah Tahir, Jamalludin Ab Rahman, Salwa Ibrahim, and Muhamad Ismail. Supervised feature selection using principal component analysis. *Knowledge and Information Systems*, 66(3):1955–1995, 2024.

[Rudin, 2019] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[Ryan *et al.*, 1998] Conor Ryan, JJ Collins, and Michael O. Neill. Grammatical evolution: Evolving programs for an arbitrary language. In Wolfgang Banzhaf, Riccardo

Poli, Marc Schoenauer, and Terence C. Fogarty, editors, *Genetic Programming*, pages 83–96, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[Sepioło and Ligęza, 2024] Dominik Sepioło and Antoni Ligęza. Towards Model-Driven Explainable Artificial Intelligence: Function identification with Grammatical Evolution. *Applied Sciences*, 14(13):5950, 2024.

[Sepioło *et al.*, 2025] Dominik Sepioło, Mikołaj Jarosławski, and Antoni Ligęza. Grammar refinement in grammatical evolution using Large Language Models. In *PP-RAI'2025: Progress in Polish Artificial Intelligence Research 6*, pages 22–28. Silesian University Publishers, 2025.

[Skrzyński and Ligęza, 2026] Jakub Skrzyński and Antoni Ligęza. Enhancing grammatical evolution with neural-inspired cfg for explainable ai. In Beata Zielosko, Rafał Doroz, Ireneusz Czarnowski, Janusz Kacprzyk, and Jacek Mańdziuk, editors, *Advances in Artificial Intelligence Research*, pages 17–29, Cham, 2026. Springer Nature Switzerland.

[Skrzyński and Horzyk, 2025] Jakub Skrzyński and Adrian Horzyk. Efficient pruning and compression techniques for convolutional neural networks to preserve knowledge and optimize performance. In Mufti Mahmud, Maryam Doborjeh, Kevin Wong, Andrew Chi Sing Leung, Zohreh Doborjeh, and M. Tanveer, editors, *Neural Information Processing*, pages 381–395, Singapore, 2025. Springer Nature Singapore.