EXPLAINABILITY 2024

The First International Conference on Systems Explainability

ISBN: 978-1-68558-215-9

November 17th – 21st, 2024

Valencia, Spain

EXPLAINABILITY 2024 Editors

Petre Dini, IARIA, USA/EU

EXPLAINABILITY 2024

Forward

The First International Conference on Systems Explainability (EXPLAINABILITY 2024), held on November 17-21, 2024 in Valencia, Spain, inaugurates a series of events dealing with models and metrics to build a documented and provable trust for the developers and users of any kind of system. Explainability helps to validate tracking between system design requirements and current implementation ensuring validation of evolving properties by continuously learning and adapting the original requirements.

Interpretability, Explainability, and Understandability are characteristics needed for any product, system, device, government regulation, or societal law to increase their trustfulness and acceptability by the end-users. Their role is to avoid bias and increase confidence in the systems' output.

Explainability favors interpretability and understandability and should be considered during the requirements, design, deployment and maintenance phases of all software, hardware, and complex systems. To a large extent, explainability is present as a user manual, software requirements tracking and code identification, validation/testing results, interactive interfaces, explanation of models, guidelines for industrial robots, and in any human-driven procedural processes. Desiderata on explainability become more complex for Artificial Intelligence (AI)-based entities/systems in terms of 'thinking' via internal mechanisms and accepting/trusting the output.

Explainability is a sought-after property of any complex 'products'. In AI-based systems, the explanation of the behavior of models for certain critical systems is mandatory. This is a complex task, considering that the behavior is the result of intricate development processes involving humans, algorithms, datasets, and other artificial entities (tools).

This conference was very competitive in its selection process and very well perceived by the international community. As such, it attracted excellent contributions and active participation from all over the world. We were very pleased to receive a large amount of top quality contributions.

We take here the opportunity to warmly thank all the members of the EXPLAINABILITY 2024 technical program committee as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and efforts to contribute to the EXPLAINABILITY 2024. We truly believe that thanks to all these efforts, the final conference program consists of top quality contributions.

This event could also not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the EXPLAINABILITY 2024 organizing committee for their help in handling the logistics and for their work that is making this professional meeting a success.

We hope the EXPLAINABILITY 2024 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in system explainability research. We also hope that Valencia provided a pleasant environment during the conference and everyone saved some time for exploring this beautiful city

EXPLAINABILITY 2024 Steering Committee

Thomas Fehlmann, Euro Project Office AG, Zurich, Switzerland

Mahdi Jalili, RMIT University, Australia
Fairouz D Kamareddine, Heriot-Watt University, Scotland

EXPLAINABILITY 2024 Publicity Chair

Francisco Javier Díaz Blasco, Universitat Politecnica de Valencia, Spain
Ali Ahmad, Universitat Politecnica de Valencia, Spain

EXPLAINABILITY 2024

Committee

EXPLAINABILITY 2024 Steering Committee

Thomas Fehlmann, Euro Project Office AG, Zurich, Switzerland
Mahdi Jalili, RMIT University, Australia
Fairouz D Kamareddine, Heriot-Watt University, Scotland

EXPLAINABILITY 2024 Publicity Chair

Francisco Javier Díaz Blasco, Universitat Politecnica de Valencia, Spain
Ali Ahmad, Universitat Politecnica de Valencia, Spain

EXPLAINABILITY 2024 Technical Program Committee

André Artelt, Bielefeld University, Germany
Esteban Bautista, Université du Littoral Côte d'Opale, France
Dalmo Cirne, Workday Inc., USA
Thomas Fehlmann, Euro Project Office AG, Zurich, Switzerland
Ming Gong, Bing Experiences | Microsoft, China
Nada Ahmed Hamed Sharaf, German International University, Cairo, Egypt
Amr Hendy, Microsoft, USA
Mahdi Jalili, RMIT University, Australia
Fairouz Kamareddine, Heriot-Watt University, Edinburgh, UK
John Kos, Georgia Institute of Technology, USA
Shudong Liu, University of Macau, Macau
Minheng Ni, Hong Kong Polytechnic University, Hong Kong
Albert Solé Ribalta, Universitat Oberta de Catalunya, Spain
Chang D. Yoo, Korea Advanced Institute of Science and Technology (KAIST), South Korea
Abdou Youssef, George Washington University, USA
Daqing Yun, Harrisburg University, USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

A Two-Dimensional Computational Model for DNA/RNA Classification

Dorota Bielińska-Wąż
Department of Radiological Informatics and Statistics
Medical University of Gdańsk
80-210 Gdańsk, Poland
email: djwaz@gumed.edu.pl

Piotr Wąż
Department of Nuclear Medicine
Medical University of Gdańsk
80-210 Gdańsk, Poland
email: phwaz@gumed.edu.pl

***Abstract*—The 2D-Dynamic Representation of DNA/RNA Sequences, a two-dimensional computational model introduced by the authors, is reviewed for its application in the classification of deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) sequences. This method falls under the bioinformatics category known as Graphical Representations of Biological Sequences. The goal of these methods is to provide tools for the graphical and numerical classification of the sequences.**

Keywords–bioinformatics; machine learning; decision trees; descriptors

I. INTRODUCTION

Graphical Representations of Biological Sequences constitute a branch of alignment-free bioinformatics methods, focusing on the graphical and numerical classification of sequences [1] [2]. Each approach reveals different aspects of similarity, and a comprehensive review can be found in [3].

This document presents a method introduced by the authors and called 2D-Dynamic Representation of DNA/RNA Sequences [4]–[10]. Specifically, this method is combined with the C5.0 decision tree algorithm [10]. Details related to the graphical representation of the sequences within this method and the numerical characteristics of the graphs ("descriptors") are described in Section II.

II. METHODS AND RESULTS

Graphically, the sequences are represented by 2D-dynamic graphs (sets of material points in a 2D space). The graphs were obtained by following a "walk" in the XY coordinate system, using the basis vectors representing the specific nucleobases: A = (-1,0), G = (1,0), C = (0,1), and T/U = (0,-1) [4] [9]. Examples of these graphs are shown in Figures 1-3.

The following descriptors of the 2D-Dynamic Representation of DNA/RNA Sequences, some of which are analogous to dynamics, are considered:

- Coordinates (μ_x, μ_y) of the centers of mass of the 2D-dynamic graphs [4]:

$$\mu_\gamma = \frac{1}{N}\sum_{i=1}^{p} m_i\gamma_i, \quad \gamma = x, y, \quad N = \sum_{i=1}^{p} m_i, \quad (1)$$

  where x_i, y_i are the coordinates of mass m_i in the Cartesian coordinate system for which $(0, 0)$ is the origin of all the sequences and N is the length of the sequence (equal to the total mass of the graph) and p is the number of the material points in the graph.
- Principal moments of inertia (I_{11}, I_{22}) of the 2D-dynamic graphs [4].

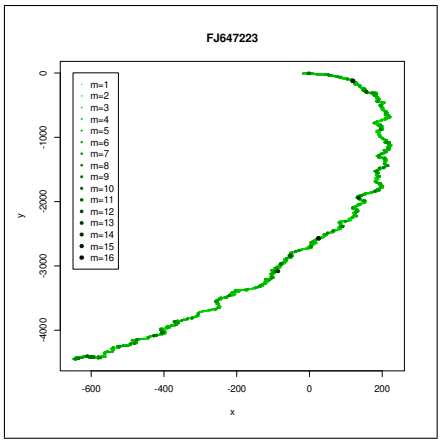


Figure 1. 2D-dynamic graph representing the complete genome sequence of embecovirus (GenBank accession number FJ647223).

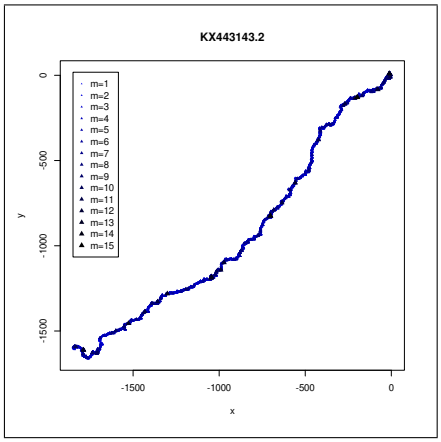


Figure 2. 2D-dynamic graph representing the complete genome sequence of deltacoronavirus (GenBank accession number KX443143.2).

The moment of inertia tensor is defined by the matrix

$$\hat{I} = \begin{pmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{pmatrix} \quad (2)$$

with elements

$$I_{xy} = I_{yx} = -\sum_{i=1}^{p} m_i x_i^\mu y_i^\mu, \quad (3)$$

$$I_{xx} = \sum_{i=1}^{p} m_i (y_i^\mu)^2, \quad (4)$$

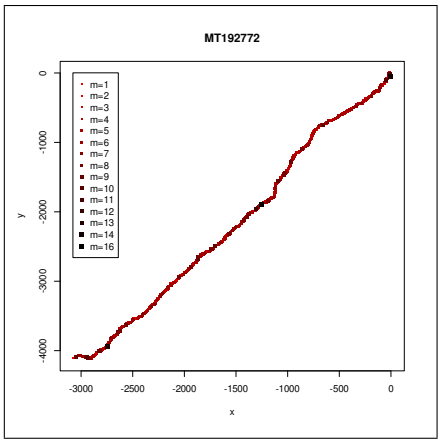


Figure 3. 2D-dynamic graph representing the complete genome sequence of the SARS-CoV-2 virus (GenBank accession number MT192772).

$$I_{yy} = \sum_{i=1}^{p} m_i (x_i^{\mu})^2, \tag{5}$$

where x_i^{μ}, y_i^{μ} denote the coordinates of mass m_i in the Cartesian coordinate system with the origin at the center of mass of the graph. Principal moments of inertia are equal to the solutions $I = I_{11}, I_{22}$ of equation

$$\begin{vmatrix} I_{xx} - I & I_{xy} \\ I_{xy} & I_{yy} - I \end{vmatrix} = 0. \tag{6}$$

- Moments of the mass-density distributions [5].

  The n-th moment of a discrete distribution ρ_E is defined as

  $$M_{E,n} = c_E \sum_i \rho_{E_i} E_i^n,$$

  $E = x, y$ and the normalization constant

  $$c_E = \left(\sum_i \rho_{E_i} \right)^{-1}.$$

  Moments normalized to a mean value equal to zero ($M'_{E,1} = 0$) are

  $$M'_{E,n} = c_E \sum_i \rho_{E_i} (E_i - M_{E,1})^n.$$

  The moments for which the variance is additionally equal to 1 ($M''_{E,2} = 1$) are also considered:

  $$M''_{E,n} = c_E \sum_i \rho_{E_i} \left[\frac{(E_i - M_{E,1})}{\sqrt{M_{E,2} - (M_{E,1})^2}} \right]^n.$$

- Angles between the x axis and the principal axes of inertia of the 2D-dynamic graphs [6].

  One of the angles smaller than $\frac{\pi}{2}$ is chosen.

- Mass overlaps of the 2D-dynamic graphs [6].
- Descriptors $(D_1^x, D_2^x, D_1^y, D_2^y)$ related to a relation between the coordinates of the center of mass and the principal moments of inertia of the 2D-dynamic graphs [7]:

  $$D_k^{\gamma} = \frac{\mu_{\gamma}}{I_{kk}}, \quad k = 1, 2; \quad \gamma = x, y. \tag{7}$$

- Graph radius [8]:

  $$g_R = \sqrt{\mu_x^2 + \mu_y^2}. \tag{8}$$

- Matrix elements of the moments of inertia tensor (I_{xx}, I_{yy}, I_{xy}) of the 2D-dynamic graphs [10].

2D-Dynamic Representation of DNA/RNA Sequences has been applied for the similarity analysis of:

- histone H4 coding sequences of different species [4]–[7];
- α-globin coding sequences of different species [4] [7];
- complete genome sequences of the Zika virus [8] [9];
- 20 most common subtypes of influenza A virus [10].

III. Conclusion

In summary, the 2D-Dynamic Representation of DNA/RNA Sequences is an effective tool for both graphical and numerical comparison of the sequences. Notably, combining this method with the C5.0 decision tree algorithm has yielded high mean accuracy in predicting the subtype of the influenza A virus, with over 90% correct predictions. This high number of correct predictions confirms the good explainability of the considered systems. Therefore, the method will be applied in the future to interpret our experimental data.

References

[1] K. E. Wade, L. Chen, C. Deng, G. Zhou, and P. Hu, "Investigating alignment-free machine learning methods for HIV-1 subtype classification", Bioinformatics Advances vol. 4, Art. No. vbae108, 2024.

[2] D. Bielińska-Wąż, P. Wąż, A. Błaczkowska, J. Mandrysz, A. Lass, and P. Gładysz, J. Karamon, "Mathematical Modeling in Bioinformatics: Application of an Alignment-Free Method Combined with Principal Component Analysis", Symmetry vol. 16, Art. No. 967, 2024.

[3] D. Bielińska-Wąż, "Graphical and numerical representations of DNA sequences: Statistical aspects of similarity", J. Math. Chem. vol. 49, pp. 2345–2407, 2011.

[4] D. Bielińska-Wąż, T. Clark, P. Wąż, W. Nowak, and A. Nandy, 2D-dynamic representation of DNA sequences, Chem. Phys. Lett. vol. 442, pp. 140–144, 2007.

[5] D. Bielińska-Wąż, W. Nowak, P. Wąż, A. Nandy, and T. Clark, Distribution moments of 2D-graphs as descriptors of DNA sequences, Chem. Phys. Lett. vol. 443, pp. 408–413, 2007.

[6] D. Bielińska-Wąż, P. Wąż, and T. Clark, Similarity studies of DNA sequences using genetic methods, Chem. Phys. Lett. vol. 445, pp. 68–73, 2007.

[7] P. Wąż, D. Bielińska-Wąż, and A. Nandy, Descriptors of 2D-dynamic graphs as a classification tool of DNA sequences, J. Math. Chem. vol. 52, pp. 132–140, 2014.

[8] A. Nandy, S. Dey, S.C. Basak, Bielińska-Wąż, and P. Wąż, Characterizing the Zika Virus Genome - A Bioinformatics Study, Curr. Comput. Aided Drug Des. vol. 12, pp. 87–97, 2016.

[9] D. Panas, P. Wąż, D. Bielińska-Wąż, A. Nandy, and S.C. Basak, 2D-Dynamic Representation of DNA/RNA Sequences as a Characterization Tool of the Zika Virus Genome, MATCH Commun. Math. Comput. Chem. vol. 77, pp. 321–332, 2017.

[10] D. Panas, P. Wąż, D. Bielińska–Wąż, A. Nandy, and S.C. Basak. An Application of the 2D-Dynamic Representation of DNA/RNA Sequences to the Prediction of Influenza A Virus Subtypes, MATCH Commun. Math. Comput. Chem. vol. 80, pp. 295–310, 2018.

3D-Dynamic Representation of DNA/RNA Sequences: A Review

Piotr Wąż
Department of Nuclear Medicine
Medical University of Gdańsk
80-210 Gdańsk, Poland
email: phwaz@gumed.edu.pl

Dorota Bielińska-Wąż
Department of Radiological Informatics and Statistics
Medical University of Gdańsk
80-210 Gdańsk, Poland
email: djwaz@gumed.edu.pl

***Abstract*—The research aims to develop new bioinformatics techniques known in the literature as Graphical Representation Methods. This methodology allows for the calculation of numerical values describing deoxyribonucleic acid (DNA) and ribonucleic acid (RNA) sequences, enabling both graphical and numerical analysis of their similarities and differences. This document provides an overview of a bioinformatics method we introduced, referred to as 3D-Dynamic Representation of DNA/RNA Sequences. In this method, the sequences are represented as sets of material points in 3D space, forming "3D-dynamic graph". Numerically, this three-dimensional dynamic graph is characterized by quantities analogous to those used in classical dynamics. The accuracy of this approach is high, allowing to distinguish sequences that differ by just one nucleobase. One application of this method is the characterization of viral genome sequences. Specifically, combining the 3D-Dynamic Representation of DNA/RNA Sequences with the random forest algorithm effectively classifies subtypes of influenza A virus strains.**

Keywords–supervised learning; bioinformatics; biostatistics; graphical methods; machine learning; random forest; Boruta algorithm

I. INTRODUCTION

This presentation describes a computational method we developed, known as the 3D-Dynamic Representation of DNA/RNA Sequences [1]–[4], which generalizes our previous 2D approach [5].

This approach is part of a broader category of techniques known as Graphical Representation Methods, which allow for both graphical and numerical comparisons of objects. Each method offers a unique perspective on similarity, and new techniques are continually being developed (for reviews, see [6]–[8]).

3D-Dynamic Representation of DNA/RNA Sequences aims to compare the sequences by representing them as sets of material points in 3D space, referred to as "3D-dynamic graphs." The distribution of these points and the calculation of their numerical characteristics ("descriptors") are described in Section II.

II. METHOD AND RESULTS

The method is based on shifts (or *walks*) in 3D space [1]. Nucleobases in a DNA/RNA sequence are represented by basis vectors: adenine A=(-1,0,1), cytosine C=(0,1,1), thymine/uracil T/U=(0,-1,1), and guanine G=(1,0,1). The walk begins at the origin point (0,0,0). This point is shifted by a basis vector corresponding to the first nucleobase in the sequence. At the end of this vector, a mass m=1 is placed, which serves as the starting point for the next shift representing the second nucleobase. This process is repeated for each nucleobase in the sequence. The resulting set of material points, which represents the entire sequence, is termed a 3D-dynamic graph (analogous to the 2D-dynamic graph used in the 2D method). Examples of 3D-dynamic graphs representing the complete genome sequences of embecovirus, the SARS-CoV-2 virus, and deltacoronavirus are shown in Figure 1. The differences between the sequences are clearly visible in the graphs. The calculations were conducted using nucleotide sequence data obtained from GenBank. FJ647223, MT192772, and KX443143.2 are the accession numbers corresponding to the sequences in this database.

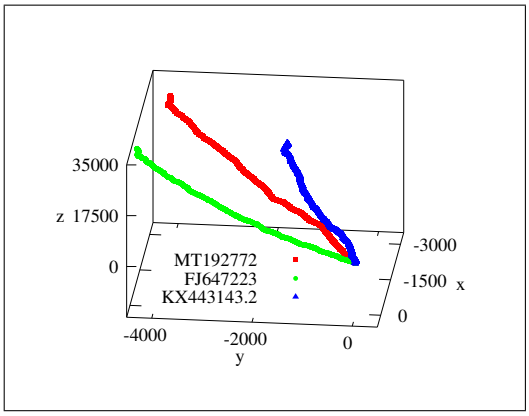


Figure 1. 3D-dynamic graphs.

We use the following descriptors (numerical characteristics of the 3D-dynamic graphs):

- Coordinates (μ_x, μ_y, μ_z) of the center of mass of the graph

$$\mu_\gamma = \frac{\sum_{i=1}^{N} m_i \gamma_i}{\sum_{i=1}^{N} m_i}, \quad \gamma = x, y, z, \tag{1}$$

where x_i, y_i, z_i are the Cartesian coordinates of mass m_i with point $(0, 0, 0)$ being the origin of the coordinate system and N is the length of the sequence. Since $m_i = 1$ for all material points, the total mass of the sequence is equal to the length of the sequence $N = \sum_{i=1}^{N} m_i$. The coordinates of the center of mass of the 3D-dynamic graph can then be expressed as:

$$\mu_\gamma = \frac{1}{N} \sum_{i=1}^{N} m_i \gamma_i, \quad \gamma = x, y, z. \tag{2}$$

- The principal moments of inertia (I_1, I_2, I_3) of the graph, where the moment of inertia tensor is defined by the matrix

$$\hat{I} = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \quad (3)$$

with elements

$$I_{aa} = \sum_{i=1}^{N} m_i \left[(b_i^\mu)^2 + (c_i^\mu)^2 \right], \quad (4)$$

and

$$I_{ab} = I_{ba} = -\sum_{i=1}^{N} m_i a_i^\mu b_i^\mu, \quad (5)$$

where $\{a, b, c\} = \{x, y, z\}$, $a \neq b \neq c$ and the coordinates $(x_i^\mu, y_i^\mu, z_i^\mu)$ of m_i are determined in the center-of-mass of the graph coordinate system. The principal moments of inertia are equal to the solutions $I = I_1, I_2, I_3$ of the characteristic equation of $\hat{I}$:

$$\begin{vmatrix} I_{xx} - I & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} - I & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} - I \end{vmatrix} = 0. \quad (6)$$

- Matrix elements of the moment of inertia tensor of the graph $(I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz})$.
- Graph radius, defined as

$$g_R = \sqrt{\mu_x^2 + \mu_y^2 + \mu_z^2}. \quad (7)$$

- Descriptors D_k^γ,

$$D_k^\gamma = \frac{\mu_\gamma}{I_k}, \quad k = 1, 2, 3; \quad \gamma = x, y, z, \quad (8)$$

that depict a relation between the coordinates of the center of mass and the principal moments of inertia of the graph.
- Normalized principal moments of inertia of the graph (r_1, r_2, r_3):

$$r_k = \sqrt{\frac{I_k}{N}}. \quad (9)$$

- The values of C_{ik}.
The relative orientation of the new and old coordinate systems can be described by cosines of appropriately defined angles:

$$C_{ik} \equiv \cos(M_i, Q_k), \quad i, k = 1, 2, 3. \quad (10)$$

M_1, M_2 and M_3 mean the planes (X, Y), (X, Z) and (Y, Z), respectively. Similarly, Q_1, Q_2, Q_3 denote the planes (Ω_1, Ω_2), (Ω_1, Ω_3), (Ω_2, Ω_3).

The descriptors derived from the 3D-Dynamic Representation of DNA/RNA Sequences have proven effective for the similarity analysis of:

- histone H4 coding sequences of different species and α-globin coding sequences of different species [1];
- β-globin genes of different species [2];
- complete genome sequences of dengue virus [3];
- 20 most common subtypes of influenza A virus [4].

Notably, it has been demonstrated that combining the 3D-Dynamic Representation of DNA/RNA Sequences with the random forest algorithm effectively classifies subtypes of influenza A virus strains [4]. In these studies, the following 22 descriptors were considered: the 3 coordinates of the center of mass $\mu = \{\mu_\gamma : \gamma = x, y, z\}$; the 6 elements of the inertia tensor $J = \{I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}\}$; the 3 principal moments of inertia $I = \{I_k : k = 1, 2, 3\}$; the graph radius g_R; and the set of 9 parameters $D = \{D_k^\gamma : k = 1, 2, 3; \ \gamma = x, y, z\}$. The relevance of these descriptors was assessed using the Boruta algorithm, which employs Breiman's random forest concept to compute normalized importance.

Recently, we extended the 3D-Dynamic Representation of DNA/RNA Sequences to a four-dimensional method, applying it to the bioinformatics characterization of the SARS-CoV-2 virus [9] and to studies on the genetic diversity of Echinococcus multilocularis in red foxes in Poland [10]. In particular, the distribution of clusters in the classification maps generated using the 4D-Dynamic Representation of DNA/RNA sequences supports the hypothesis that SARS-CoV-2 may have originated in bats and pangolins [9].

III. Conclusion

In summary, 3D-Dynamic Representation of DNA/RNA Sequences allows for both graphical and numerical comparisons of the sequences, with enhanced classification effectiveness when combined with the random forest algorithm [4]. It is especially important to focus on developing tools that could be used to characterize unidentified viruses. In the future, we plan to evaluate the explainability of the systems under consideration using new descriptors of 3D-dynamic graphs, such as those that describe the direction of the sum of eigenvectors in the 3D space.

References

[1] P. Wąż and D. Bielińska-Wąż, "3D-dynamic representation of DNA sequences", J. Mol. Model. vol. 20, Art. No. 2141, 2014.

[2] P. Wąż and D. Bielińska-Wąż, "Non-standard similarity/dissimilarity analysis of DNA sequences", Genomics vol. 104, pp. 464–471, 2014.

[3] D. Bielińska-Wąż, D. Panas, and P. Wąż, "Dynamic representations of biological sequences", MATCH Commun. Math. Comput. Chem. vol. 82, pp. 205–218, 2019.

[4] D. Bielińska-Wąż, P. Wąż, and D. Panas, "Applications of 2D and 3D-Dynamic Representations of DNA/RNA Sequences for a Description of Genome Sequences of Viruses", Comb. Chem. High T. Scr. vol. 25, pp. 429–438, 2022.

[5] D. Panas, P. Wąż, D. Bielińska-Wąż, A. Nandy, and S.C. Basak, "2D–Dynamic Representation of DNA/RNA Sequences as a Characterization Tool of the Zika Virus Genome", MATCH Commun. Math. Comput. Chem. vol. 77, pp. 321–332, 2017.

[6] A. Nandy, M. Harle, and S. C. Basak, "Mathematical descriptors of DNA sequences: development and applications", Arkivoc vol. ix, pp. 211–238, 2006.

[7] M. Randić, M. Novič, and D. Plavšić, "Milestones in Graphical Bioinformatics", Int. J. Quant. Chem. vol. 113, pp. 2413–2446, 2013.

[8] S. Mizuta, "Graphical Representation of Biological Sequences", In Bioinformatics in the Era of Post Genomics and Big Data; I.Y. Abdurakhmonov, Ed.; IntechOpen: London, UK, 2018.

[9] D. Bielińska-Wąż and P. Wąż, "Non-standard bioinformatics characterization of SARS-CoV-2", Comput. Biol. Med. vol. 131, Art. No. 104247, 2021.

[10] D. Bielińska-Wąż, P. Wąż, A. Lass, and J. Karamon, "4D-Dynamic Representation of DNA/RNA Sequences: Studies on Genetic Diversity of *Echinococcus Multilocularis* in Red Foxes in Poland", Life vol. 12, Art. No. 877, 2022.

Explain Yourself

Expanding and optimizing models to enable fast Shapley value approximations

Holger Ziekow, Peter Schanbacher, Valentin Göttisheim
Faculty of Business Information Systems
Furtwangen University
Furtwangen, Germany
email: {Holger.Ziekow, Peter.Schanbacher, Valentin.Goettisheim}@hs-furtwangen.de

***Abstract* — This paper addresses the problem of providing fast and accurate approximations of Shapley values for neural networks by embedding the approximation directly into the network architecture. The approach is tested on a synthetic and a real world dataset. The results demonstrate that integrating Shapley value approximations into the loss function enables making a trade-off between explainability and prediction accuracy, optimizing both aspects. This method yields accurate approximations while improving the model's explainability, making it more stable and easier to explain in practical applications.**

Keywords - Explainable AI; Machine Learning; Neural Networks; Shapley value approximation.

I. INTRODUCTION

In various applications, understanding and explaining the behavior of neural networks is crucial for both internal management decision-making and meeting the requirements of regulators and external stakeholders. As neural networks are increasingly deployed in critical areas such as finance, healthcare, and autonomous systems, the need for transparency and explainability becomes paramount. Stakeholders need to trust that the models are making decisions based on relevant and understandable factors, and they must be able to justify these decisions to regulatory bodies and customers alike [1].

A powerful tool for gaining insights into the relevance of attributes in these models is the use of Shapley values [2]. Originating from cooperative game theory, Shapley values provide a fair distribution of the total gain generated by a coalition of players, attributing a value to each player's contribution. When applied to neural networks, Shapley values help users understand how each input feature contributes to the model's prediction. They are prized for their desirable properties, such as fairness, efficiency, and consistency, making them an ideal choice for feature attribution. A significant challenge with Shapley values is that their exact evaluation is computationally expensive, with the complexity growing exponentially with the number of input features [3]. This computational burden makes them impractical for large-scale applications involving high-dimensional data. To mitigate this, researchers have developed various approximation methods. Notably, the authors in [4] introduced polynomial-time approximations, which significantly reduce the computational load while still providing useful insights into feature importance.

This work advances this field by demonstrating that the approximation of Shapley values can be seamlessly integrated into the training process of neural networks. Specifically, a method is proposed where the outputs of interest from the neural network are extended to include these approximated Shapley values. This integration occurs during the training phase, ensuring that the model not only learns to make accurate predictions but also provides explanations for these predictions concurrently.

A key benefit of this integration is that it enables a direct trade-off between model accuracy and Shapley value approximation. In addition, this approach enables improved explainability of the model as well as the immediate availability of explanations.

By integrating Shapley value approximations during training, the neural network converges to a state that is inherently easier to explain. For instance, the network's responses to changes in input features become more stable. This smoothing effect is often a desirable property, especially in domains where stakeholders need to understand the model's behavior in intuitive terms. It prevents scenarios where minor changes in input result in disproportionately large and unexpected changes in the output, which can be challenging to justify to customers and regulators [1]. An explainable model enhances trust and facilitates better decision-making.

Additionally, the approximated Shapley values are produced as a direct result of the model's predictions. This means that for every prediction the model makes, an accompanying explanation is immediately available. This capability is appealing in applications requiring high-frequency predictions and where each decision needs to be justified on the spot.

The approach is particularly valuable in applications where the model undergoes a single training phase followed by numerous predictions, each requiring an explanation. This ensures that the model not only performs well in terms of predictive accuracy but also remains transparent and explainable throughout its operational lifecycle. By embedding the approximation of Shapley values into the training process, the approach strikes a balance between computational efficiency and the need for clear,

understandable explanations, meeting the demands of both operational efficiency and regulatory compliance.

The remainder of the paper is structured as follows: Section 2 discusses the related work and the inclusion of Shapley values into the model's prediction is laid out in Section 3. Section 4 presents an analysis with the data and model applied. The results are discussed in section 4. Section 5 summarizes and concludes.

II. Related Work

Shapley values, originating from cooperative game theory, have become a fundamental tool for feature attribution in machine learning models [2]. They offer a fair distribution of the total gain generated by a coalition of players, attributing a value to each player's contribution [3][5]. However, their exact computation is computationally expensive, leading to the development of various approximation methods [6]. This section reviews these methods, highlighting the limitations they present, and the gaps the proposed approach aims to address.

Feature-removal approaches are central to feature contributions in Shapley value calculations [6]. They involve systematically removing features and assessing the impact on the model's output. The primary types are: (1) Baseline Shapley values where missing features are replaced with values from a baseline sample, such as zeros, means, or medians. This approach is simple to implement and interpret; however, the choice of baseline can be arbitrary and may not accurately represent the data distribution [5][7]. (2) Marginal Shapley values calculate the marginal expectation of the model output by treating absent features as random variables following their marginal distribution. It involves evaluating the model with subsets of features including and excluding the feature of interest. It provides a more accurate estimate of feature importance by considering the marginal distribution of features. However, it is computationally more expensive as it requires multiple model evaluations for different subsets of features [7]. (3) Conditional Shapley values which define the game by the conditional expectation of the model output, where absent features are treated as following a conditional distribution given the observed features. It considers the interdependencies between features [7]. This most accurately accounts for the conditional dependencies between features, providing a realistic assessment of feature importance. However, it is highly complex and computationally intensive due to the need for estimating conditional distributions, which can be challenging, especially in high-dimensional data.

To address the computational challenges of exact Shapley value calculations, various approximation strategies have been developed. These strategies can be broadly categorized into model-agnostic approximations, which are applicable to any model type, and model-specific approximations, which are tailored to specific model structures. Model-agnostic approximations include methods such as interactions-based method for explanation (IME) [9] and KernelSHAP [5][10]. IME utilizes stochastic sampling to provide unbiased estimates of Shapley values. While broadly applicable to various models, it is computationally intensive. KernelSHAP also employs a sampling-based approach, reducing computational load but still requiring significant resources.

In contrast, model-specific approximations are tailored to particular model structures. TreeSHAP [7] leverages the inherent structure of decision trees to compute exact Shapley values efficiently. It offers faster and more precise calculations but is limited to tree-based models. Similarly, LinearSHAP [11] computes Shapley values exactly for linear models with linear time complexity. It performs well for linear relationships, however, is not suitable for other models. While approximation methods like KernelSHAP and IME provide useful insights with reduced computational demands, they suffer from high variance and are still resource intensive. Assumption-based methods like TreeSHAP and LinearSHAP offer solutions with lower computational costs but are restricted to specific model types.

Some research has focused on considering Shapley value approximations into the model architecture itself to balance accuracy and computational efficiency. For instance, ShapNets [12] are designed to facilitate easier estimation of Shapley values through specific network architectures, enhancing both explainability and performance. Deep Approximate Shapley Propagation [4] leverages uncertainty propagation to estimate Shapley values, providing deterministic results with moderate computational requirements.

The proposed approach distinguishes itself by embedding Shapley value approximations directly into the neural network training process. This integration ensures that the model's predictions are inherently more explainable due to more stable responses to input feature changes. Additionally, it allows for the immediate availability of explanations with each prediction, a crucial advantage in settings requiring frequent and justifiable decisions. By embedding the Shapley value approximation into the network architecture, the proposed method achieves a balance between computational load and the need for clear, understandable explanations. It also enables an explicit trade-off between model performance and quality of Shapley value approximations. The proposed integrated approach offers a novel solution that enhances both explainability and efficiency, meeting the demands of real-world applications requiring transparency and accountability.

III. Modeling Shapley Values

Shapley values are a well-established method to understand the impact of an attribute on the outcome [5]. Consider a data set of N attributes and a model f mapping each subset S of the attributes to real numbers (i.e., a prediction). The Shapley value quantifies the importance of attribute i to the prediction. To determine the effect, a model $f_{S\cup\{i\}}$ using data $x_{S\cup\{i\}}$ for a subset S of features including

feature i and a model f_S using data x_S without feature i. Now for all possible subsets $S \subseteq F\backslash\{i\}$ the impact of withholding feature i is calculated. The Shapley values are calculated based on the weighted average of all possible differences.

$$\phi_i(x_S) = \sum_{S \subseteq F\backslash\{i\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!}\left[f_{S\cup\{i\}}\left(x_{S\cup\{i\}}\right) - f_S(x_S)\right]$$

Computing Shapley values requires the evaluation of all possible feature subsets, which makes it infeasible for common practical applications with many features to consider. Shapley values sampling is most frequently used to approximate the Shapley values [13]. Despite the approximation, it still requires considerable calculation time. The standard approach to predict outcome y based on input x, is to minimize the objective $f = \arg\min_f E\left[\left(y - f(x)\right)^2\right]$. This work aims to predict the Shapley values of the features as well, hence optimizing function $g: \mathbb{R}^N \to \mathbb{R}^{N+1}, g(x) \to \left(y, \phi_1(x), .., \phi_n(x)\right)$ such that we minimize:

$$\mathrm{g} = \arg\min_{\mathrm{g}} \mathrm{E}\left[\left(\mathrm{y} - \mathrm{g}_0(\mathrm{x})\right)^2 + \lambda \sum_{\mathrm{i}=1}^{\mathrm{N}} \left(\phi_\mathrm{i}(\mathrm{x}) - \mathrm{g}_\mathrm{i}(\mathrm{x})\right)^2\right]$$

The hyperparameter λ can be used for a trade-off between the standard approach (λ=0) and a joint prediction of outcome y and the Shapley values ϕ_i (λ>0). The hyperparameter λ controls the balance between prediction accuracy and Shapley value approximation. At λ = 0, the model optimizes accuracy, while increasing λ improves explainability by incorporating Shapley values, albeit with some loss in accuracy. Higher λ values shift the focus more toward generating accurate Shapley values.

IV. Experimental Setup

As a test model, a neural network with a three-node input layer, a hidden layer of 16 neurons, another hidden layer of 8 neurons and a four-neurons output layer (see Figure 1) is built. The output contains y_j as well as the three Shapley values $\phi_1(x_j)$, $\phi_2(x_j)$, $\phi_3(x_j)$ for $x_j = (x_{0j}, x_{1j}, x_{2j})$. For the hidden layers, a leaky ReLu is used ($\alpha = 0.1$). The MSE is optimized using the ADAM [14] optimizer.

The model is trained once with minimizing the MSE of the output of interest y_j only and no weight on accurate Shapley value approximations (λ=0). A second model is trained for the joint prediction of the output of interest as well as the Shapley values (λ=1). A third model is trained with joint prediction of the output of interest and a very high weight on Shapley value approximations (λ=1000). A batch size of one was chosen for pragmatic reasons. In each forward pass we compute the target Shapley values of the model with an existing technique. In our tests we used KernelExplainer from the SHAP library [5]. However, this may be replaced with any other method. We use these values to compute the error for $\phi_1(x_j)$, $\phi_2(x_j)$, $\phi_3(x_j)$.

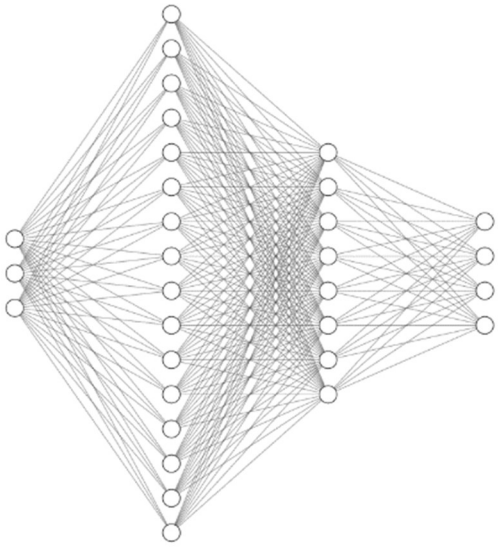


Figure 1. Architecture for the neuronal network (created with https://alexlenail.me/NN-SVG/index.html)

For bigger λ values we expect an increase of the MSE based on the outcome y_j, as the introduction of the Shapley values leads to a biased prediction. We also expect reduced errors for the Shapley value approximations as λ increases. Furthermore, we expect simpler relations between feature values and their corresponding Shapley values, which are easier to approximate. This should be apparent when plotting the feature values against the targeted Shapley values (in our tests computed with KernelExplainer from the SHAP library [5].

A. Experiments with synthetic data

For illustration and initial analysis, we use a synthetic dataset generated as follows. The target variable y_j is created using the linear relationship:

$$y_j = 2 \cdot x_{0j} + \frac{1}{2}\epsilon_j, j \in \{1, \ldots, 1000\}$$

where x_{0j} is the first feature, and ε_j represents independent and identically distributed (i.i.d.) noise drawn uniformly from the interval [0, 1]. The second feature x_{1j} is also i.i.d. and uniformly distributed, generated independently from the same interval. The third feature x_{2j} is then derived from a non-linear transformation of x_{1j} and y_j as follows:

$$x_{2j} = \left(x_{1j} + y_j\right)^{\frac{1}{4}}, j \in \{1, \ldots, 1000\}$$

We use 80% of the generated data as the training set and 20% as the test set. The synthetic data was designed to exhibit both simple and complex relationships between the features (x_{0j}, x_{1j}, x_{2j}) and the target variable y_j. This setup allows us to demonstrate the desired trade-off between prediction

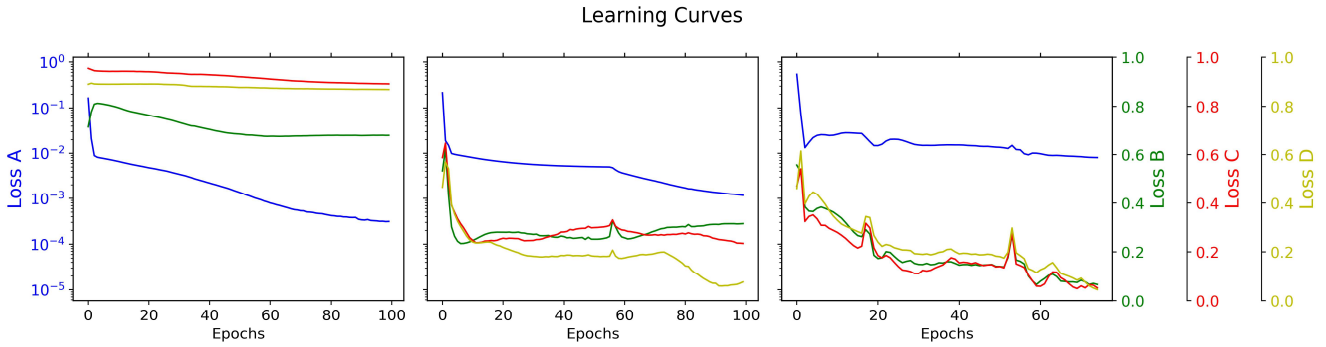


Figure 2. MSE for $\lambda \in \{0,1,100\}$ (left to right) models for the outcome of interest y (blue) and the corresponding Shapley values (green, red, orange).

accuracy and the approximability of Shapley values. The model and training procedure are implemented as described above. The number of epochs was chosen based on the learning curves observed across all tests, ensuring that training did not stop prematurely due to a sudden error spike in any model. The resulting learning cures are shown in Figure 2. The model outputs A, B, C, D, represent the model prediction y (i.e., y=A) and the predicted Shapley values for the features 1, 2, 3.

As expected, higher λ values drive down the errors for Shapley value predictions and increase the prediction error for the target A. In detail the MSE for the model λ=0 is 0.0003, while the MSE for the λ=100 model is 0.001. It is also observed – as expected – that the partial dependency plots show increasingly simpler structures (see Figure 3). The resulting curves become more smooth and less scattered. This makes them easier to approximate and easier to interpret by humans.

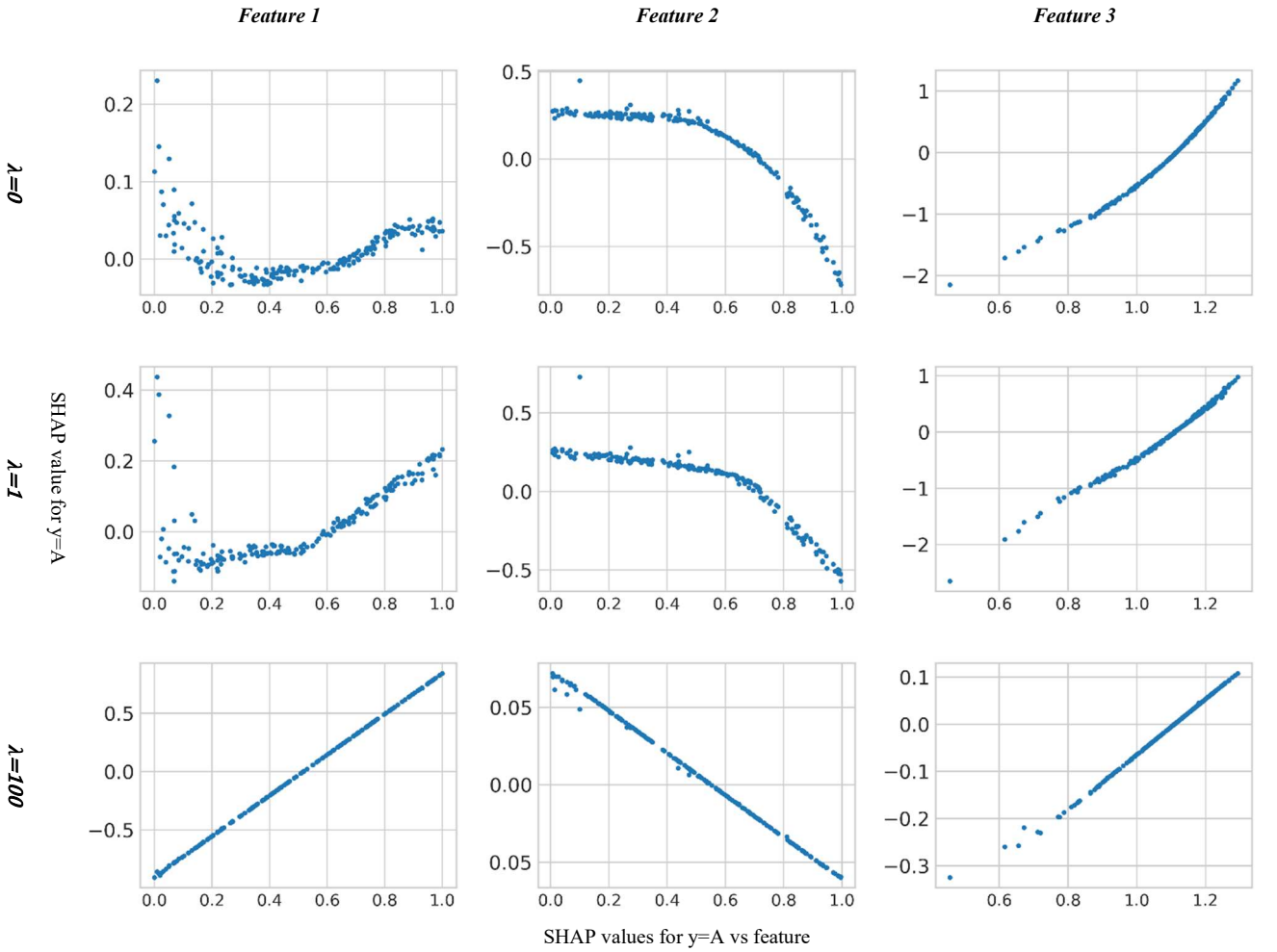


Figure 3. Shapley values of features 1,2,3 (left to right) of models $\lambda \in 0,1,10$ (top to bottom)

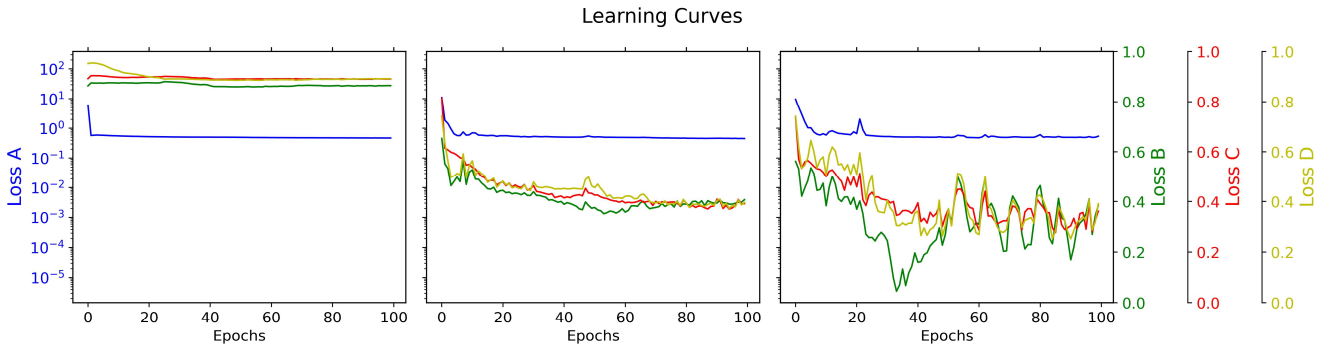


Figure 4. MSE for $\lambda \in \{0, 10, 1000\}$ (left to right) models for the outcome of interest y (blue) and the corresponding Shapley values (green, red, orange).

The results demonstrate the desired tendency towards more explainable models with higher values for λ. Overall, the tests verify the feasibility of the proposed approach and demonstrate the desired effects.

B. *Experiments with real data*

A publicly available data set from openml.org was chosen to verify the applicability of the approach on real data. Specifically, the data set named wine-quality-red was used to predict wine quality [15]. The network structure remained the same as described above, with three features for predicting the target. The target variable includes 6 levels of quality, and the learning problem is treated as a regression problem. The selected features are 'sulphates', 'alcohol', and 'total_sulfur_dioxide'. Feature selection was done based on exploratory analysis for identifying features with non-linear relations to the target. This was done to give room for a trade-off between model accuracy and simplicity of the Shapley value approximation.

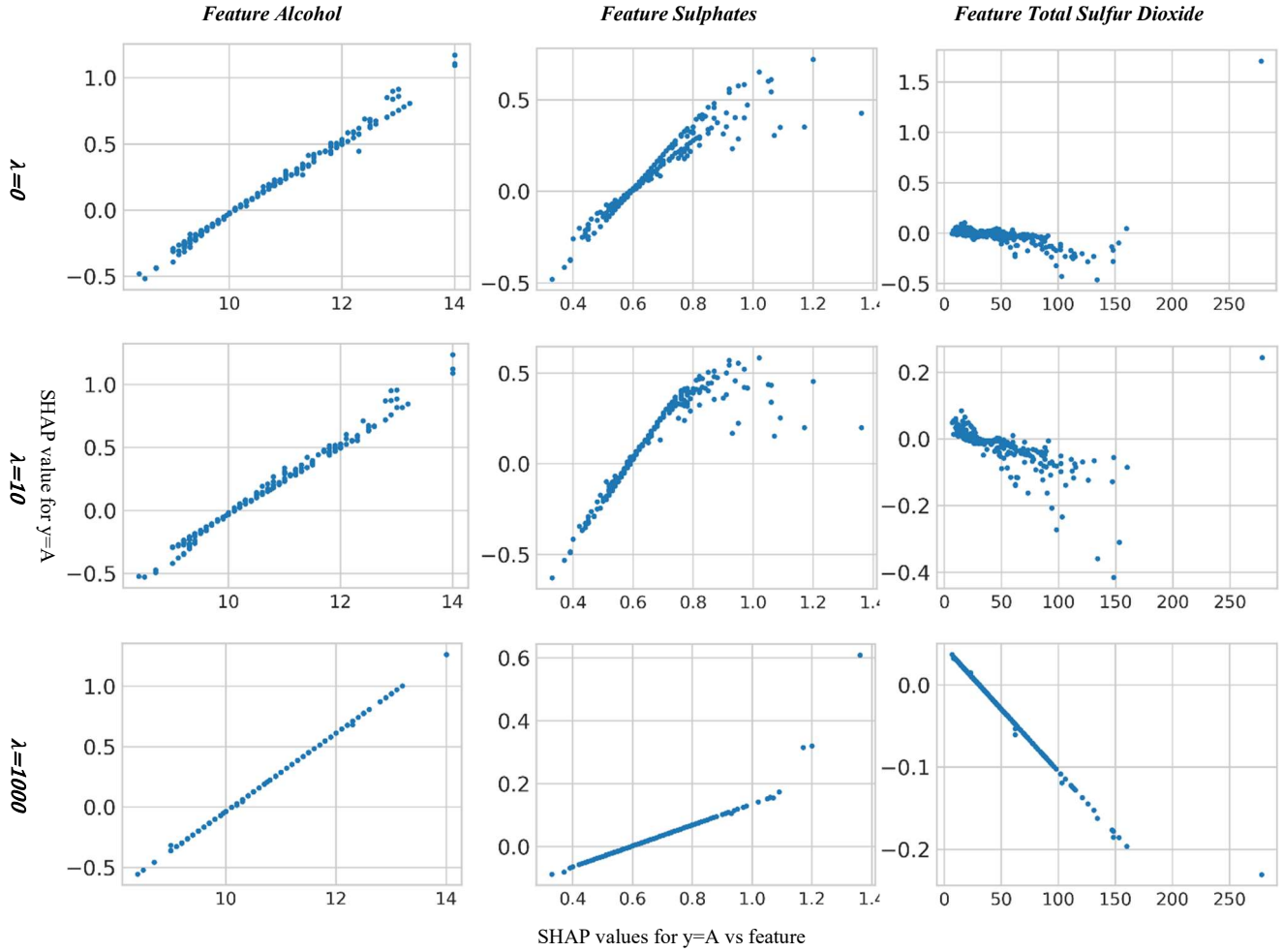


Figure 5. Shapley values of features 1,2,3 (left to right) of models $\lambda \in 0,1,10$ (top to bottom)

Model and training procedure followed the procedure described above, with 100 training epochs. Again, the number of epochs was chosen based on the learning curves of all tests (i.e., ensuring not to stop at a sudden error spike for any model). The resulting learning curves are shown in Figure 4 The model outputs A, B, C, D, represent the model prediction y (i.e., y=A) and the predicted Shapley values for the features 'sulphates', 'alcohol', and 'total_sulfur_dioxide'.

The experiment with real data show the same general effects as the experiments with synthetic data. Specifically, higher λ values reduce the errors in Shapley value predictions but increase the prediction error for the target variable y. For instance, for epoch 100 the MSE the for model . λ=0 is 0.460, while for the model with. λ=10, the MSE decreases to 0.45. However, for . λ=1000, an increase of the MSE to 0.54 can be observed. The learning curves for all models exhibit similar behavior, while the Shapley value approximation shows significant improvement in smoothness and explainability. Again, we observe that the partial dependency plots show increasingly simpler structures (see Figure 5). These findings confirm the applicability of the approach with real data.

V. CONCLUSION AND FUTURE WORK

Explaining neural networks remains a challenging task, often due to the complexity and non-linear nature of these models. Often minor changes in input data can lead to significantly different model outcomes, which complicates explaining these changes to users. It was found that training models with a focus on Shapley values results in more stable and explainable outputs. This approach enhances the consistency of explanations derived from Shapley values, making the model's behavior more predictable and understandable. Contrary to initial expectations, incorporating Shapley values into the training process did not lead to a significant decline in predictive performance, as measured by the mean squared error of the outcome of interest. This suggests that it is possible to maintain accuracy while improving explainability.

Future work could explore adjusting batch sizes to balance convergence and estimation accuracy, as larger batch sizes, while smoothing convergence, may reduce the precision of Shapley value approximations. Moreover, increasing λ improves explainability, it may reduce sensitivity to rare or extreme cases. And scaling to high-dimensional data poses challenges, suggesting more efficient methods for Shapley approximations should be developed. Additionally, expanding experiments to include more diverse datasets could further validate the approach and confirm its generalizability across different domains.

Nevertheless, the proposed model offers the advantage of providing direct explanations for its predictions. This feature is particularly valuable for internal stakeholders, such as management, and external stakeholders, such as regulators, who often require transparent and understandable model explanations.

Based on these findings, this paper recommends adopting our approach for AI models that have to be rarely updated but are frequently used for prediction tasks. This methodology ensures that the model not only performs well but also delivers reliable explanations in the form of Shapley values, thereby meeting the growing demand for transparency in AI systems.

REFERENCES

[1] European regulation on artificial intelligence, “EU AI Act”, [Online], https://www.europarl.europa.eu/doceo/document/TA-9-2024-0138-FNL-COR01_EN.pdf, [retrieved: 10, 2024].

[2] L. Shapley, “Notes on the n-Person Game -- II: The Value of an n-Person Game.”, Santa Monica, Calif.: RAND Corporation, 1951.

[3] L. Merrick and A. Taly, “The Explanation Game: Explaining Machine Learning Models Using Shapley Values” *Machine Learning and Knowledge Extraction*, Springer International Publishing, CD-MAKE 2020, pp. 17-38, Dublin, Ireland, August 25–28, 2020.

[4] M. Ancona, C. Oztireli, and M. Gross, “Explaining Deep Neural Networks with a Polynomial Time Algorithm for Shapley Value Approximation”, *Proceedings of the 36th International Conference on Machine Learning*, PMLR, pp. 272-281, 2019.

[5] S. Lundberg and S. Lee, “A Unified Approach to Interpreting Model Predictions”, *Advances in Neural Information Processing Systems*, pp. 4765–4774, 2017.

[6] H. Chen, I. Covert, and S. Lundberg, “Algorithms to estimate Shapley value feature attributions”, *Nature Machine Intelligence 5*, pp. 590–601, 2023.

[7] M. Sundararajan and A. Najmi, “The Many Shapley Values for Model Explanation”, *Proceedings of the 37th International Conference on Machine Learning*, in Proceedings of Machine Learning Research, pp. 9269-9278, 2020.

[8] S. Lundberg, G. Erion, H. Chen, et al., “From local explanations to global understanding with explainable AI for trees”, *Nature Machine Intelligence 2*, pp. 56–67, 2020.

[9] E. Strumbelj and I. Kononenko, “An Efficient Explanation of Individual Classifications using Game Theory”, *Journal of Machine Learning Research 11*, pp. 1–18, 2010.

[10] I. Covert and S. Lee, “Improving KernelSHAP: Practical Shapley Value Estimation Using Linear Regression”, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, in Proceedings of Machine Learning Research, pp. 3457-3465, 2021.

[11] H. Chen, J. Janizek, S. Lundberg, and S. Lee, “True to the Model or True to the Data?”, *ArXiv, abs/2006.16234,* 2020.

[12] R. Wang, X. Wang, and D. Inouye, “Shapley explanation networks”, *In Proc. International Conference on Learning Representations*, ICLR, 2021.

[13] J. Castro, D. Gamez, and J. Tejada, “Polynomial calculation of the shapley value based on sampling”, *Computers and Operations Research*, pp. 1726 – 1730, 2009.

[14] D. Kingma and J. Ba, “Adam: A method for stochastic optimization”, *ICLR: international conference on learning representations,* pp. 1-15, 2014.

[15] OpenML, “Red Wine Quality Dataset. Dataset”, [Online] https://openml.org/search?type=data&status=active&id=40691, [retrieved: 10, 2024].

Explainable Facial Emotion Recognition with the use of Vision Transformers

Isidoros Perikos
Computer Engineering & Informatics Department
University of Patras,
Computer Technology Institute and Press Diophantus
Patras, Greece
perikos@ceid.upatras.gr

Ioannis C. Kollias
Computer Engineer & Informatics Department
University of Patras
Patras, Greece
st1064886@ceid.upatras.gr

Vaggelis Kapoulas
Computer Technology Institute and Press "Diophantus",
Patras, Greece
kapoulas@cti.gr

Michael Paraskevas
Electrical and Computer Engineering Department
University of Peloponnese,
Computer Technology Institute and Press "Diophantus",
Patras, Greece
mparask@cti.gr

***Abstract*—Facial Emotion Recognition (FER) is very important in the field of human-computer interaction and it can greatly help computer systems to interpret and react to human emotions. The analysis of facial expressions and the accurate recognition of their emotional content are highly desired and assistive in a wide spectrum of domains. In this paper, we present a work on the recognition of facial expressions using a hybrid framework that incorporates Vision Transformers (ViT) with Temporal Convolution Networks. The proposed ViT's goal is to extract intricate facial features, whereas the Temporal Convolution Network component effectively captures temporal relationships and aims to enhance the accuracy of facial expression classification. In addition, the LIME technique was used to illustrate the decision-making procedure of the framework utilized. Our framework can achieve an accuracy of 72% on FER2023 dataset, with a strong emphasis on the explanatory power and generalizability of the model.**

Keywords-Facial Emotion Recognition; Vision Transformers (ViT); Explainability; Temporal Convolutional Network (TCN);

I. INTRODUCTION

Emotion recognition from facial expressions forms the backbone of inferences about human intentions and mental states, making it quintessential in human communication [17]. As interactions with machines become increasingly prevalent, teaching computers to perceive human emotions has the potential to revolutionize human-technology interaction. Applications range from healthcare to marketing, where emotionally aware systems can respond actively, leading to more personalized and effective experiences. For instance, Facial Emotion Recognition (FER) in healthcare can assist with early diagnostics in mental health by analyzing even subtle emotional cues, offering the potential to identify conditions like depression and anxiety much earlier than traditional methods. In marketing, FER enables real-time emotional analysis, allowing companies to present tailored product offerings that align with the consumer's emotional state, thereby enhancing the user experience.

Building accurate and reliable FER systems is quite challenging. Emotions are dynamic and change depending on context, which introduces complexity for FER systems [19]. Additionally, the way individuals express emotions can vary significantly based on factors, such as age, gender, ethnicity, and cultural background [20]. External conditions, such as lighting, facial occlusions (e.g., glasses, masks), and head poses further complicate accurate emotion recognition [22]. Moreover, most existing datasets in the literature, while diverse, often fail to account for all these variations, resulting in models that struggle to generalize effectively to real-world scenarios.

Given the sensitivity of the applications, the accuracy and robustness of FER models are paramount. For instance, incorrect emotion detection in healthcare could lead to misdiagnoses, potentially resulting in harmful treatment plans. In fields like customer service or security, undetected or poorly detected emotions can degrade user experiences or even lead to safety issues. As a result, high accuracy, generalization, and reliability are fundamental requirements for trustable FER systems, particularly in critical areas like healthcare, law enforcement, and mental health.

Recent advancements in deep learning and particularly in the formulation of advanced transformer models, have opened new many new possibilities for improving both the accuracy and generalization of FER systems [18]. While initially designed for natural language processing, transformers have demonstrated their potential for image-based tasks by capturing complex patterns and long-range dependencies. Unlike traditional Convolutional Neural Networks (CNNs), which excel at capturing local features, transformers leverage self-attention mechanisms, allowing them to focus on different parts of the face to recognize subtle emotional cues. This makes them well-suited for addressing the nuanced and dynamic nature of emotions. Moreover, the ability of transformers to process sequential and contextual data presents an opportunity to improve FER in environments where emotions fluctuate rapidly [21].

In this paper, we present a work on the recognition of facial expressions using a hybrid framework that incorporates Vision Transformers (ViT) with Temporal

 ISBN: 978-1-68558-215-9

Convolution Networks. The proposed ViT's goal is to extract intricate facial features, whereas the Temporal Convolution Network component effectively captures temporal relationships and aims to enhance the accuracy of facial expression classification. In addition, the LIME technique was used to illustrate the decision-making procedure of the framework utilized.

The paper is structured as follows. Section II presents related works. Section III presents the transformer-based approach to analyze facial expressions and recognize their emotional content. After that, Section IV presents the experimental study and the results collected. Section V presents the explainability implementation on the images. Finally, Section V concludes the paper and provides the main directions that future work will explore.

II. RELATED WORKS

In recent years, various methods were investigated for FER, and deep learning models, especially recently developed Vision Transformers, show great promise because they allow for the modeling of complex patterns in facial expressions [15][16].

In the work of Chaudhari et al. [1] on the ViTFER, the authors implemented the face emotion recognition system with the help of a vision transformer. The hybrid dataset used by the authors comprised three datasets: FER2013, AffectNet, and CK+48, and is referred to as AVFER. This model has been a fine-tuned Vision Transformer combined with ResNet-18, which was compared. In order to balance samples between classes, data augmentation was performed. In particular, the Vision Transformer (ViT) models proposed in this work with Sharpness-Aware Minimizer reached as high an accuracy as 53.10% in this hybrid dataset and outperform ResNet-18.

VGGNet-based Convolutional Neural Network architecture implemented by Khaireddin et al. [2] achieved best results were realized with a fine-tuning of the hyperparameters and the various techniques of optimization. Using this model, trained on FER2013, gave an accuracy of 73.28% on FER2013, ranking it among the best single-network results. It contains a total of 35,887 grayscale images in size of 48x48, labeled under seven classes of emotions.

Another approach combines Deep CNN with Haar Cascade to capture facial features [3]. This hybrid model, designed for real-time emotion classification, applied pre-processing and data augmentation techniques to optimize training on FER2013, achieved 70% accuracy with significantly reduced training time of 2098.8s.

For ViT-based models, a ViT-CNN hybrid model demonstrated the strength of merging local feature extraction from CNNs with global attention mechanism of ViTs [4]. This model outperformed traditional CNNs by effectively capturing both local and global features, achieving 72.1% accuracy on FER2013.

A more recent contribution is by Wang et al. [5], who, in their submission to the ABAW4 competition, propose an ensemble deep model, CNN-Transformer for facial affect recognition. By construction, the model combines strengths from CNNs that perform well in local spatial feature extraction and strengths from Transformers that capture long-range dependencies and global contextual information. Their method outperformed others with well-balanced performance in a wide range of tasks of facial affect recognition. Their validation set experimental results indicated that their model significantly outperformed the baseline, with a higher F1 score of 0.618 on the LSD task; therefore, this verified the effectiveness of the hybrid design.

Li et al. [6] developed a more powerful hybrid model that combined CNN with a Vision Transformer in facial expression recognition. Their framework leverages the CNN part to extract multiscale local features while the Vision Transformer extracts global relations using the attention mechanism. Besides, they devised a feature integration method and a patch-dropping strategy to further enhance its efficiency and improve the accuracy of recognition. This approach increased the performance significantly in the FER2013 dataset, with an accuracy of about 71.8%, outperforming most models purely based on CNNs or Transformers.

Wang et al. [7] focused on the Vision Transformers with attention mechanisms to further improve the accuracy of emotion recognition. The work underlined the capabilities of transformers to model global context and relationships in facial images, which thus improved results in emotion classification on the FER2013 dataset. Their proposed model attained an accuracy of 74.3%, indicating the effect of an attention-based architecture to capture variations in the facial features.

Kollias et al. [8] present the comprehensive survey on the recognition of face behavior in the wild and analyze different models with respect to operation in unconstrained conditions. Their work has targeted FER challenges in natural conditions and gives a gradual transformation of interest towards transformer-based models and hybrid networks to minimize these challenges, specifically for lesser-controlled facial expression scenarios.

Zhang et al. [9] proposed a real-time face emotion recognition system that combined the attention-based CNN and Transformer components. It was optimized to perform well in real time without a loss in accuracy in detecting fast-changing emotions. Tested on FER2013, it yielded 74.6% accuracy. Thus, hybrid models have also been established for time-critical tasks to be practical.

Zhang et al. [10] developed the attention dual graph convolutional network, which takes facial landmarks as nodes in a graph and models the relationships between them. This is a novel approach, which allows for more ordered representation of the facial features, and because of this reason, it allows for better performance on FER2013 with 90.1%. In fact, their graph-based representations combined with attention mechanisms bring high performance to the model dealing with FER tasks.

Khan et al. [11] proposed the model architecture that hybridized both: the ViT and CNN models. In their model, CNN was used for efficient feature extraction, while with the use of ViTs the possibility of modeling long-range dependencies in facial images can be brought about

underneath. The system achieved a recognition accuracy of 76.8% on FER2013, illustrating the viability of hybrid models applied to emotion recognition tasks.

III. METHODOLOGY

In this section, we present the approach used for facial emotion recognition and detail the underlying architecture of the hybrid ViT and TCN model, which forms the backbone of this study. The design aims to improve FER accuracy by combining powerful feature extraction techniques with temporal modeling capabilities. Additionally, the LIME technique is employed to enhance the model's explainability, providing insights into its decision-making process.

A. Hybrid framework

We propose a fine-tuned version of customized model named ViTCN. It has a balanced architecture, considering both complexity and performance for particular challenges that come associated with face emotion recognition. More importantly, to further strengthen the model, it is based on explanatory technique Local Interpretable Model-agnostic Explanations (LIME) in analyzing and interpreting model prediction with a view to decision-making. To complement the ViT's spatial feature extraction, a TCN was integrated to capture temporal relationships across sequences of facial features. The TCN architecture consists of eight convolutional layers, with one initial layer followed by seven convolutional layers in a loop. Each layer uses 1D convolutions, a kernel size of 3, and padding of 1 to process the sequential input data.

1) Data Preprocessing and Augmentation

Data preprocessing involved converting the grayscale images from the FER2013 dataset to RGB format and resizing them from 48x48 to 224x224 to match the input size required by the ViT model. Various data augmentation techniques, including random horizontal flipping, random cropping, color jittering, and random rotation, were employed to increase dataset variability and prevent overfitting. The transformed images were then normalized and converted into vectors, which were loaded into PyTorch's dataloader for training, validation, and testing.

2) Model Architecture

To create this hybrid model required the definition of its two parts, the first part being ViT and the second part being TCN. The model uses a pre-trained Vision Transformer (*ViTModel*) from the Hugging Face model hub [12][13]. The pre-trained ViT base model utilizes patch size 16x16, with an output hidden dimension of 768. The classifier head of the ViT model is replaced with *nn.Identify()*, meaning the ViT is only used for feature extraction (specifically, the *last_hidden_state* is used). The output sequence, after the features from the ViT have been extracted, passes through a Temporal Convolutional Network. The network consists of 8 convolutional layers: the first initial layer and seven convolutional layers arranged in a loop. Each layer of a TCN consists of one-dimensional convolution with a kernel size fixed to 3 and padding fixed to 1, Rectified Linear Unit (ReLU) non-linearity introducing activation function and dropout rate for regularization at 0.3. The output from TCN acts as an input to *F.adaptive_avg_pool1d(x,1)*, which reduces the sequence length to 1; hence the output size becomes batch_size, channels.

A linear layer (*self.fc)* maps the pooled TCN output to the number of emotion classes. The output from ViT, of shape *(last_hidden_state, shape.(batch_size, seq_len, hidden_dim))* is permuted to change dimensions which is necessary for compatibility with the *Conv1d* layers in the TCN.

3) Hyper-parameters

First, the optimization model relies on the Adam optimizer, as this has adaptive learning rates and momentum, hence being quite suitable when dealing with sparse gradients in large Vision Transformers. In this regard, a starting learning rate of 10^{-4} will be selected to give moderate model parameter updates for stable convergence during training. Apart from that, weight decay with a factor of 10^{-4} was used. In essence, this is an L2 regularization, one that penalizes large weights and hence reduces overfitting; this makes the model more generalized on unseen data.

CrossEntropyLoss criterion was used to assess the performance of the model, which was trained with the help of this criterion. This loss function is quite well adapted for multi-class classification tasks like FER, since it computes the divergence between the estimated probability distribution and the true distribution of the target labels. By minimizing this loss, the model gets gradually trained to provide class probabilities that are close to the actual labels of emotions, thus increasing its accuracy.

A *ReduceLROnPlateau* learning rate scheduler was added to the training pipeline, dynamically adjusting the learning rate based on the performance on the validation set. This 'min' mode scheduler would reduce the learning rates when there was no significant improvement of the validation loss. The reduction factor used here was 0.5, halving the current best learning are where the model's progress started showing a plateau. This will prevent early reductions by giving the model tree epochs of stagnating validation loss before the learning rate is adjusted, as specified by the patience parameter set to 3. The minimum threshold for reducing the learning rate was kept at 10^{-6} to avoid excessive reduction that might impede learning.

Early stopping was used to prevent overfitting and improve model efficiency. This approach monitored the validation loss across epochs and stopped training if any improvement was not witnessed during a certain window. Early stopping was configured to be patient for 7 epochs, meaning it would stop training if the validation loss did not improve for 7 consecutive epochs. Early stopping fired by their model, it reset the weights of the model back to the state representing the epoch with the lowest validation loss. The mentioned technique not only avoided overfitting but also economized computational resources by saving a lot of superfluous epochs of training.

IV. EVALUATION STUDY

The FER2013 dataset is used as a benchmark for developing and testing FER models. It contains a large, diverse set of labeled images across seven basic emotions, making it valuable for overcoming challenges arising from

individual and cultural variability. Despite its widespread use, models trained on FER2013 often fail to perform robustly in general scenarios, a challenge that necessitates further improvements in model architectures and training approaches.

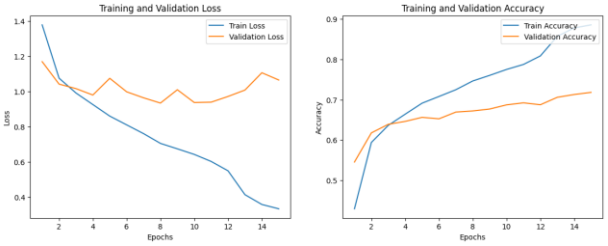


Figure 1. Metric-graphs Results

In Figure 1, the graphs of loss and accuracy for each training and validation epoch are illustrated. We can see that the use of parameters to deal with overfitting was necessary. The evaluation results are quite interesting. We assess the performance of the framework in terms of precision, recall and F1-Score which are reported in Table I. The performance of our model reaches a macro average precision of 74%, with the values of the individual metrics being particularly good.

TABLE I. PERFORMANCE RESULTS

	Classification Report		
	precision	***recall***	***f1-score***
Angry	0.64	0.67	0.65
Disgust	0.89	0.73	0.80
Fear	0.63	0.52	0.57
Happy	0.91	0.89	0.90
Sad	0.57	0.65	0.61
Surprise	0.86	0.79	0.82
Neutral	0.66	0.71	0.68
Macro Avg	0.74	0.71	0.72
Weighted Avg	0.72	0.72	0.72

In addition, the confusion matrix was created, showing the correlations between the actual and predicted labels. The diagonal is identified while it is observed that the fear and sad labels have the highest confusion. Th confusion matrix is illustrated in Figure 2. The confusion matrix provides valuable insight into the performance of the model across different emotion classes. Notably, the model exhibits high accuracy for detecting "Happy" expressions, with a correct prediction rate of 89%, indicating strong performance in recognizing this emotion. However, the matrix also reveals challenges in distinguishing between certain emotions. For instance, the model struggles with differentiating "Fear" and "Sad" expressions, as it correctly identifies "Fear" only 52% of the time, frequently confusing it with "Sad" and "Surprise." Similarly, "Disgust" is often misclassified as "Angry" (18%), suggesting overlap in the facial features associated with these emotions.

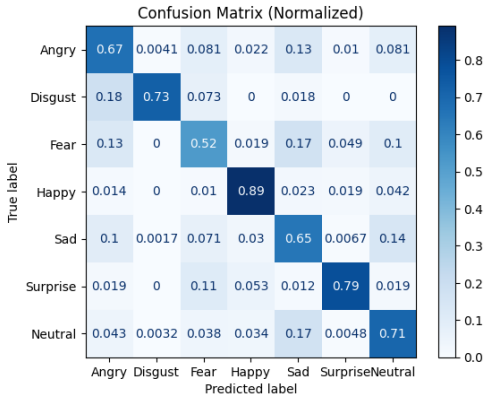


Figure 2. Confusion Matrix

This confusion is indicative of subtle similarities between these emotional expressions that the model may be finding difficult to distinguish. While emotions like "Surprise" and "Neutral" are predicted with relatively high accuracy (79% and 71%, respectively), further improvements could be made for classes like "Fear" and "Disgust." These results suggest that the model would benefit from additional training data, particularly for the underperforming classes, and possibly more refined feature extraction methods to improve its ability to differentiate between similar emotions.

V. EXPLAINABILITY

Explainability in deep learning refers to methods used in an attempt to interpret how models make the decisions. More importantly, for tasks like FER, it is desirable to ascertain that the model decides on the classes of emotions based on meaningful facial features-for example, eyes and mouth. Furthermore, explainability is crucial for building trust in FER systems, as well as for improving the model.

LIME is an explainability technique that provides us insight into model decisions by perturbing parts of the input and observing changes in predictions [14]. This works by creating locally faithful explanations, enabling us to see which regions of the image most drive the model's decision. This is particularly helpful to identify which facial features it relies on to classify emotions-skipping the possibility it learns nonsensical patterns. In this regard, the implementation of the LIME explanation first defines a prediction function, taking a batch of images, preprocesses them by resizing and normalizing, and then passes them through the model in order to compute probabilities for emotion classifications. This prediction function is going to be used by LIME when assessing how model predictions are affected by certain perturbations to the input image.

The key explanation procedure starts by taking an input image and generating slight variations or perturbations of the image. These perturbed images are passed through the model using the prediction function. Subsequently, LIME follows the output response of the model to these variations and selects the most important regions in the image that contributed to the classification decision. It highlights the

boundaries in order to represent areas that had the strongest influence on the prediction. This allow us to interpret visually which parts of the face were of most significance when determining the predicted emotion. Specifically, the explanations generated by LIME illustrate how the model focuses on specific facial regions when classifying emotions These visualizations provide intuitive insights into the model's decision-making process, highlighting the areas of the face that are most influential in predicting each emotion. The yellow-highlighted regions, produced by LIME, confirm that the model is correctly focusing on the most salient facial features typically associated with these emotions, thereby enhancing the interpretability of the model's predictions. The following examples demonstrate the application of LIME in classifying various emotional expressions.

In the example case shown in Figure 3, which illustrates the "Disgust" emotion, the regions of interest are primarily concentrated around the nose and the central part of the face—key areas typically involved in disgust expressions.

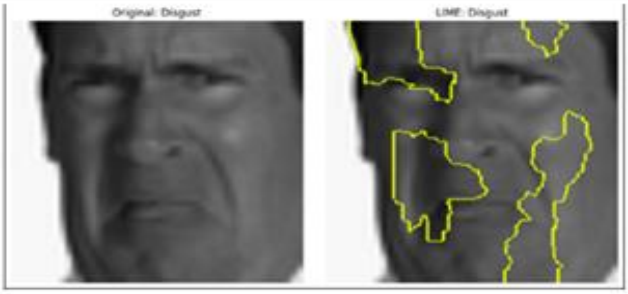


Figure 3. LIME Explanation for Disgust Emotion Prediction.

The LIME output clearly highlights the wrinkling of the nose and the characteristic mouth shape, often frowning or pursed lips. These visual cues further confirm that the model is accurately focusing on the facial features that define disgust. This aligns with human perception, where attention is naturally drawn to the upper face, particularly the nose, when interpreting disgust, as this expression generally centers on the middle of the face. Similarly, in the case for the "Anger" emotion depicted in Figure 4, LIME places emphasis also on the regions around the forehead, eyebrows, and the mouth. As expected, these are the prime areas to consider while one identifies anger, since they have features like furrowed eyebrows and a tense, open mouth both of which are strong indicators of frustration or aggression.

Figure 4. LIME Explanation for Anger Emotion Prediction.

Given the model's attention towards these regions, it convincingly demonstrates that the model has learned to pick out these important features, which adds more credibility to its prediction. Such a visual explanation also agrees with human intuition, since intuitively we associate furrowed eyebrows and tensed facial expressions with anger.

In the example case of the "Sad" emotion in Figure 5, the LIME explanation places significant emphasis on the forehead, eyebrows, and cheeks. The model particularly focuses on key indicators of sadness, such as sagging eyebrows and a drooping mouth. Additionally, the strong attention to the downward gaze suggests that the model is effectively capturing the lowered head posture often associated with sadness. This alignment with typical facial cues reinforces that the model is accurately identifying the relevant features needed for emotion classification.

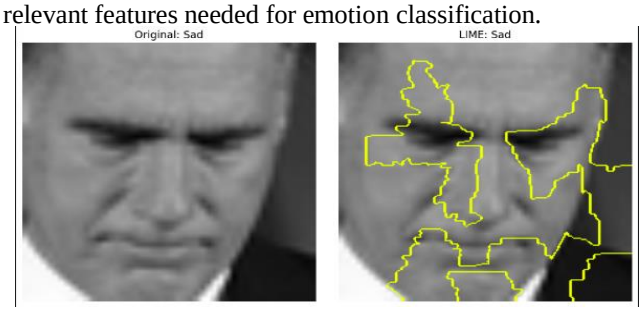


Figure 5. LIME Explanation for Sad Emotion Prediction.

In Figure 6, a case is illustrated for LIME explanation created for a "Surprise" emotion. The LIME explanation has highlighted the forehead, eyes, and mouth regions in this image. Of course, these are important to a surprise expression because wide eyes and a slightly open mouth are common visual keys to this emotion, which the model has learned to focus on appropriately for the expression of surprise. Emphasis on the wide-eyed look and the form of the mouth also aligns with human perception for surprise, further grounding the model's prediction.

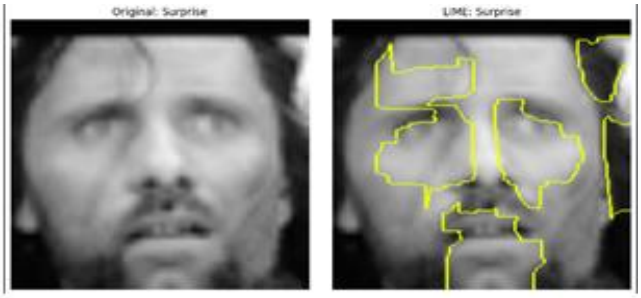


Figure 6. LIME Explanation for Surprise Emotion Prediction.

These LIME visualizations not only show that the model is focusing on relevant and emotion-specific facial features but also offer an interpretable illustration for its predictions. The fact that the LIME highlights align with commonly understood human expressions adds further credibility to the model-that it makes decisions based on appropriate facial cues.

VI. CONCLUSIONS AND FUTURE WORK

Facial expressions form a universal language of emotions, which can instantly express a wide range of emotional states and feelings. The accurate analysis of facial expressions and the precise recognition of their emotional content are highly desired and assistive in a wide spectrum of domains and applications. Although it is natural for humans to interpret facial expressions naturally with little or even no effort, the accurate and robust facial expression recognition by computer systems is still a great challenge. Our work presents a hybrid model for FER by combining both the Vision Transformer and Temporal Convolution Network models. Therefore, this work was successful at achieving high accuracy along with good generalization, Optimization in Vision Transformer was able to extract fine details of a face while embedding TCNs allowed it to identify temporal relationships within the data. In addition, the LIME technique was used to illustrate the decision-making procedure of the framework utilized. Specifically, our framework visualizes explanations with, which explained important facial regions that are responsible for emotions classified within the decision making of the transformer model. The results were very encouraging and indicate that the approach is efficient and accurate in analyzing facial expressions and recognizing their emotional content.

Future research will focus on enhancing generalization by expanding both the size and diversity of the dataset. Additionally, advancing explanation techniques could further increase trust in FER systems, particularly for applications in healthcare and psychological diagnostics. Future directions could involve integrating additional explainability methods, such as SHAP to provide deeper insights into the model's decision-making processes. This combination with LIME, would allow for a clearer understanding of the importance of individual facial regions in each emotion recognition.

ACKNOWLEDGEMENT

This work was partially supported by the Project entitled ”Upgrade and enhancement of the digital services ecosystem of the Panhellenic School Network / Expansion of the myschool system functionalities and completion of the environment for providing advanced digital services to all members of the educational community” with Project Code OPS 5168211

REFERENCES

[1] A. Chaudhari, Y. Khaireddin, and Z. Chen, "Facial emotion recognition: State of the art performance on FER2013". *arXiv Preprint arXiv:2105.03588*, 2021

[2] O.C Oguine, K.J. Oguine, H.I. Bisallah, and D. Ofuani, "Hybrid facial expression recognition (FER2013) model for real-time emotion classification and prediction". *arXiv Preprint arXiv:2206.09509*. 2022

[3] H. Li, M. Sui, F. Zhao, Z. Zha, and F. Wu, "Facial expression recognition using a hybrid ViT-CNN aggregator". In *Springer*, 2021

[4] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual transformers: Token-based image representation and processing for computer vision", *arXiv Preprint arXiv:2006.03677*, 2020

[5] L. Wang, H. Li, and C. Liu, "Hybrid CNN-Transformer model for facial affect recognition in the ABAW4 challenge". *ArXiv*, 2022

[6] N. Li, Y. Huang, Z. Wang, Z. Fan, X. Li, and Z. Xiao, "Enhanced hybrid vision transformer with multi-scale feature integration and patch dropping for facial expression recognition", *Sensors*, *24*(13), 4153, 2024

[7] G. Wang, Y. Zhao, C. Tang, C. Luo, and W. Zeng, "When shift operation meets Vision Transformer: An extremely simple alternative to attention mechanism", *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(2), 2423-2430, 2022

[8] D. Kollias, A. Schulc, and S. Zafeiriou, "Face behavior recognition in the wild: A survey", *IEEE Transactions on Affective Computing*, *13*(4), 2244-2263, 2022

[9] J. Zhang, C. Li, G. Liu, M. Min, C. Wang, J. Li, Y. Wang, H. Yan, Z. Zuo, W. Huang, and H. Chen, "A CNN-Transformer hybrid approach for decoding visual neural activity into text", *Computer Methods and Programs in Biomedicine*, *214*, 106586. 2021

[10] S. Zhang, Y. Zhang, Y. Zhang, Y. Wang, and Z. Song, "A dual-direction attention mixed feature network for facial expression recognition". *Electronics*, *12*(17), 3595,2023

[11] A. Khan, Z. Rauf, A. Sohail, A.R. Khan, H.Asif, A. Asif, and U. Farooq,"A survey of the vision transformers and their CNN-transformer based variants". *Artificial Intelligence Review*, *56*, 1-54, 2023

[12] B. Wu, C. Xu, X. Dai, W. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual Transformers: Token-based image representation and processing for computer vision". *arXiv preprint*, 2020

[13] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database". In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 248-255). IEEE, 2009

[14] E. Hvitfeldt, T.L. Pedersen, and M. Benesty, "*LIME: Local interpretable model-agnostic explanations*", 2022

[15] F.Z. Canal, T.R. Müller, J.C. Matias, G.G. Scotton, A.R. de Sa Junior, E. Pozzebon, and A. Sobieranski, "A survey on facial emotion recognition techniques: A state-of-the-art literature review", *Information Sciences*, *582*, 593-617, 2022

[16] Y. Li, J. Wei, Y. Liu, J. Kauttonen, and G. Zhao, "Deep learning for micro-expression recognition: A survey", *IEEE Transactions on Affective Computing*, *13*(4), 2028-2046, 2022

[17] S. C. Leong, Y.M. Tang, C.H. Lai, and C.K.M. Lee, "Facial expression and body gesture emotion recognition: A systematic review on the use of visual data in affective computing". *Computer Science Review*, *48*, 100545, 2023

[18] H. Ge, Z. Zhu, Y. Dai, B. Wang, and X. Wu, "Facial expression recognition based on deep learning". *Computer Methods and Programs in Biomedicine*, *215*, 106621, 2022

[19] A.V. Savchenko, L.V. Savchenko, and I. Makarov, "Classifying emotions and engagement in online learning based on a single facial expression recognition neural network". *IEEE Transactions on Affective Computing*, *13*(4), 2132-2143, 2022

[20] S. Kumar, S. Rani, A. Jain, C. Verma, M.S. Raboaca, Z. Illés and B.C. Neagu, "Face spoofing, age, gender and facial expression recognition using advance neural network architecture-based biometric system". *Sensors*, *22*(14), 5160, 2022

[21] L. Xiong, J. Zhang, X. Zheng, and Y. Wang, "Context Transformer and Adaptive Method with Visual Transformer for Robust Facial Expression Recognition", *Applied Sciences*, *14*(4), 1535, 2024

[22] I. Perikos, M. Paraskevas, and I. Hatzilygeroudis, "Facial expression recognition using adaptive neuro-fuzzy inference systems", In *2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS)* (pp. 1-6). IEEE, 2018

An XAI Approach on the Capacity of Transformers to Learn Time Dependencies in Time Series Forecasting

Alberto Mino Calero, Adil Rasheed, and Anastasios M. Lekkas
Department of Engineering Cybernetics, Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology (NTNU)
NO-7034, Trondheim, Norway
e-mail: {alberto.m.calero | adil.rasheed | anastasios.lekkas}@ntnu.no

***Abstract*—Despite the traction gained by transformers on time series forecasting tasks, partially based on their success in natural language processing to learn contextual information, their suitability in this domain has been questioned. Several works have found significant performance problems when comparing these models with simpler options leading some people to think transformers are not as fit for time series forecasting as considered before. However, it remains unclear if they can capture long-term dependencies, similarly as they do with contextual information. This study takes on that line of questioning using an explainable artificial intelligence approach. By making use of Shapley additive explanations, the attribution scores assigned to input features are computed, showcasing that a transformer model optimized for time series forecasting is unable to learn long-term time dependencies, and mostly considers the last time steps from the inputs. The analysis, based on interpretable knowledge based on the computed Shapley values, is done in a multivariate forecasting setting that resembles a complex real-world problem, proving that the model is unsuited for the task.**

Keywords-Transformers; Time Series; Deep Learning; XAI; SHAP.

I. Introduction

Time series analysis and forecasting have a major role in research [1] and real-world applications, such as climate science [2], healthcare [3], biology [4], and economics [5]. As technology has progressed, new methods and algorithms have been used to improve the models used. From statistical and regression-based models such as autoregressive integrated moving average (ARIMA) [6][7], the field has seen a transition to machine learning approaches, such as support vector machines [8], nonparametric models like functional decomposition [9], and nonparametric bayesian models [10].

More recently, deep learning has gained more traction over other machine learning techniques thanks to their advantage of being universal approximators [11][12] under the right conditions. With their great versatility and representational power, many neural network architectures have been used for time series forecasting, such as multi-layer perceptions (MLPs) [13], convolutional neural networks (CNNs) [14], and networks designed to solve sequential data problems like recurrent neural networks (RNNs) [15][16] and long-short term memory (LSTM) networks [17]. More recently, transformers [18] have become one of the main technologies used for time series forecasting [19]–[23] thanks to their outstanding capabilities in modeling contextual information and handling sequence data, particularly in natural language processes (NLP) tasks. Despite their popularity, all deep learning models have the disadvantage of being black boxes, a feature highly undesired in safety- and business-critical applications. Even transformers, despite the self-attention mechanism that allows them to weigh the importance of different parts of their input sequential data to make their forecasts, distinguishing them from other types of neural networks, are virtually uninterpretable.

Due to the abrupt increase in the use of transformers for time series forecasting, several works have questioned their effectivity, with results that show how remarkably simple linear models can, indeed, outperform them. To answer this question, performance has been the main element examined to the best of our knowledge, like in [24], using the metrics mean square error (MSE) and mean absolute error (MAE) to reason why transformers may have design flaws to tackle time series forecasting, having been designed for natural language processing (NPL) tasks. Although the inputs of transformers are sequences, like in time series forecasting, one key part of their success is their ability to extract contextual information from the text sequences, which seems not to translate very well to learning time dependencies.

Thanks to the revitalized interest in eXplainable AI (XAI) motivated by the surge in using neural networks, many solutions provide interpretable knowledge and explanations based on many algorithms that exploit either external elements, with primarily model agnostic techniques, or internal elements, which primarily focus on the gradients that connect input and output in these models or attention mechanism in case of transformers. Similarly to other deep learning models, interpretability in transformers can be tackled from those perspectives, which may give insight into their efficacy in time series forecasting by showing if they are learning long-term time dependencies, or if they rather only focus on the last instances from the inputs, leading to short-term dependencies and discardment of the rest of the time series input.

In this work, an XAI-based methodology is proposed to analyze what time dependencies have been learned by a transformer-based model making use of Shapley additive explanations (SHAP) [25] to compute attribution scores of the inputs. By aggregating these scores and examining them with respect to the time series input and output sequences, it is possible to look into how these scores change as the model predicts further into the future. By doing so, an analysis of these changes offers a way to determine what kind of temporal

 ISBN: 978-1-68558-215-9

patterns the models have learned, and how long back into the past they are able to look for their predictions.

The paper is organized as follows. Section II presents a brief overview of the state-of-the-art concerning XAI applicable to transformer-based models both broadly and specific to time series forecasting. Next, Section III introduces the methodology used in this work. Details referring to the dataset and the transformer model trained are provided, followed by a description of SHAP, the implementation choice for the computation of the Shapley values, and the framework used to analyze them to assess the suitability of the model for the task. Then, the results and analysis are presented in Section IV, which conveys the findings of this work that the model is unable to learn long-term time dependencies. Finally, V presents the conclusions and suggests future lines of work aimed to better examine this family of models and their suitability for time series forecasting tasks through the lenses of XAI.

II. State-of-the-art on explainability of transformers

Transformers are a type of deep neural network (DNN) characterized by using a self-attention mechanism and an encoder-decoder architecture. Although one self-attention head can learn which parts of the inputs are more important when making predictions, hence holding interpretable knowledge, they typically have many of such self-attention heads, in addition to a very high number of parameters combined with non-linear activation functions, and so they are initially incomprehensible. To tackle them and attempt to explain their inference, there are two main families of algorithms: attribution score and attention-based methods.

In attribution-score methods, the algorithms assign a relevance score to each of the inputs that were processed by the model to reach a certain output. This provides interpretable knowledge that exposes how high or low the contribution of each input feature is for any output. These types of methods rely mostly on input-output mappings, and as such are considered "external", and in many cases are model agnostic, meaning they can be applied to any machine learning model.

Some examples of attribution score-based methods are integrated gradients (IG) [26], layer-wise relevance propagation (LRP) [27] or SHAP [25]. IG works by integrating the gradients of the model's output with respect to the input features along a path from a baseline input to the actual input. LRP propagates the prediction backward in a neural network with designed local propagation rules subjected to a conservation property so that what is received by a neuron is redistributed in equal amounts to the previous layer. SHAP calculates the relevance scores by distributing the outcome of the model among the input features to get the relevance scores, which is done using Shapley values from coalitional game theory. These methods can, usually, be applied to any deep learning model, while others such as SHAP are model agnostic, hence applicable to any machine learning model. Thanks to this versatility, they have been successfully applied on numerous occasions.

As one key aspect of the success of transformers comes from their attention mechanism, this inbuilt attention mechanism can help to look into their behavior. Although still not interpretable, analyzing where transformers focus their attention may provide a better understanding of how they infer. Along this line, several works take advantage of that to build methods that throw some light on their interpretability to provide, to a certain degree, XAI in the context of transformers in the form of attention-based methods. These methods usually provide visual information about which elements of the input sequence the model has learned to pay attention to and rely directly on the attention parameters, although the existence of multiple attention heads complicates the process.

Attention mechanism-based methods require considering the importance of how to properly take advantage of the multi-head self-attention mechanism (MHSA) to get valuable information. In [28], an updated version of LRP designed to work with transformers and MHSA was developed. Instead of propagating scores backward through all layers, their approach focuses on attention head relevance, and although their technique has the initial goal of serving as a means to prune unimportant attention heads, they successfully identify different interpretable roles within the MHSA in the context of automatic translation.

Attention rollout and attention flow are the solutions proposed in [29] to quantify the information flow approximating the attention to the inputs. First, attention rollout traces the flow of information from input tokens to hidden embeddings in the higher layers by propagating attention weights through several network layers. This is done by recursively multiplying the minimum, maximum, or mean of the attention heads of each block with the attention of the previous blocks. Then, attention flow treats the resultant graph as a flow network using the attention weights as edge capacities. By doing this, the maximum attention flow between any of the layers to any of the input nodes can be computed with any maximum flow algorithm, which can serve as an approximation of the attention to input nodes. Nevertheless, the method's speed is prohibiting, hence rather unfeasible for evaluations at a large scale.

Beyond attention [30], based on LRP, proposes assigning attribution scores using the deep Taylor decomposition principle to generate the initial scores and then propagate them through the layers taking into consideration both attention layers and skip connections, both integral parts of transformers architectures. The initial scores are computed via LRP for all attention heads and all layers by integrating the relevance score of each attention head and its gradient with respect to the input features. The weighted attention scores they propose to adopt the notion of positive relevance, hence considering only the positive values that result from the computation, and the method was tested in the NLP model BERT [31] and a visual transformer model.

Beyond intuition [32] is another proposal designed to ap-

proximate the contribution of the input tokens. This solution relies on the partial derivative of the loss function with respect to each token in two steps. First, in the attention perception phase, the relationships between input and output for each attention block are unfolded, resulting in the development of two recurrence formulas involving head-wise and token-wise attention maps. The second step is reasoning feedback, which applies the IG algorithm to the last attention map with a noise-decreasing strategy to reduce the gradient self-induced noise.

Gradient self-attention maps (Grad-SAM) [33] uses a gradient-based method applied to the attention matrices to generate rankings for the tokens used in the layers of the encoder and it was tested on the NLP model BERT. This method offers another way to understand how the model reaches an individual prediction by highlighting the inputs that contribute to it the most, and it is based on computing the partial derivatives of the model output with respect to the self-attention blocks. This approach does not ignore negative values resulting from this computation, but to avoid the accumulation of negatives that may cancel the information carried by positive values, they propose using the ReLU function to zero out negatives.

In the context of visual transformers [34], the survey from [35] paints a picture of the current state-of-the-art methods. Understanding visual information has the technical advantage of humans being able to identify patterns easily, and it is only necessary to visualize what the models are paying attention to, to make sense of their inference process. Traditional attribution-score methods, such as layer-wise relevance propagation [27], SHAP [25], local interpretable model-agnostic (LIME) [36], or gradient-weighted class activation mapping (Grad-CAM) [37] can suffice. On the other hand, since the attention mechanism is designed to allow transformers to focus on the relevant part of the input sequence, other methods focus on this element, and in computer vision tasks attention can be directly visualized just by looking into the raw attention of the transformers, or through some function, such as Grad-SAM [33] or beyond intuition [32].

In time series classification [38], attribution and attention methods are the most popular strategies to assign relevance scores to every time point in a time series. Another type of explanation can be extracted based on subsequences by identifying the parts of a sequence responsible for a certain classification outcome, such as PatchX [39] and data-agnostic classification method via shapelets (DASH) [40]. The third general type of explanation in this domain is based on instances and relies on the complete time series to extract reasons for the classification outcomes, such as multi-operator temporal decision trees [41] and dual prototypical shapelet networks [42].

In time series forecasting, [43] presents an experimental comparison between XAI methods based on saliency maps applied to several deep learning architectures including transformers. The methods tested include IG [26], SmoothGrad [44], DeepLIFT [45], gradient and deep SHAP [25], and feature ablation [46], among others less up-to-date. The conclusions point out a general lack of high-quality interpretable knowledge from these methods when applied to multivariate time series data while succeeding when the time series is represented as images or univariate and propose a two-step temporal saliency rescaling to improve the results. They hypothesize that the main reason for this lack of quality is the combination of temporal and feature domains. Previous studies [47] provided comparisons between LIME, LRP, DeepLIFT, Saliency, and SHAP for time series classification with DNNs, CNNs, and ResNet, with SHAP performing more robustly than the rest on ResNet. They showed a decrease in accuracy when perturbing subsequences in univariate time series classification datasets, which was assumed to be indicative of the alteration of parts that were important for the models' internal inference, and SHAP seemed to perform better than the rest for the more advance Resnet architecture.

III. Methodology

The methodology applied in this work is described in this section, and it is based on the goal of analyzing if a machine learning model, in this case a Transformer, is able to learn time dependencies in a time series forecasting setup using attribution scores computed with the Shapley additive explanations method. Henceforth, it can be generalized to any other model as long as the XAI method used to compute the attribution scores is model agnostic, such as SHAP, or is applicable to the model in question.

A. Data

In this work, the data comes from a collection of the currency exchange rates of eight countries, including Australia, British, Canada, Switzerland, China, Japan, New Zealand, and Singapore, between 1990 and 2016, gathered daily. This data has been used for time series analysis benchmarks of different models, transformer-based architectures included, in works such as [48] and [24]. The dataset has eight input features and 7588 total time steps with a time resolution of one day. Regarding the data splitting, a standard distribution was used, forming subsets of 70%, 10%, and 20% of the total samples for training, validation, and testing, respectively.

B. Transformer model

This work analyzes the transformer implementation used in [20], [24]. The implementation details contain several differences with respect to the actual vanilla transformer [18]. The importance of the differences becomes fundamental to analyzing the models through XAI lenses because of a series of technical issues and can be summarized as follows.

- Concerning the architecture, this transformer version contains an additional temporal embedding that receives and handles the information about the time marks of each sequence element.
- For the inputs, the transformers take 3 elements in addition to the expected time series sequence. First, the decoder takes the last L_{dec} pieces of the time series input. This reduced time series acts as an enhanced "start

sequence" token, supplying additional information for the decoder. Then, the time marks of the time series input on one hand, and the time marks of the enhanced start sequence token followed by the time marks that characterized each time step of the expected prediction horizon on the other. These two time mark sequences are used by the encoder and the decoder respectively, and are treated as the other two inputs. The time mark sequences encode information about the dates in a four-values format. This format is a hyperparameter, and in these experiments, the selected "hourly" encodes and normalizes the hour of the day, the day of the week, the day of the month, and the day of the year.

- In regards to the outputs and in connection to the training and inference processes, the model performs a direct multi-step prediction, hence providing the whole expected sequence in a single step.

1) Training and performance: Regarding training, 10 models were optimized for a prediction horizon $Z_p = 96$, chosen from the set of values $96, 192, 336, 720$ as it yielded the best performance. Since it is a multivariate forecasting task and 10 models were trained, each model's performance mean squared error MSE_{exp} is computed with (2), where y is the prediction value, $\hat{y}$ the true value, N_f the total number of features predicted, N_p the number of predictions, and the subscripts i, j, and k are feature, prediction, and experiment, respectively (also in (1)). Then, the aggregated MSE (MSE_{agg}) is simply an average of the MSE_{exp} of the 10 models following (1).

$$MSE_{agg} = \frac{\sum_{k=1}^{N_{exp}} MSE_{exp_k}}{N_{exp}} \tag{1}$$

$$MSE_{exp} = \frac{\sum_{j=1}^{N_p} \sum_{i=1}^{N_f} (y_{i,j} - \hat{y}_{i,j})^2}{N_f N_p} \tag{2}$$

As the loss function, the MSE is used as the standard for regression tasks, which is averaged among the 8 output features since the multivariate forecast of the model. The models are trained with different seeds to get rid of possible underperforming combinations of initialized parameters, using the adaptive moment estimation (ADAM) optimizer with input sequences of length $Z_i = 96$, again yielding the best results from the set $96, 192, 336, 720$. The models were trained in an NVIDIA RTX 3090 GPU. For the rest of the hyperparameters, the default values from [24] were initially used, and for this work, a random search for optimization purposes was performed over batch size, learning rate, training epochs, and early stopping hyperparameters, arriving at respective values of 32, 0.0001, 10, and early stopping with patience 3.

Since the goal is to ascertain if transformers can learn time dependencies, the best-performing model is the one chosen to be analyzed by SHAP. Note that this work will not examine model performance in depth as it falls out of the scope.

C. SHAP

The original kernel and deep implementations from [25], referred to hereinafter as Kernel and Deep, were used to examine the inner workings of transformers with the SHAP approach as computationally efficient algorithms to approximate Shapley values, particularly so in higher dimensions. As shown by [25], the Shapley values approximated with SHAP using Kernel perform slightly better than those approximated with Deep (DeepLift-based SHAP). Because of the substantially high memory and computation requirements of Kernel, a base test was made to compare both results and determine how much Kernel suffers by leaving out part of the training data as background data for the algorithm, to make an informed decision about which algorithm to use for the analysis of the transformers with SHAP. The L2 distance and symmetric mean absolute percentage error (SMAPE) were used as distance measurements. L2 serves as a standard distance measurement between real value vectors to capture important deviations in feature attribution, while SMAPE is a scale-independent and commutative measurement that offers a better understanding of the distance regardless of the small scale of the values.

With the larger amount of data manageable with the computation resources used in this work for a single time step of the prediction horizon (32 training sequences for Deep, 16 for Kernel), an $L2 = 0.0036$ and $\textit{SMAPE} = 175.48\%$ were reached between Deep and Kernel. With the minimum reduction of the background data for the analysis of the whole prediction horizon (32 training sequences for Deep, 4 for Kernel), an $L2 = 0.0046$ and $\textit{SMAPE} = 184.38\%$ were computed instead. These are calculated for the feature attribution of the first time step forecast. While the difference between *SMAPE* error is only 4.95%, the percentage difference in $L2$ ascends to 24.81%, suggesting the Kernel implementation suffers when the background data to integrate out features decreases. Henceforth, the choice was made to use the results of the DeepLift-based Shapley values (DeepSHAP) for the analysis.

For the use of the original DeepSHAP algorithm implementation [25], the critical adaptations of this algorithm to function on the model and data used for this work were implemented in the form of wrappers and data refactorization to meet the requirements of the DeepExplainer interface. To achieve this, the model is wrapped in a function that only needs the time series input sequence so that SHAP focuses on extracting the feature importance of this data instead of over every input the model actually needs. This is done by making the rest of the inputs independently accessible by the model outside the DeepSHAP algorithm. Therefore, the time series input sequence is used as input data for the model to calculate the Shapley values. Regarding the rest of the inputs, the time series sequence subset needed by the decoder is handled by the wrapper, which already receives the whole time series input sequence. The time marks, on the other hand, must remain unchanged during the computation because it is not desirable in this setting to alter the possible time dependencies that the model might have learned and may be related to the temporal embeddings. For this reason, this part of the inputs is accessible by the model through external variables, but not modified by SHAP.

The data refactorization involves the input sequence and the output but does not affect the process other than reshaping the sequences so that DeepSHAP can process them. The time series input sequences are reshaped to a format admissible by SHAP before the algorithm is called and shaped back to the format required by the model within the model wrapper sent to the SHAP algorithm. Regarding the outputs, the wrapper is configured to output a single time step within the prediction horizon, which can be selected beforehand. Therefore, to analyze the whole prediction horizon SHAP needs to be computed for each time step. These changes allow the extraction of the attribution scores of the features examined in this work in a compatible way with SHAP, while also reducing the computation requirements, which would otherwise be prohibited in terms of memory due to the high memory use, which increases exponentially with the dimensionality of the data. To analyze the impact of each input feature of each time step of the input sequence, each feature from each time step needs to be considered as an individual input, hence the dimensionality of the input features analyzed by SHAP is the number of input features times the length of the input sequence. Note none of these changes have any effect on the actual transformer model architecture or parameters behind the wrapper, hence the inference process performed by the transformer remains unaltered.

D. Shapley values analysis

In the analysis, the features that significantly impact the model outputs according to the attribution scores computed by Shapley values are examined. These values are computed by the SHAP algorithm using the full training set as background data as described in [25]. The Shapley values are studied by looking at the features with a higher accumulated impact on the outputs from several individual time steps of the prediction horizon. This is done by considering each feature from each time step of the input sequence on its own and accumulating their impact by adding the absolute values that each input feature has for all the outputs in the chosen prediction horizon slice. This way, it can be seen which input features and from which input time steps have a higher impact on the model's outputs.

Secondly, the evolution of the attribution scores is investigated throughout time steps from input and output sequences. To look at the overall evolution of the impact of all the input features and time steps, looking separately into specific output features of the model allows to obtain more general information about how the impact of input features from different time steps varies depending on the prediction horizon.

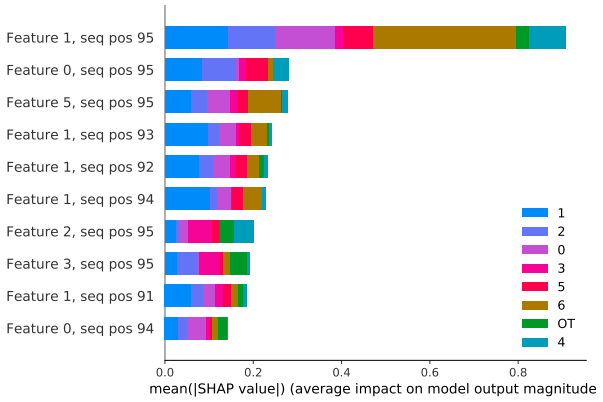
Ultimately, the global distribution of the accumulated impact from all features is studied. This is done by applying a sum reduction over the absolute values of the impact of all the input features over all the output features. Since the impact from the features computed by attribution scores that SHAP provides measures how much changes in the values cause changes in the prediction, using sum reduction over their absolute values will provide practical information about the patterns learned by the model without mismanaging or losing the SHAP values meaning. Hence, looking at the impact that each time step of the input sequence has over each time step of the prediction horizon yields the most global vision. By analyzing this, it is possible to ascertain what time dependencies the model has learned from a wider point of view, as well as detect possible anomalies or interesting behaviors.

IV. Results and analysis

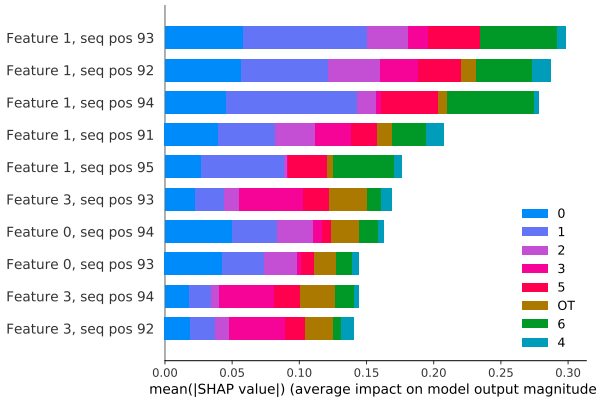
The 10 instances of the transformer trained reached an average performance of $MSE = 0.748$, with the best-performing model chosen for the analysis with SHAP having a performance of $MSE = 0.601$.

The input features with the highest impact on the model's output for several time steps located at the beginning and ending parts of the prediction horizon can be seen in Figure 1. In Figure 1a, it can be seen how the Shapley values reveal that feature attribution is placed directly on the time steps from the input sequence that are directly adjacent to the starting point of the prediction horizon. This behavior of the attribution scores replicates in the whole prediction horizon the model is trained for ($Z_p = 96$) except in the last time step. Note that the second to last time step also behaves similarly, as displayed by Figure 1b, although with more evenly distributed attribution scores across a wider range of input features from the last elements of the input sequence. The only deviation from this pattern can be seen in Figure 1c, which displays the attribution score for the last time step of the prediction horizon, and where it can be observed that 6 out of the 10 most impactful features come from the intermediate input sequence elements instead of the last. Nevertheless, as this does not replicate in any other time step of Z_p, it seems likely to be an outlier prediction situation. Note that there is no evident anomaly in the input sequence that explains this event either.

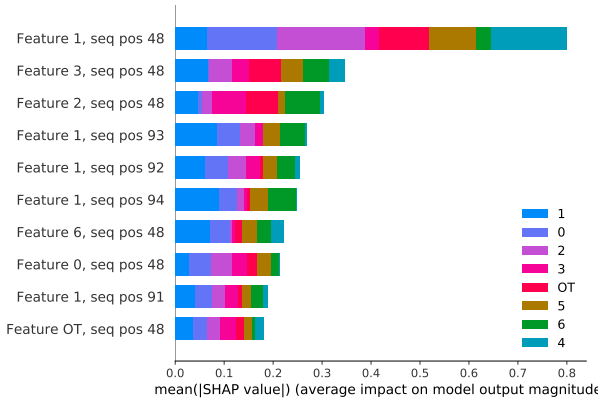
The distribution of the attribution scores across the entirety of the input sequence is shown in Figure 2 for four time steps across diverse lengths of the Z_p. As can be seen, the model is influenced mostly by the last elements of the input sequence length, except in the last time step. These attribution scores, however, present some changes after the first forecasted time steps, especially observable in Figure 2b and Figure 2c, with the attribution scores being slightly more evenly distributed. For some output features, such as Feature 1 and 2, this happens across the second half of the input sequence, while for others, such as Feature 3 and 4, it is distributed between the ending and the beginning of the input sequence. Despite these changes, notice that most of the importance is still placed in the last time steps of the input sequence, especially so if the comparison is made between the initial, intermediate, and ending parts of the input sequence. Moreover, as was displayed by Figure 1c, Figure 2d shows that there is a notable anomaly on the attribution score placement for the last forecasted time step, where most of the impact, and across all output features, is placed on an intermediate, single time step (48). Regardless of the shifts in the feature importance distribution as the model



(a) Predicted time step $Z_p = 0$



(b) Predicted time step $Z_p = 94$



(c) Predicted time step $Z_p = 95$

Figure 1. SHAP values of the 10 features with the highest impact for multiple times steps of the prediction horizon. Seq. pos. indicates the time step of the input feature, i.e. its location in the time series input sequence, while the name of each feature is enumerated from 0 to 6 plus the OT feature. Note that OT is one of the eight countries, named differently to be used as the default output target for the univariate implementation.

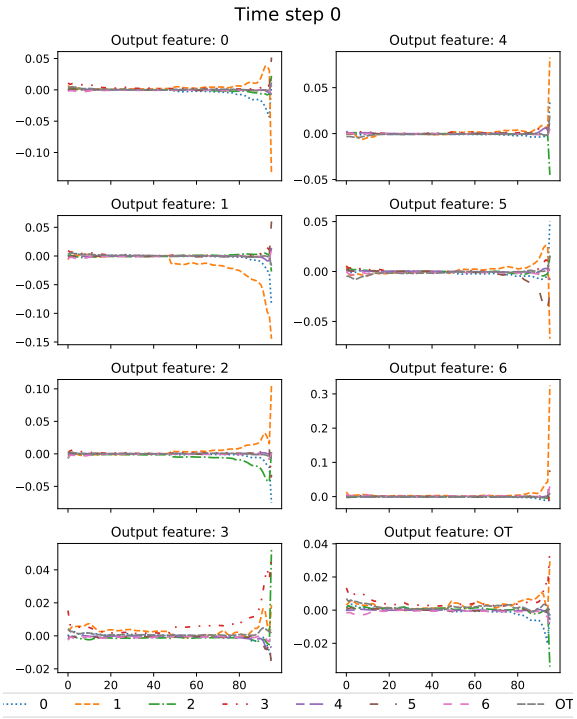
predicts further into the future, the anomaly only happens in the last forecasted time step, complicating the analysis of the event unless it can be explained as an outlier in the prediction, which is a known issue in certain domains as pointed out in [49], [50].

The distribution of the impact from all the input features over all the outputs, for all the time steps from both input sequence and output prediction horizon can be seen in Figure 3. In the figure, it can be easily seen that the transformer model is mostly only affected by the last five elements of the input sequence. As most of the influence is placed upon only a small fraction of the inputs, essentially the last time steps from the input, and across the whole prediction horizon, it can be verified that the model has not learned any long-term time dependencies, which suggests transformers might be fundamentally flawed in dealing with long time series forecasting (LTSF) tasks from an XAI perspective. It can be noted that two anomalies occur in two instances of time. One, as previously spotted in Figure 2d, appears on the last time step of the prediction horizon, while the other does so in the very first, and can also be seen in more detail in Figure 2a. The second anomaly is only in the accumulated value of the impact of the features, which is significantly higher (about three times bigger) with respect to its neighbors on the plot. Nevertheless, it is not an anomaly in the general behavior. On the contrary, the first anomaly of the predicted time steps is indeed of value but also of behavior. Nevertheless, being the only behavioral break of the pattern analyzed across 9216 data points of 3, it is hardly significant enough, and seems most likely an outlier in the prediction or the SHAP values computed.

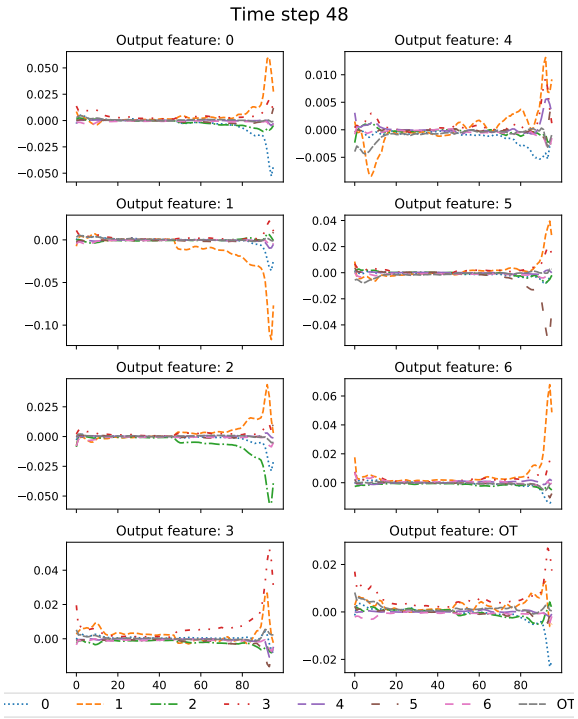
V. Conclusion and Future Work

In this work, a framework is proposed to look into the suitability of transformers for time series forecasting using SHAP to analyze the behavior of the model. It is found that the transformer was not able to learn long-term time dependencies from its training, and according to the attribution scores computed all the impact is placed upon the last time steps of the inputs. These findings suggest that transformers-based models might be particularly prone to disregard most of the time series data used for their training, hence making them not good choices for this particular task. In consequence, developers should be aware that transformers appear to fail to capture the long-term dynamics of time series, and the issue should be investigated when proposing transformer-based models for this type of task, in addition to considering the use of other types of models for LTSF tasks. Furthermore, for applications where the performance of the transformer-based model is the only feature looked for, a similar analysis may be useful in optimizing the training, given that similar conclusions are reached where most of the input sequence has a very low impact on the forecasts, suggesting that decreasing the length of the input sequence would not significantly impact the performance.

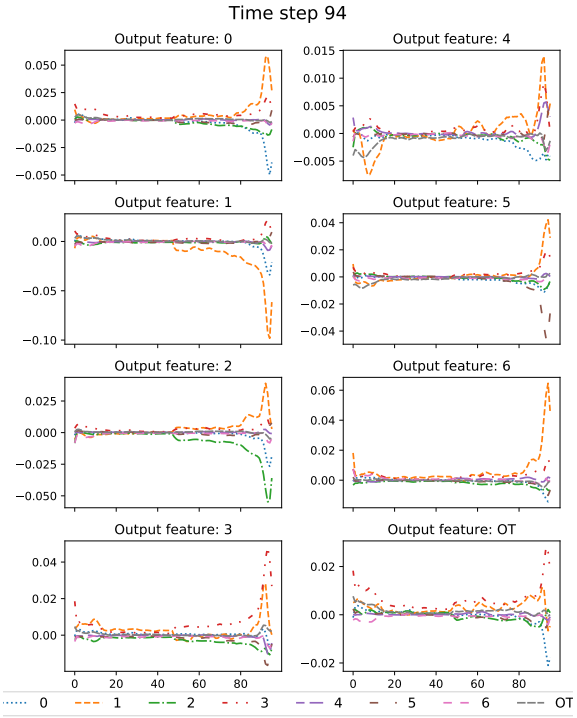
Facing the lack of analysis of this particular issue from an XAI perspective, experimenting further with these models in a wider variety of time series datasets and types of models may yield interesting findings. It can lead to better establishing if the findings of this work are a symptom of a flaw in transformer-based models' design to address LTSF tasks, or if it is just a case of a flawed model not being able to learn from a



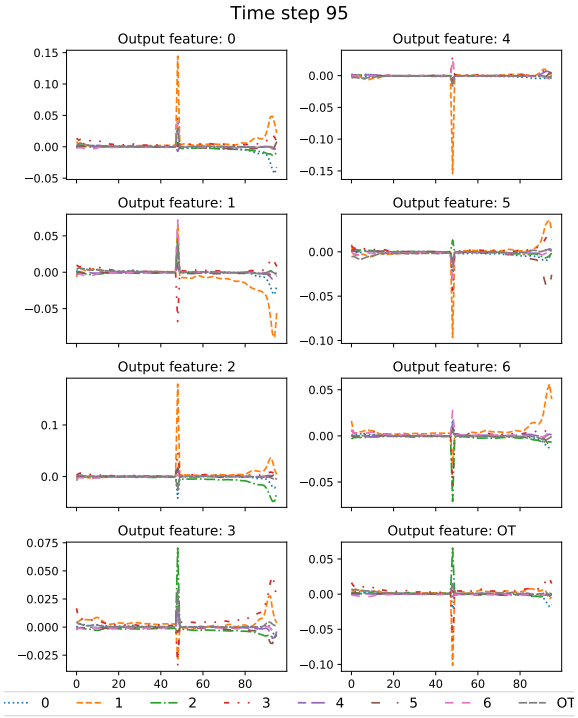
(a) Forecasted time step 0. Most of the input influence is placed over the last elements of the input time series sequence.



(b) Forecasted time step 48. The behavior is similar to Figure 1a, but the influence is less concentrated on the last time steps.



(c) Forecasted time step 94. The behavior is consistent with every previous forecasted time step, and nearly identical to Figure 2b.



(d) Forecasted time step 95. The behavior changes and most of the influence is placed on the middle input features of the time series. Note that it is the only time step where this behavior occurs.

Figure 2. Distribution of the attribution scores across all the elements of the input sequence for the forecasted time steps 0, 48, 94, and 95. Each subfigure depicts the impact evolution across the input time steps for each of the 8 output features, which are divided into different plots, each one depicting 8 signals representing each one of the 8 input features from the input time series

 ISBN: 978-1-68558-215-9

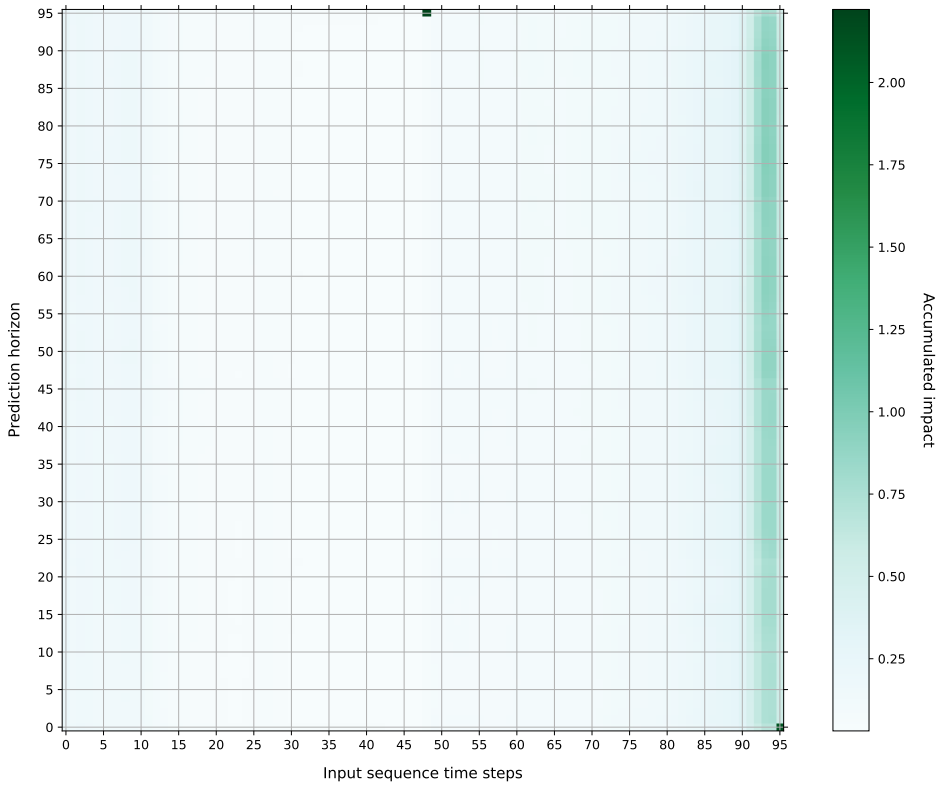


Figure 3. Accumulated impact of all the input features aggregated by the aggregation of the absolute values of their attribution scores over all the output features for each time step of the input sequence and of the output prediction horizon.

dataset. Furthermore, analyzing more recent transformer-based models' designs from this XAI perspective can lead to more impactful and relevant findings regarding model performance and suitability for the LTSF task. Additionally, looking more deeply into these findings with different XAI methods might yield a solution to exactly where the problem is with these types of models, so they can be adapted and improved.

ACKNOWLEDGEMENTS

This work has been funded by the Research Council of Norway through the EXAIGON project, project number 304843.

REFERENCES

[1] G. Mahalakshmi, S. Sridevi, and S. Rajaram, "A survey on forecasting of time series data," in *2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16)*, Jan. 2016, pp. 1–8. DOI: 10.1109/ICCTIDE.2016.7725358.

[2] M. Mudelsee, *Climate time series analysis. Classical statistical and bootstrap methods.* (Atmospheric and Oceanographic Sciences Library), Second Edition. Springer, 2014, vol. 51, ISBN: 978-3-319-04449-1. DOI: 10.1007/978-3-319-04450-7.

[3] E. J. Topol, "High-performance medicine: The convergence of human and artificial intelligence," *Nature Medicine*, vol. 25, no. 1, pp. 44–56, Jan. 2019, Publisher: Nature Publishing Group, ISSN: 1546-170X. DOI: 10.1038/s41591-018-0300-7.

[4] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 379, no. 2194, p. 20 200 209, Feb. 2021, Publisher: Royal Society. DOI: 10.1098/rsta.2020.0209.

[5] J. Feyrer, "Trade and Income—Exploiting Time Series in Geography," *American Economic Journal: Applied Economics*, vol. 11, no. 4, pp. 1–35, Oct. 2019, ISSN: 1945-7782. DOI: 10.1257/app.20170616.

[6] A. C. Harvey, "ARIMA models," in *Time series and statistics*, J. Eatwell, M. Milgate, and P. Newman, Eds., London: Palgrave Macmillan UK, 1990, pp. 22–24, ISBN: 978-1-349-20865-4. DOI: 10.1007/978-1-349-20865-4_2.

[7] K. Yunus, T. Thiringer, and P. Chen, "ARIMA-Based Frequency-Decomposed Modeling of Wind Speed Time Series," *IEEE Transactions on Power Systems*, vol. 31, no. 4, pp. 2546–2556, Jul. 2016, Conference Name: IEEE Transactions on Power Systems, ISSN: 1558-0679. DOI: 10.1109/TPWRS.2015.2468586.

[8] K. W. Lau and Q. H. Wu, "Local prediction of non-linear time series using support vector regression," *Pattern Recognition*,

vol. 41, no. 5, pp. 1539–1547, May 2008, ISSN: 0031-3203. DOI: 10.1016/j.patcog.2007.08.013.

[9] M. G. Frei and I. Osorio, "Intrinsic time-scale decomposition: Time–frequency–energy analysis and real-time filtering of non-stationary signals," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 463, no. 2078, pp. 321–342, Aug. 2006, Publisher: Royal Society. DOI: 10.1098/rspa.2006.1761.

[10] S. J. Gershman and D. M. Blei, "A tutorial on Bayesian nonparametric models," *Journal of Mathematical Psychology*, vol. 56, no. 1, pp. 1–12, Feb. 2012, ISSN: 0022-2496. DOI: 10.1016/j.jmp.2011.08.004.

[11] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, Jan. 1991, ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T.

[12] Y. Lu and J. Lu, "A Universal Approximation Theorem of Deep Neural Networks for Expressing Probability Distributions," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 3094–3105.

[13] T. Kolarik and G. Rudorfer, "Time series forecasting using neural networks," *ACM SIGAPL APL Quote Quad*, vol. 25, no. 1, pp. 86–94, 1994, ISSN: 0163-6006. DOI: 10.1145/190468.190290.

[14] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, Feb. 2017, Conference Name: Journal of Systems Engineering and Electronics, ISSN: 1004-4132. DOI: 10.21629/JSEE.2017.01.18.

[15] G. Grudnitski and L. Osburn, "Forecasting S&P and gold futures prices: An application of neural networks," *Journal of Futures Markets*, vol. 13, pp. 631–643, Sep. 1993. DOI: 10.1002/fut.3990130605.

[16] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent Neural Networks for Time Series Forecasting: Current status and future directions," *International Journal of Forecasting*, vol. 37, no. 1, pp. 388–427, Jan. 2021, ISSN: 0169-2070. DOI: 10.1016/j.ijforecast.2020.06.008.

[17] U. M. Sirisha, M. C. Belavagi, and G. Attigeri, "Profit Prediction Using ARIMA, SARIMA and LSTM Models in Time Series Forecasting: A Comparison," *IEEE Access*, vol. 10, pp. 124 715–124 727, 2022, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3224938.

[18] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., Jun. 2017, p. 11.

[19] H. Zhou *et al.*, "Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, ISSN: 2374-3468, 2159-5399 Issue: 12 Journal Abbreviation: AAAI, vol. 35, May 2021, pp. 11 106–11 115. DOI: 10.1609/aaai.v35i12.17325.

[20] H. Wu, J. Xu, J. Wang, and M. Long, *Autoformer: Decomposition Transformers with Auto-Correlation for Long-Term Series Forecasting*, arXiv:2106.13008 [cs], Jan. 2022. DOI: 10.48550/arXiv.2106.13008.

[21] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. Hoi, *ETSformer: Exponential Smoothing Transformers for Time-series Forecasting*, arXiv:2202.01381 [cs], Jun. 2022. DOI: 10.48550/arXiv.2202.01381.

[22] Y. Zhang and J. Yan, "Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting," in *ICLR Proceedings 2023*, Sep. 2022, p. 21.

[23] Q. Wen *et al.*, *Transformers in Time Series: A Survey*, arXiv:2202.07125 [cs, eess, stat], May 2023.

[24] A. Zeng, M. Chen, L. Zhang, and Q. Xu, "Are Transformers Effective for Time Series Forecasting?" *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 9, pp. 11 121–11 128, Jun. 2023, Number: 9, ISSN: 2374-3468. DOI: 10.1609/aaai.v37i9.26317.

[25] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[26] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17, Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.

[27] S. Bach *et al.*, "On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation," en, *PLOS ONE*, vol. 10, no. 7, e0130140, Jul. 2015, Publisher: Public Library of Science, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0130140.

[28] E. Voita, D. Talbot, F. Moiseev, R. Sennrich, and I. Titov, "Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds., Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 5797–5808. DOI: 10.18653/v1/P19-1580.

[29] S. Abnar and W. Zuidema, "Quantifying Attention Flow in Transformers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 4190–4197. DOI: 10.18653/v1/2020.acl-main.385.

[30] H. Chefer, S. Gur, and L. Wolf, *Transformer Interpretability Beyond Attention Visualization*, en, arXiv:2012.09838 [cs], Apr. 2021.

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North*, Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[32] J. Chen, X. Li, L. Yu, D. Dou, and H. Xiong, "Beyond Intuition: Rethinking Token Attributions inside Transformers," en, *Transactions on Machine Learning Research*, Oct. 2022, ISSN: 2835-8856.

[33] O. Barkan *et al.*, "Grad-SAM: Explaining Transformers via Gradient Self-Attention Maps," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, Virtual Event Queensland Australia: ACM, Oct. 2021, pp. 2882–2887, ISBN: 978-1-4503-8446-9. DOI: 10.1145/3459637.3482126.

[34] A. Dosovitskiy *et al.*, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, arXiv:2010.11929 [cs], Jun. 2021. DOI: 10.48550/arXiv.2010.11929.

[35] R. Kashefi, L. Barekatain, M. Sabokrou, and F. Aghaeipoor, *Explainability of Vision Transformers: A Comprehensive Review and New Perspectives*, en, arXiv:2311.06786 [cs], Nov. 2023.

[36] M. T. Ribeiro, S. Singh, and C. Guestrin, *"Why Should I Trust You?": Explaining the Predictions of Any Classifier*, arXiv:1602.04938 [cs, stat], Aug. 2016.

[37] R. R. Selvaraju *et al.*, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," in *2017 IEEE International Conference on Computer Vision (ICCV)*, ISSN: 2380-7504, Oct. 2017, pp. 618–626. DOI: 10.1109/ICCV.2017.74.

[38] A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti, "Explainable AI for Time Series Classification: A Review, Taxonomy and Research Directions," *IEEE Access*, vol. 10,

pp. 100 700–100 724, 2022, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3207765.

[39] D. Mercier, A. Dengel, and S. Ahmed, "PatchX: Explaining Deep Models by Intelligible Pattern Patches for Time-series Classification," in *2021 International Joint Conference on Neural Networks (IJCNN)*, arXiv:2102.05917 [cs], Jul. 2021, pp. 1–8. DOI: 10.1109/IJCNN52387.2021.9533293.

[40] R. Guidotti and A. Monreale, "Designing Shapelets for Interpretable Data-Agnostic Classification," Jul. 2021, pp. 532–542. DOI: 10.1145/3461702.3462553.

[41] V. Shalaeva, S. Alkhoury, J. Marinescu, C. Amblard, and G. Bisson, "Multi-operator Decision Trees for Explainable Time-Series Classification," en, in *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations*, J. Medina *et al.*, Eds., Cham: Springer International Publishing, 2018, pp. 86–99, ISBN: 978-3-319-91473-2. DOI: 10.1007/978-3-319-91473-2_8.

[42] W. Tang, L. Liu, and G. Long, *Interpretable Time-series Classification on Few-shot Samples*, arXiv:2006.02031 [cs, stat], Jul. 2020. DOI: 10.48550/arXiv.2006.02031.

[43] A. A. Ismail, M. Gunady, H. C. Bravo, and S. Feizi, *Benchmarking Deep Learning Interpretability in Time Series Predictions*, arXiv:2010.13924 [cs, stat], Oct. 2020. DOI: 10.48550/arXiv.2010.13924.

[44] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, *SmoothGrad: Removing noise by adding noise*, arXiv:1706.03825 [cs, stat], Jun. 2017. DOI: 10.48550/arXiv.1706.03825.

[45] A. Shrikumar, P. Greenside, and A. Kundaje, *Learning Important Features Through Propagating Activation Differences*, arXiv:1704.02685 [cs], Oct. 2019. DOI: 10.48550/arXiv.1704.02685.

[46] H. Suresh *et al.*, "Clinical Intervention Prediction and Understanding using Deep Networks," *ArXiv*, May 2017.

[47] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, and D. A. Keim, *Towards a Rigorous Evaluation of XAI Methods on Time Series*, en, arXiv:1909.07082 [cs], Sep. 2019.

[48] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, "Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks," in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, ser. SIGIR '18, New York, NY, USA: Association for Computing Machinery, Jun. 2018, pp. 95–104, ISBN: 978-1-4503-5657-2. DOI: 10.1145/3209978.3210006.

[49] Harald Martens and Tormod Næs, "5.3.2 Outliers in prediction (goodness of fit)," in *Multivariate calibration*, John Wiley & Sons, Aug. 1992, p. 440, ISBN: 978-0-471-93047-1.

[50] J. A. Fernández Pierna, F. Wahl, O. E. de Noord, and D. L. Massart, "Methods for outlier detection in prediction," *Chemometrics and Intelligent Laboratory Systems*, Chemometrics 2002 S.I. Vol. 63, no. 1, pp. 27–39, Aug. 2002, ISSN: 0169-7439. DOI: 10.1016/S0169-7439(02)00034-5.

A Medical Decision Support System for Explainable Multimodal Detection of Non-Small Cell Lung Cancer Using Clinical and PET Data

Nikolaos I. Papandrianos, Anna A. Feleki, Ioannis D. Apostolopoulos, Elpiniki I. Papageorgiou
Department of Energy Systems, University of Thessaly, Gaiopolis Campus, 41500
Larissa, Greece
npapandrianos@uth.gr, annafele1@uth.gr, elpinikipapageorgiou@uth.gr

Jose Maria Alonso Moral
Centro Singular de lInvestigación en Tecnoloxías Intelixentes (CiTIUS), Universidade de Santiago de Compostella, 15782,
Santiago de Compostela, Spain
josemaria.alonso.moral@usc.es

Nikolaos D. Papathanasiou, Dimitrios J. Apostolopoulos
Department of Nuclear Medicine, University General Hospital of Patras, University of Patras, 265-00
Patras, Greece
nikopapath@upatras.gr, dimap@upatras.gr

Javier Andreu-Perez
Centre for Computational Intelligence, School of Computer Science and Electronic Engineering, University of Essex, Colchester, United Kingdom
j.andreu-perez@essex.ac.uk

***Abstract*— Non-small cell lung cancer is a prevalent form of lung cancer, with Solitary Pulmonary Nodules (SPNs) as a key indicator. Early detection and accurate diagnosis are critical for effective treatment. While Convolutional Neural Networks (CNNs) have been successful in diagnosing SPNs from Computed Tomography (CT) and Positron Emission Tomography (PET) imaging, they lack explainability. To address this, we applied DeepFCM, a multimodal approach that combines Fuzzy Cognitive Maps (FCMs) with CNNs, integrating clinical and PET imaging data to predict SPN malignancy. Clinical data include patient characteristics (i.e., gender, age, Body Mass Index, Glucose Levels) and SPN characteristics (diameter, Standardized Uptake Value (SUV)max, location, type, and margins). Predictions from the RGB-CNN, trained on PET images, are used as additional inputs for DeepFCM. Initially defined by nuclear experts using fuzzy sets, concept interconnections were adapted with Particle Swarm Optimization (PSO) and Genetic Algorithm (GA). DeepFCM is integrated into a Medical Decision Support System (MDSS) to enable data-driven predictions for NSCLC. To improve explainability, Gradient-weighted Class Activation Mapping (Grad-CAM) highlights significant image regions, while DeepFCM illustrates the relationships between each feature to NSCLC diagnosis. Natural Language Generation (NLG) is applied to explain the DeepFCM decision-making process by demonstrating each feature's impact on the diagnosis in human-understandable language.** *(Abstract)*

Keywords-Fuzzy Cognitive Maps; Non-small Cell Lung Cancer; Particle Swarm Optimization; Genetic Algorithm.

I. INTRODUCTION

Non-Small Cell Lung Cancer (NSCLC) constitutes approximately 85% of all lung cancer cases globally [1]. NSCLC can often be presented as a Solitary Pulmonary Nodule (SPN) on imaging studies, necessitating further evaluation to determine if the nodule is benign or malignant, which presents challenges. Most individuals with early-stage lung cancer do not exhibit typical symptoms. However, once symptoms like cough and hemoptysis appear, many patients have already progressed to the middle or late stages of lung cancer, with metastasis occurring in some cases [2]. Deep Learning (DL) methodologies like CNN have been applied and published to detect SPN malignancies. In [3], the authors proposed an ensemble-based prediction model for NSCLC recurrence following surgical resection. The method integrated three neural network models, each trained separately on clinical data, handcrafted radiomic (HCR) features, and deep learning radiomic (DLR) features derived from CT images. The outputs of these models were combined using an ensemble analyzer to make the final prediction. Data from two institutions were utilized, involving standardized Computed Tomography (CT) images and relevant clinical features, excluding incomplete cases. The proposed ensemble model demonstrated superior accuracy using only single data types, achieving an 11.69% higher accuracy than the staging baseline. In [4], the VGG19 model was applied to classify CT and Positron Emission Tomography (PET) images, using the extracted features from VGG19 for further analysis. These outputs, along with additional SPN characteristics, were fed into an XGBoost model, which conducted the final diagnosis by merging imaging data and clinical features to enhance diagnostic precision. 402 patient cases were used with human annotations for internal validation and 96 histopathologically confirmed cases for external evaluation. The model achieved a 97% agreement with human experts and showed 85%

 ISBN: 978-1-68558-215-9

diagnostic accuracy on the external dataset. The most important predictors identified in this study were the Standardized Uptake Value (SUV)max value and the nodule diameter. Additionally, in [5], the authors explored multimodal learning on the «CoLlAborative multi-sources Radiopathomics approach for personalized Oncology in non-small cell lung cancer» (CLARO) dataset for NSCLC, combining clinical data and imaging from a patient cohort to predict overall survival. A late fusion ensemble approach optimally integrated classifiers from different modalities, including a ResNet34 and a VGG11-BN for the imaging modality and a TABNET for the clinical modality, by solving a multiobjective optimization problem to maximize performance and diversity. Results indicate the proposed multimodal ensemble outperforms unimodal models, achieving 75% accuracy, 77.7% F1-score, and 84% recall. Furthermore, in [6], a two-stage multimodal learning framework was developed for diagnosing pulmonary nodules in PET/CT images. Pulmonary parenchyma segmentation was applied in the first stage with a pre-trained U-Net model. The second stage focused on extracting image-level and feature-level characteristics by utilizing a 3D Inception-Residual Net (ResNet) with a convolutional block attention module and a dense-voting fusion mechanism. The model's performance was validated on real clinical data, achieving mean scores of 89.98% accuracy, 89.21% precision, 84.75% recall, 93.38% specificity, 86.83% F1 score, and 0.9227 area under the curve (AUC). In [7], a stacked 3D CNN model was implemented to classify SPN in PET/CT images. 113 participants were included. Data augmentation was applied to increase the size of the training dataset, with random rotation, and by applying Gaussian noise, to differentiate the augmented images. Grad-CAM was applied as a post-hoc explainability technique to get insights from the CNN model. The 3D CNN attained a sensibility of 80.00%, a specificity of 69.23%, and an accuracy of 73.91%. Four-fold cross-validation was performed as an evaluation method.

In a preliminary previous work [8], DeepFCM was implemented in the context of a research-funded project named EMERALD [9] for the diagnosis of NSCLC assesing PET images with the diameter of the SPN and SUVmax variable as two only clinical features. RGB-CNN was constructed from scratch, and trained on PET images, where the CNN predictions for each image instance were included as an additional input concept. The FCM-weight analysis revealed the interconnections between various concepts, illustrating how they influence each other. In addition, DeepFCM was employed for the effective diagnosis of Coronary Artery Disease (CAD) with Polar map images, along with clinical and demographic information about the patients presenting the FCM-weighted analysis of concepts [10] and in [11], where the results were enhanced with the incorporation of visual (Grad-CAM) and textual (supported by language models) explanations. This way, we go a step ahead from eXplainable Artificial Intelligence (XAI) methodologies towards Trustworthy AI [12].

This study aims to develop DeepFCM with the following set of clinical features, which include patient demographic information like gender, age, Body Mass Index (BMI), Glucose Level (GLU) value, and definite parameters such as SPN location, type, and margins, along with PET image data for NSCLC diagnosis. Robust XAI techniques are employed to facilitate understanding of the model, including Grad-CAM, which explains the decision-making process of CNN results. Moreover, Natural Language Generation (NLG) techniques translate DeepFCM outputs into human-readable linguistic pieces of information, further enhancing the overall clarity and transparency of the model's predictions.

The remainder of the paper is organized as follows: Section II presents the methods and methodology, including an overview of the patient data and the DeepFCM approach. Section III details the research results, while Section IV provides the concluding remarks.

II. MATERIAL AND METHODS

This section details the data acquisition process for PET data, followed by the steps of the proposed DeepFCM methodology.

A. Patient Data

The PET/CT image data was recorded in the Clinical Sector of the Department of Nuclear Medicine at the University Hospital of Patras using a hybrid PET/CT scanner (Discovery iQ3 sl16, General Electric Healthcare). This system uses three detector rings with a 15cm field of view to reconstruct 35 axial images at 4.25mm intervals. 3D volumes are acquired to represent the whole body using various bed positions. At the same time, the patient was in a supine position. Two experienced human readers (N.P., 10 years of experience, D.J.A., 30 years of experience) characterized the SPN malignancy with patient follow-up. The study's nature waives the requirement for obtaining patients' informed consent. From 2020 to 2023, more than 800 PET/CT scans were reviewed to identify potential participants. Patients without detected SPNs or with SPNs with a diameter greater than 30mm were excluded. 456 patients with a single SPN were qualified. The total benign cases were 222 and the total malignant cases were 234. Experts annotated CT scan slices, noting the finding's type, location, margins, diameter, and SUVmax and SPN diameter along with demographic information about each patient (gender, age, BMI). The SUVmax and diameter parameters were extracted from the PET scan. Each SPN finding is represented by a single 2D slice in which the full extent of the nodule is visible.

B. Deep Fuzzy Cognitive Map

In this research paper, we demonstrate the capabilities of our MDSS, specifically highlighting the DeepFCM method for diagnosing NSCLC using PET images alongside the clinical characteristics. The FCM-based model processed the values as input concepts, leveraging FCM's ability to convert input knowledge into system concepts with established causal relationships among them [13]. Expert knowledge is provided in the form of fuzzy sets with linguistic values defining the input-output interconnections among concepts. The linguistic values are transformed into numerical values to be utilized in the algorithm [14]. As interpretability techniques, Grad-CAM was employed to interpret CNN

predictions, while NLG was used to translate the DeepFCM outputs into human-readable explanations. Figure 1 presents the full methodological framework of the DeepFCM approach used in this study, detailing both the training process and the inference mechanism within the MDSS functionality. Following, the steps of the DeepFCM methodological process are analyzed.

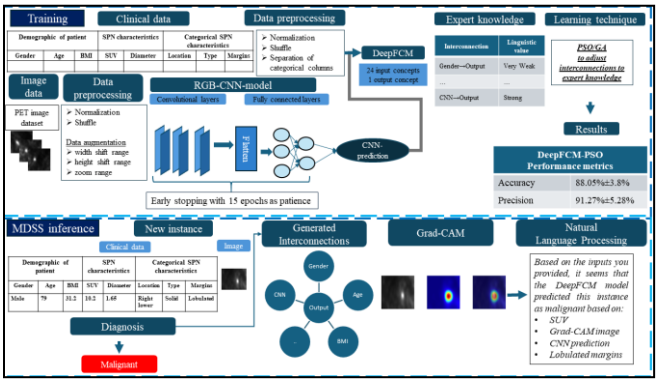


Figure 1. DeepFCM Methodological Pipeline Framework for NSCLC Diagnosis Using Clinical Data and PET Imaging Data

The clinical data includes patient demographic information about the patient such as gender, age, and BMI, GLU, as well as characteristics of the SPN such as SUVmax value, nodule diameter, and three SPN categorical variables type, location, and margins each segmented by their respective values.

First, age was preprocessed with the Min-Max normalization technique [15] and the variables BMI, GLU, SUV, and diameter were divided with their maximum value of 70, 192, 30, and 3 accordingly to be rescaled into the spectrum [0,1]. The SPN categorical variables, including location, type, and margins, were separated into individual columns, with one-hot encoding process. The separated columns generated from categorical variables with SPN characteristics along with the demographic of the patient result in a total of twenty-three clinical characteristics. RGB-CNN was constructed from scratch and trained with the PET dataset; being able to extract a prediction for each image instance. The RGB-CNN predictions were incorporated as an additional input concept alongside clinical data, collectively forming twenty-four distinct input concepts for the DeepFCM model. This model leverages both clinical values and CNN-derived predictions, integrating them into a cohesive framework that enhances interpretability and insight into the diagnostic process. By incorporating clinical and imaging data, DeepFCM generates results that are not only accurate but also transparent, enabling a clear understanding of how each concept impacts the final diagnosis. Through CNN's robust feature extraction from imaging data, combined with the FCM's transparent framework for mapping interconnections, DeepFCM delivers an insightful diagnostic tool that provides clinicians with a nuanced view of the decision-making process. This combined approach strengthens the system's capacity to guide decisions, making DeepFCM a comprehensive and interpretable tool in the context of medical diagnostics [11]. A 10-fold cross-validation approach was implemented to ensure the generalizability of results by partitioning the dataset into 10 batches, where each batch served as the testing fold while the remaining nine served as the training folds [16].

In this study, Particle Swarm Optimization (PSO), and Genetic Algorithm (GA) were incorporated into the DeepFCM learning process to adjust the interconnections among concepts and thus be in line with the provided expert knowledge. PSO [17] is a population-based approach with particles exploring the search space. PSO can be integrated into the FCM learning process by treating the interconnections (weights) between concepts as particles in the search space. Each particle (weight) adjusts its position based on its own best-known position and the best-known positions of its neighbors, iteratively optimizing the FCM weight matrix to minimize the error between predicted and actual outcomes. GA integrates with FCM learning by encoding the weights between concepts as chromosomes, which evolve over multiple generations. Through selection, crossover, and mutation, GA searches for the optimal weight matrix that best aligns with expert knowledge, improving the predictive power of the FCM model [18]. Both optimization methods calculate the weights (interconnections) among DeepFCM concepts to improve classification performance. Even though they perform similarly in the benchmark classification metrics, the GA applied for DeepFCM learning emerges with lower computational latency.

Overall, the learning process creates a weight matrix to minimize the error function, used by DeepFCM for NSCLC diagnosis. For each case, DeepFCM uses the selected weight matrix to provide a diagnosis and at the same time to visualize the input-output relationships, enhancing transparency in the decision-making process.

Regarding XAI techniques, Grad-CAM is used to interpret RGB-CNN predictions, by highlighting the most influential regions that signify the prediction. Grad-CAM implemented by Selvaraju [19] leverages the feature maps produced by the final convolutional layer of the CNN to identify the most relevant regions in the image that contribute to the model's prediction. By computing the gradients of the target class concerning these feature maps, Grad-CAM generates a heatmap that highlights the areas of the image most influential in the decision-making process, providing visual insights into the model's focus [19]. In addition, textual explanations were generated with GPT-4, a pre-trained large language model, which has an Application Programming Interface (API) provided by OpenAI [20], as an NLG technique. Namely, a prompt is provided to GPT-4, containing the user-inputted variable values, the DeepFCM result, the CNN prediction, and the corresponding DeepFCM weight values, along with instructions about how to realize the structure of the requested textual explanation. This enables GPT-4 to generate a comprehensive natural explanation of the decision-making process, offering clear insights into how DeepFCM arrived at its diagnosis.

III. RESULTS

This section presents the classification results along with the included XAI techniques, with the DeepFCM, generated interconnections, heatmap image, and NLG reasoning, to interpret the DeepFCM decision-making process. Additionally, it demonstrates the functionality of MDSS in classifying NSCLC diagnoses using PET and clinical data.

A. Classification Results

For classifying NSCLC data into two categories, benign and malignant, CNN models and DeepFCM were applied and compared with the literature state of the art for similar cases. Table I provides a summary of the performance metrics across different investigated models. Initially, the results of the RGB-CNN model, which was trained exclusively on PET image data are presented. Next, DeepFCM results are illustrated, following the proposed multimodal approach integrating clinical and imaging data (see section above), optimized using PSO and GA. Finally, the proposed models are compared against state-of-the-art models cited in the literature review [3]-[5]. Mean values and standard deviations illustrate the consistency of results for each model. Additionally, confidence intervals (CIs) are provided for the second to last model, offering insight into the precision and reliability of its performance estimates.

TABLE I. DEMONSTRATION OF RESULTS

Accuracy	*Loss*	*Sensitivity*	*Specificity*	*Precision*
RGB-CNN model				
83.12%±6.43%	0.3	92.26±6.18%	91.91±9.21%	91.31±5.75%
Proposed study (DeepFCM-PSO and DeepFCM-GA)				
88.14%±3.8%	**0.12**	**88.36±5.23%**	**87.29±7.48%**	**91.27±5.28%**
87.08%±5.96%	0.13	84.56±12.29%	85.38±6.83%	87.79±6.16%
Literature study [3]				
73.23%±6.0%	-	80.08±6.4%	-	75.71±4.8%
Literature study [4]				
85.21 (95% CI: 83.74–86.68)		81.23 (95% CI: 79.22–83.24)	95.37 (95% CI: 92.99–97.75)	
Literature study [5]				
75%±16.2%	-	84%±15.17%	-	-

In particular, RGB-CNN achieved 83.12% accuracy, while DeepFCM's multimodal approach improved the classification accuracy, with PSO reaching 88.14% and GA 87.08%. Incorporating additional clinical information, the overall performance has been enhanced. DeepFCM models showed smaller deviations in the calculated metrics, indicating consistency. State-of-the-art multimodal approaches attained 73.23% [3], 85.21% [4], and 75% [5] accuracy, highlighting the improvements achieved by the proposed model.

B. MDSS illustrative example

We present the DeepFCM results through MDSS for NSCLC diagnosis using the DeepFCM-PSO model, which achieved the best metrics. The DeepFCM graph demonstrates the interconnections among concepts, Grad-CAM provides visual CNN explanations, while GPT-4 translates outputs into clear, understandable interpretations.

1) MDSS Diagnosis

Figure 2 showcases the DeepFCM diagnosis along with the generated DeepFCM graph, illustrating the interconnections between concepts. The patient refers to a 63-year-old male patient with a BMI of 27.8, a GLU value of 89, an SUVmax of 10.2, and an SPN diameter of 2.7 cm located in the right lower lobe. The type of SPN is semi-solid with lobulated margins.

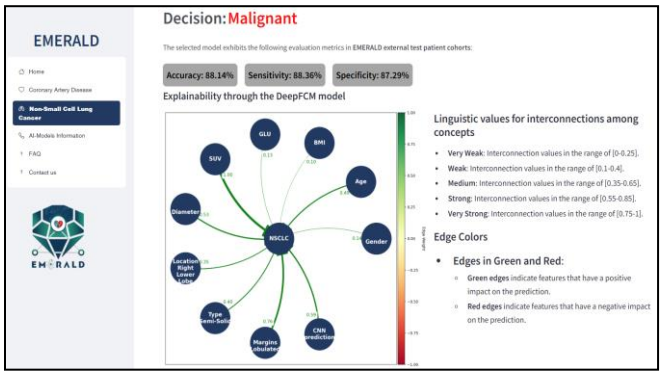


Figure 2. MDSS screenshot: Illustrative example with DeepFCM-PSO Integration for NSCLC Diagnosis Using PET Imaging and Clinical Data.

2) Grad-CAM

Figure 3 showcases the Grad-CAM application through MDSS, with JET colormap to highlight impactful regions in red and less impactful ones in blue. The figure includes the cropped ROI image, the heatmap indicating key areas, and the overlay combining both. The CNN accurately classified the malignant lesion, and Grad-CAM effectively localized and highlighted in red the malignant SPN region, providing visual justification for the RGB-CNN model's prediction of malignancy.

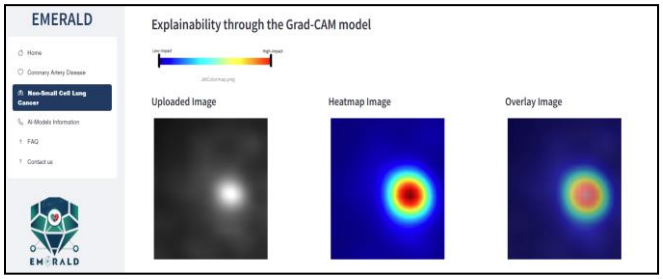


Figure 3. Grad-CAM Application Integrated into MDSS as an XAI Technique for PET Image Analysis.

3) Textual explanations

In Figure 4, we present the results from the GPT-based textual explainer integrated within the MDSS, which offers a

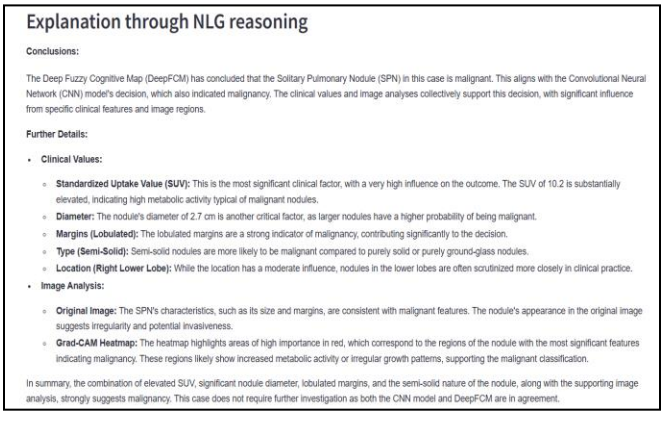
clear and detailed analysis of the clinical factors contributing to the malignancy prediction. A prompt was carefully constructed, incorporating several key components: the CNN and DeepFCM predictions, the input clinical values, the heatmap image generated through Grad-CAM, and the DeepFCM-generated interconnections. This combination allows the explainer to highlight the most influential factors, while seamlessly integrating image analysis and model predictions.

DeepFCM has accurately classified the SPN as malignant, supported by both clinical and imaging data. Key clinical factors such as high SUV (10.2), significant nodule diameter (2.7 cm), and lobulated margins play pivotal roles in this diagnosis. Additionally, the image analysis, including the original scan and Grad-CAM heatmap, reinforces the malignancy prediction by highlighting critical regions associated with increased metabolic activity and irregular growth patterns. The integration of clinical insights, CNN predictions, and visual heatmap evidence provides a comprehensive and explainable diagnosis for the nuclear doctor, ensuring consistency and clarity in the interpretation of the results.

IV. CONCLUSIONS

DeepFCM's capabilities establish it as a fundamental tool for achieving accurate SPN diagnoses in PET images and clinical data while providing explainability of results, enhancing DeepFCM's suitability to be incorporated into doctors' diagnosis process. DeepFCM stands out as a transparent tool for effective NSCLC diagnosis by incorporating explainable methodologies.

DeepFCM's interconnections between clinical and imaging data, using PSO and GA, highlight key features like SUV, nodule diameter, and the SPN type. This alignment validates its accuracy in prioritizing factors influencing NSCLC malignancy. Its ability to reveal meaningful clinical-imaging connections solidifies its potential as a reliable diagnostic tool, aiding clinicians in identifying critical factors for accurate NSCLC diagnosis.



Explanation through NLG reasoning

Conclusions:

The Deep Fuzzy Cognitive Map (DeepFCM) has concluded that the Solitary Pulmonary Nodule (SPN) in this case is malignant. This aligns with the Convolutional Neural Network (CNN) model's decision, which also indicated malignancy. The clinical values and image analyses collectively support this decision, with significant influence from specific clinical features and image regions.

Further Details:

- **Clinical Values:**
  - **Standardized Uptake Value (SUV):** This is the most significant clinical factor, with a very high influence on the outcome. The SUV of 10.2 is substantially elevated, indicating high metabolic activity typical of malignant nodules.
  - **Diameter:** The nodule's diameter of 2.7 cm is another critical factor, as larger nodules have a higher probability of being malignant.
  - **Margins (Lobulated):** The lobulated margins are a strong indicator of malignancy, contributing significantly to the decision.
  - **Type (Semi-Solid):** Semi-solid nodules are more likely to be malignant compared to purely solid or purely ground-glass nodules.
  - **Location (Right Lower Lobe):** While the location has a moderate influence, nodules in the lower lobes are often scrutinized more closely in clinical practice.
- **Image Analysis:**
  - **Original Image:** The SPN's characteristics, such as its size and margins, are consistent with malignant features. The nodule's appearance in the original image suggests irregularity and potential invasiveness.
  - **Grad-CAM Heatmap:** The heatmap highlights areas of high importance in red, which correspond to the regions of the nodule with the most significant features indicating malignancy. These regions likely show increased metabolic activity or irregular growth patterns, supporting the malignant classification.

In summary, the combination of elevated SUV, significant nodule diameter, lobulated margins, and the semi-solid nature of the nodule, along with the supporting image analysis, strongly suggests malignancy. This case does not require further investigation as both the CNN model and DeepFCM are in agreement.

Figure 4. Explanation of DeepFCM Prediction for NSCLC Analysis Using NLG Reasoning.

This approach aligned with nuclear experts' assessments and helped non-specialists understand the model's logic. Grad-CAM detected the SPN region in the PET image, which RGB-CNN classified as malignant. The heatmap highlighted key high metabolic activity areas within the nodule, visually explaining the model's decision. This validated the model's focus on relevant regions, offering clinicians a clear understanding of CNN's classification process, and enhancing transparency in diagnosis.The study has limitations, primarily due to the dataset being sourced from a single hospital, which affects its representativeness and generalizability across different regions and healthcare settings. This may limit the broader applicability of the findings. Further improvement could be achieved by incorporating diverse datasets to enhance model robustness and applicability.

MDSS incorporating DeepFCM is a valuable tool for accurate SPN diagnosis in PET images and clinical data. By providing clear, explainable results, MDSS enhances the diagnostic process, making it a vital asset in clinical settings. This system not only improves the accuracy of diagnoses but also ensures that the reasoning behind each diagnosis is transparent and understandable, fostering trust in AI-driven healthcare solutions.

DATA AVAILABILITY

The dataset used in this study is not publicly available due to hospital medical data privacy constraints. However, it can be provided by the corresponding author upon reasonable request. The source code is available in [21].

ACKNOWLEDGEMENT

The research project was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.) under the "2nd Call for H.F.R.I. Research Projects to support Faculty Members & Researchers" (Project Number: 3656).

REFERENCES

[1] I. D. Apostolopoulos, D. J. Apostolopoulos, and G. S. Panayiotakis, "Solitary Pulmonary Nodule malignancy classification utilising 3D features and semi-supervised Deep Learning," in 2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA), Jul. 2022, pp. 1–6. doi: 10.1109/IISA56318.2022.9904334.

[2] E. Gershman, R. Guthrie, K. Swiatek, and S. Shojaee, "Management of hemoptysis in patients with lung cancer," Annals of Translational Medicine, vol. 7, no. 15, p. 358, Aug. 2019, doi: 10.21037/atm.2019.04.91.

[3] G. Kim, S. Moon, and J.-H. Choi, "Deep Learning with Multimodal Integration for Predicting Recurrence in Patients with Non-Small Cell Lung Cancer," Sensors (Basel), vol. 22, no. 17, p. 6594, Aug. 2022, doi: 10.3390/s22176594.

[4] I. D. Apostolopoulos, N. D. Papathanasiou, D. J. Apostolopoulos, N. Papandrianos, and E. I. Papageorgiou, "A Multi-Modal Machine Learning Methodology for Predicting Solitary Pulmonary Nodule Malignancy in Patients Undergoing PET/CT Examination," Big Data and Cognitive Computing, vol. 8, no. 8, Art. no. 8, Aug. 2024, doi: 10.3390/bdcc8080085.

[5] C. M. Caruso et al., "A Multimodal Ensemble Driven by Multiobjective Optimisation to Predict Overall Survival in

Non-Small-Cell Lung Cancer," Journal of Imaging, vol. 8, no. 11, Art. no. 11, Nov. 2022, doi: 10.3390/jimaging8110298.

[6] T. Li et al., "Fully automated classification of pulmonary nodules in positron emission tomography-computed tomography imaging using a two-stage multimodal learning approach," Quant Imaging Med Surg, vol. 14, no. 8, pp. 5526–5540, Aug. 2024, doi: 10.21037/qims-24-234.

[7] V. M. Alves, J. dos Santos Cardoso, and J. Gama, "Classification of Pulmonary Nodules in 2-[18F]FDG PET/CT Images with a 3D Convolutional Neural Network," Nucl Med Mol Imaging, vol. 58, no. 1, pp. 9–24, Feb. 2024, doi: 10.1007/s13139-023-00821-6.

[8] A. Feleki et al., "Deep Fuzzy Cognitive Map methodology for Non-Small Cell Lung Cancer diagnosis based on Positron Emission Tomography imaging," in 2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA), Jul. 2023, pp. 1–6. doi: 10.1109/IISA59645.2023.10345912.

[9] "HOME," EMERALD. Accessed: Oct. 27, 2024. [Online]. Available: https://emerald.uth.gr/

[10] A. Feleki, I. D. Apostolopoulos, K. Papageorgiou, E. I. Papageorgiou, D. J. Apostolopoulos, and N. I. Papandrianos, "A Fuzzy Cognitive Map Learning Approach for Coronary Artery Disease Diagnosis in Nuclear Medicine," in Fuzzy Logic and Technology, and Aggregation Operators: 13th Conference of the European Society for Fuzzy Logic and Technology, EUSFLAT 2023, Jun. 2023, pp. 14–25. doi: 10.1007/978-3-031-39965-7_2.

[11] A. Feleki et al., "Explainable Deep Fuzzy Cognitive Map Diagnosis of Coronary Artery Disease: Integrating Myocardial Perfusion Imaging, Clinical Data, and Natural Language Insights," Applied Sciences, vol. 13, p. 11953, Nov. 2023, doi: 10.3390/app132111953.

[12] S. Ali et al., "Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence," Inf. Fusion, vol. 99, no. C, Nov. 2023, doi: 10.1016/j.inffus.2023.101805.

[13] E. I. Papageorgiou, N. I. Papandrianos, D. Apostolopoulos, and P. Vassilakos, "Complementary use of Fuzzy Decision Trees and Augmented Fuzzy Cognitive Maps for Decision Making in Medical Informatics," 2008 International Conference on BioMedical Engineering and Informatics, pp. 888–892, May 2008, doi: 10.1109/BMEI.2008.275.

[14] L. S. Jayashree, R. Lakshmi Devi, N. Papandrianos, and E. I. Papageorgiou, "Application of Fuzzy Cognitive Map for geospatial dengue outbreak risk prediction of tropical regions of Southern India," Int. Dec. Tech., vol. 12, no. 2, pp. 231–250, Jan. 2018, doi: 10.3233/IDT-180330.

[15] S. G. K. Patro and K. K. Sahu, "Normalization: A Preprocessing Stage," Mar. 19, 2015, arXiv: arXiv:1503.06462. doi: 10.48550/arXiv.1503.06462.

[16] D. Berrar, "Cross-Validation," 2018. doi: 10.1016/B978-0-12-809633-8.20349-X.

[17] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: an overview," Soft Comput, vol. 22, no. 2, pp. 387–408, Jan. 2018, doi: 10.1007/s00500-016-2474-6.

[18] A. Lambora, K. Gupta, and K. Chopra, "Genetic Algorithm-A Literature Review," in 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Oct. 2019, pp. 380–384. doi: 10.1109/COMITCon.2019.8862255.

[19] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," Int J Comput Vis, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: 10.1007/s11263-019-01228-7.

[20] E. A. M. van Dis, J. Bollen, W. Zuidema, R. van Rooij, and C. L. Bockting, "ChatGPT: five priorities for research," Nature, vol. 614, no. 7947, pp. 224–226, Feb. 2023, doi: 10.1038/d41586-023-00288-7.

[21] "emeraldUTH · GitHub." Accessed: Oct. 28, 2024. [Online]. Available: https://github.com/emeraldUTH

Analyzing Complex Models by Orthogonal Input-Output Decompositions

Pavel Loskot
ZJU-UIUC Institute
Haining, China
pavelloskot@intl.zju.edu.cn

***Abstract*—Devising mathematical models with high accuracy, but otherwise low interpretability as well as explainability is no longer sufficient. This paper proposes a universal, model-agnostic method for achieving a certain degree of explainability of complex mathematical models including the models that are used, for example, in computer simulations. Specifically, the proposed method prescribes how to effectively perform Sobol's outer decomposition of a complex model by exploiting masking of the model inputs by a default value. The masking can not only divide the inputs into multiple orthogonal subspaces, but it also determines the granularity, at which the model explainability is studied. The outputs corresponding to every masked input can be orthogonalized by some existing methods including, for example, the Gram-Schmidt process. It enables readily finding the optimum linear combining of these orthogonal outputs. It should be noted that such a model decomposition is not intended to improve the accuracy by ensemble modeling, but the goal is to uncover an inherent structure and properties of complex models.**

***Keywords*—*Explainability; model-agnostic; multivariate function; orthogonal decomposition; Sobol's decomposition*.**

I. Introduction

Mathematical modeling has become indispensable in many scientific and engineering disciplines. In engineering, for example, mathematical models are necessary for offering credible explanations about how the systems should be designed in any particular way, or why the given system has a particular performance. The widespread adoption of mathematical models is steadily driven by the modeling economics. Thus, designing ever more complex engineering systems and elucidating understanding of many physical and biological systems have become much cheaper and faster when they are studied as mathematical models rather than performing often costly and tedious laboratory or field experiments. Mathematical modeling allows guided searches and systematically exploring very large spaces of domain knowledge by leveraging computing technology and algorithms. The expanding ecosystem of mathematical models leads to much higher information gains, and it also facilitates automated discovery of new knowledge.

Mathematical models can appear in several basic forms. The first obvious form are mathematical expressions, which can be manipulated using algebraic rules and calculus. It is the only form, which guarantees reproducability. The second form are algorithms and computer simulations represented implicitly or explicitly by hierarchical mathematical structures. They are mainly used for computing the outputs for given input values. The third form are the sets of input-output data values. The datasets can be used to infer other forms of mathematical models with a varying degree of accuracy. Deep learning models are particularly popular nowadays due to their universality to fit different kinds of datasets within the same model structure. Such a fundamental property can be attributed to compositional sparsity of computable functions [1].

The model development and analysis often requires understanding how the model outputs are derived from its inputs. It includes understanding how the input-output transformation is affected by the model structure as well as by different sets of model parameters. Such a task has been traditionally referred to as sensitivity analysis [2], and it is key in providing the model interpretability. It is generally accepted that there is a trade-off between the model interpretability and the model accuracy [3], although this assumption is currently being debated. The model interpretability enables a variety of model-related tasks such as calibrating, optimizing, selecting, validating and simplifying the models, and making the models more robust by reducing the model uncertainties.

The basic strategy for performing a local sensitivity analysis of the model exploits derivatives in multiple input dimensions [2]. The global sensitivity analysis can be obtained by expanding the model outputs directly, or by expanding the model output means or variances in terms of the input statistics [2], [4]. The surrogate or meta models can be particularly effective in reducing the computational costs, and providing the faster convergence. The challenge is how to preserve the key properties of the original model [4].

Furthermore, since the models are usually used to provide a certain functionality within high-level applications, it may be easier as well as sufficient to examine the model explainability [5]. Unlike interpretability, explainability does not require understanding a complete model structure. Instead, explainability focuses on a more narrow objective of identifying, which model inputs are more important in determining the model outputs, This problem is also referred to as input factors screening, or attribution problem. The most popular methods for achieving the model explainability are model-agnostic. They include permutation importance of features, various dependency plots (e.g., individual conditional expectation plots, and partial dependency plots), local interpretable model-agnostic explanations (LIME) [6], and Shapley additive explanations (SHAP) [7].

 ISBN: 978-1-68558-215-9

In general, orthogonalizations can be used to improve the convergence, and even the performance of models [8]. The orthogonalization injects separability into the model, which then leads to a reduced complexity and increased robustness, since the model components have less effect on each other. The orthogonal model components can be added or removed without requiring to change the existing model structure. In the literature, various orthogonalization methods of linear and non-linear models were considered. For example, the Gram-Schmidt process together with variable projections was adopted in [9] to fit the data with a linear combination of non-linear models. The Gram-Schmidt process was also used in [10] to improve learning of deep neural networks. The mixing of orthogonalized linear models was examined in [11] to improve the scale and the convergence of data fitting. The properties of orthogonalized linear regression were studied in [12], [13]. A faster convergence of statistical parameter estimation was obtained by exploring orthogonal statistical moments in [14]. The methods for achieving explainability in deep neural networks were reviewed in [15], although without assuming any orthogonalization strategies.

The main problem with the above works is they implicitly assume that the model under consideration is a linear combination of multiple sub-models. This excludes a large number of other models, for example, the models that are used in computer simulations, where explicit mathematical description is often very limited. In this paper, we overcome such a limitation by defining the model components using multiple input projections with the model being considered, so it does not matter how the given model is specified.

More specifically, our objective is to explore an inherent structure of complex models by orthogonal projections of their inputs and outputs. The proposed method obviates the need to mathematically manipulate the model, so it is completely model agnostic. The only requirement is that the model being investigated is already "good enough". The proposed method is primarily motivated by Sobol's decomposition of multivariate functions [16]. The proposed method is similar to SHAP method except that it does not require obtaining multiple models, for example, retraining multiple machine learning models, for different input subspaces. Moreover, both input and output subspaces can be made orthogonal. It enables exploiting other useful properties, and providing insights into the dependency and importance of model inputs and outputs.

The rest of the paper is organized as follows. Section II reviews common decomposition strategies of multivariate functions. Section III introduces the proposed decomposition method for explainability of complex models. Numerical example is investigated in Section IV. Conclusion and future work are summarized in Section V.

II. Decompositions of Multivariate Functions

Function decompositions allow uncovering their latent structure, reducing the computational complexity by enabling divide & conquer strategies, obtaining approximations, which are amenable to optimizations and analysis, and most importantly, they can also provide explainability. In the literature, function decomposition is also sometimes referred to as function factorization, and function expansion, respectively, depending on the specific objectives assumed.

Consider a multivariate vector function,

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}(x_1,\ldots,x_I) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ \vdots \\ f_O(\boldsymbol{x}) \end{bmatrix} \in \mathcal{R}^O,\ \boldsymbol{x} \in \mathcal{R}^I \quad (1)$$

representing a mapping between the Euclidean vector spaces, $\mathcal{R}^I \mapsto \mathcal{R}^O$. There are two basic function decomposition strategies. In particular, the decomposition into n product-factors can be written as,

$$\boldsymbol{f}(\boldsymbol{x}) = \prod_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{s}_i),\ \boldsymbol{s}_i \subseteq \{x_1,\ldots,x_I\} \quad (2)$$

whereas the decomposition into n sum-factors is defined as,

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{s}_i),\ \boldsymbol{s}_i \subseteq \{x_1,\ldots,x_I\}. \quad (3)$$

These decompositions are very useful for effectively performing, for example, the function marginalization and maximization over all except a small number of independent input variables. Moreover, it is usually easier to express a complicated support region, $\mathcal{A}$, of the function, $\boldsymbol{f}(\boldsymbol{x})$, $\forall \boldsymbol{x} \in \mathcal{A}$, using a scalar decision function, $A(\boldsymbol{x})$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x})A(\boldsymbol{x}) = \begin{cases} \boldsymbol{f}(\boldsymbol{x}), & \boldsymbol{x} \in \mathcal{A} \\ 0, & \boldsymbol{x} \notin \mathcal{A} \end{cases} \quad (4)$$

which enforces zero function values, when the inputs are outside the support region. Since $A(\boldsymbol{x})$ is itself a multivariate function, it can be decomposed using the same factorizations.

The multivariate Taylor expansion [17] was assumed in [4] to obtain a polynomial expansion of stochastic functions in multiple dimensions, i.e.,

$$\boldsymbol{y} \approx \sum_{i=1}^{n} a_i\, |\boldsymbol{x} - \mathrm{E}[\boldsymbol{x}]|_1^i \quad (5)$$

where $\mathrm{E}[\cdot]$ denotes the expectation, and $|\cdot|_1$ is the absolute value of a sum of the vector or matrix elements. The main issue with the Taylor-based function expansion is that it is very localized, so one has to decide about which point in the input space the function is to be approximated.

In the literature, there are different versions of the universal approximation theorem [18]. Specifically, this theorem claims that certain broad classes of multivariate functions can be approximated to an arbitrary accuracy by compounding a sufficient number of linear transformations followed by a dimension-wise non-linearity (activation function), i.e.

$$\boldsymbol{y} \approx = \cdots\ \boldsymbol{\sigma} \circ (\boldsymbol{A}_i, \boldsymbol{b}_i) \circ \cdots \boldsymbol{\sigma}(\boldsymbol{A}_1\boldsymbol{x} + \boldsymbol{b}_1). \quad (6)$$

The corresponding computing structure is known as a multilayer perceptron (MLP).

Alternatively, the Kolmogorov-Arnold theorem claims that multivariate scalar functions defined on a unit hypercube can

be approximated to an arbitrary accuracy by assuming the decomposition [19],

$$f(\boldsymbol{x}) = \sum_{i=0}^{2n} \Phi \left(\sum_{j=1}^{n} \phi_{i,j}(x_j) \right). \tag{7}$$

The caveat is that, in practice, the non-linear functions, Φ, and, $\phi_{i,j}$, can be very peaky, or otherwise ill-shaped. The corresponding computing structure is known as the Kolmogorov-Arnold network (KAN) [20].

Another function decomposition, which will be assumed in the following section to enable explainability of complex models, is due to Sobol [2], [16]. It is referred to as Sobol's decomposition, and it is an example of the sum-factors decomposition (3). In particular, a multivariate vector function, $\boldsymbol{f}$, can be systematically expanded as,

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{f}_0 + \sum_{i=1}^{I} \boldsymbol{f}_i(\boldsymbol{x}_i) + \sum_{\substack{i,j=1 \\ i \neq j}}^{I} \boldsymbol{f}_{i,j}(\boldsymbol{x}_i, \boldsymbol{x}_j) \cdots + \sum_{i=1}^{I} \boldsymbol{f}_{\{1:I\}\setminus i}(\boldsymbol{x}) \tag{8}$$

where $\boldsymbol{f}_0$ is a constant vector, $\boldsymbol{x}_i$ denotes the i-th variable of the I-dimensional input vector, $\boldsymbol{x}$, and, $\{1:I\} \setminus i$, removes the index i from the index set, $\{1,2,\ldots,I\}$. Expansion (8) also allows for symmetric component functions, i.e., the functions that are invariant to permutations of their arguments. More importantly, Sobol's decomposition is not unique, however, without any further constraints, it is exact.

There is yet another model-agnostic method for approximating multivariate functions, which turned out to be very effective in many practical scenarios involving complex models that are expensive to evaluate. The basic idea of this method is to approximate the model by the realization of a multi-dimensional Gaussian process [21] assuming only a few points where the function values are known. However, this method is not considered in this paper.

III. Orthogonal Input-Output Decompositions

Kolmogorov-Arnold decomposition (7) assumes only univariate component functions, whereas Sobol's decomposition (8) combines component functions having different number of variables. Assuming the latter, it can be argued that the summands in (8) that are dependent on larger number of input variables are more accurate approximations of the given function, $\boldsymbol{f}(\boldsymbol{x})$, then other component functions having smaller number of variables. However, the component functions with more variables are not only more difficult to obtain, but they are also less interpretable.

In practice, a good trade-off between the decomposition interpretability, complexity, and accuracy can be achieved by assuming non-empty variable subsets, $\boldsymbol{s}^{(i)} \subset \{x_1,\ldots,x_I\}$, $i = 1,\ldots,N$, such that the subsets, $\boldsymbol{s}^{(i)}$, fully cover all input variables, $\boldsymbol{x}$. Here, it is proposed to only consider the following terms in decomposition (8) in order to obtain an interpretable approximate representation of the original function, i.e., let,

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{f}_0 + \sum_{i=1}^{N} \boldsymbol{f}_i(\boldsymbol{s}^{(i)}) + \sum_{\substack{i,j=1 \\ i \neq j}}^{N} \boldsymbol{f}_{i,j}(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)}). \tag{9}$$

The structural interpretation of decomposition (9) is obvious; it is a fully connected graph consisting of N vertices as depicted in Figure 1. The vertices are assigned the component functions, $\boldsymbol{f}_i(\boldsymbol{s}^{(i)})$, while the edges between the vertices are assigned the pairwise component functions, $\boldsymbol{f}_{i,j}(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)})$.

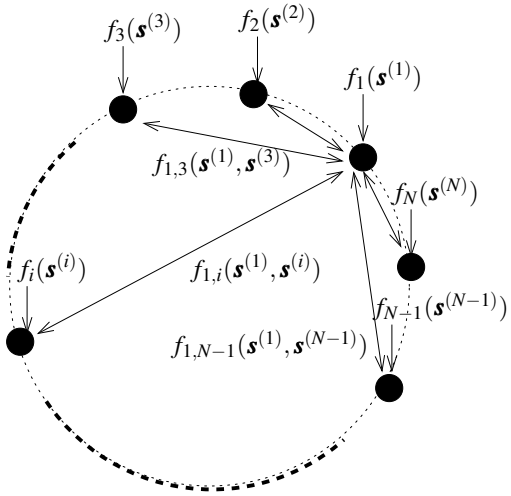


Figure 1. Truncated Sobol's decomposition of a multivariate function as an interpretable and universal representation with good approximation accuracy.

In the sequel, we assume that the multivariate vector function, $\boldsymbol{f}$, represents a complex model, $M(\boldsymbol{x};\boldsymbol{\Omega})$, which is parameterized by a set of parameters, $\boldsymbol{\Omega}$, i.e.,

$$\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\Omega}) \equiv M(\boldsymbol{x};\boldsymbol{\Omega}) \equiv M_{\boldsymbol{\Omega}}(\boldsymbol{x}). \tag{10}$$

The parameters, $\boldsymbol{\Omega}$, represent additional input dimensions of the model. In practice, conditioning on parameters defines a whole class of models, $M_{\boldsymbol{\Omega}}(\boldsymbol{x})$, so the input dimensions are only represented by $\boldsymbol{x}$. For example, the model, M, can be a deep learning model with learnable parameters, $\boldsymbol{\Omega}$. The overall process of extracting the proposed model structure as decomposition (9) is shown in Figure 2.

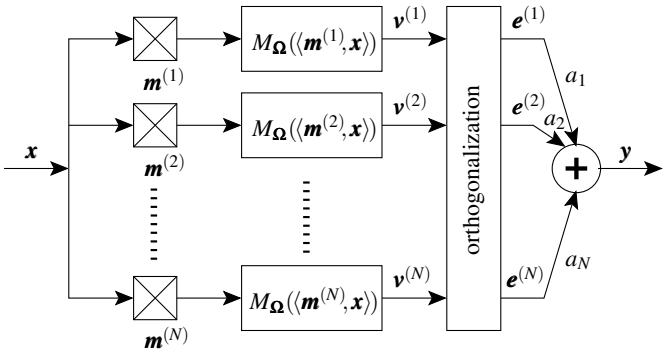


Figure 2. Model-agnostic decomposition (9) for extracting the input-output structure of complex models, $M_{\boldsymbol{\Omega}}(\boldsymbol{x})$.

A. Orthogonal Inputs

It is desirable to consider the projections of model inputs into multiple independent subspaces, since the information

contents of a sum of independent components is then equal to the sum of the information contents of these components. The vector subspaces are independent, provided that they are mutually orthogonal. Thus, assume the subsets of input variables, $\boldsymbol{s}^{(i)}$, satisfying,

$$\begin{cases} \boldsymbol{s}^{(i)} \cap \boldsymbol{s}^{(j\neq i)} = \emptyset & \text{(disjoint)} \\ \cup_i \boldsymbol{s}^{(i)} = \boldsymbol{x} & \text{(full coverage)} \end{cases}. \tag{11}$$

If the model, M, is complex, and expensive to evaluate (a typical case for deep learning models), the question arises how to effectively find the component functions, $\boldsymbol{f}_i$, and, $\boldsymbol{f}_{i,j}$, in decomposition (9). A possible solution is to combine the given multi-dimensional complex model with the orthogonal subspace projections of its inputs as shown next.

Let the input variables be arranged in a column vector, $\boldsymbol{x}$. Define the binary mask vectors, $\boldsymbol{m}^{(i)}$, $i = 1,\ldots,I$, of the same length as $\boldsymbol{x}$, i.e., $\boldsymbol{m}^{(i)} \in \{0,1\}^I$. The zeros in $\boldsymbol{m}^{(i)}$ indicate, which components in $\boldsymbol{x}$ should be masked by setting them to some default value, for example, to zero. Then, by slightly abusing the notation when using the same symbol for a set as well as for its vector representation, the orthogonal (i.e., mutually exclusive) variable subsets, $\boldsymbol{s}^{(i)}$, can be represented as vectors,

$$\boldsymbol{s}^{(i)} = \langle \boldsymbol{m}^{(i)}, \boldsymbol{x} \rangle = (\boldsymbol{m}^{(i)})^T \cdot \boldsymbol{x} = \sum_{j=1}^{I} \boldsymbol{m}_j^{(i)} \boldsymbol{x}_j \tag{12}$$

where $\langle \cdot,\cdot \rangle$ denotes the inner product of two vectors. The corresponding output vectors for every masked input are,

$$\boldsymbol{v}_i = M_{\boldsymbol{\Omega}}(\langle \boldsymbol{m}^{(i)}, \boldsymbol{x} \rangle), \quad i = 1,2,\ldots,N. \tag{13}$$

B. Orthogonal Outputs

After computing the N output vectors for all N masked inputs, it is useful to explore their structure too. In particular, the outputs can be assumed to be deterministic vectors in an O-dimensional vector space. In such a case, they can be orthogonalized by the Gram-Schmidt procedure, provided that, $O \geq N$. In particular, the orthogonal vectors, $\boldsymbol{e}_i$, are obtained as linear projections of vectors, $\boldsymbol{v}^{(i)}$, using the recurrence,

$$\boldsymbol{e}^{(i)} = \boldsymbol{v}^{(i)} - \sum_{j=1}^{i-1} \langle \boldsymbol{v}^{(i)}, \boldsymbol{e}^{(j)} \rangle \, \boldsymbol{e}^{(j)}, \; i = 2,3,\ldots,N \tag{14}$$

with the initial vector, $\boldsymbol{e}^{(1)} = \boldsymbol{v}^{(1)}$. Note that orthogonalization (14) can be expressed as a linear transformation of vectors, $\boldsymbol{v}^{(i)}$. The caveat is that the linear transformation must be recomputed for every new set of vectors, $\boldsymbol{x}$.

In practice, often, $N \gg O$, which rules out the Gram-Schmidt procedure (more precisely, the vectors with O components can span the subspaces in at most O dimensions). In such a case, other orthogonalization strategies are possible that exploit various matrix factorizations. In this paper, the vectors, $\boldsymbol{v}^{(i)}$, are decorrelated by first finding their empirical correlation matrix, $\boldsymbol{C}^{\boldsymbol{v}}$. The elements of $\boldsymbol{C}^{\boldsymbol{v}}$ are the average inner products. They can be computed recursively as soon as the k-th set of vectors, $\boldsymbol{v}^{(i)}(k)$, has been obtained, i.e.,

$$\boldsymbol{C}_{i,j}^{(\boldsymbol{v})}(k) = \frac{k}{k+1} \boldsymbol{C}_{i,j}^{(\boldsymbol{v})}(k-1) + \frac{1}{k+1} \langle \boldsymbol{v}^{(i)}(k), \boldsymbol{v}^{(j)}(k) \rangle. \tag{15}$$

Since the matrix, $\boldsymbol{C}^{(\boldsymbol{v})}$, is guaranteed to be positive-definite, it can be decomposed into a product of the square matrix, $\boldsymbol{D} \in \mathcal{R}^{N\times N}$, i.e., , $\boldsymbol{C}^{(\boldsymbol{v})} = \boldsymbol{D}^T \boldsymbol{D}^T$, for example, using a singular value decomposition (SVD). The matrix, $\boldsymbol{D}$, can be then used to decorrelate, i.e., orthogonalize the vectors, $\boldsymbol{v}^{(i)}(k)$, as,

$$\boldsymbol{E}(k) = [\boldsymbol{e}^{(1)}(k) \; \cdots \; \boldsymbol{e}^{(N)}(k)] = \boldsymbol{V}(k) \boldsymbol{D}^{-T}(k) \in \mathcal{R}^{O\times N} \tag{16}$$

where the vectors at instant, k, are gathered into matrices, $\boldsymbol{E}(k)$, and, $\boldsymbol{V}(k)$, respectively.

The resulting vectors, $\boldsymbol{e}^{(i)}$, are orthogonal, so that,

$$\langle \boldsymbol{e}^{(i)}, \boldsymbol{e}^{(j)} \rangle = \begin{cases} E_i, & i = j \\ 0, & i \neq j \end{cases} \tag{17}$$

where the squared (Euclidean) length of these vectors is,

$$E_i = \langle \boldsymbol{e}^{(i)}, \boldsymbol{e}^{(i)} \rangle = \left\| \boldsymbol{e}^{(i)} \right\|^2. \tag{18}$$

The values, E_i, can be again computed recursively as,

$$E_i(k) = \frac{k}{k+1} E_i(k-1) + \frac{1}{k+1} \left\| \boldsymbol{e}^{(i)}(k) \right\|^2. \tag{19}$$

Recall also that, when the vectors, $\boldsymbol{v}^{(i)}$, are assumed to be random (the $N \gg O$ case), the inner products are computed as empirical means (cf. eq. (15)).

Finally, the orthogonal vectors, $\boldsymbol{e}^{(i)}$, are linearly combined to create the output vector, $\boldsymbol{y}$, as,

$$\boldsymbol{y}(k) = \sum_{i=1}^{N} a_i \boldsymbol{e}^{(i)}(k). \tag{20}$$

It is immediately obvious, why to make the vectors, $\boldsymbol{v}^{(i)}$, orthogonal. Thus, by multiplying both sides of (20) by the vector, $\boldsymbol{e}^{(j)}$, and averaging, the combining coefficients can be computed one-by-one as,

$$\sum_k \langle \boldsymbol{e}^{(j)}(k), \boldsymbol{y}(k) \rangle = \sum_k \sum_{i=1}^{N} a_i \langle \boldsymbol{e}^{(j)}(k), \boldsymbol{e}^{(i)}(k) \rangle = a_j \bar{E}_j \tag{21}$$

$$\Rightarrow \quad a_j = \langle \boldsymbol{e}^{(j)}, \boldsymbol{y} \rangle / E_j = \langle \boldsymbol{e}^{(j)}, M_{\boldsymbol{\Omega}}(\boldsymbol{x}) \rangle / E_j \tag{22}$$

where the desired output vector, $\boldsymbol{y} = M_{\boldsymbol{\Omega}}(\boldsymbol{x})$.

C. Obtaining Decomposition (9)

The procedure described so far can be readily used to obtain the first two summands in decomposition (9), i.e., $(N+1)$ functions, $\{\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$. Even though it is possible to also obtain, at the same time, the second-order functions, $\boldsymbol{f}_{i,j}$, the joint orthogonalization of all $(N(N+1)/2)$ vectors, $\boldsymbol{v}^{(i)} = M_{\boldsymbol{\Omega}}(\langle \boldsymbol{m}^{(i)}, \boldsymbol{x} \rangle)$, and, $\boldsymbol{v}^{(i,j)} = M_{\boldsymbol{\Omega}}(\langle \boldsymbol{m}^{(i,j)}, \boldsymbol{x} \rangle)$, is rather cumbersome. In order to overcome this difficulty, decomposition (9) can be performed in two steps. In particular, after obtaining the zero and the first-order functions, $\{\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_N\}$, the $N(N-1)/2$ second order functions, $\boldsymbol{f}_{i,j}$, are obtained in the second step in order to approximate the left-hand side of the expression,

$$\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}_0 - \sum_{i=1}^{N} \boldsymbol{f}_i(\boldsymbol{s}^{(i)}) \approx \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \boldsymbol{f}_{i,j}(\boldsymbol{s}^{(i)}, \boldsymbol{s}^{(j)}). \tag{23}$$

This process can be continued to obtain the $N(N-1)(N-2)/6$ third-order functions, $\boldsymbol{f}_{i,j,k}$, and so on. However, and importantly, the input masks for higher-order functions are no longer assumed to be disjoint (i.e., orthogonal), as they are created by combining the first-order masks, $\boldsymbol{m}^{(i)}$; for example,

$$\begin{aligned} \boldsymbol{m}^{(i,j)} &= \boldsymbol{m}^{(i)} + \boldsymbol{m}^{(j)} \\ \boldsymbol{m}^{(i,j,k)} &= \boldsymbol{m}^{(i)} + \boldsymbol{m}^{(j)} + \boldsymbol{m}^{(k)}. \end{aligned} \tag{24}$$

IV. Numerical Example

Machine learning is assumed as an example of a complex model to illustrate the proposed explainability method. The well-known MNIST dataset [22] of hand-written digits is used with a basic MLP classifier. The training samples are gray-scale images of (28×28) pixels with pixel values between 0 and 1. The MLP has two hidden layers with 30 and 20 neurons, respectively, and 10 softmax outputs. The MLP was trained on all samples over only 10 epochs. The trained model reached the training accuracy of 97.56%, and the testing accuracy of 94.31%. The model was implemented using Python class, "MLPClassifier", from the Python module, "sklearn.neural_network".

The goal is to decompose the trained MLP model into the explainable structure in Figure 2. There are two key aspects to investigate. First, we can compare the training samples from the MNIST dataset, on which the MLP model was trained, against the randomly generated inputs. Second, the orthogonal masks can be compared with the masks generated at random. In both cases, we compute the mean-square error (MSE) between the trained MLP output, and the combined output of the expanded model in Figure 2.

In the experiment concerning different distributions of inputs, in addition to training and testing samples from the MNIST dataset, the same number of (28×28) independent random inputs were generated from a uniform distribution. The resulting MSE values are summarized in Table I, for $N = 7, 14, 28, 56$ and 112 subspace projections, respectively. The MSE values were averaged over K input samples and N model components as follows:

$$\begin{aligned} \overline{\mathrm{MSE}}_0 &= \frac{1}{K}\sum_{k=1}^{K} \|\boldsymbol{y}(k)\|^2 \\ \overline{\mathrm{MSE}}_1 &= \frac{1}{NK}\sum_{i=1}^{N}\sum_{k=1}^{K} \left\|\boldsymbol{v}^{(i)}(k) - \boldsymbol{y}(k)\right\|^2 \\ \min \mathrm{MSE}_1 &= \min_{1\le i\le N} \frac{1}{K}\sum_{k=1}^{K} \left\|\boldsymbol{y}^{(i)}(k) - \boldsymbol{y}(k)\right\|^2 \\ \overline{\mathrm{MSE}}_2 &= \frac{1}{K}\sum_{k=1}^{K} \|\tilde{\boldsymbol{y}}(k) - \boldsymbol{y}(k)\|^2 \end{aligned} \tag{25}$$

where $\boldsymbol{y} = M_{\boldsymbol{\Omega}}(\boldsymbol{x})$ is the output of the trained model, $\boldsymbol{v}^{(i)}$ is the output of the model using the i-th mask as its inputs, and $\tilde{\boldsymbol{y}}$ denotes the overall combined output of the decomposed model. Note also that $\overline{\mathrm{MSE}}_0$ values are independent of N.

Examining the MSE values in Table I, it is obvious that MSE_1 values are comparable to $\overline{\mathrm{MSE}}_0$ values, i.e., masking the inputs can substantially reduce the classifier accuracy in exchange for better interpretability. More importantly, combining all N outputs of the N model replicas with masked inputs can greatly improve the accuracy. Thus, assuming not only the first-order functions in decomposition (9) is expected to further improve the approximation accuracy. Surprisingly, the number of masks N considered seems to have a little effect on the MSE. For training samples from the MNIST dataset, using random or systematic masks make a little difference. However, for randomly generated inputs, the original MLP model and the expanded model have very similar MSE values.

TABLE I. COMPARISON of MSE VALUES

N	inputs	masks	$\overline{\mathrm{MSE}}_0$	$\overline{\mathrm{MSE}}_1$	min MSE_1	$\overline{\mathrm{MSE}}_2$
7	train.	rand.	0.953	0.821	0.717	0.791
		systm.	0.953	1.2085	0.912	0.767
	rand.	rand.	0.416	0.432	0.336	0.275
		systm.	0.418	0.892	0.466	0.253
14	train.	rand.	0.953	0.921	0.805	0.739
		systm.	0.953	1.1028	0.835	0.739
	rand.	rand.	0.417	0.430	0.330	0.270
		systm.	0.417	0.727	0.362	0.252
28	train.	rand.	0.953	0.936	0.866	0.741
		systm.	0.953	1.0348	0.844	0.751
	rand.	rand.	0.417	0.442	0.322	0.258
		systm.	0.416	0.584	0.279	0.233
56	train.	rand.	0.953	0.947	0.850	0.753
		systm.	0.953	0.986	0.877	0.749
	rand.	rand.	0.417	0.437	0.330	0.269
		systm.	0.418	0.511	0.328	0.233
112	train.	rand.	0.954	0.950	0.890	0.731
		systm.	0.954	0.972	0.878	0.742
	rand.	rand.	0.418	0.465	0.351	0.280
		systm.	0.417	0.474	0.315	0.232

Next, we display the correlation matrix, $\boldsymbol{C}^{\boldsymbol{v}}$, and, the probabilities, $\boldsymbol{P}_{i,j}$, that the MLP output predictions for the masked inputs agree with the predictions obtained when the masks are combined. We again consider four cases as in Table I. Specifically, the probabilities, $\boldsymbol{P}_{i,j}$, are defined for the pairs of masks, $\boldsymbol{m}^{(i)}$, and, $\boldsymbol{m}^{(j)}$, $1 \le i \ne j \le N$, as the ratios of the number of instances when the decisions for the two masked inputs are the same as the decision when the two masks are combined, i.e., for the mask, $\boldsymbol{m}^{(i)} + \boldsymbol{m}^{(j)}$. The calculated matrices, $\boldsymbol{C}^{\boldsymbol{v})}$, and, $\boldsymbol{P}$, are shown in Figure 3.

In Figure 3, there is an apparent line marking the main diagonal. All sub-figures are symmetric about the main diagonal. Increasing N not only increases the resolution, but also the variety of calculated values. There are clearly observable patterns (the squares of various sizes and color shades) indicating that orthogonal input subspaces might lead to similar output decisions due to inherent cross-correlations. The probabilities (right-column sub-figures) related to the final output decisions appear to have visually richer patterns than the pairwise cor-

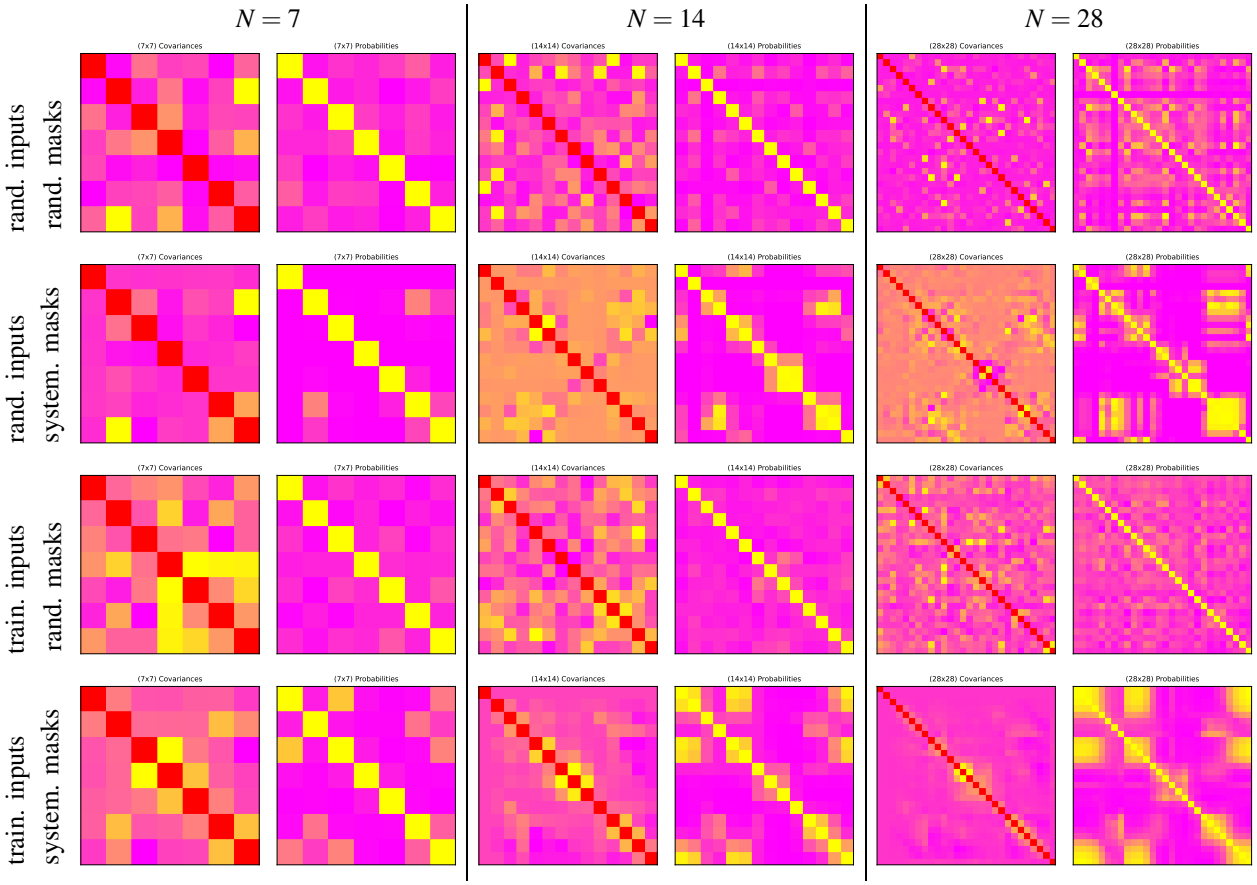


Figure 3. The calculated correlations $\boldsymbol{C}^{\boldsymbol{v}}$ (left columns), and the decision probabilites (defined in the text; right columns). For comparison purposes, the units are arbitrary, and the darker colors represent smaller values.

relations (left-column sub-figures). Interestingly, the patterns for $N = 28$ start emerging as being in 3D. There is a clear difference between the models processing random inputs, and processing the inputs, on which they were trained. This can be exploited for detecting the out-of-sample distributions. Similar claims can be made about systematic (orthogonal) masks vs. random masks. In both cases, the differences become more recognizable when N is increased.

V. Conclusion and Future Work

Sobol's decomposition of multivariate functions was adopted to expand complex models, and to support their input-output explainability. This was achieved by masking the inputs, which is equivalent to projections into orthogonal subspaces. The component model outputs can be orthogonalized in order to facilitate their linear combination. The number of components in model expansion is a trade-off between computational complexity and explainability. For MLP classifier trained on the MNIST dataset, a good value of the number of components for (28×28) inputs seems to be $N = 28$.

The proposed method opens up many opportunities for future research. For instance, providing explainability for complex models requires that explainability is sufficiently simple, or at least much simpler than the model to be explained. For large number of inputs and outputs, the number of possible orthogonal projections, and thus, the number of possible explanations, is exponentially large. It requires to analyze how to choose these projections for a particular explainability objective, which can be defined, for example, as an optimization problem. Moreover, the inputs can be averaged out from the model instead of being masked. There are several other methods for orthogonalizing vectors that can be considered involving, e.g., matrix factorizations (QR, Cholesky, PCA) and lattice reductions. There may be other strategies how to define the components of complex models. The models with orthogonal components can be used as complex basis functions for generating samples with the desired properties instead of focusing on explainability. The dependencies between the model components that are not orthogonal can be studied as structural causal models using statistical methods of causal inferences. Also, the vector space of model parameters can be orthogonalized similarly as the vector space of model inputs in order to perform the sensitivity analysis. Considering Sobol's expansion itself, it is useful to investigate how additional higher-order terms can be used to create higher-order graphs having beyond pairwise interactions. There is also a need to obtain the approximation bounds for Sobol's decomposition, which does seem to have been provided in the literature.

 ISBN: 978-1-68558-215-9

REFERENCES

[1] T. Poggio and M. Fraser, "Compositional sparsity of learnable functions," *Bulletin AMS*, vol. 61, pp. 438–456, May 2024.

[2] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola, *Global Sensitivity Analysis: The Primer*. John Wiley & Sons, Ltd, Chichester, England, 2008.

[3] U. Johansson, C. Sönströd, U. Norinder, and H. Boström, "Trade-off between accuracy and interpretability for predictive in silico modeling," *Future Medicinal Chemistry*, vol. 3, no. 6, pp. 647–663, May 2011.

[4] P. Loskot, "Polynomial representations of high-dimensional observations of random processes," *Mathematics*, vol. 9, no. 123, pp. 1–24, Jan. 2021.

[5] Y. Liang, S. Li, C. Yan, M. Li, and C. Jiang, "Explaining the black-box model: A survey of local interpretation methods for deep neural networks," *Neurocomputing*, vol. 419, pp. 168–182, Sep. 2021.

[6] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?" explaining the predictions of classifier," in *KDD*, 2016, pp. 1135–1144.

[7] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *NIPS'17*, 2017, pp. 4768–4777.

[8] J. A. Mumford, J.-B. Poline, and R. A. Poldrack, "Orthogonalization regressors in fMRI models," *PLOS One*, vol. 10, no. 4:e0126255, 2015.

[9] G.-Y. Chen, M. Gan, F. Ding, and C. L. P. Chen, "Modified Gram–Schmidt method-based variable projection algorithm for separable nonlinear models," *IEEE Trans. Neural Networks and Learning Systems*, vol. 30, no. 8, pp. 2410–2418, Aug. 2019.

[10] Y.-P. Zhao, Z.-Q. Li, P.-P. Xi, D. Liang, L. Sun, and T.-H. Chen, "Gram–Schmidt process based incremental extreme learning machine," *Neurocomputing*, vol. 241, pp. 1–17, Jun. 2017.

[11] M. Clyde, H. Desimone, and G. Parmigiani, "Prediction via orthogonalized model mixing," *J. American Statistical Association*, vol. 91, no. 435, pp. 1197–1208, 2012.

[12] Z. Mikulášek, "The benefits of the orthogonal LSM models," Nov. 2007, arXiv:0711.4510v1 [astro-ph].

[13] S. J. Pallavi, S. Dilbag, K. Manjit, and H. Lee, "Comprehensive review of orthogonal regression and its applications in different domains," *Archives Comput. Methods Engineer.*, vol. 29, pp. 4027–4047, 2022.

[14] L. Mackey, V. Syrgkanis, and I. Zadik, "Orthogonal machine learning: Power and limitations," in *PMLR*, vol. 80, 2018, pp. 3375–3383.

[15] G. Ras, N. Xie, M. Gerven, and D. Doran, "Explainable deep learning: A field guide for uninitiated," Apr. 2021, arXiv:2004.14545v2 [cs.LG].

[16] A. Saltelli, M. Ratto, S. Tarantola, and F. Campolongo, "Sensitivity analysis for chemical models," *Chemical Reviews*, vol. 105, pp. 2811–2827, May 2005.

[17] G. B. Folland, "Higher-order derivatives and Taylor's formula in several variables," 2010, Math 425A Course notes.

[18] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366c, Jan. 1989.

[19] G. G. Lorentz, "Metric entropy, widths, and superpositions of functions," *American Mathematical Monthly*, vol. 69, no. 6, pp. 469–485, 1962.

[20] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljaččić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," Jun. 2024, arXiv:2404.19756 [cs.LG].

[21] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *PMLR*, vol. 28(1), 2013, pp. 115–123.

[22] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST dataset," 1998, https://yann.lecun.com/exdb/mnist/.

 ISBN: 978-1-68558-215-9

The Graph Model of Combinatory Logic as a Model for Explainability

Thomas Fehlmann
Euro Project Office AG
8032 Zurich, Switzerland
E-mail: thomas.fehlmann@e-p-o.com

Eberhard Kranich
Euro Project Office
47051 Duisburg, Germany
E-mail: eberhard.kranich@t-online.de

***Abstract*—Directed graphs such as neural networks can be described by arrow terms linking a finite set of incoming nodes to some response nodes. Scott and Engeler [1] have shown that its powerset is a model for combinatory logic. This algebra is called Graph Model of Combinatory Logic. Combinatory logic is Turing-complete; thus, the model explains both traditional programming as well as neural networks such as the brain. The graph model would yield a performant AI-tool if used as a blueprint for implementing AI. A chain of thoughts would come for free, and explainability with it. However, its performance would make such a tool impractical and useless. We propose a combined approach for adding explainability to AI. It is the strategy humans use when they try to explain their ideas. First, we use the generative power of neural networks to produce an idea or solution. Next, we create a chain of thoughts that explains such ideas to others. AI could follow the same strategy. Anything generated by an AI engine can be analyzed as a sequence of set of arrow terms that explain the line of thinking, provided the AI engine had been trained properly. Improper training, biases, and hallucinations would become detectable. Since the target is known, guided search can find suitable arrow terms in predictable time. The architecture of this proposed AGI engine consists of three distinct elements: a well-trained artificial neural network, a deduction engine for the arrow term sets, and a search engine for fact checking.**

Keywords—Chain-of-Thought (CoT); Artificial General Intelligence (AGI); Artificial Neural Networks (ANN); Combinatory Logic; Quality Function Deployment (QFD).

I. INTRODUCTION

A. Short History of AI and a its Philosophical Background

In the early 20th century, there were some shocking events taking place in mathematical logic and natural science. Gödel [2], when trying to solve some of Hilbert's 23 problems, detected that predicate logic, something with a long history dating back to the ancient Greeks, is undecidable. This insight gave birth to theoretical computer science, including the theory of computation, founded by Turing [3]. For a modern compilation, see Raatikainen [4].

Schönfinkel and Curry [5] developed *Combinatory Logic* to avoid the problems introduced when using logical quantifiers, and Church invented *Lambda Calculus* as a rival formalism [6]. Scott and Engeler developed the *Graph Model* [1], based on *Arrow Terms*, and proved that this is a model of combinatory logic. This means that you can combine sets of arrow terms to get new arrow terms, and that combinators, accelerators, and constructors can be used to create new elements of algebra.

Graphs in the form of neural networks appeared already at the origins of *Artificial Intelligence* (AI). Its first instantiation in modern times was the *Perceptron*, a network of neurons postulated by Rosenblatt [7]. It later became a directed graph [8]. Rosenblatt was also the first who postulated concepts, among perception and recognition, as constituent parts of AI [7, p. 1].

Since its origins, AI has experienced difficulties; however, today it seems to have become mainstream as far as there are many AI applications that provide value for the user. In some areas, training an AI model is much simpler and more rewarding than finding and programming an algorithm.

AI-powered visual recognition systems excel in recognizing and classifying objects, following the ideas established by Rosenblatt [7]. However, they are weak at recognizing temporal dependencies and unable to combine learnings, despite attempts to develop methods with sequential data and the ability to capture temporal patterns. AI lacks what humans use in such cases: a concept.

Logical skills such as inference and deduction provide quite a challenge, as exemplified by the ARC Price challenge, a sort of intelligence test for AI models, proposed by Chollet [9]. A *Large Language Model* (LLM) easily summarizes texts or books but it still does not understand what is written in it, in the sense that the US National Council of English Teachers calls Literacy, see [10], [11].

Regarding LLM or any other variant of *Artificial Neural Networks* (ANN), we refer to the rapidly evolving literature. As an entry point, Gerven & Bothe's classification might be a good start [12]. *Natural Neural Networks*, in analogy to ANNs, are abbreviated by NNN.

IBM defines *Explainable Artificial Intelligence* (XAI) as a set of processes and methods that allows human users to comprehend and trust the results and output created by machine learning algorithms [13]. This is a bold attempt to use statistical correlations as a basis for reasoning. From a theoretical perspective, this is unlikely to work, because of Gödel [2]; however, from an engineering prospective, it is an attempt to work around undecidability. Dallanoce compiled a list of available processes and methods for XAI [14].

Artificial General Intelligence (AGI) is a type of artificial intelligence (AI) that falls within the lower and upper limits of human cognitive capabilities across a wide range of cognitive tasks. The creation of AGI is a primary goal of AI research and companies such as OpenAI and Meta, but what exactly AGI refers to is controversial [15].

B. Research Questions

The aim of this paper is to recall prior work in logic and AI to understand how neural networks work. To do this, we investigate into the following three research questions:

RQ 1: How are neural networks and especially ANNs linked to the graph model?
RQ 2: Does a chain of thoughts relate to a sequence of arrow schemes?
RQ 3: Can arrow schemes explain AI?

The motivation for this is that we are currently experiencing the fourth AI hype in sixty years and that its acceptance in society is currently transitioning from admiration to rejection. Because the nature of AI is poorly understood not only by society but also by the AI research community. We believe that the graph model is an excellent way to understand what intelligence is, both natural and artificial. However, it is not an answer to how to construct XAI.

C. Paper Structure

We first explain combinatory logic (section II) and the motivation for building a model (section III). Then we compare ANNs with graphs and explain how arrow schemes represent what an ANN does and have an outlook on the architecture of intelligent systems (section IV).

II. COMBINATORY LOGIC

Here has been a lack of attention and consequently of publications on *Combinatory Logic.* Nevertheless, it explains quite a bit what artificial intelligence can do and what not.

A. Combinatory Logic and Axiom of Choice

Combinatorial Logic is a notation that eliminates the need for quantified variables in mathematical logic, and thus the need to explain what the meaning of existential quantifiers $\exists x \in M$ is, see Curry [5] and [16]. Eliminating quantifiers is an elegant way to avoid the *Axiom of Choice* [17] in its traditional form. Combinatory Logic can be used as a theoretical model for computation and as design for functional languages (Engeler [18]); however, the original motivation for combinatory logic was to better understand the role of quantifiers in mathematical logic.

It is based on *Combinators* which were introduced by Schönfinkel in 1920. A combinator is a higher-order function that uses only functional application, and earlier defined combinators, to define a result from its arguments.

The combination operation is denoted as $M \bullet N$ for all combinatory terms M, N. To make sure there are at least two combinatory terms, we postulate the existence of two special combinators **S** and **K**.

They are characterized by the following two properties (1) and (2):

$$\mathbf{K} \bullet P \bullet Q = P \tag{1}$$

$$\mathbf{S} \bullet P \bullet Q \bullet R = P \bullet Q \bullet (P \bullet R) \tag{2}$$

P, Q, R are terms in combinatory logic. The combinator **K** acts as projection, and **S** is a substitution operator for combinatory terms. Equations (1) and (2) act like axioms in traditional mathematical logic.

Like an assembly language for computers, or a Turing machine, the **S**-**K** terms become quite lengthy and are barely readable by humans, but they work fine as a foundation for computer science. The power of these two operators is best understood when we use them to define other, handier, and more understandable combinators.

The identity combinator for instance is defined as

$$\mathbf{I} := \mathbf{S} \bullet \mathbf{K} \bullet \mathbf{K} \tag{3}$$

Indeed, $\mathbf{I} \bullet M = \mathbf{S} \bullet \mathbf{K} \bullet \mathbf{K} \bullet M = \mathbf{K} \bullet M \bullet (\mathbf{K} \bullet M) = M$. Association is to the left. Moreover, **S** and **K** are sufficient to build a Turing-machine. Thus, combinatory logic is Turing-complete. For a modern proof, consult Barendregt [19, pp. 17-22].

B. Functionality by the Lambda Combinator

Curry's *Lambda Calculus* [20] is a formal language that can be understood as a prototype programming language. The **S**-**K** terms implement the lambda calculus by recursively defining the *Lambda Combinator* $\mathbf{L_x}$ for a variable x as follows:

$$\begin{gathered} \mathbf{L_x} \bullet x = \mathbf{I} \\ \mathbf{L_x} \bullet Y = \mathbf{K} \bullet Y \text{ if } Y \text{ different from } \mathbf{x} \\ \mathbf{L_x} \bullet M \bullet N = \mathbf{S} \bullet \mathbf{L_x} \bullet M \bullet \mathbf{L_x} \bullet N \end{gathered} \tag{4}$$

The definition holds for any term **x** of combinatory logic. Usually, on writes suggestively $\lambda x.M$ instead of $\mathbf{L_x} \bullet M$, for any combinatory term M. *Lambda Terms* $\lambda x.M$ offer the possibility of programmatic parametrization. Note that $\lambda x.M$ is a combinatory term, as proofed by (4), and that this introduces a kind of variable in combinatory logic with a precisely defined binding behavior.

The Lambda combinator allows writing programs in combinatory logic using a higher-level language. When a Lambda term gets compiled, the resulting combinatory term is like machine code for traditional programming languages.

C. The Fixpoint Combinator

Given any combinatory term Z, the *Fixpoint Combinator* **Y** generates a combinatory term $\mathbf{Y} \bullet Z$, called *Fixpoint of* Z**,** that fulfills $\mathbf{Y} \bullet Z = Z \bullet (\mathbf{Y} \bullet Z)$. This means that Z can be applied to its fixpoint as many times as wanted and still yields back the same combinatory term.

In linear algebra, such fixpoint combinators yield an eigenvector solution $\mathbf{Y} \bullet Z$ to some problem Z.

According to Barendregt in his textbook about Lambda calculus [19, p. 12], the fixpoint combinator can be written as

$$\mathbf{Y} := \lambda f.\left(\lambda x.f \bullet (x \bullet x)\right) \bullet \left(\lambda x.f \bullet (x \bullet x)\right) \quad (5)$$

Translating (5) into an **S**–**K** term demonstrates how combinatory logic works, see the authors' paper from 2022 [21].

When translated into arrow terms, the fixpoint combinator contains loops. Fixpoint operations are related to infinite loops, programming constructs that never end in some normal form. Applying **Y**, or any equivalent fixpoint combinator to a combinatory term *Z*, usually does not terminate. An infinite loop can occur, and must sometimes occur, otherwise Turing would be wrong and all finite state machines would reach a finishing state [3].

III. The Graph Model of Combinatory Logic

The graph model is a versatile model for knowledge in all its instantiations. It is highly recursive and Turing-complete, which means, it also describes conventional algorithmic programming.

A. A Logic Needs a Model

A *Model* for a logical structure is a set-theoretic construction that has the properties postulated for the logic and can be proved to be non-empty. Then it means that logic makes sense as far as it describes some structure that really exists. If a non-empty model exists, then the logic exists in the sense that it can be used to prove something about the model.

Let $\mathcal{L}$ be a non-empty set. Engeler [1] defined a *Graph* as the set of ordered pairs:

$$\langle \{a_1, a_2, \dots, a_m\}, b \rangle \quad (6)$$

with $a_1, a_2, \dots, a_m, b \in \mathcal{L}$. We write $\{a_1, \dots, a_m\} \to b$ for the ordered pair to make notation mnemonic, i.e., referring to directed graphs, and call them *Arrow Terms*. These terms describe the constituent elements of directed graphs with multiple origins and a single node. We extend the definition of arrow terms to include all formal set-theoretic objects recursively defined as follows:

$$\begin{gathered}\text{Every element of } \mathcal{L} \text{ is an arrow term.} \\ \text{Let } a_1, \dots, a_m, b \text{ be arrow terms.} \\ \text{Then } \{a_1, \dots, a_m\} \to b \text{ is also an arrow term.}\end{gathered} \quad (7)$$

The left-hand side of an arrow term is a finite set of arrow terms, and the right-hand side is a single arrow term. This definition is recursive. Elements of $\mathcal{L}$ are also arrow terms. The arrow, where present, should suggest the ordering in a graph, not logical imply.

B. Einstein-Notation for Arrow Terms

To avoid the many set-theoretical parenthesis, the following notation, called *Arrow Schemes*, is applied, in analogy to the Einstein notation [22, p. 6]:

- a_i for a finite set of arrow terms, i denoting some Choice Function selecting finitely many specific terms out of a set of arrow terms a.
- a_1 for a singleton set of arrow terms; i.e., $a_1 = \{a\}$ where a is an arrow term.
- $\emptyset$ for the empty set, such as in the arrow term $\emptyset \to a$.
- $a_i + b_j$ for the union of two observation sets a_i ,b_j.

(8)

The application rule for M and N now reads:

$$M \bullet N = (a_i \to b) \bullet N = \{b | \exists a_i \to b \in M;\ a_i \subset N\} \quad (9)$$

$(a_i \to b) \subset M$ is the subset of level 1 arrow terms in M. With these conventions, $(a_i \to b)_j$ denotes a *Concept*, i.e., a non-empty finite set of arrow terms with level 1 or higher, together with two choice functions i, j. Each set element has at least one arrow.

The choice function i chooses specific observations a_i out of a (larger) set of observations a. This is what Zhong describes as *Grounding* when linking observations to real-world objects [23]. In AI, grounding is crucial for linking AI engines to the real world. If a denotes knowledge, i.e., an infinite set of arrow terms of any level, a_i can become part of a concept consisting of specific arrow terms referring to some specific object, specified by the choice function i. Choice functions therefore have the power of focusing knowledge on specific objects in specific areas. That makes choice functions interesting for intelligent systems and AI.

There is a conjunction of choice functions, thus $a_{i,j}$ denotes the union of a finite number of grounded arrow schemes:

$$a_{i,j} = a_{i,1} \cup a_{i,2} \cup \dots \cup a_{i,m} = \bigcup_{k=1}^{m} a_{i,k} \quad (10)$$

There is also cascading of choice functions. Assume $N = \left(a_j \to b\right)_k$, then:

$$\begin{gathered}M = \left(\left(\left(a_j \to b\right)_k \to b_i\right)_l \to c\right) \text{ and} \\ M \bullet N = \left(b_{i_l} \to c\right)\end{gathered} \quad (11)$$

The choice function might be used for grounding an arrow scheme to observations.

An arrow scheme without outer indices represents a potentially infinite set of arrow terms. Thus, writing a, we mean knowledge about an observed object. Adding an index, a_j, indicates such a grounded object together with a choice function j that chooses finitely many specific observations or knowledge.

While on the first glimpse, the Einstein notation seems just another way of denoting arrow terms, for representing such data in computers it means that the simple enumeration of finite data sets is replaced by an intelligent choice function providing grounding that must be computed and can be either programmed or guessed by an intelligent system.

For practical applications, the choice function is an important part of deep learning. It means learning by generalization. The more choices you get on the left-hand side, the more knowledge you acquire. The ARC price competition for instance is easily solvable if we can generalize our choice functions good enough, to draw conclusions from the samples into general rules. However, generalization is not easily available with current AI technology. *Controlling Combinators*, see section IV.B, are a workaround.

C. The Graph Model of Combinatory Logic

The algebra of observations represented as arrow terms is a combinatory algebra and thus a model of combinatory logic. The following definitions demonstrate how the graph model implements Curry's combinators **S** and **K** fulfilling equations (1) and (2), following [5].

- $\mathbf{I} = a_1 \to a$ is the Identification, i.e., $(a_1 \to a) \bullet b = b$
- $\mathbf{K} = a_1 \to \emptyset \to a$ selects the 1st argument:
  $\mathbf{K} \bullet b \bullet c = (b_1 \to \emptyset \to b) \bullet b \bullet c = (\emptyset \to b) \bullet c = b$
- $\mathbf{KI} = \emptyset \to a_1 \to a$ selects the 2nd argument:
  $\mathbf{KI} \bullet b \bullet c = (\emptyset \to c_1 \to c) \bullet b \bullet c = (c_1 \to c) \bullet c = c$
- $\mathbf{S} = \left(a_i \to \left(b_j \to c\right)\right)_1 \to (d_k \to b)_i \to \left(a_i + b_{j,i} \to c\right)$

(12)

Therefore, the algebra of observations is a model of combinatory logic. The interested reader can find complete proofs in Engeler [1, p. 389].

The *Lambda Theorem* from Barendregt [20] says that with **S** and **K,** an abstraction operator can be constructed that adds algorithmic skills to knowledge represented as arrow schemes, following equation (4).

As the name "graph model" suggests, arrow terms are an algebraic way of describing neural networks. Thus, something that nature uses to acquire and work with knowledge.

Figure 1 illustrates the effect of the combination according to equation (9). It becomes apparent that the graph model describes graphs indeed, with loops. Repeatedly applying equation (9) leads to what we perceive as the "response of a neural network". Combination of knowledge and combinators thus play a significant role in AI.

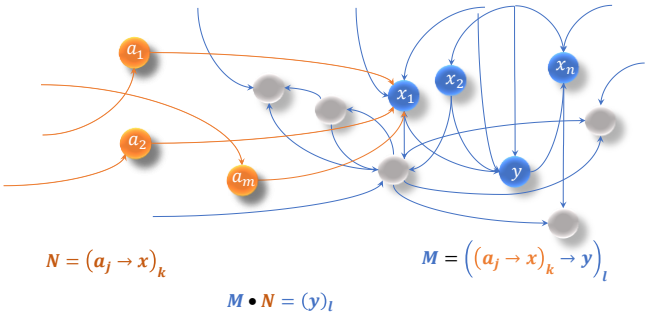


Figure 1: Neural Network become a Combinatorial Algebra

However, Figure 1 is not only a picture of an abstract graph. It can also be understood as a part of an ANN – or of an NNN. Engeler associated neuroscience with the graph model in 2019, by explaining how a brain works [24]. He used the graph model of combinatory logic as an algebraic representation of NNN.

IV. TOWARDS INTELLIGENT SYSTEMS

Barceló et. al. has shown in 2019 that modern neural network architectures are Turing-complete [25]. This is also a property of the graph model but not of every ANN. We propose an architecture for intelligent systems that incorporates conventional algorithmic programming.

A. How Arrow Schemes describe ANNs

While it is obvious how an NNN is represented by arrow schemes, this is not equally clear for ANNs. The reason is that directed graphs contain loops while looping in ANNs is very restricted. There exist certain architectures for ANNs that allow for loops, within narrow limits, a typical *Multi-Layered Perceptron* (MLP) as used for LLMs does not [12].

Consequently, an ANN has only a limited ability to emulate an NNN.

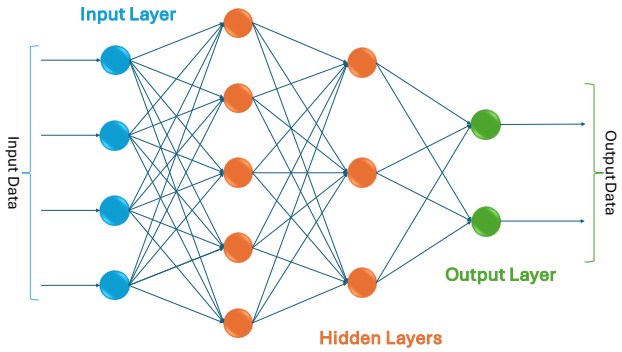


Figure 2: Multi-Layered Perceptron as an ANN

In principle, every arrow scheme $a_i \to b$ describes one node in a directed but not loop-free graph. Some arrow schemes describe algorithmic concepts such as in equation (12) or as explained in equation (5). Other arrow schemes simply connect observations a_i to some response b. General knowledge has many facets.

It would be wonderful if we had the ability to look at an LLM and identify arrow schemes for each node. This would add full explainability to AI, but unfortunately, this is not the case. Theoretically, this is because neither combinatory terms nor arrow schemes have normal forms. Very often there is a wide variety of solutions that are equivalent but widely different in effectiveness.

This makes explainability of AI difficult. The lack of normal forms blocks all attempts to find the one sequence of arrow schemes that explains what AI is doing. AI engineers have no other choice than trying to train their ANNs such that the response meets expectations but without exactly knowing what happens. It is comforting, however, that they share the same sad fate with neuroscientists. It is astonishing how long-forgotten theoretical results such as the lack of a normal form in combinatory logic yield economically highly relevant results, nowadays, in the evolving AI ecosystem. Consult Lachowski [25] for a survey of the performance challenges that occur around combinatory logic.

However, there is a famous saying that nothing is too difficult for the engineer ("Inventor of Anything"). Recent findings suggest that AI is capable of recognizing chains of thought that lead to the observation of a specific response [26]. This complements earlier findings that describe CoT as a prompting technique [27]. Thus, there exist AI architectures that allow to identify at least some arrow schemes that describe what AI does. It is not necessarily the full truth, as is

also not the case when humans explain their thoughts to colleagues. But it should be enough to persuade them.

Having a complete sequence of arrow schemes describing some ANN would lead to explainable AI that even is able to get certified for safety-critical applications. However, there is a problem with hidden layers. While the *Quality Function Deployment* (QFD) method uses identifiable topics for each layer [28], an ANN has none; they are hidden indeed. Thus, much of the intermediate reasoning also remains hidden. RQ 2 remains at least partially unanswered. If the input data and response only can be captured by arrow schemes, the intermediate steps must be guessed based on domain knowledge, but it is not known what exactly the AI engine actually did consider. AI might change behavior and create hazardous changes to the hidden layers; low-rank adaptation (LoRA) of large language models is an attempt to limit such change [29]. In QFD, on the contrary, intermediate stages are identifiable based on their topic; for example, when deploying customer needs, we first go to user stories and then to testable features.

Another approach to better explainable AI is already well established: *Retrieval-Augmented Generation* (RAG) might avoid hallucinations for LLMs [30] by referencing knowledge databases and including them into the generation of responses. RAG impacts the architecture of intelligent systems by connecting neural networks to knowledge databases [31]. RAG corresponds to grounding arrow schemes using the choice function; RAG is indispensable for explainable AI.

This is the motivation for looking at AI architectures. In some way, it must be complemented by functionality that controls the behavior of AI. Only with such control an AI-engine can perform safety-critical tasks. When certifying an AI-engine for safety, it is no longer necessary to convert all nodes of an ANN into arrow schemes, but we can focus on the overall result. If an AI fails on such tasks, we do not have a white-box trace of all nodes, or arrow schemes, that have contributed to this failure, but we are at least as good as with traditional safety-preserving methods and techniques.

B. *The Architecture of Intelligent Systems*

Intelligent systems using AI are based upon *Controlling Combinators*. Controlling combinators are derived from the idea behind fixpoint combinators, see equation (5). A *Controlling Operator* $\mathbf{C}$ acts on a controlled object X by its application $\mathbf{C} \bullet X$. Control means that knowledge represented by arrow schemes in X is completely known and described.

Accomplishing control can be formulated by (13):

$$\mathbf{C} \bullet X = X \tag{13}$$

The equation (13) is a theoretical statement, referring to a potentially infinite loop. For solving practical problems, X must be approximated by finite subterms.

Thus, the control problem is solved by a *Control Sequence* $X_0 \subseteq X_1 \subseteq X_2 \subseteq \cdots$, a series of finite subterms and the controlling operator $\mathbf{C}$, starting with an initial X_0 and determined by (14):

$$X_{i+1} = \mathbf{C} \bullet X_i, i \in \mathbb{N} \tag{14}$$

This is called *Focusing*. The details can be found in Engeler [24, p. 299]. The controlling operator $\mathbf{C}$ gathers all faculties that may help in the solution. The inclusion operator in equation (14) is explained by the graph model. The control problem is a repeated process involving substitution, like finding the fixpoint of a combinator, and thus increasing the number of arrow schemes, and especially of choice functions, in the resulting focusing process.

Controlling combinators both collect and use empirical data for continuous training. Such an intelligent system incorporates the necessary functional processes for fine-tuning based on feedback received.

For more details, we refer to the authors' paper about solving the control problem [32].

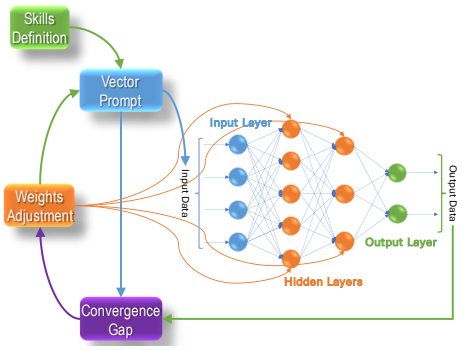


Figure 3: Self-learning Intelligent Systems based on an ANN

The program scheme we use in Figure 3 for the controlling combinator depends on the *Convergence Gap*; the measurable variation between actual behavior and expectations and requirements regarding an AI-enabled intelligent system. Both come as (large) vectors and thus the Euclidean distance is easily computable. Expectations and correct answers might also come from an external knowledge database, allowing the intelligent system to learn autonomously.

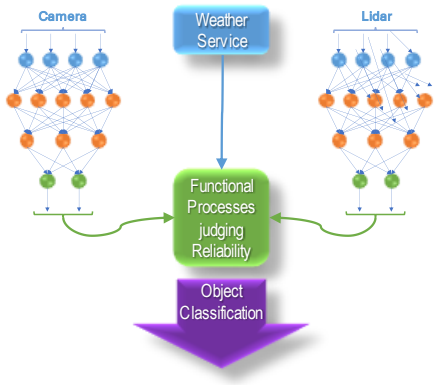


Figure 4: An Intelligent System that selects the most reliable AI response.

The architecture for RAG now extends. Instead of embedding the reference into response generation [31], and hoping it works, we set up functional processes for comparing LLM results with evidence from the knowledge database and calculate the convergence gap.

The convergence gap of such a system fully explains its behavior. Under well-defined conditions, such a system can be certified, even for safety critical tasks.

It is also possible to add more than one AI engine to an intelligent system, compare results and go forward with the most reliable one. Insufficient training, biases, and hallucinations therefore would become detectable.

Figure 4 shows an example of an intelligent system design that relies on two separate visual recognition engines analyzing the same scenario, one through a camera and the other through a Lidar. Such an architecture requires that the reliability of each artificial intelligence engine be known, under certain conditions, such as weather. In this way, the intelligent system can explain why it selected one or the other response.

Obviously, if both AI-engines produce an identical response, this increases overall reliability of the response of the intelligent system quite a bit.

The graph model delivers the metrics for defining controlling combinators by inclusion, and it also allows to combine knowledge and thus reliability correctly, by equation (9). This is discussed in another paper of the authors [33]. This remark should also explain why we do not use the term "Loss Function" that originates from *Signal Theory* and originally described the loss of fidelity in analog sound transmission. Since the discovery of the *Fast-Fourier Transform* (FFT) [34], one understands that A/D-convergence is not a loss, but an acquisition of enough knowledge to reach some threshold. Deep learning uses the same principles.

V. Conclusions and Future Work

We therefore have shown that

RQ 1: ANNs can be represented in the graph model of combinatory logic;
RQ 2: CoT do not exactly relate to a sequence of arrow schemes, as they do not cover hidden layers in ANNs;
RQ 3: Arrow schemes do not explain AI but explain how AI can be controlled.

The graph model of combinatorial logic does not provide an alternative for implementing AI, but it is an excellent guide and theoretical foundation for what can be done with AI, for explaining AI, but also for learning where AI meets its limits.

The current step forward is collecting several designs of intelligent systems, finding methods for measuring reliability and defining suitable convergence gaps. This work in progress of the authors will be shared with interested parties [35]; the authors have no institution or sponsor to help with this.

It remains the idea that AI could be explained by searching for arrow schemes that provide the same responses. Since combinatory logic does not have normal forms, this seems feasible. It could be used as a validation process for AI. However, for now, this is a future research project.

Acknowledgment

The authors would like to thank Lab42 in Davos for asking excellent questions and promoting the ARC Challenge [9], now ARC Prize [36], and to the anonymous IARIA reviewers who helped to improve this paper quite a bit. Special thanks to Erwin Engeler for his suggestions and availability for valuable discussions with his former student.

References

[1] E. Engeler, "Algebras and Combinators," *Algebra Universalis*, vol. 13, pp. 389-392, 1981.

[2] K. Gödel, "Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I," *Monatshefte für Mathematik und Physik*, vol. 38, no. 1, p. 173–198, 1931.

[3] A. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 42, no. 2, pp. 230-265, 1937.

[4] P. Raatikainen, "Gödel's Incompleteness Theorems," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., 2020.

[5] H. Curry and R. Feys, Combinatory Logic, Vol. I, Amsterdam: North-Holland, 1958.

[6] A. Church, "The Calculi Of Lambda Conversion," *Annals Of Mathematical Studies 6*, 1941.

[7] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton (Project PARA)," Cornell Aeronautical Laboratory, Inc., Buffalo, 1957.

[8] M. Minsky and S. Papert, Perceptrons: An Introduction to Computational Geometry, 2nd edition with corrections ed., Cambridge, MA: The MIT Press, 1972.

[9] F. Chollet, "On the Measure of Intelligence," arXiv:1911.01547 [cs.AI], Cornell University, Ithaca, NY, 2019.

[10] K. Wijekumar, B. J. Meyer and P. Lei, "High-fidelity implementation of web-based intelligent tutoring system improves fourth and fifth graders content area reading comprehension," *Computers & Education*, vol. 68, pp. 366-379, 2013.

[11] A. Peterson, "Literacy is More than Just Reading and Writing," National Council of Teachers of English, 23 March 2020. [Online]. Available: https://ncte.org/blog/2020/03/literacy-just-reading-writing/. [Accessed 4 July 2024].

[12] M. v. Gerven and S. Bohte, "Artificial Neural Networks as Models of Neural Information Processing," in *Frontiers in Computational Neuroscience*, Lausanne, 2017.

[13] IBM TechXchange Community, "What is explainable AI?," IBM Reserach, [Online]. Available: https://www.ibm.com/topics/explainable-ai. [Accessed 23 10 2024].

[14] F. Dallanoce, "Explainable AI: A Comprehensive Review of the Main Methods," Medium.com, San Francisco, California, 2022.

[15] M. R. Morris, J. Sohl-Dickstein, N. Fiedel, T. Warkentin and A. Dafoe, "Levels of AGI for Operationalizing Progress on the Path to AGI," arXiv:2311.02462v4 [cs.AI], Cornell University, 2024.

[16] H. Curry, J. Hindley and J. Seldin, Combinatory Logic, Vol. II, Amsterdam: North-Holland, 1972.

[17] T. M. Fehlmann and E. Kranich, "Intuitionism and Computer Science – Why Computer Scientists do not Like the Axiom of Choice," *Athens Journal of Sciences*, vol. 7, no. 3, pp. 143-158, 2020.

[18] T. M. Fehlmann and E. Kranich, "A General Model for Representing Knowledge - Intelligent Systems Using Concepts," *Athens Journal of Sciences*, vol. 11, pp. 1-18, 2024.

[19] H. Barendregt and E. Barendsen, Introduction to Lambda Calculus, Nijmegen: University Nijmegen, 2000.

[20] H. P. Barendregt, "The Type-Free Lambda-Calculus," in *Handbook of Math. Logic*, vol. 90, J. Barwise, Ed., Amsterdam, North Holland, 1977, pp. 1091 -1132.

[21] T. M. Fehlmann and E. Kranich, "The Fixpoint Combinator in Combinatory Logic - A Step towards Autonomous Real-time Testing

of Software?," *Athens Journal of Sciences*, vol. 9, no. 1, pp. 47-64, 2022.

[22] T. M. Fehlmann, Autonomous Real-time Testing – Testing Artificial Intelligence and Other Complex Systems, Berlin, Germany: Logos Press, 2020.

[23] V. Zhong, J. Mu, L. Zettlemoyer, E. Grefenstette and T. Rocktäschel, "Improving Policy Learning via Language Dynamics Distillation," arXiv:2210.00066v1, Cornell University, 2022.

[24] E. Engeler, "Neural algebra on "how does the brain think?"," *Theoretical Computer Science*, vol. 777, pp. 296-307, 2019.

[25] Ł. Lachowski, "On the Complexity of the Standard Translation of Lambda Calculus into Combinatory Logic," *Reports on Mathematical Logic*, vol. 53, pp. 19-42, 2018.

[26] Z. Jin and W. Lu, "Self-Harmonized Chain of Thought," arXiv:2409.04057v1 [cs.CL], Cornell University, 2024.

[27] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le and D. Zhou, "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," arXiv:2201.11903v6 [cs.CL], Cornell University, 2023.

[28] T. M. Fehlmann and E. Kranich, "How to Explain Artificial Intelligence to Humans - Learning from Quality Function Deployment," in *Systems, Software and Services Process Improvement - CCIS 2179*, vol. Part 1, Murat Yilmaz, Paul Clarke, Andreas Riel, Richard Messnarz, Christian Greiner and Thomas Peisl, Eds., Munich, Springer Communications in Computer and Information Science 2179, EuroSPI 2024, pp. 48-63.

[29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685v2 [cs.CL], Cornell University, Ithaca, NY, 2021.

[30] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang and H. Wang, "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997v5 [cs.CL] , Cornell University, 2024.

[31] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," arXiv:2005.11401v4 [cs.CL], Cornell University, 2021.

[32] T. M. Fehlmann and E. Kranich, "Making Artificial Intelligence Intelligent - Solving the Control Problem for Artificial Neural Networks by Empirical Methods," in *Human Systems Engineering and Design (IHSED2024)*, Split, Croatia, 2024.

[33] T. M. Fehlmann and E. Kranich, "Measuring Knowledge - An Attempt to Define a Measurement Principle for Learning Machines," *ATINER Journal of Science*, 2024.

[34] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, pp. 297-301, 17 August 1964.

[35] T. M. Fehlmann, "Intelligent Systems," Euro Project Office AG, 9 May 2024. [Online]. Available: https://web.tresorit.com/l/AXX78#FaBkGqfY2cF_JsVmX70_ng.

[36] Lab 42, "ARC Price," Mike Knoop, Davos{San Francisco, 2024.