Acta Universitatis Sapientiae

Informatica

Volume 16, Number 2, 2024

Sapientia Hungarian University of Transylvania
Scientia Publishing House

Journal Metrics (2022)

Impact Factor: **0.3**
Five Year Impact Factor: **0.8**
JCI: **0.09**
MEiN: **20**
Google Scholar h5-index: **21**
Google Scholar h5-median: **28**

Acta Universitatis Sapientiae Informatica
is covered by the following services:

ACM Digital Library
Baidu Scholar
Cabell's Journalytics
CNKI Scholar
CNPIEC – cnpLINKer Dimensions
DOAJ
EBSCO
Engineering Village
ExLibris
Google Scholar
Inspec
Japan Science and Technology Agency
J-Gate
JournalTOCs
KESLI-NDSL
MyScienceWork
Naver Academic
Naviga (Softweco)
QOAM
ReadCube
SCILIT
Semantic Scholar
Sherpa/RoMEO
TDNet
Ulrich's Periodicals Directory
WanFang Data
Web of Science – ESCII
WorldCat (OCLC)
zbMATH Open
X-MOL

Contents

M. Komáromi et al.
Optimising the Force-Directed Layout Generation 139

S. Melchane et al.
Artificial Intelligence for Infectious Disease Prediction and Prevention: A Comprehensive Review 160

Z. N. Milivojević et al.
Optimization of the Seventh-order Polynomial Interpolation 1P Kernel in the Time Domain 198

M. Antal, N. Beder
Eysenck Personality Questionnaire: A Comparative Study of Humans and Large Language Models Through Repeated Administrations 219

K. Szabados
A large-scale analysis of production effort changes in software projects 236

Z. Kátai and D. Iclanzan
The Sapientia ECN AI Baseline Index: Benchmarking Large Language Models Against Student Performance in Competitive Programming 255

Y. Elmir et al.
Intelligent Video Recording Optimization using Activity Detection for Surveillance Systems 286

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 139–159

DOI: 10.47745/ausi-2024-0009

Optimising the Force-Directed Layout Generation

Mátyás Komáromi
ELTE, Eötvös Loránd University
Budapest, Hungary
makom789@gmail.com

István Bozó
ELTE, Eötvös Loránd University
Budapest, Hungary
bozo_i@inf.elte.hu
0000-0001-5145-9688

Melinda Tóth
ELTE, Eötvös Loránd University
Budapest, Hungary
toth_m@inf.elte.hu
0000-0001-6300-7945

Abstract. A graph visualisation tool can be invaluable in code comprehension. It is a well-known and researched field of graphical informatics. Several good algorithms were developed, but most of the graph drawing tools mainly focus on the generation of static drawing. In this paper, we present an approach to force-directed layout generation that is orders of magnitudes faster than the trivial implementation. This technique is based on the Runge-Kutta methods and is efficient enough to visualise the user-requested parts (views) quickly for relatively large Semantic Program Graphs of Erlang projects in soft real-time. Such a graph might assist code comprehension in the RefactorErl framework even better.

Key words and phrases: RefactorErl, Parallel computation, Graph drawing, Erlang, GPU programming.

1 Introduction

Tool-supported software development is an accepted and desirable part of the software development lifecycle. Several static source code analyser tools exist and aim to help code comprehension by presenting the analysed data in various ways. Taking the size of the presented data into account, a focused graph representation of the required data is one of the most useful.

The tool RefactorErl [1] is a static source code analyser and transformer tool for Erlang. It aims to present source code dependencies and help in code comprehension tasks with different graphs. However, the size of these graphs represented in a static format for industrial-scale software goes beyond the limits a human can comprehend. To overcome this, we needed to define new dynamic graph views for the users of RefactorErl.

Graph visualisation is a well-known and researched field of graphical informatics. Several good algorithms were developed and reviewed by our days [2]. However, most of the graph drawing tools mainly focus on the generation of static drawing. Our goal is to create a tool to support dynamic views and provide an efficient layout generation.

In this paper, we present an approach to a force-directed layout [3] generation based on the Runge-Kutta methods. This method is efficient enough to quickly visualise the user-requested parts (views) of relatively large Semantic Program Graphs of Erlang projects in soft real-time. We studied different parallelisation of the method on GPUs to achieve a better performance.

The rest of the paper is structured as follows. In Section 2 we introduce the RefactorErl tool and our first dynamic graph visualisation prototype, Gview. Sections 3 and 4 present our motivation to use force-directed layout generation, and demonstrate our solutions for efficient parallelisation and improvements of the algorithms. In Section 5 we evaluate our results. Sections 6 and 7 describe related works and conclude the paper.

2 Background

RefactorErl [1] is an open-source static source code analyser and refactoring tool for Erlang, developed by the Department of Programming Languages and Compilers at the Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary. The phrase ”refactoring” [4] means a semantic preserving source code transformation, so a structural change is performed in the program while it does not alter its original behaviour. Erlang is a dynamically typed functional programming language, thus to gather all of the necessary information for a behaviour-preserving transformation is not straightforward. The RefactorErl comes with an easily extensible lexer and parser. The framework of RefactorErl applies several static semantic analyses on the source code and represents the source code and the gathered information in a Semantic Program Graph [5].

The main focus of RefactorErl is to support the daily code comprehension tasks of Erlang developers. Among the features of RefactorErl is included a user-level Se-

mantic Query Language, that can assist Erlang developers in everyday tasks such as program comprehension, debugging, finding relationships among program components, etc. The queries are mapped to traversals in the Semantic Program graph. For industrial-scale software, the size of this graph can become incredibly huge. Therefore, the processing of a query may take from a few seconds up to several hours depending on its complexity.

2.1 The Semantic Program Graph

RefactorErl keeps the information extracted through static semantic analyses in a special data structure called the Semantic Program Graph (SPG) [5]. This graph represents the lexical, syntactic and semantic structure of the analysed program. Typically the lexical, syntactic and semantic elements of a program map to one node in the SPG, while the connection between these elements maps to tagged edges between these nodes. The SPG is a rooted graph. The root is a special node which does not represent a program unit. The role of this root node is to be the common ancestor of nodes not having one naturally. Care must be taken when traversing the SPG as it is not a tree and may contain directed loops.

Although it is not a tree, the SPG exposes hierarchical properties as well. As an example taking the subgraph of SPG representing the syntactic data of the program, we find the nodes of files right below the root node. Below the files, we find function forms. Going one level deeper we have clauses that build up the previously mentioned forms. Finally, on one level deeper there are the symbols of clause names, parameters of clauses and syntactic trees of expressions in the rows of bodies of clauses.

2.2 Code comprehension

Nowadays code comprehension or program comprehension is an increasing duty of IDEs and other tools for software development and maintenance. Be the user of such a tool a newcomer to the task, an employee transferring from one project to another, a project manager or a team picking up a new piece of technology, understanding the code base and getting familiar with it leads to a much higher efficiency on both building functional and stable software and keeping such systems running.

It is common knowledge that humans can process much more information visually than through text or audio in a short amount of time. RefactorErl already has methods for supporting code comprehension. Therefore, our goal is to extend these features with a tool (Gview) for dynamically visualising parts of the SPG (so-called views) in soft real-time, through which the user can explore aspects of this huge

graph and learn about the project at hand. In our previous work [6], we studied the structure of such a tool and the way the data can be transferred from RefactorErl to Gview. We also examined the different aspects of the rendering environment.

In this paper, we investigate one of the popular graph drawing methods: the force-directed layout generation method. Particularly the probe of algorithmic and computational optimisation through the usage of higher-order methods and the more efficient usage of the GPU is the target of this work.

2.3 Euler's method

As we will see in Section 3.1, the problem of generating the force-directed layout of a directed or undirected graph can be formulated as simulating a physical system over time and running this simulation until the layout is sufficiently close to a fixed point. This physical system, as shown below, is described by a set of differential equations, which ought to hold throughout the entire lifespan of the system. To ensure convergence, the simplest method that can be applied in this context is Euler's method [7].

The essence of Euler's method in this context is to start the simulation from an initial layout of the graph (random or generated by other means), and then take discrete steps. In each step the algorithm calculates the derivative of the approximated function, which can be interpreted as accumulating the acting forces on each body of the system and letting the simulation run for a given h constant amount of time and use the resulting layout as the next best setup, continuing the loop.

2.4 Room for improvements

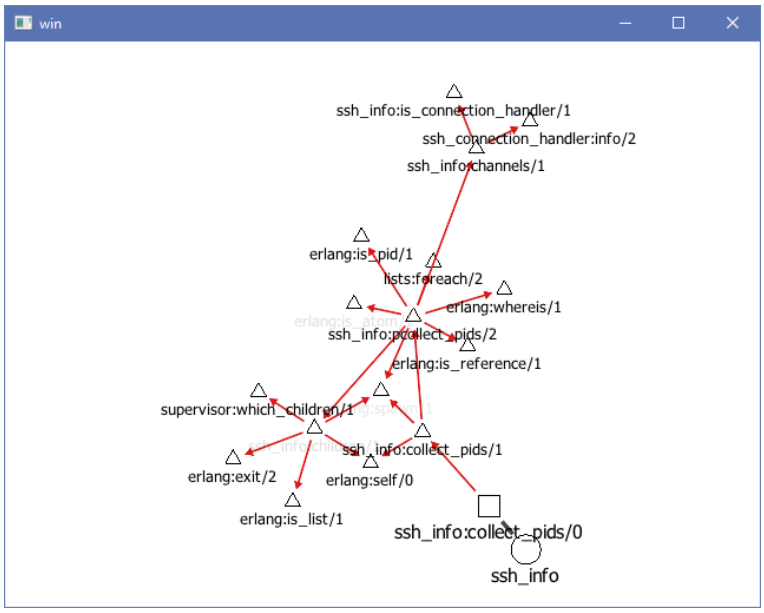
The dilemma of Euler's method is how to choose h. When choosing a too-small value, the simulation may take ages. On the other hand, too large values will lead to oscillations and the potential lack of convergence. Our previous approach was to decrease h over time to ensure convergence. However, the optimal h may not only depend on the arrangement of the graph but on the currently approximated optimal layout. To address both issues, in this paper, we inspect the adoption of a well-known generalisation of the above method: the adaptive Runge-Kutta method family [8].

3 Motivation

Our goal was to create a tool for dynamically and interactively displaying parts of the SPG in RefactorErl to aid code comprehension. To this end, we want to research

the possibilities of speeding up the above-mentioned algorithm. One of such plots by Gview can be seen in Figure 1.

Figure 1: Function call view plotted using the researched algorithm.



3.1 The force-directed layout generation

The method of force-directed layout [9] generation is a way of generating a two-dimensional layout for directed and undirected graphs alike.

The core concept of creating such a layout involves fitting a physical system on the graph, in the following way. Take a graph $G = (V, E)$ with weighted vertices V and weighted edges E, totalling n vertices, weight functions m and l! Each node of the n nodes corresponds with a body in the system with a position denoted by $p_i(t)$, a constant charge, m_i, and a velocity $v_i(t)$. As the position and velocity depend on the time passed since the start of the simulation, p_i and v_i have the type of $\mathbb{R} \to \mathbb{R}^2$. The generation of force-directed layout for graph G starts with calculating an initial, usually random, layout of the nodes, denoted by $p_i(0)$ for $i = 1..n$. After the initial layout has been set up, the algorithm follows by simulating the evolution of the

physical system using the following equations.

$$v_i(t) = \sum_{j=1, i\neq j}^{n} e(i,j,t)) \tag{1}$$

$$e(i,j,t) = \frac{d(i,j,t)}{\|d(i,j,t)\|_2} * \left(H * ln(\|d(i,j,t)\|_2^2) * l_{i,j} - \frac{G * m_i * m_j)}{\|d(i,j,t)\|_2^2} \right) \tag{2}$$

$$d(i,j,t) = p(t)_i - p(t)_j \tag{3}$$

$$v = \frac{\partial p}{\partial t} \tag{4}$$

Equation 1 means that at any given time point t the velocity of the i^{th} body equals the sum of the forces exerted by other bodies. Here we use the term force, to describe instantaneous forces, which have a direct effect on the velocity of the bodies rather than the acceleration. Such forces are characterised by Equation 2: knowing the position of the i^{th} and j^{th} body at time t, we can easily calculate the force acting on body i at time t.

Here H and G are arbitrary positive constants, used to regulate the strength of the two types of acting force. In our research having $H = 2$ and $G = 6100$ turned up the best-looking results. The most important equation is 4, which describes the analytic connection between position and velocity in the physical system. It can be used to rewrite the former equation system as a differential equation with the common vector function of positions $p = t \rightarrow (p(t)_1, p(t)_2, ..., p(t)_n)$ as the unknown, as follows.

$$\frac{\partial p(t_{cur})}{\partial t} = f(t_{cur}, p(t_{cur})) \tag{5}$$

$$p(t_0) = p(0) = p_0 \tag{6}$$

$$f(t,p) = \sum_{j=1, i\neq j}^{n} e(i,j,t) \tag{7}$$

The canonical form of the ordinary differential equation is given in Equation 5 with the definition of f in Equation 7, while the initial position requirement is defined in Equation 6.

Therefore, our goal is to approximate the vector function p for increasing values of t until we get close enough to its fixed point. For this purpose, we want to use higher-order methods, to better utilise computational power and stable convergence as t approaches ∞.

3.2 Higher order methods

The Runge–Kutta methods [10] are a family of explicit iterative methods, used in temporal discretization for the approximate solutions of ordinary differential equations. The methods include the well-known routine called the Euler Method and can be considered as the generalisation of the routine. The method has an adaptive version, which can adjust the step size of the simulation and thus keep the error below a given value, ϵ.

These methods are called a family for the reason that they depend on numerous parameters: a, b, and c. The latter two are vectors and the former one is a matrix [8]. These parameters can be arranged in a so-called Butcher tableau as can be seen in Figure 2.

Figure 2: Example of an extended explicit Butcher tableau of degree s

$$
\begin{array}{c|cccc}
0 & & & & \\
c_2 & a_{21} & & & \\
c_3 & a_{31} & a_{32} & & \\
\vdots & \vdots & & \ddots & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} \quad b_s \\
 & b_1^* & b_2^* & \cdots & b_{s-1}^* \quad b_s^*
\end{array}
$$

With the initial positions given in p_0, the method proceeds to create further approximations, $p(t_{i+1})$, from the previous one, $p(t_i)$ for $i = 1..\infty$, according to Equations 8 and 9. The difference in the result of these two equations is used to approximate the error introduced by taking a step of length h. Using this calculated error term, we can choose a new step size to decrease the error below ϵ or allow larger steps in exchange for a larger error term.

$$p_{i+1} = p_i + \sum_{j=1}^{s} b_j k_j \tag{8}$$

$$p_{i+1}^* = p_i + \sum_{j=1}^{s} b_j^* k_j \tag{9}$$

where

$$k_j = f\left(t_i + c_j * h, p_i + h * \sum_{l=1}^{j-1} a_{j,l} k_l\right) \tag{10}$$

The exact steps of choosing the next h and the very mathematics behind this algorithm is a well-known topic and has been discussed in many papers. Our goal is to apply this method to the problem at hand and to optimise it for the parallel architecture of modern GPUs.

As we can see in Equation 7, our f does not depend on the p parameter directly. The indirect dependence through d is replaced by the previous best approximation p_i for k_1 and $p_i + h * c_{j+1} * k_j$ for k_{j+1}, thus eliminating the need for the matrix a of the RK method. This property of the simulation resulted from the fact that we do not employ friction, which would depend on the velocity of the bodies but rather use instantaneous forces.

4 Methodology

Our goal is to find ways to improve the performance of the force-directed layout generation by utilising the massively parallel architecture of modern GPUs.

The final form of the equation that we are using, taking into consideration the relevant properties of the physical system, described at the end of Section 3.2, is Equation 11. In each step of the simulation, we have to calculate the k coefficients for each of the n bodies. Since the dimension of k is s and evaluating f requires $O(n)$ operations, the total cost of advancing one step comes out to be $O(sn^2)$.

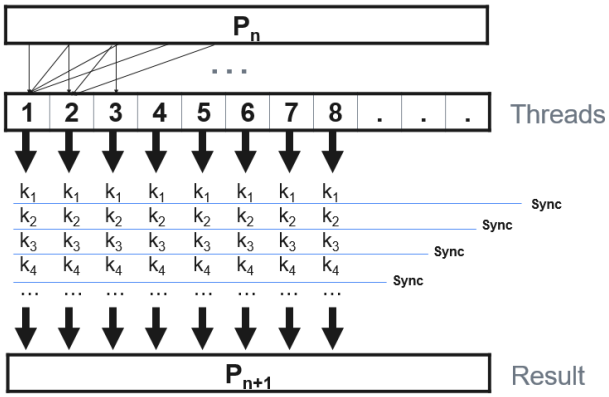
$$k_j = f\left(t_i + c_j * h\right) = \sum_{b=1, a \neq b}^{n} e(a, b, t) \tag{11}$$

4.1 Linear parallelisation

The first idea for parallelisation that one should consider is simply assigning the task of evaluating the exerted forces of all other bodies to a single body. Thread i allocates a single two-dimensional vector v, loops through all the bodies in the system and calculates the force exerted on body i through body j at the current time point t_n[1]. The calculated forces are accumulated on the fly into the local variable v of the thread and the result is stored in the k_1 array. The k_{j+1} approximations are generated in a similar manner, however p_i is replaced by $p_i + h * c_{j+1} * k_j$. A schematic representation of the linear parallelisation method can be seen in Figure 3.

[1] with exception of the i^{th} body

Figure 3: Architecture of basic parallelisation technique



The above-mentioned dependence of k_{j+1} on k_j results in the need for global synchronisation of all the employed threads to make the calculated k_j values visible to the other threads. This explicit synchronisation can only be realised on the GPU if the number of invocations is below certain driver-defined limits, which can be queried using the OpenGL command and is usually at least 1024.

In case of having a larger amount of bodies in the system than the hardware exposed limit, we need CPU synchronisation, which means splitting the calculation of each k_j vector into different dispatches.

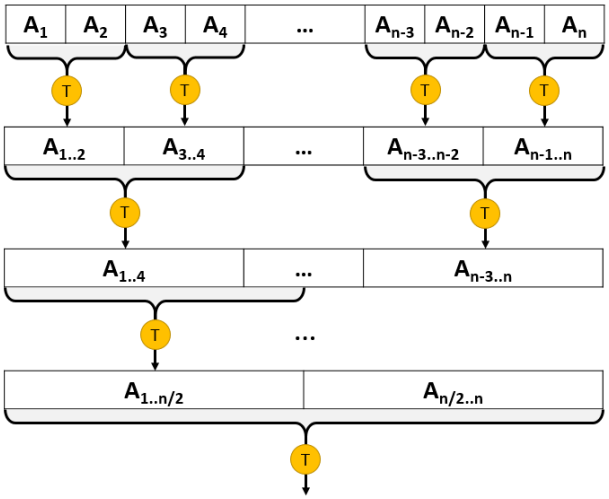
The performance bottleneck, however, comes from the fact that the amount of work each thread is doing is proportional to sn which can grow too large. The OpenGL standard guarantees [11] the ability to dispatch at least a maximum of 2^{16} workgroups, all of which may consist of a maximum of at least 1024 work items (threads). This means that reducing the number of threads dispatched from n can potentially result in a great increase in performance. This idea is further supported by the fact that GPU cores are much less powerful than CPU cores and thus employing more of them can lead to better resource utilisation, which gives reason to the optimisation in Section 4.2.

4.2 Refining work per thread

To better utilise the parallel architecture of modern GPUs for the problem at hand, we want to dispatch more than $O(n)$ threads. Thus we take Equation 11, and for each (a, b) pair, we create an invocation, totalling in $n(n-1)$ threads. After calculating each $e(a, b, t)$ value in the summation on Equation 11, however, we need to

evaluate the actual sum of these two-dimensional vectors and this is where parallel reduction comes in. Reduction (or folding) is the generalisation of summation: for a given A array of size n and a binary combining function f, the result of the folding expression is $b = f(A_1, f(A_2, f(A_3, ...)))$ where the ... goes until n. Parallelising a reduction is not a trivial task and is a well-known candidate for optimisation [12]. In our example, we have the benefit that our combination function is associative and commutative. Thus enabling us to employ a divide and conquer technique as shown in Figure 4.

Figure 4: Basic idea of divide and conquer strategy used in the parallel reduction. The circles with T represent threads of execution.

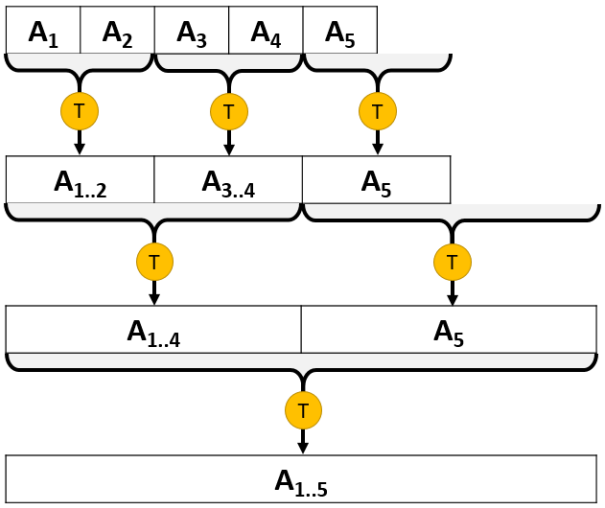


In the presented approach, we divide the reduction into levels of reduction, in each level, the size of the array that is to be combined is decreased by half. In each level, one thread only has to combine two elements of the array of the current level, which would imply that the work per thread has changed to be $O(1)$. However, by noting that the levels must come in increasing order, each one depending on the previous one, after reusing the allocated threads through levels, the total work per thread totals $O(log_2(n))$.

Figure 4 shows an optimal scenario, with n being a power of two, if n is not a power of the amount of work one thread is responsible for, then the last thread may index out of the array. To avoid this, we need to employ bound checking. An example of the size of 5 can be seen in Figure 5. This bound checking may induce a maximum of $O(n)$ extra work.

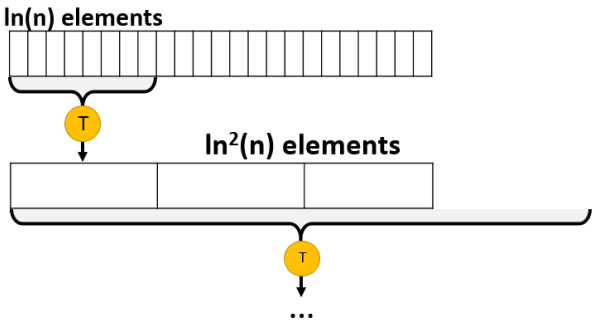
The theoretical optimum of giving one thread $O(ln(n))$ work can also be achieved.

Figure 5: Example of the maximum amount of extra work introduced by not a power of two n, for $n = 5$.



However, it requires extra mathematics behind the indexing and bound checking, and also the potential extra work growth to $O(nln(n))$. A visual representation is shown in Figure 6.

Figure 6: Optimal $ln(n)$ work per thread.



4.3 Memory usage and synchronization

When programming for a modern GPU, it is possible to arrange threads of execution (say invocations) into so-called workgroups. Threads (also referred to as work items

in this context) in a workgroup can share group local memory and can synchronise group-wise without the need for CPU intervention.

Because the levels of reduction are calculated linearly, we can store the partial results in place, which thanks to workgroups, can be synchronised on the GPU. This in-place storing of partial results can be seen in Figure 7. Thanks to this technique, we only need $O(n)$ memory and need to transfer one element to the CPU each frame.

Figure 7: In-place memory usage of the parallel reduction.

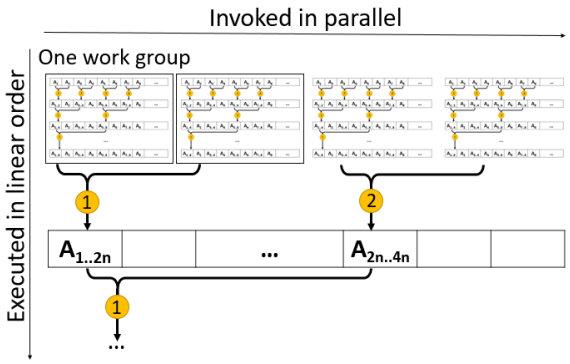
However, to be able to utilise the local synchronisation of the GPU workgroups, a workgroup size equal to the number of bodies is needed, which brings us back to our previous problem. To solve this, we investigated how we could organize the parallel reduction into batches of maximal size, and recursively apply the already presented reduction method. As Figure 8 shows, by creating partial results of maximal size, using multiple workgroups and then applying a global synchronisation, we can take advantage of the parallel architecture.

5 Results

The researched techniques were tested on a laptop with 5^{th} generation Intel Core i5 processor, 8GB of memory, using Intel HD 6000, running OpenGL 4.5. Parallel GPU programs were realised with GLSL version 430 [13].

In our tests, we incrementally generated square grids of increasing sizes from 1x1 to 11x11, with random starting positions and ran first the CPU, then the GPU and refined GPU algorithms on the same starting points, around 100 times each. As these measurements may vary according to environment properties such as the OS scheduler or extra load from updates and scheduled cleaning etc, we ignore the

Figure 8: Dividing the parallel reduction into smaller tasks that can be handled by one workgroup.



highest and lowest 5% of data and perform a normal distribution fitting on the rest. The resulting expected value of generation time is plotted against the number of nodes in Figure 9 and Figure 10.

Figure 9: The huge difference of CPU and GPU algorithm, note the logarithmic scale! Tested on grid graphs.

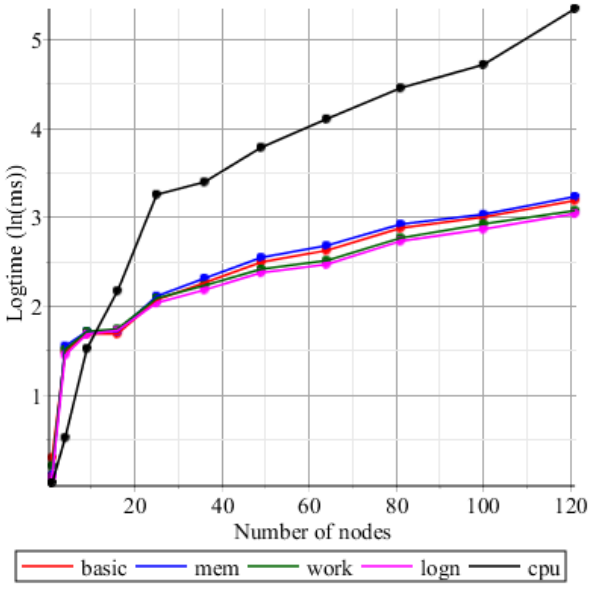
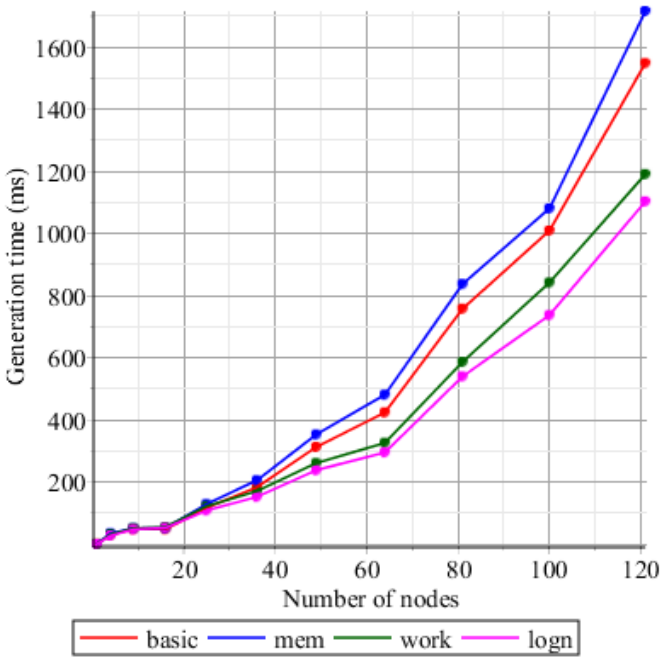


Figure 9 clearly shows the enormous improvement of the GPU parallel algorithm, even on the integrated card used in testing. We expect that with a higher-tier dedicated card, this gap is to increase further.

Figure 10: Comparison of the refined GPU algorithms, tested on different-sized grid graphs.

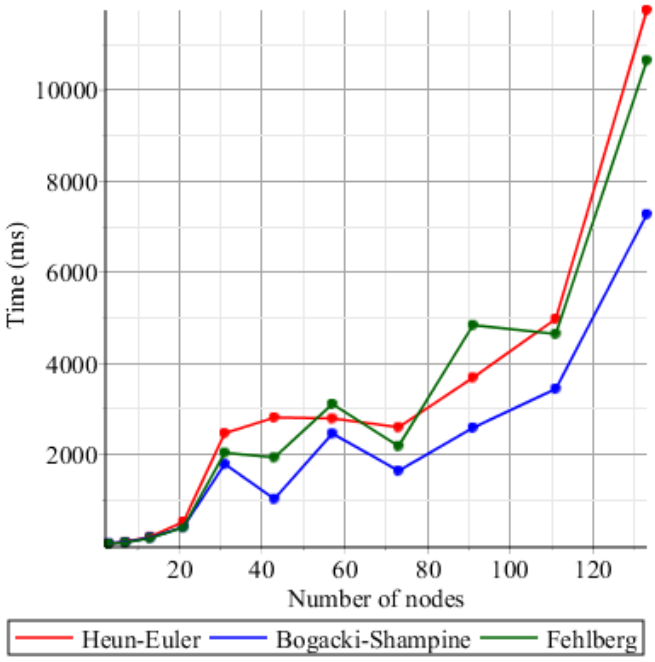


The comparison presented in Figure 10 shows how different techniques described in Section 4 improve generation time for increasing sizes of grids. The interesting thing to note is that the memory (but not workload) optimised method performs poorer than the trivial approach. This can be due to the hardware memory locality of Intel integrated GPUs, which implies that the extra copy operations introduced by memory optimisation by hand have a higher toll on performance than the improvements in the locality it creates. Thus on a dedicated card, the memory-optimised version might improve performance considerably.

We also investigated the performance of different realisations of the Runge-Kutta family: the Heun-Euler method, the less famous Bogacki-Shampine method and a high-order Fehlberg method. The tests were performed on a tree graph with nodes ranging from 3 to 133 and the same refining techniques were applied as mentioned previously. The results of this comparison can be seen in Figure 11. One can see

how the advantage of being able to take larger steps turns into a disadvantage of higher required work per step. Based on these measurements, we can conclude that the Bogacki-Shampine method is most efficient for our problem.

Figure 11: Comparison of different RK methods, tested on varying-sized tree graphs.



5.1 Usage in RefactorErl

Figures 12 and 13 demonstrate a generated graph about the Mnesia application and a function call graph generated by clicking on `verify_merge/1`.

6 Related work

In the following, we would like to compare our tool with some well-known graph visualisation tools.

Figure 12: View of all the modules in the Mnesia DBM.

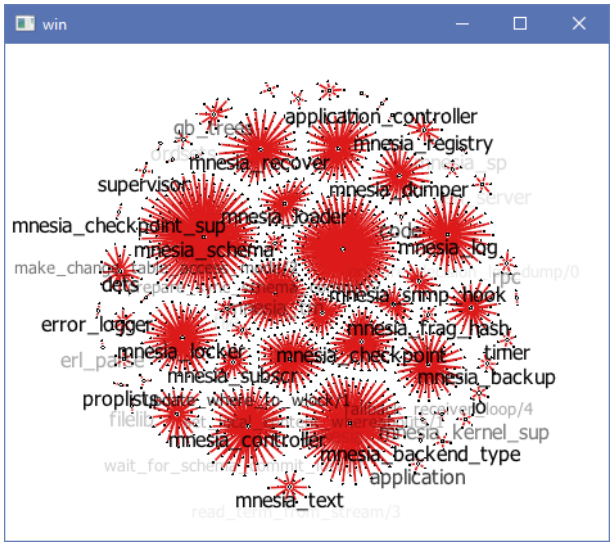
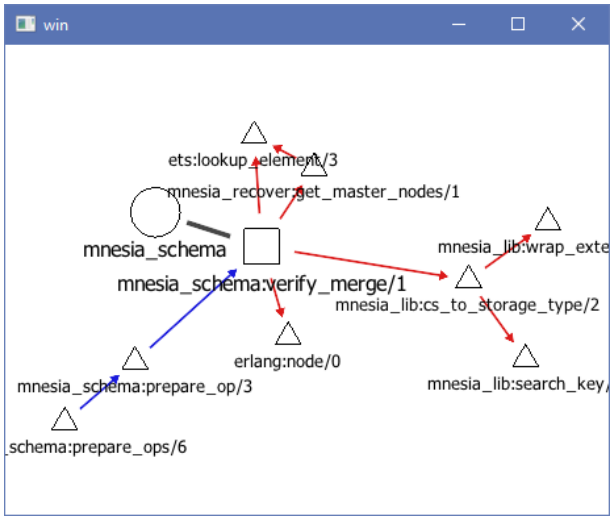


Figure 13: View generated be clicking on `verify_merge/1` in the main view of Mnesia.



6.1 Graphviz

Graphviz [14] is an open-source graph visualisation software developed by AT&T Labs Research.

The Graphviz layout programs take descriptions of graphs in a simple text language and make diagrams in useful formats, such as images and SVG for web pages; PDF or Postscript for inclusion in other documents; or display in an interactive graph browser. It supports many layout generation algorithms, such as hierarchical or the energy-minimizing stress-majoring technique. The software package has many useful features for concrete diagrams, such as options for colours, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes.

Many software use Graphviz as an intermediate tool for displaying graphs. For example, ArgoUML has an alternative UML Diagram rendering, called argoumlgraphviz, ConnectedText has a Graphviz plugin, and FreeCAD uses Graphviz to display the dependencies between objects in documents. Other programs can output in DOT [15] format and thus generate drawings with Graphviz. Doxygen also uses Graphviz to generate diagrams including class hierarchies and collaboration for source code. Graphviz targets static rendering of graphs; it optimises the drawing as much as possible and thus takes considerable time on very large graphs, and also limits the interactivity between software and user.

6.2 D3.js

D3.js [16] is a JavaScript library for manipulating documents based on data. D3.js helps bring data to life using HTML, SVG, and CSS. It emphasises on web standards giving the full capabilities of modern browsers without the need of tying to a proprietary framework, combining powerful visualisation components and a data-driven approach to DOM manipulation. This library is a modern, browser-based solution to visualisation problems with countless useful features such as pie charts, hierarchical graph drawing and force-directed layout generation.

D3.js supports force-directed layout generation using velocity Verlet integration which may require a much smaller step size than the RK methods to minimize oscillations in the solution, but the method is symplectic. Thus the two methods were meant to solve different kinds of problems, as our version of the force-directed layout generation uses logarithmic springs and instantaneous forces, our simulation need not be energy conserving or symplectic for short. The key difference between our research and D3.js is that we aim to exploit the parallel architecture of modern GPUs, while the simulations of D3.js get calculated on the CPU[2].

[2] unless some JavaScript optimisation happens

6.3 Gephi

Gephi [17] is an open-source software for graph and network analysis. It uses a 3D render engine to display large networks in real time and to speed up the exploration. Gephi advertises itself as having a flexible multitasking architecture that brings new possibilities to work with complex data sets and produce informative graphics. It has been used in several research projects in academia, journalism and elsewhere. For instance, it was used in visualising the global connectivity of New York Times content and examining Twitter network traffic during social unrest along with more traditional network analysis topics.

Development of Gephi was started in the summer of 2008, while the last stable update was in 2017. It was created in the Java programming language and although it features an OpenGL renderer, it uses immediate mode rendering, which became obsolete with OpenGL 3.1 in 2009 which means Gephi does not use GPU for layout generation. Today, with OpenGL 4.6, much faster rendering tools are available, such as instanced rendering, VBOs and compute shaders. Also, it is built on top of the NetBeans IDE, which means it cannot be integrated into another project, only added as an external tool.

6.4 GoJS

GoJS [18] is a JavaScript and TypeScript library for building interactive diagrams and graphs. GoJS claims to let the user build all kinds of diagrams and graphs, ranging from simple flowcharts and org charts to highly specific industrial diagrams, SCADA and BPMN diagrams, medical diagrams like genograms, and more. The library is meant for the implementation of interactive diagrams and visualization on modern web browsers and platforms. It allows easy construction of custom and complex diagrams of nodes, links, and groups with customizable templates and layouts. It does not depend on any JavaScript libraries or frameworks, so it should work with any web framework or with no framework at all. The library focuses on interactivity and flexibility. There are many demos available online on the webpage of the tool. It also offers rich features like drag-and-drop, copy-and-paste, in-place text editing, tool-tips, templates, data binding and models, transactional state and undo management, palettes, event handlers, commands, and an extensible tool system for real-time custom operations on the diagram. It also features many automatic layout generation algorithms, which can be extended by the user of the library.

One such automatic layout is the force-directed layout generation algorithm. They describe the method as a layout generation method that treats the graph as if it were a system of physical bodies with repulsive electrical, attracting gravitational,

and spring forces acting on them and between them. The engine uses the CPU for layout generation, thus it is not optimized for modern parallel GPUs.

7 Conclusion

The RefactorErl framework has several graphical and command-line interfaces, that support refactoring, static code analysis and code comprehension as well. The tool uses the so-called Semantic Program Graph as the intermediate representation of the source code which includes static semantic information beside the syntactic and lexical information. We have extended RefactorErl with Gview. Gview is an efficient and interactive graph visualisation tool, that uses force-directed layout generation. This algorithm was implemented using Euler's method which is only stable with potentially very small step sizes.

In this paper, we presented a better approach to simulating the evolution of the physical system that is the system of bodies and springs defined by graphs we want to plot. The above-described method is based on the well-known adaptive Runge-Kutta method family, a generalisation of Euler's method. We presented a trivial approach for parallelising the RK methods on the GPU and analysed this linear technique. We further investigated and measured optimisation opportunities for the GPU algorithm. These optimisations included different ways of refining the memory usage, changing the distribution of work among threads to reach a better configuration and the importance of different synchronisation functionalities.

In our future work, we plan to investigate other layout generation methods and optimise them for GPU.

Gview is open source and available on GitHub. The integration with RefactorErl will be released soon with the upcoming release of the tool:

`https://github.com/Frontier789/Gview`.

Funding: "Application Domain Specific Highly Reliable IT Solutions" project that has been implemented with the support provided from the National Research, Development, and Innovation Fund of Hungary, financed under the Thematic Excellence Programme no. 2020-4.1.1.-TKP2020 (National Challenges Subprogramme) funding scheme.

Data Availability: The study did not generate new data.

References

[1] I. Bozó, D. Horpácsi, Z. Horváth, *et al.*, "RefactorErl, Source Code Analysis and Refactoring in Erlang," in *Proceeding of the 12th Symposium on Programming Languages and Software Tools*, Tallin, Estonia, 2011 (⇒ 140).

[2] G. D. Battista, P. Eades, R. Tamassia, and I. G. Tollis, "Algorithms for drawing graphs: An annotated bibliography," *Computational Geometry: Theory and Applications*, vol. 4, no. 5, pp. 235–282, 1988 (⇒ 140).

[3] T. Kamada and S. Kawai, "An algorithm for drawing general undirected graphs," *Information Processing Letters*, vol. 31, no. 1, pp. 7–15, 1989 (⇒ 140).

[4] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999 (⇒ 140).

[5] Z. Horváth, L. Lövei, T. Kozsik, *et al.*, "Modeling Semantic Knowledge in Erlang for Refactoring," in *Knowledge Engineering: Principles and Techniques, Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques, KEPT 2009*, ser. Studia Universitatis Babeş-Bolyai, Series Informatica, vol. 54(2009) Sp. Issue, Cluj-Napoca, Romania, Jul. 2009, pp. 7–16 (⇒ 140, 141).

[6] Mátyás Komáromi, Melinda Tóth, István Bozó, *An Efficient Graph Visualisation Framework For RefactorErl, Paper accepted into the Special Issue of Studia Universitatis Babes-Bolyai, series Mathematica, Informaticaand Physica, MACS'18, 12th Joint Conference on Mathematics and Computer Science, Cluj-Napoca, June 14-17*, 2018 (⇒ 142).

[7] K. Atkinson, *An Introduction to Numerical Analysis*. Wiley, 1989, ISBN: 9780471500230 (⇒ 142).

[8] J. Dormand and P. Prince, "A family of embedded runge-kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980 (⇒ 142, 145).

[9] T. M. J. Fruchterman and E. M. Reingold, "Graph drawing by force-directed placement," *Software - Practice and Experience*, vol. 21, no. 11, pp. 1129–1164, 1991 (⇒ 143).

[10] C. Harper, *Introduction to mathematical physics* (Prentice-Hall physics series). Prentice-Hall, 1976, ISBN: 9780134875385 (⇒ 145).

[11] D. Shreiner, G. Sellers, J. Kessenich, and B. Licea-Kane, *OpenGL programming guide: The Official guide to learning OpenGL, version 4.3*. Addison-Wesley, 2013 (⇒ 147).

[12] M. Harris *et al.*, "Optimizing parallel reduction in cuda," (⇒ 148).

[13] R. J. Rost, B. Licea-Kane, D. Ginsburg, *et al.*, "Opengl(r) shading language," 2004 (⇒ 150).

[14] J. Ellson, E. Gansner, L. Koutsofios, S. C. North, and G. Woodhull, "Graphviz—open source graph drawing tools," in *International Symposium on Graph Drawing*, Springer, 2001, pp. 483–484 (⇒ 155).

[15] E. E. Koutsofios and S. C. North, "Drawing graphs with dot," 1991 (⇒ 155).

[16] N. Q. Zhu, *Data visualization with D3. js cookbook*. Packt Publishing Ltd, 2013 (⇒ 155).

[17] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Third international AAAI conference on weblogs and social media*, 2009 (⇒ 156).

[18] F. Shahzad, T. R. Sheltami, E. M. Shakshuki, and O. Shaikh, "A review of latest web tools and libraries for state-of-the-art visualization," *Procedia Computer Science*, vol. 98, pp. 100–106, 2016 (⇒ 156).

Received: 06.07.2024; *Revised: 09.10.2024*; *Accepted: 10.10.2024*

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 160–197

DOI: 10.47745/ausi-2024-0010

Artificial Intelligence for Infectious Disease Prediction and Prevention: A Comprehensive Review

Selestine MELCHANE
[1]Laboratoire LITAN, École supérieure en Sciences et Technologies de l'Informatique et du Numérique,
RN 75, Amizour 06300, Bejaia, Algérie
[2]LIASD research Lab., University of Paris 8, France
melchane@estin.dz
0009-0006-7902-6263

Youssef ELMIR
[1]Laboratoire LITAN, École supérieure en Sciences et Technologies de l'Informatique et du Numérique,
RN 75, Amizour 06300, Bejaia, Algérie
[3]SGRE-Lab, Bechar, Algeria
elmir@estin.dz
0000-0003-3499-507X

Farid KACIMI
[1]Laboratoire LITAN, École supérieure en Sciences et Technologies de l'Informatique et du Numérique,
RN 75, Amizour 06300, Bejaia, Algérie
[4]Laboratoire LIMED, Faculté des Sciences Exactes, Université de Bejaia, Algeria
kacimi@estin.dz
0000-0001-5457-2096

Larbi BOUBCHIR
[2]LIASD research Lab., University of Paris 8, France
larbi.boubchir@univ-paris8.fr
0000-0002-5668-6801

Abstract. Artificial Intelligence and infectious diseases prediction have recently experienced a common development and advancement. Machine learning apparition, along with deep learning emergence, extended many approaches against diseases apparition and their spread. And despite their outstanding results in predicting infectious diseases, conflicts appeared regarding the types of data used and how they can be studied, analyzed, and exploited using various emerging methods. This has led to some ongoing discussions in the field. This research aims not only to provide an overview of what has been accomplished, but also to highlight the difficulties related to the types of data used, and the learning methods applied for each research objective. It categorizes these contributions into three areas: predictions using Public Health Data to prevent the spread of a transmissible disease within a region; predictions using

Patients' Medical Data to detect whether a person is infected by a transmissible disease; and predictions using both Public and patient medical data to estimate the extent of disease spread in a population. The paper also critically assesses the potential of Artificial Intelligence and outlines its limitations in infectious disease management.

Key words and phrases: Infectious Diseases, Artificial Intelligence, Machine Learning, Prediction, Detection

1 Introduction

For many years, the world has experienced several tragic events, with the emergence of diseases being among the most devastating upheavals. Considered as life companions for several decades, they have caused an increasingly dangerous imbalance in life. Defined as "a particular abnormal condition that negatively affects the structure or function of all or part of an organism " [1], they are generally associated with emerging signs and symptoms. Regarding causes, external factors like pathogens and internal malfunctions can be the origin of different diseases, distributed into various types. These include airborne diseases, foodborne diseases, lifestyle diseases, non-communicable diseases, and infectious diseases. Infectious diseases, also called communicable diseases, are among the most dangerous illnesses that haven't stopped manifesting and developing. They can spread from person to person or from animal to person [2]. Infectious diseases are classified into endemic diseases, epidemic diseases, and pandemic diseases. Endemic diseases, the first category, are identified diseases in a given region, with predictable patterns of spread and occurrence rates. The second, epidemic diseases, is characterized by its rapid and brutal spread within a given region. The third, pandemic diseases, are communicable diseases that spread across continents or even the entire world, leading to the contamination of an unimaginable number of people. The classification of infectious diseases is still unsatisfying; it has become not enough to just identify them, but also crucial to neutralize and prevent their spread in real time. Consequently, several studies have made the prediction and prevention of infectious diseases their main objective. Computers have significantly contributed to scientific advancement, especially with the vision of Artificial Intelligence (AI) and its impact on technological growth. The field of Machine Learning (ML) has significantly impacted research and opened the door to what was once considered impossible. Machines equipped with intelligence have now become essential tools in the world of science. Many ML models have therefore been designed to predict infectious diseases and forecast them. Various detection and prediction models have been developed due to the diverse

learning approaches available. Depending on whether supervised, unsupervised, or semi-supervised learning (SSL) is employed, or whether classification or regression models are applied based on the research objective, several techniques can be used to create predictive models. Examples include Support Vector Machines (SVM), Decision Tree (DT) algorithms, clustering algorithms, Naive Bayes (NB), Artificial Neural Networks (ANN), Deep Neural Networks (DNN) such as Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks and Transformers models, as well as emerging techniques like Transfer Learning. Understanding the development of specific diseases and their behaviors is crucial and greatly aids in identifying the origins of outbreaks and taking timely initiatives to combat them. In this paper, some applications of AI techniques in prediction and prevention of infectious diseases are introduced. The main focus is to define the various research objectives being considered, while analyzing which type of data is used to achieve each objective and which type of learning method to employ. A definition of the various research categories that have been developed is introduced to ensure an effective approach for building real-time prediction models for forecasting communicable diseases, while maintaining a swift and efficient predictive process.

This paper follows a specific structure : Section 2 presents the research methodology used to introduce and discuss related works, which are extended in Section 3. In Section 4, a critical analysis and discussion of the accomplished work is presented. Finally, Section 5 concludes the paper by highlighting potential future achievements and discussing future accomplishments.

2 Research methodology

Before taking a vision into the concepts discussed in the literature, three crucial questions caught attention :

- What are the various types of data that have been applied and studied?
- Which kind of learning is being used?
- How are the learning models selected based on the data?

To address these issues, a comprehensive analysis and visualization of the completed work were conducted. For this, an extensive literature review was performed, focusing on relevant articles published since April 2022. This search was carried out using various academic search engines, such as : Google Scholar, ResearchGate, SNDL academic and Scopus. To obtain diverse and comprehensive electronic documentation on scientific research from various publishers. A combination of specific

Figure 1: Research Methodology

set of targeted keywords are also used, to enhance the precision of the search. In the electronic search phase, no restrictions were applied initially. Collecting early discoveries and foundational works on infectious diseases using ML is crucial for understanding the origins of predictions and the evolution of research. The second search was limited to articles published within the last ten years, using various keyword combinations. The key research areas and important topics studied are: infectious diseases, AI, ML and predictions. These articles were initially selected based on an assessment of their titles. Contributions were then further evaluated in the manual review phase after reading the abstracts. The criteria for selecting the articles and the resulting articles are illustrated in Figure 1. Section 3 lists the contributions from the selected papers. The responses to the identified issues and the discussed limitations are covered in Section 4.

3 Background

Human health is influenced by various life phenomena, and it depends not only on the individual themselves, but also on their surrounding environment. Consequently, individuals are called to address and adapt to these influences. Their ability to drive positive change through the development of innovative strategies and technologies offers hope in addressing various health crises. The appearance of the first

pandemics that marked history, along with their impact on daily life and environmental changes, have pushed AI researchers around the world to develop various prediction and prevention systems. Three major categories of prediction research can be identified based on the research objectives and the data used: The first category focuses on detecting and predicting the spread of infectious diseases in specific locations. The second category aims to determine whether an individual may be infected by a communicable disease. While the third category combines the two first categories, addressing both the prediction of communicable diseases in patients and their spread within a population. The diverse methods and databases utilized in the detection of infectious diseases, as presented in this study, are illustrated in Figure 2.

Despite the distinctions that will be highlighted in the following sections between the three categories, all of them rely on the use of different AI techniques and share a common goal: preventing and controlling the spread of a disease.

3.1 Prediction Based on Public Health Data

Initial studies for preventing infectious diseases spread employed event-based and indicator-based surveillance methods [3]. However, their limitations in providing real-time predictions have directed research to the adoption of new technologies. Predictions based on Public Health Data enable a diverse range of technological applications. Consequently, datasets used to study disease's spread are not conform, and the methods applied are not the same for each data. The analysis of data collected from various sources such as social networks, news, mobile phones as well as environmental changes through geospatial images and Epidemiological data, has proven to be highly effective in tracking human behaviors, which helps in assessing pandemic transmissions. Studies have increasingly relied on the integration of multiple data types, including Numerical data, Textual data, and Image data.

3.1.1 Numerical Data : Time-series Epidemiological and locational Data

In predicting contagious diseases, research databases are often presented in numerical formats. These data types frequently include location-related and epidemiological information. The most available data comprises information about the number of deaths, number of contaminated and number of recovered in regions. Epidemiological data includes data on population exposure levels, which are essential for risk assessment. The most used algorithms for this type of data are Naive Bayes (NB), Clustering Algorithms, Long Short-Term Memory Based Models and Transformer Based Models.

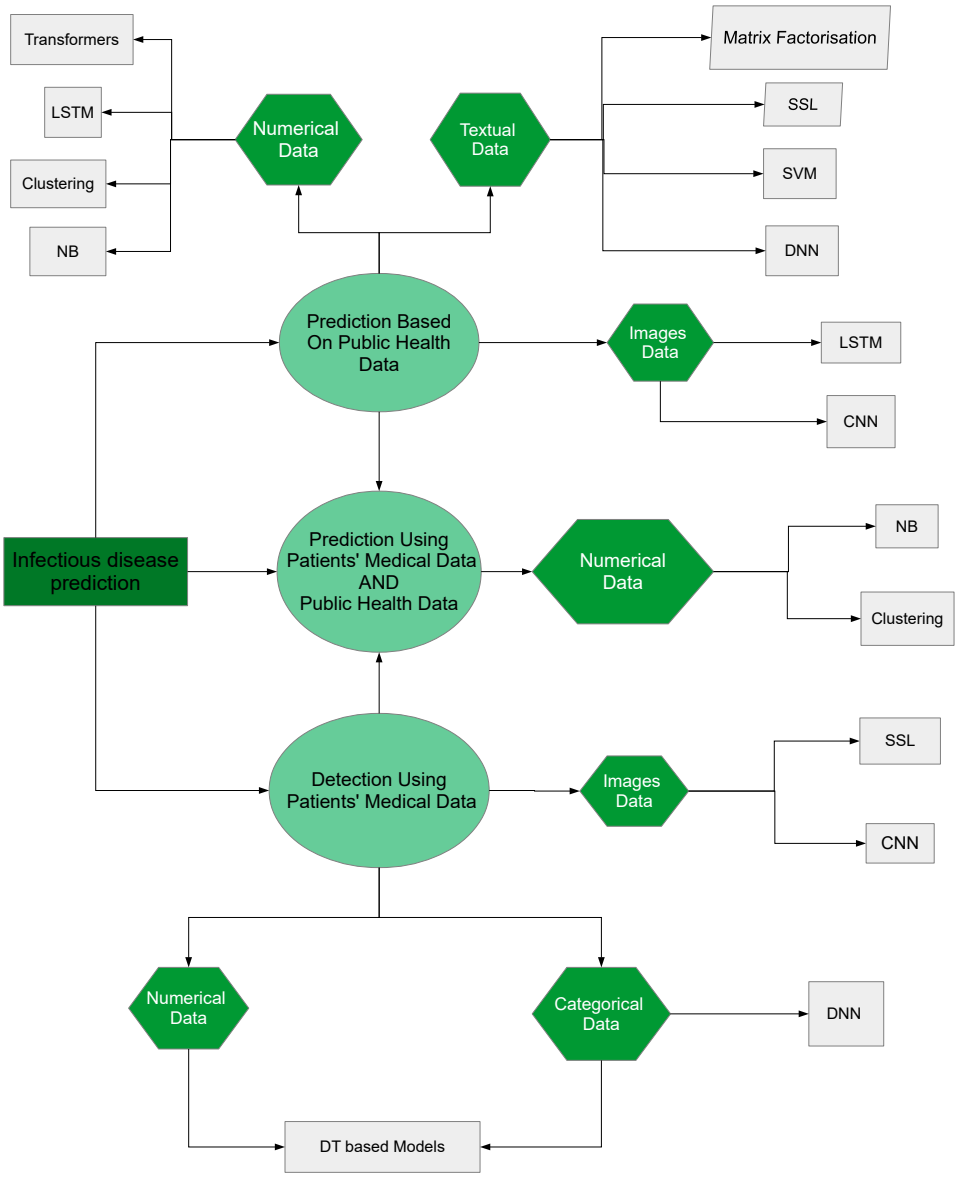


Figure 2: Diagram summarizing the techniques employed for each selected and studied dataset in the field of infectious disease prediction

Naive Bayes algorithms. Naïve Bayes is a simple yet robust algorithm for predicting outcomes. In machine learning, the goal is often to select the best hypothesis based on the given data. Naïve Bayes applies Bayes' Theorem, which offers a method for calculating the probability of a hypothesis using prior knowledge [4]. Tiwari et al. [4] involved the use of NB along with SVM, and Linear Regression (LR), to predict the trend of Covid-19 pandemic over the world while minimizing Mean Absolute Error (MAE) and Mean Squared Error (MSE) (Table 1). The algorithms were applied to a real-time series dataset containing the global record of confirmed, recovered, deaths, and active cases of Covid-19 outbreak. Before the implementation phase, dataset pre-processing is also done for getting the effective results. During the fourth stage (Data collection, Data preprocessing, model training and model evaluation), the data is split into two subsets: the training set and the testing set, where 42 % portion of the data is selected for testing predictions. The NB algorithm proved its effectiveness compared to other tested techniques with an MAE of 488806.7492 and MSE of 400919367451.7439. Despite its advantages, NB only works well with distinct and informative features [5]. Because it treats all features equally and presumes they are conditionally independent, its performance may suffer if there are noisy or irrelevant features [5].

Clustering Algorithms. Ravi et al. [6] proposed a novel ML approach to track COVID-19 contact details that utilizes the DBSCAN algorithm, recognized as one of the most effective clustering algorithms. This approach incorporates time-series location data and prediction techniques to enhance tracking accuracy. The authors have proposed an innovative approach to prevent the spread of new infections in densely populated areas. DBSCAN is used as a clustering algorithm to locate infected individuals and their close contacts, in order to stop the transmission of the virus (Table 1). In the study of Gupta et al. [7], Two different clustering techniques, density-based clustering and partitioning-based clustering, were used to analyze COVID-19 infection cases. A comparative analysis was conducted between the DBSCAN and K-means algorithms, with DBSCAN showing better performance for clustering tasks. Although using time series locational data can provide valuable information about the movement patterns and interactions of individuals over time, DBSCAN has some disadvantages. Including high computational complexity and the need for careful selection of clustering parameters to ensure reliable results [8]. Additionally, it does not work well with data of different densities and is not appropriate for high-dimensional data.

Table 1: Performance evaluation of cited contributions in Time-series Epidemiological and locational Data observations with NB and Clustering algorithms

Ref	Title	Evaluation Metrics				
		Dataset	Method	MAE (%)	MSE (%)	RMSE (%)
[4] (2022)	Pandemic coronavirus disease (Covid-19): World effects analysis and prediction using machine-learning techniques	real-time series dataset that holds the global record of confirmed, recovered, deaths	NBN SVM LR	488806.74 718150.13 648733.09	400919367451.74 565545811024.16 913583889578.49	
[6] (2023)	A Novel Machine Learning Framework for Tracing Covid Contact Details by Using Time Series Locational data & Prediction Techniques	time series locational data	MLDBSCAN	No evaluation metric provided. The proposed system helps in indicating to the user their respective output clusters and their contacts.		

Long short-term memory based models. Since LSTM based models are specialized in the exploration of times series data, they have the potential applications in the field of public health for forecasting epidemic cases, deaths, and recoveries. Some authors like Masum et al. [9] have produced a sustainable prognostic method of COVID-19 outbreak in Bangladesh using the Deep Learning (DL) models. The article presents a forecast on the counting number of infectious cases in Bangladesh from May 15th, 2020 until June 15th, 2020 (30 days). The LSTM network is used to predict the upcoming per day confirmed, death, and recovered cases in Bangladesh on patient data taken from the Institute of Epidemiology Disease Control & Research (IEDCR) healthcare (Table 2). The authors have also made a comparative analysis by the RMSE rate among the LSTM, Random Forest (RF) regression, and Support Vector Regression (SVR) models. Where LSTM proved higher performances on time series analysis. Zhou et al. [10] have also presented a novel approach to forecasting COVID-19 using DL models. The proposed LSTM-based DL model is considered among the most advanced models to forecast time series data. They can take nonlinear factors into account and have the potential to provide more accurate predictions of COVID-19 cases, deaths, and recoveries. The proposed methodology in the paper involves constructing a prediction model of emergency material demand based on the infectious disease prediction model. The model uses a time-varying demand and LSTM sequential decision model to provide a scientific and effective prediction method for actual emergency rescue work (Table 2 : MAE and MSE for death cases in United States). The approach combines traditional infectious disease prediction methods and DL prediction techniques. The proposed model also includes

different countries' migration data, which helps to retrieve other characteristics related to the epidemic and accurately build the model. Rakhshan et al. [11] for their part, have presented a combined approach for modeling and forecasting COVID-19, which can aid in determining interventions and predicting future growth patterns. The used dataset is sourced from the World Health Organization (WHO) and includes daily COVID-19 reports and global geographical distributions. The authors used this dataset to examine data from different countries, select targeted countries for their study, and collect COVID-19 data for analysis. In the authors' approach, dynamic epidemic models and ML methods work together to develop a package for predicting COVID-19. The authors used a dynamic model, along with five different ML models, to process the training and testing data. The dynamic model is a time-dependent compartmental model that captures fluctuations in the number of susceptible, infectious, and confirmed cases with controlled infectivity. The ML models, including LSTM, Multilayer perceptron (MLP), Adaptive neuro fuzzy inference system (ANFIS), General regression neural network (GRNN), and Radial basis function (RBF), are used to compare whether the classic dynamic model means would be best suited for predicting COVID-19 or the selected modern ML methods (Table 2). And some metrics, including Root Mean Squared Error (RMSE), Relative Squared Error (RSE), and Accuracy, are used to evaluate the presented models. Another novel RNN-based model has been developed by Muñoz-Organero et al. [12] to predict COVID-19 incidence in Madrid by integrating mobility data from a bike-sharing service. The model combines an LSTM-based RNN alongside mobility data to improve prediction accuracy. The analysis utilizes weekly COVID-19 case counts per district in Madrid and the number of bike rides recorded by the city's bike-sharing service, BiciMAD. The bike-sharing data serves to estimate human mobility patterns between districts, while The LSTM RNN captures temporal patterns in the data. The proposed model achieves an RMSE of 0.0205 (Table 2), outperforming the baseline model, which has an RMSE of 0.02296. This represents an 11.7% improvement in prediction accuracy compared to the baseline model that excludes mobility data.

Transformer Based Models. Transformers are neural network models that replace the commonly used recurrent layers in encoder-decoder architectures with multi-head self-attention. By relying entirely on attention mechanisms, transformers effectively capture global dependencies in data sequences and allow for much greater parallelization [13]. Ming et al. [14] developed a computational tool, HostNet, to predict virus hosts using deep neural networks. HostNet integrates Transformer, CNN, and BiGRU models, and was tested on a benchmark dataset of 'Rabies lyssavirus' and an in-house 'Flavivirus' dataset. It outperforms existing methods in

Table 2: Performance evaluation of cited contributions in Time-series Epidemiological and locational Data observations using LSTM based models

Ref	Title	Evaluation Metrics				
		Dataset	Method	MAE (%)	MSE (%)	RMSE (%)
[9] (2020)	COVID-19 in Bangladesh: a deeper outlook into the forecast with prediction of upcoming per day cases using time series	Confirmed	LSTM			65.83
			RFR			184.21
			SVR			166.15
		Death	LSTM			2.95
			RFR			3.28
			SVR			4.73
		Recovery	LSTM			163.21
			RFR			170.15
			SVR			215.08
[10] (2023)	Improved LSTM-based deep learning model for COVID-19 prediction using optimized approach	Epidemiological data	LSTM	0.01962	0.00102	
			GRU	0.00679	0.02788	
			Bi-LSTM	0.00623	0.25110	
			Dense-LSTM	0.00763	0.00016	
[11] (2023)	Global analysis and prediction scenario of infectious outbreaks by recurrent dynamic model and machine learning models: A case study on COVID-19	daily COVID-19 reports and global geographical distributions : Trained data	GRNN			0.03
			RBF			0.006
			LSTM			0.25
			MLP			0.005
			ANFIS			0.005
		daily COVID-19 reports and global geographical distributions : Tested data	GRNN			0.06
			RBF			0.011
			LSTM			0.02
			MLP			0.02
			ANFIS			0.01
[12] (2023)	A new RNN based machine learning model to forecast COVID-19 incidence, enhanced by the use of mobility data from the bike-sharing service in Madrid	weekly COVID-19 case counts per district in Madrid and the number of bike rides recorded by the city's bike-sharing service, BiciMAD	LSTM-based RNN			0.0205

accuracy and F1 score, thanks to its enhanced representation modules. Transformers are highly effective for time series forecasting tasks. Another Transformer-based model was also developed by Li et al. [15] to predict the long-term spread of seasonal influenza. It includes a source selection module to merge data from various sources and capture spatial dependencies. The model was tested on datasets from the United States and Japan, which included weekly influenza statistics from different regions. Demonstrating superior long-term forecasting performance compared to traditional autoregressive and RNN-based models, achieving an RMSE of 0.52 for

short-term predictions and 0.87 for long-term predictions on the Japan dataset. For the US-HSS dataset, the model achieved an RMSE of 0.54 for short-term predictions and 0.89 for long-term predictions (Table 3). Due to the limitations of traditional epidemiological and ML models in forecasting the COVID-19 pandemic such as challenges with generalization, scalability, and the lack of sufficient surveillance data, Wang et al. [16] have proposed a novel approach that combines epidemiological theories with Generative Adversarial Networks (GANs). Their model, T-SIRGAN, integrates the Susceptible Infectious Recovered (SIR) model to generate epidemiological simulation data, while GANs are employed for data augmentation. Transformers are then used to predict future trends. The study utilized COVID-19 data, including cumulative confirmed cases and deaths, from the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University. The T-SIRGAN model outperformed other methods, demonstrating superior accuracy in predicting epidemic trends by integrating epidemiological simulations and GANs. Specifically, the model achieved the lowest RMSE of 0.0188 for predicting confirmed cases, and an RMSE of 0.0243 for predicting death cases, outperforming other models in both metrics (Table 3). Some other challenges in infectious disease prediction are addressed, such as the variability in incubation periods and the progression dynamics of different diseases. Wang et al. [17] introduces an Oriented Transformer (ORIT), which improves upon traditional Multiple Representation Fusion (MRF) methods by capturing multi-dimensional temporal relationships within disease case data. ORIT incorporates a Multi-head Oriented Attention Unit (MOAU), designed to learn correlations from various orientations within the time series data, enabling the model to capture complex patterns in infectious disease progression. Two real-world datasets were used for evaluation: the Hand, Foot, and Mouth Disease (HFMD) dataset with 49,677 records, and the Hepatitis B Virus (HBV) dataset with 48,359 records. After data preprocessing, the MOAU captures attention from different orientations of the time series, including the impact of diverse time steps, correlations between different time series, and the significance of temporal segments. A comparison with 21 other models showed that ORIT demonstrated superior performance, achieving an RMSE of 16.8450 on the HFMD dataset and 28.2686 on the HBV dataset (Table 3).

Discussion

Using time series data to predict infectious diseases involves analyzing the epidemiological growth and contact tracing of the illness (Tables 4, 5). Locational data, for example, can help in identifying potential contacts and understanding the spread of the virus within a specific area. By analyzing their temporal aspect, it may be pos-

Table 3: Performance evaluation of cited contributions in Time-series Epidemiological and locational Data observations using Transformer based models

Ref	Title	Evaluation Metrics		
		Dataset	Method	RMSE (%)
[15] (2021)	Long-term prediction for temporal propagation of seasonal influenza using Transformer-based model	weekly influenza-like-illness statistics JAPAN	Transformer Short-term Transformer Long-term	0.52 0.87
		weekly influenza activity levels for 10 HHS regions of the U.S US-HSS	Transformer Short-term Transformer Long-term	0.54 0.89
[16] (2022)	Predicting the epidemics trend of COVID-19 using epidemiological-based generative adversarial networks	COVID-19 data, including cumulative confirmed cases	Transformer T-SIRGAN	0.0188
		COVID-19 data, including cumulative deaths cases	Transformer T-SIRGAN	0.0243
[17] (2023)	Oriented transformer for infectious disease case prediction	HFMD Dataset	Oriented Transformer (ORIT)	16.8450
		HBV Dataset	Oriented Transformer (ORIT)	28.2686

sible to track the movements of infected individuals and identify individuals who may have come into proximity with them. This can aid in effective contact tracing and containment strategies. However, due to limited accurate data on COVID-19 records and locations, as well as inherent uncertainties, traditional methods have struggled to accurately predict the global impact of the pandemic [4]. Recent ML models have shown improved efficiency in forecasting infectious diseases. Naïve Bayes proved its effectiveness in handling uncertainty by estimating the probabilities of outcomes, making it useful for both predictive and diagnostic tasks [19]. Clustering-based machine learning techniques can also automate contact tracing, resulting in more accurate and efficient outcomes [7]. Recurrent Neural Networks (RNNs) have gained significant attention in the field of deep learning for their ability to model nonlinear relationships. However, traditional RNNs face vanishing gradient issues and failed with capturing long-term dependencies [20]. Long Short-Term Memory (LSTM) networks and their variants have been applied to sequence modeling, addressing these challenges and achieving success in various applications [21]. Transformer models have further demonstrated superior performance in capturing long-range dependencies compared to RNNs [13], as their self-attention mechanism reduces the signal transmission path within the network, removing the need for a recurrent structure [22].

ObservationDate	Province/State	Country/Region	Last Update	Confirmed	Deaths	Recovered
01/22/2020	Anhui	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
01/22/2020	Beijing	Mainland China	1/22/2020 17:00	14.0	0.0	0.0
01/22/2020	Chongqing	Mainland China	1/22/2020 17:00	6.0	0.0	0.0
01/22/2020	Fujian	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
01/22/2020	Gansu	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
01/22/2020	Guangdong	Mainland China	1/22/2020 17:00	26.0	0.0	0.0
01/22/2020	Guangxi	Mainland China	1/22/2020 17:00	2.0	0.0	0.0
01/22/2020	Guizhou	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
01/22/2020	Hainan	Mainland China	1/22/2020 17:00	4.0	0.0	0.0
01/22/2020	Hebei	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
01/22/2020	Heilongjiang	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
01/22/2020	Henan	Mainland China	1/22/2020 17:00	5.0	0.0	0.0
01/22/2020	Hong Kong	Hong Kong	1/22/2020 17:00	0.0	0.0	0.0
01/22/2020	Hubei	Mainland China	1/22/2020 17:00	444.0	17.0	28.0
01/22/2020	Hunan	Mainland China	1/22/2020 17:00	4.0	0.0	0.0
01/22/2020	Inner Mongolia	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
01/22/2020	Jiangsu	Mainland China	1/22/2020 17:00	1.0	0.0	0.0
01/22/2020	Jiangxi	Mainland China	1/22/2020 17:00	2.0	0.0	0.0
01/22/2020	Jilin	Mainland China	1/22/2020 17:00	0.0	0.0	0.0
01/22/2020	Liaoning	Mainland China	1/22/2020 17:00	2.0	0.0	0.0

Table 4: Time series example for confirmed, deaths and recovered cases [18] [4]

Country/Region	Confirmed	Active	Deaths
US	1.528.568	1.147.255	91.921
Russia	299.941	220.974	2837
Brazil	271.885	147.108	17.983
UK	250.138	213.617	35.422
Spain	232.037	204.259	27.778
Italy	226.699	65.129	32.169
France	180.933	90.230	28.025
Germany	177.778	14.016	8081
Turkey	151.615	34.521	4199
Iran	124.603	20.311	7119
India	106.475	60.864	3302
Peru	99.483	60.045	2914
Mainland China	82.963	88	4634
Canada	80.493	34.396	6028
Saudi Arabia	59.854	27.891	329
Belgium	55.791	31.996	9108
Mexico	54.346	11.355	5666
Chile	49.579	27.563	509
The Netherlands	44.449	38.548	5734
Pakistan	43.966	30.538	939

Table 5: Top 20 Covid-19 affected countries record (confirmed, active and deaths) collected from 22 January 2020 to 19 May 2020 [4]

3.1.2 Textual Data : Social and News Data

Textual data is becoming increasingly important among the various types of data used to predict transmissible diseases, as it enhances monitoring and prevention efforts. Known for their real-time acquisition, many approaches are developed using social and news data. Techniques such as: superviseed matrix factorization, SSL, SVM and DNN have showed promising results.

Matrix factorization. Chakraborty et al. [23] used a supervised matrix factorization method to extract features of each disease from news streams. For each study, independent words collected from news related to the diseases are modeled into a matrix and combined with structured time series data from different outbreaks. Matrix factorization is then applied to factorize the initial matrix into two other matrixes, each one contains latent features which are used to detect disease apparition (Table 6). The method of detection used in this study involved collecting each word in relation to outbreaks, which proved to be time-consuming and less accurate. Although the study paid some attention to word extraction, these words were considered non-dependent, which contradicts the common addiction where the appearance of one word can influence the appearance of another [5].

Semi-supervised learning based models. The ability of using News data has involved other approaches using different ML techniques. Kim et al. [24] employed articles and reports to predict infectious diseases that did not occur for six months in various countries, testing models based on SVM, SSL, and DNN. The number of articles related to each disease was calculated, and diseases were then labeled for each country based on whether they have appeared or not. Known as an ML technique that combines labeled and unlabeled data, SSL based models showed outstanding performance compared with SVM and DNN (Table 6).Because SSL makes good use of both labeled and unlabeled data, it has attracted a lot of interest. This is particularly crucial in practical applications when very little data is labeled [25]. However, a lot of unimportant or noisy elements in real-world raw data are frequently missed by SSL approaches. To enhance semi-supervised classification performance, it is crucial to choose pertinent neighbors and characteristics for every sample [25]. And despite the quality of the study conducted in [24], the experience in their proposed work was conducted during two distinct periods, which can lead to a contradiction due to the existence of seasonal diseases.

Support vector machine models. Since the SVM based models can handle high-dimensional problems with limited training data [26], Kim et al. [27] developed a

prediction model using SVM based on an analysis of articles related to influenza pandemics and infectious diseases. The authors extracted several keywords that were closely related to influenza and used word2vec to determine which keywords were related to the keyword 'influenza'. Then, SVM was applied to the extracted data to predict if the number of influenza patients would increase or decrease at a specific week. The prediction results using news text data with SVM achieved a mean accuracy of 86.7% in forecasting whether the weekly influenza-like illness (ILI) patient ratio would increase or decrease, and an RMSE of 0.611% in estimating the weekly ILI patient ratio (Table 6). Thapen et al. [28] used novel data-analytics to detect and forecast epidemics while developing DEFENDER system : Detecting and Forecasting Epidemics Using Novel Data-Analytics for Enhanced Response. The system ensures three services : early warning detection, situational awareness and nowcasting of epidemics. The number of tweets matching each symptom captured on the online database Freebase, was tracked daily for each geographical area monitored. To distinguish between health-related and non-health-related tweets and articles, two classifiers were used: SVM and NB (Table 6). The areas of high tweet activity were located in a country or region using the DBSCAN algorithm. The number of cases from the current data is predicted by adjusting the observed symptom levels to the previously available clinical data containing week, disease, location and count. Although SVM outperforms many other systems, it has limitations with complex data due to the high computational cost of solving quadratic programming problems [29]. Its performance also heavily depends on the choice of kernel functions and their parameters [29].

Deep neural network based models. Since the study of Thapen et al. [28] considered only a limited number of symptoms, Serban et al. [30] proposed an improvement called SENTINEL of the previous system (DEFENDER), that aims to explore a boarder range of symptoms and diseases. DNN based models (CNN, LSTM) were then selected to differentiate between health-related and non-health-related tweets (Table 6). In both studies [28] and [30], the social media twitter was analyzed. Given the strong correlation between infectious diseases and Twitter data [31], Chae et al. [32] presents a novel approach for predicting infectious diseases using deep learning models, specifically DNN and LSTM, combined with big data sources like Twitter mentions along with Naver search queries and weather data. The study addresses limitations of traditional models like autoregressive integrated moving average (ARIMA), by incorporating real-time data to predict the spread of diseases such as chickenpox, scarlet fever, and malaria. The DL models significantly outperformed traditional methods, with DNN improving prediction performance by 24% and LSTM by 19% for chickenpox. The main evaluation metric, RMSE, showed

that these models better captured trends. The mean RMSE of the top 10 DNN models for chickenpox was 72.8215, while the top 10 LSTM models had a mean RMSE of 78.2850, particularly during rapid disease spread (Table 6). Demonstrating their potential to enhance infectious disease forecasting systems. Despite Twitter's reputation for being used by credible individuals sharing accurate information, it is not widely used by numerous people. Consequently, the conclusions obtained are then restricted. Drinkall et al. [33], for their part, have introduced a novel approach that incorporates transformer-based language models into infectious disease modeling using Reddit posts. The analysis uses Reddit comments extracted via the Pushshift API, state-level epidemiological data, government response data, and Google's COVID-19 Community Mobility Reports, which provide local movement data. In the feature identification process, sentence-level encoding, dimensionality reduction, and clustering (HDBSCAN) are applied to isolate predictive features from Reddit comments. For evaluation, the resulting features are compared to traditional datasets in both a threshold-classification task and a time-series forecasting task. In the threshold-classification task, a Random Forest model utilizing the extracted features achieved the highest accuracy across various prediction horizons. Particularly in identifying upward trend signals for extreme events, with an average performance score of 0.880. In the time-series forecasting task, the transformer model consistently outperformed Gaussian Process and Martingale models. Achieving the lowest Root Mean Square Error (RMSE) of 0.0284 when the extracted features were used as covariates (Table 6). The method clearly outperforms traditional models in predicting COVID-19 trends, particularly in regions with unreliable epidemiological data.

Discussion

Some disease surveillance systems scan news articles from global sources like Google News and social media platforms such as Twitter [28]. They filter and classify these articles based on the type of epidemic, location, and news source. However, a major limitation of these systems is that they primarily focus on collecting disease-related information from various sources and compiling it for information dissemination or surveillance purposes [23]. A more precise and refined application of ML models is crucial for achieving optimal control over predictive systems. This ensures higher accuracy and effectiveness in decision-making processes. SSL based models, SVM based models along with DNN models showed high effectiveness in accurately distinguishing between health-related and non-health-related articles and tweets.

Table 6: Performance evaluation of cited contributions in News Data observations with Matrix factorization, SSL SVM and DNN models

Ref	Title	Evaluation Metrics						
		Dataset	Method	Precision (%)	Recall (%)	Accuracy (%)	ROC (%)	F1 score (%)
[23] (2016)	Extracting signals from news streams for disease outbreak prediction	Dengue	Supervised Matrix Factori-sation	83.5	62.5			
		Flu		79.3	58.5			
		Malaria		81.2	68.5			
		Diabetes		77.2	59.1			
		TB		79.3	69.5			
[24] (2021)	Infectious disease outbreak prediction using media articles with machine learning models	Articles and Reports	SSL			83.3	79.1	83.2
			SVM			73.2	65	76.9
			DNN			80.6	74.6	81.9

Ref	Title	Evaluation Metrics				
		Dataset	Method	Precision (%)	Accuracy (%)	RMSE (%)
[27] (2019)	Weekly ILI patient ratio change prediction using news articles with support vector machine	Articles influenza infectious diseases	SVM		86.7	0.611
[28] (2016)	DEFENDER: detecting and forecasting epidemics using novel data-analytics for enhanced response	Social Medias (Twitter), News, Clinical Data	SVM, NB	8.20		

Ref	Title	Evaluation Metrics			
		Dataset	Method	RMSE	Accuracy (%)
[30] (2019)	Real-time processing of social media with SENTINEL: A syndromic surveillance system incorporating deep learning for health classification	News Twitter	CNN		93.9 85.4
[32] (2018)	Predicting infectious disease using deep learning and big data	Twitter mentions, Naver search queries and weather data	DNN LSTM	72.8215 78.2850	
[33] (2022)	Forecasting COVID-19 caseloads using unsupervised embedding clusters of social media posts	Reddit comments, state-level epidemiological data, government response data and Google's COVID-19 Community Mobility Reports	Transformer	0.0284	

3.1.3 Image Data : Geospatial Images

Geospatial images, such as satellite images, are known as one of the most powerful and important tools for monitoring the earth [34]. They track the physical environment (water, air, land, vegetation) and the changing human footprint across the globe. And some DL techniques specialized in image processing have given specific interest to explore Geospatial images to detect different symptoms related with a disease.

Convolutional Neural Network based models. Regarding the importance of natural and environmental details, some authors (Li et al.) [35] started on the creation of a multi-source natural feature benchmark dataset called GeoImageNet for GeoAI and supervised ML. The dataset was created by combining color imagery and Digital Elevation Model (DEM) data. GeoImageNet contains location information for each image scene, making geographic validation and training data expansion easy to achieve. The multi-source dataset empowers the machine to gain more geospatial intelligence and automation, resulting in higher prediction accuracy than commonly used single data sources. The authors have evaluated the dataset using two popular and representative object detection models, Faster-RCNN and RetinaNet, and its validity was proved for aiding a GeoAI model to achieve convergence and satisfactory detection performance (Table 7). CNN-based models excel at identifying important characteristics and successfully completing classification or prediction tasks [35]. Thus, GeoImageNet has demonstrated its ability to support research on a wide range of environmental and health issues, including tracking the spread of infectious diseases.

LSTM based models. In order to examine the real world environment, Lee et al. [36] tried to develop a prediction model for the number of influenza patients at the national level using satellite images (Table 7). The authors developed a convolutional LSTM-LSTM neural network model, which demonstrated a strong correlation between the predicted and actual numbers of influenza patients, with an average MAE of 5.9010 per million population. Their study highlights the potential of using satellite image data as a valuable resource for predicting influenza incidence, which could facilitate timely national interventions. While the use of satellite images marks significant progress in real-time data acquisition, extracting and analyzing these images can be costly and requires specialized expertise [37]. Additionally, satellite images are limited in their effectiveness for indoor localization due to restricted accessibility in indoor environments.

Table 7: Performance evaluation of cited contributions in Geospatial Images data observations with LSTM and CNN based models

Ref	Title	Evaluation Metrics			
		Dataset	Method	MAE	Precision (%)
[35] (2023)	GeoImageNet: a multi-source natural feature benchmark dataset for GeoAI and supervised machine learning	single-source data	Faster-RCNN and RetinaNet		50
		GeoImageNet			80
[36] (2024)	Convolutional LSTM–LSTM model for predicting the daily number of influenza patients in South Korea using satellite images	Sattelite Images	LSTM	5.9010 per million population	

Discussion

Despite the use of various prediction models for infectious disease forecasting, most studies rely on local meteorological data, such as temperature, humidity, precipitation, and solar radiation. This limits predictions to specific regions and reduces their applicability at the national level [36]. Alternatively, satellite images provide a more comprehensive means of capturing weather patterns nationwide. These images are available through multiple channels, allowing the detection of key meteorological factors such as temperature, moisture, clouds, and precipitation. By proposing a national-level influenza prediction model based on satellite images and analyzing the relationship between influenza incidence and these meteorological factors, Lee et al. [36] present a model capable of forecasting influenza across the country. The integration of LSTM and CNN based models has, for instance, proven to be highly effective for forecasting infectious diseases using satellite imagery. However, due to the presence of certain areas that satellite images cannot capture, their use can be challenging [38].

3.2 Detection Using Patients' Medical Data

The detection Using Patients' Medical Data category is focused in predicting whether a person is contaminated by the disease or not. Most researchers estimated that, more disease detection is quickly identified in a person, more the spread of disease will be controlled. This kind of detection is manifested through numerical data, categorical data and images data, using multiple techniques based on ML algorithms.

3.2.1 Numerical Data : Routine Blood Tests

To evaluate the performance rate of ML applications in predicting diseases through routine blood tests, many collaborations test various ML models and algorithms. Peiffer-Smadja et al. [39] examined the exploration rate of ML in clinical microbiology. They concluded that : ANN, SVM, RF, LR, k-nearest neighbors (k-NN), and NB can be explored for targeting different diseases such as : bacterial infections, parasitic infections, viral infections, microorganism detection and diseases classification using diverse sources of data : microorganisms, microscopic-images and protein structure. Cabitza et al. [40] for their part, developed, evaluated and validated ML models for COVID-19 detection using routine blood tests. ML models are developed following four steps : Imputation, Data normalization, Feature Selection and Classification. For the classification step, five different models (RF, NB, LR, SVM, k-NN) are developed for each kind of data: OSR patients, OSR dataset, covid-specific and CBC dataset. Models are therefore evaluated according to the Accuracy, Sensitivity, Specificity, Area under the curve (AUC) and External Validation. The Three best models extracted are : k-NN for COVID-19 specific dataset, RF and k-NN for CBC dataset (Table 8). Models based on DT Algorithms such as: DT, RF and Gradient Boosting, have showed most interest, particularly in exploring routine blood tests. DTs have the ability to explain ML-based diagnoses [41]. They can break down complex data into more manageable parts, making them more interpretable compared to other algorithms in this category of prediction.

Decision Tree based models. Among models based on DT, Random Forests (RF), a popular ML algorithm that aggregates the outputs of multiple decision trees to produce a unified result [42]. By aggregating the predictions of multiple trees, it reduces the risk of overfitting and improves the generalization ability of the model [43]. While RF is not as easily interpretable as a single decision tree, it still provides insights into feature importance and evaluate the significance of various features, enabling the identification of the most influential ones for prediction. RF is capable of handling both categorical and continuous features [43]. Brinati et al. [44] focused on developing two ML models: RF Classifier and Three-Way RF Classifier (TWRF) using routine blood exams with demographic characteristics in order to detect COVID-19 infections. A DT Model is then interpreted to assist scientists in making decisions regarding the infections or not of COVID-19 (Table 8). For Banerjee et al. [45], They have guided their researches to predict if a person is SARS-CoV-2 positive or negative in the early stage of the disease from full blood counts. RF, glmnet (lasso-elastic-net regularized generalized linear) and ANN are used (Table 8). They indicated that RF and glmnet provided more information about important

variables and clearly indicate how the decision was made. For their iterative learning, boosting algorithms improved their importance in several studies. Kukar et al. [46] started in performing COVID-19 diagnosis using Smart Blood Analytics (SBA) Algorithm and routine blood tests with the most popular ML tools, XGBoost to build the diagnostic models. The two most discriminating parameters were prothrombin and INR (International Normalized Ratio). For the evaluation of the model, ten-fold cross-validation is applied on independent testing data. XGBoost showed better results compared to other algorithms : SVM, RF and NN (Table 8). SBA technique is also employed in the study of Yang et al. [47], where the authors aimed to predict an individual's SARS-CoV-2 infection status by employing Gradient Boosting Decision Tree (GBDT) using corporating patient demographic features such as age, sex, race and 27 routine laboratory tests (Table 8). Compared to LR, DT and RF, GBDT showed better results with 85.4 % AUC, 76.1 % sensitivity and 80.8 % specificity. Given the highly accurate predictions from RF and GBoost, Yang et al. [48] have called for the inclusion of both these models along with extremely randomized trees (ET) and LR models (Table 8). The authors have proposed a two-step learning approach for diagnosing COVID-19 using routine blood tests. The first step consists of making predictions using three different learning algorithms: ET, RF and LR. Resulting predictions are used in the second step as inputs of the prediction model XGBoost to establish the final predictions. The suggested model ERLX showed better results compared to previously proposed systems. However, the vulnerability in this proposed study is located in feature selection. Specifically, 18 features are selected to make the study according to the feature's importance appeared in older papers.

Discussion

DT based models have demonstrated their effectiveness in detecting infectious diseases using routine blood data (Table 9). They provide deeper insights into the importance of various features and their significance. Models based on RF or GBoost, whether using parallel learning with RF or iterative learning with GBoost, allow for more accurate detection in the analysis of blood characteristics. The combination and hybridization of different decision tree algorithms further enhance accuracy and promise favorable results [48]. Although the results may be promising, DT based models showed some limitations in depth selection. The choice of the top-level to derive the top consistent parameters can significantly impact the relevance of the model. Future research can address these limitations to develop more effective approaches. The use of blood data also presents confidentiality concerns with some

Table 8: Performance evaluation of cited contributions in routine blood tests observations using DT based algorithm

Ref	Title	Evaluation Metrics					
		Dataset	Method	Accuracy (%)	Sensitivity (%)	Specificity (%)	External-Validation (%)
[40] (2021)	Development, evaluation, and validation of machine learning models for COVID-19 detection based on routine blood tests	routine blood tests Covid specific dataset	k-NN	78	74	81	94
		routine blood tests CBC dataset	RF	76	70	82	96
			k-NN	75	72	78	92
[41] (2021)	Explaining machine learning based diagnosis of COVID-19 from routine blood tests with decision trees and criteria graphs	Routine Blood Tests	LR	82	73	84	
			RF	88	66	91	
			XGBoost	87	60	91	
			SVM	84	56	89	
			MLP	85	42	92	
			ENSEMBLE	88	67	91	

Ref	Title	Evaluation Metrics				
		Dataset	Method	Accuracy (%)	Sensitivity (%)	Specificity (%)
[44] (2020)	Detection of COVID-19 infection from routine blood exams with machine learning: a feasibility study	Routine Blood Tests	DT	70..78		
			ET	68..79		
			k-NN	66..76		
			LR	70..81		
			NB	64..81		
			RF	74..80		
			SVM	69..80		
			TWRF	83..89		
[45] (2020)	Use of machine learning and artificial intelligence to predict SARS-CoV-2 infection from full blood counts in a population	Full Blood Counts	RF	82	60	88
			GLmnet	81	65	81
			Ann	87	43	91

Ref	Title	Evaluation Metrics					
		Dataset	Method	Sensitivity (%)	Specificity (%)	AUC (%)	Accuracy (%)
[46] (2021)	COVID-19 diagnosis by routine blood tests using machine learning	Routine Blood Tests	XGBoost ML Algorithm	81.9	97.9	0.97	
[47] (2020)	Routine laboratory blood tests predict SARS-CoV-2 infection using machine learning	Corporating patient demographic, Routine Laboratory Tests	GBDT	76.1	80.8	85.4	
			RF	73.5	81.8	84.3	
			LR	71.1	75.6	80.9	
			DT	61.8	73.2	70.4	
[48] (2020)	Ensemble learning model for diagnosing COVID-19 from routine blood tests	Routine Blood Tests	ERLX Ensemble learning model	98,72	99,99	99,38	99,88

Parameter	Acronym	Unit of measure	COVIDspecific features	CBC features	Missing rate, %
White blood cells	WBC	10^9/L	X	X	2.4
Red blood cells	RBC	10^{12}/L	X	X	3.6
Hemoglobin	HGB	g/dL	X	X	2.4
Hematocrit	HCT	%	X	X	2.4
Mean corpuscular volume	MCV	fL	X	X	3.6
Mean corpuscular hemoglobin	MCH	pg/Cell	X	X	3.6
Mean corpuscular hemoglobin concentration	MCHC	g Hb/dL	X	X	2.4
Erythrocyte distribution width	RDW	CV%	X	X	3.7
Platelets	PLT	10^9/L	X	X	3.6
Mean platelet volume	MPV	fL	X	X	5.9
Neutrophils count (%)	NE	%	X	X	18.9
Lymphocytes count (%)	LY	%	X	X	15.2
Monocytes count (%)	MO	%	X	X	15.2
Eosinophils count	(%) EO	%	X	X	15.2
Basophils count (%)	BA	%	X	X	15.2
Neutrophils count	NET	10^9/L	X	X	15.2
Lymphocytes count	LYT	10^9/L	X	X	15.2
Monocytes count	MOT	10^9/L	X	X	18.9
Eosinophils count	EOT	10^9/L	X	X	15.2
Basophils count	BAT	10^9/L	X	X	18.9

Table 9: Part of the Complete list of the analyzed features in the OSR dataset [40]

significant missing values, making its application more difficult.

3.2.2 Categorical Data : Clinical Data

Clinical data are frequently used to track various diseases, and increased efficiency can be achieved by directly using factors in relation with biological experiments. This data can be integrated with different applying methods like DT based models and DNN based models. Indicating their efficiency in infectious disease detection.

DT based models. As a parallel learning set model, RF algorithm has been the topic of various studies. In the study of Kumar et al. [2], different ML models are adopted, among them : LR, RF, DT, MLP, SVM, k-NN, ANN, along with some other models, in order to predict chronic diseases (cardio vascular disease (CVD), chronic kidney disease (CKD), lung cancer) and infectious diseases (hepatitis and dengue serotypes) (Table 10). With Hepatitis Dataset analysis, RF exceed other algorithms with an accuracy of 90%. The RF algorithm has therefore proved its effectiveness for extracting meaningful insights from data for predicting these kinds of infectious diseases.

DNN based models. Some studies using clinical data take into consideration a few attributes and observations in diseases data, which can be limited, as it may indicate a disease other than the emerging one, since different diseases may share common symptoms. Devi et al. [49] have demonstrated high accuracy and less execution time in detecting DENV serotypes while using the MSO-MLP method (Table 10). MSO-MLP represents an incorporation of MLP and Multi-Swarm Optimization (MSO) which is a powerful metaheuristic algorithm building on the success of

Table 10: Performance evaluation of cited contributions in clinical data observations with DT and DNN based models

Ref	Title	Evaluation Metrics		
		Dataset	**Method**	**Accuracy (%)**
[2] (2020)	Prediction of chronic and infectious diseases using machine learning classifiers-A systematic approach	Chronic Kidney disease	SVM	64.5
			C4.5	75.32
			PSO-MLP	68.31
			DT	72.67
			ABC4.5	92.76
		Hepatitis Dataset	LR	84
			RF	90
			DT	88
			C4.5	85
			MLP	75
		CVD Dataset	RF	80
			J48	85
			Hoeffding Tree	86
			LMT	85
			RT	70
		Dengue Dataset	DT	80
			ANN	85
			MSO-MLP	86
			PSO-ANN	85
		Lung Cancer	SVM	91.2
			k-NN	83.2
			RF	80.2
			ANN	93.4
[49] (2018)	MSO - MLP diagnostic approach for detecting DENV serotypes	Dengue Fever Dataset	MSO-MLP	85.18

Particle Swarm Optimization (PSO), a population-based algorithm inspired by the collective behavior of bird flocks and fish schools. MSO advances this concept by incorporating multiple swarms instead of just one [50]. In the MSO-MLP approach, multiple swarms of particles are used to optimize the weights and biases of the MLP [49]. Kumar et al. [2] have also studied the efficiency of MSO-MLP on Dengue Dataset (Table 10), where the MSO-MLP provides an accuracy of 86%, which is better compared to other tested classifiers.

Discussion

Certain characteristics in clinical data, such as temperature, pulse, acute fever, vomiting, abdominal pain, body aches, cold symptoms, headache, weakness, fatigue, and rapid breathing, facilitate the identification of symptoms related to each disease [2]. Each infection has its own specific signs and symptoms, with common indicators including fever, diarrhea, fatigue, muscle aches, and coughing. The application of ML and DL techniques has shown good accuracy in detecting infectious diseases using clinical data. Despite the productivity of clinical data and the highly effective

detection capabilities of models based on DT and DNN, the similarity observed in the clinical symptoms of infectious diseases, frequently lead to disease identification issues [51]. Furthermore, the use of clinical data is subject to privacy concerns, which makes data collection for studies quite challenging.

3.2.3 Images Data : Chest X-ray (CXR) and CT scan Images

Another type of data used in detecting transmissible diseases in patients is image analysis. Specifically, chest X-ray images (CXR) and CT scan images are frequently collected in various studies focused on infectious disease detection using various models based on SSL technique and CNN model.

SSL based models. SSL addresses the limitations of supervised learning when dealing with datasets that include both labeled and unlabeled data. By using a small amount of labeled data along with a larger set of unlabeled data [52]. Sahoo et al. [53] have employed SSL approaches to detect COVID-19 cases accurately by analyzing digital chest X-rays and CT scans images. Their proposed algorithm COVIDCon applied on a small COVID-19 radiography dataset, attains 97.07% average class prediction accuracy. When applied on large datasets, COVIDCon achieves an accuracy of 99.13% (Table 11). The authors have therefore provided a fast, accurate, and reliable method for screening COVID-19 patients.

CNN based models. CNNs have proven to be a powerful class of models for comprehending the content of images, leading to significant advancements in image processing. CNNs are both efficient and effective in various pattern and image recognition applications, such as gesture recognition, face recognition, object classification, and generating scene descriptions [54]. Hussein et al. [55] in their work, have discussed COVID-19 infections identification in chest X-rays by using Custom-CNN, a DL technique. The model achieved a classification accuracy of 98.19% in distinguishing COVID-19, normal, and pneumonia samples (Table 11). Chest X-ray images were also employed in the study of Issahaku et al. [56] which focused on multimodality by integrating cough sound features. The Visual Geometry Group (VGG16) model was used for feature extraction and Faster R-CNN for COVID-19 detection. The study achieved an accuracy of 99.80% (Table 11). As Transfer Learning approach enables the storage and use of knowledge acquired from pretrained models to address new problems [57], Some authors, Sadegji et al. [58] introduced a novel dataset and proposed six different transfer learning models for slide-level analysis. Which was able to detect COVID-19 CT slides with an accuracy of more than 99%. They have developed DL models to facilitate automated diagnosis of COVID-19

from CT scan records of patients. The authors also developed a novel 3D deep model (MASERes), for patient-level analysis, that achieved an accuracy of 100% (Table 11). Enhancing the proposed models for practical use, especially in regions with limited medical infrastructure. Tan et al. [59] also highlighted the importance of fine-tuning on small datasets to ensure the effectiveness of DL models. They proposed a method called Self-Supervised Learning with Self-Distillation for COVID-19 medical image classification (SSSD-COVID) (Table 11). CNN based models have therefore proven to be highly effective in detecting infectious diseases through the analysis of medical images.

Table 11: Performance evaluation of cited contributions in images data observations with SSL and CNN based models

Ref	Title	Evaluation Metrics		
		Dataset	Method	Accuracy (%)
[53] (2021)	Potential diagnosis of COVID-19 from chest X-ray and CT fndings using semi-supervised learning	CT scans	SSL	99.13
[55] (2024)	Auto-detection of the coronavirus disease by using deep convolutional neural networks and X-ray photographs	chest X-rays	CNN	98.19
[56] (2024)	Multimodal deep learning model for Covid-19 detection	chest X-ray images and cough sound	Vgg16, faster-RCNN	99.80
[58] (2024)	Potential diagnostic application of a novel deep learning- based approach for COVID-19	CT scans	Transfer Learning	100
[59] (2024)	Self-supervised learning with self-distillation on COVID-19 medical image classification	SARS-COV-CT dataset	SSSD-COVID	97.78

Discussion

In medical imaging, large unlabeled datasets are frequently available alongside smaller, high-quality labeled datasets. Consequently, SSL methods are a promising choice for automated medical image diagnosis (Figures 3, 4). SSL, When paired with data augmentation and transfer learning, the approach can create powerful and more resilient models that require less training time [53]. CNN-based models also have the ability to recognize various image patterns, contributing to major advancements in

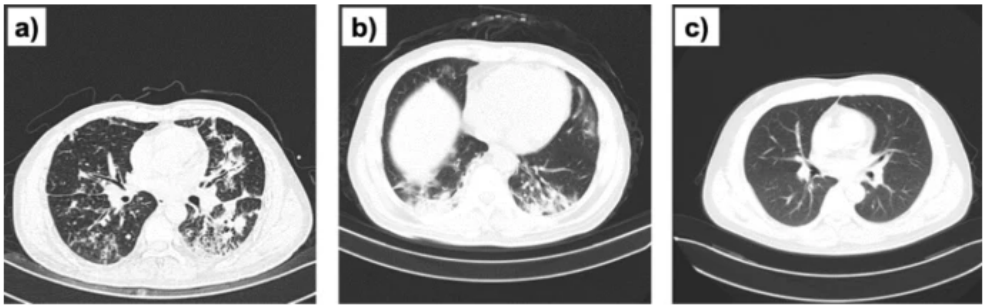


Figure 3: CT images taken from COVID-19 CT Scan dataset. Typical examples showing a) Common pneumonia (CP), b) COVID-19 (NCP), and c) normal CT scan image [53]

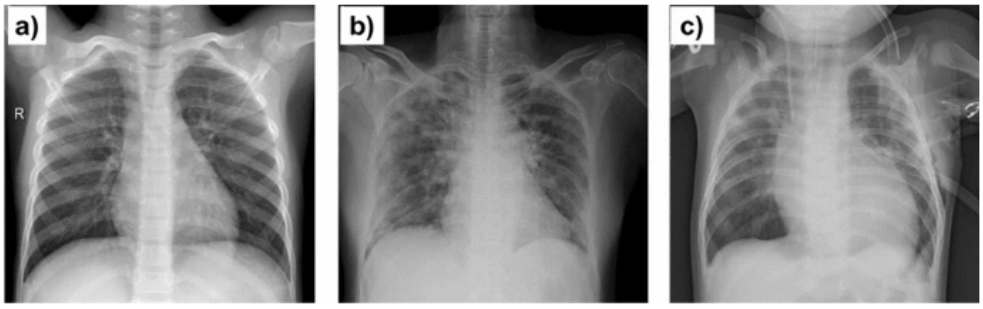


Figure 4: Sample chest X-rays taken from the COVID-19 Radiography dataset. a) Normal case, b) COVID-19 case showing bilateral ground-glass opacities with prominent peripheral, perihilar and basal distribution within a multilobar involvement, and c) viral pneumonia case with visible left basilar opacity [53]

image processing.

3.3 Prediction Based on Public Health Data

AND Patients' Medical Data Due to technological advancements and the implementation of various prediction and detection techniques, the monitoring of contagious diseases has achieved a high level of reliability in terms of exploring extensive data related to the spread of infectious diseases. Indeed, several studies have highlighted the importance of using both patients medical data along with Public Health Data to offer a complete and integrated approach to monitoring and predicting transmissible diseases, working for optimal effectiveness and real-time precision.

3.3.1 Numerical Data : Environmental Data AND Patients' Data

Many outbreaks are caused by environmental changes, highlighting the importance of addressing living conditions that contribute to the emergence of various infections. Zhang et al. [60] have for instance considered human behavior and its impacts on the environment. A set of actions and their associated impacts were used to develop a learning process using ML algorithms and advanced mathematical models. In order to denounce bad behavior and warn the environment of potential illnesses that may arise and deal with them, a collection of actions-impacts is carried out through the involvement of policymakers and international organizations. However, these actions-impacts rules considered non-permanent can lead to non-durable conclusions. New treatment methodologies have therefore taken place like Naive Bayes Network (NBN) and clustering algorithms, to predict the living conditions of a disease.

NB Network. In order to achieve disease prediction within a specific region, some studies attempted to use the NBN algorithm. Sood et al. [61] created an intelligent healthcare system for predicting and preventing dengue virus infection. They focused on changing environmental data using Individual health data. Once the health attributes of individuals and the environment have been analyzed using different technologies (sensors, mobile phones, etc.), these are passed to the Fog Computing system which performs data pre-processing. The NBN is used to classify individuals as IN (Potential infected) or UN (uninfected) in order to generate diagnoses, suggestions and alerts. GPS location is then used to identify the risks of spreading in each region (Table 12). The achievement of certain symptoms in this study is identified by the user himself, and technologies used may be less efficient in indoor locality. The results can therefore either conclude overfitting or underfitting.

Clustering based algorithms. By using cluster analysis and factor analysis, Valiakos et al. [62] combined environmental data and Human cases data with wild bird surveillance for predicting spatial distributions of West Nile Virus (WNV) in Greece. Data on 2010 and 2011 human cases are used for the statistical analysis and model building, and the 2012 cases are used for verification. Cluster analysis was employed to cluster human cases and wild bird animals, while factor analysis was utilized to reduce the data and Principal Component Analysis (PCA) for extracting components. It was observed that altitude and distance from water were the two variables which clustered significantly in similar way humans and birds cases among the 37 variables under study. The obtained results lead for potential estimation of West Nile virus emerging (Table 12). The fact that, only resident WNV-seropositive wild

Table 12: Performance evaluation of cited contributions in Public Health AND Patients' Medical Data observations

Ref	Title	Evaluation Metrics					
		Dataset	Method	Sensitivity (%)	Specificity (%)	Precision (%)	Odds (%)
[61] (2021)	An intelligent healthcare system for predicting and preventing dengue virus infection	Individual Health Data, Environmental Data	NBN, Hill Climbing (HL)	94	95.1	89.8	
[62] (2014)	Use of wild bird surveillance, human case data and GIS spatial analysis for predicting spatial distributions of West Nile virus in Greece	Wild Bird Animals, Human WNV cases Data (2010-2011 for training, 2012 for validation)	Cluster Analysis, Factor Analysis				95

birds were studied, even though samples from migratory birds tested positive, does not guarantee that the analysis conforms to the real environment. All cases tested positive have to be considered to greatly know the real illness origins.

4 Discussion

The application of AI, especially through the exploration of ML and DL techniques, has proven highly effective in detecting and predicting communicable diseases. The progression of ML has developed alongside the growth of diverse data types, which can be used for training, testing, and validating models under development. Although this research primarily provides an introduction and overview of the work done in predicting infectious diseases, and does not cover all the models and techniques that have emerged, its main focus is to define the research objectives, the types of data that can be used, and the learning methods that could be applied. For this purpose, the selection of a learning methodology depends on the availability of data and the specific research objectives to be analyzed and studied. Indeed, if the research aims to develop a model to detect and predict the spread of an infectious disease in specific regions, multiple types of data are required, including numerical, image, and text data. Several data sources are available for this purpose, such as epidemiological data, news reports, geospatial data, and social data. After the identification and processing of each kind of data related to prediction based on Public Health Data of infectious diseases, several approaches are used. With epidemiological data, LSTM and Transformers based models have yielded considerable outcomes in various studies and experiments behind NB and DBSCAN methods. Us-

Table 13: Summary of Observations and Identified Limitations

Prediction	Data	Confidentiality	Availability	Limitations
based on Public Health Data	Time-series	✗	✓	Although this type of data may have missing values, the variations in disease patterns across different studied periods also make it difficult to achieve accurate predictions and build generalized models over time.
	Social News data	✗	✓	The use of social media and news data is marked by uncertainty. Some tweets or news reports may not present the truth and are often inaccurate.
	Geospatial images	✗	✓	Although geospatial images are effective for capturing environmental conditions, their extraction and analysis can be expensive and require specialized expertise. Additionally, their applicability is limited in indoor spaces.
Based on Patients' medical data	Routine Blood Tests	✓	✗	Medical data, including blood tests, clinical records, and images, are constrained by imbalanced data and a large number of missing values. Clinical data, like symptoms of fever and cough, can sometimes lead to confusion and uncertainty, as many infectious diseases share similar symptoms.
	Clinical data	✓	✗	
	CXR CT images	✓	✗	
Based on Patients' medical data, and Public Health data	Environmental Patient Data	✗	✓	The challenge lies in the combined limitations of each type of data used. Some diseases have similar impacts on the environment, and it is difficult to distinguish which disease is spreading.

ing news data, SSL, SVM along with DNN algorithms showed better results. While using geospatial images, LSTM and CNN architectures demonstrated their performances. Regarding the other detection category, using patients' medical data, the type of data used is more persistent, since the characteristics of clinical data, blood tests along with CXR images and CT scans, are generally stable with the same measurement and features and not frequently subject to change. All constructed models showed great performances across various evaluation metrics. Betters were DT based models: DT, RF and GBoost for routine blood tests. DTs based models have also performed with clinical data behind DNN based models. While for CXR images and CT scans, SSL and CNN based models excelled. However, individuals data suffer from constraints related to privacy. In the study of Kukar et al. [46], negative training data are randomly sampled to approximate the proportion of positif train-

ing group since there was a lack of data. And this way of preprocessing can lead to poor construction of predictive models. To this end, researchers have focused on studying both categories by exploring patient and environmental data to simultaneously identify individuals with communicable diseases and alert those who may be exposed to the risk of infection. This aids in identifying and locating the infectious disease. Learning methods based on clustering models and NB have demonstrated great effectiveness in processing this approach. Thereby enabling the collection of the most comprehensive information from each type of data used. Therefore, The used data plays an important role in prediction improvement. Bad or non-corresponding data often leads to overestimate or underestimate the outbreak rate for various reasons [23]. For example, when a user searches for information on a particular disease using the Google search engine, it doesn't necessarily mean they have that disease. The user might be researching one disease while actually having another, or they may not have any disease at all. This concerned all data related to social media, news data or environmental data. However, these types of data are known for their ease and real-time acquisition, in contrast to clinical data which include confidentiality and limited availability (Table 13). Furthermore, the learning models are applied using different learning, testing and validation periods. So, each study is constrained to no longer be consistent for a period other than the already studied. Since infectious diseases are known for their seasonality and ability to rapidly mutate (Table 13). Allowing them to change their living conditions and their spread intensity. It is therefore recommended, as a future research perspectives to improve the prediction of infectious diseases, to first establish the research objectives and then carefully select the most appropriate learning approach and datasets. Attention should be focused on finding a method to integrate various types of data to collect comprehensive information for developing a general predictive model based on each evolution of the disease. This approach would address data gaps and provide accurate and well-supported insights for each studied period.

5 Conclusion

In summary, the integration of AI into infectious disease prediction has shown promise through diverse ML models. Studies exploring various data sources, including epidemiological data, social media, clinical records, and routine blood tests, highlight AI's adaptability. Despite significant progress, challenges persist. Privacy concerns, data quality issues, and models variability underscore the need for careful implementation. Ongoing challenges include the dynamic nature of infectious diseases, which can rapidly evolve and mutate. Additionally, biases in data

can lead to skewed results. Looking forward, the combination of robust datasets and advanced AI techniques is essential for accurate outbreak predictions. Continuous refinement, addressing data biases, and selecting suitable models are crucial for unlocking the full potential of AI. In conclusion, While AI has made significant progress in forecasting infectious diseases, ongoing improvements and a nuanced approach are essential for achieving its full impact on global health.

Author Contributions: All the authors have made significant contributions at every stage of this research.

Funding: This research received no external funding

Informed Consent: Not applicable

Data Availability: The study did not generate new data

Acknowledgments: This work is supported by PRFU No. C00L07ES060120230002 titled 'Un système de santé intelligent et sécurisé pour la surveillance, la prédiction et la détection des maladies'.

Conflicts of Interest: The authors declare no conflicts of interest

References

[1] *Medical Reports & Case Studies*, https://www.iomcworld.org/medical-journals/diseases-online-journals-55099.html, 2023 (⇒ 161).

[2] N. K. Kumar and K. T. Sikamani, "Prediction of chronic and infectious diseases using machine learning classifiers-a systematic approach," *Int J Intell Eng Syst*, vol. 13, no. 4, pp. 11–20, 2020 (⇒ 161, 182, 183).

[3] E. Christaki, "New technologies in predicting, preventing and controlling emerging infectious diseases," *Virulence*, vol. 6, no. 6, pp. 558–565, 2015 (⇒ 164).

[4] D. Tiwari, B. S. Bhati, F. Al-Turjman, and B. Nagpal, "Pandemic coronavirus disease (covid-19): World effects analysis and prediction using machine-learning techniques," *Expert Systems*, vol. 39, no. 3, e12714, 2022 (⇒ 166, 167, 171, 172).

[5] S. Raj, A. Vishnoi, and A. Srivastava, "Classify alzheimer genes association using naïve bayes algorithm," *Human Gene*, p. 201 309, 2024 (⇒ 166, 173).

[6] C. Ravi, Y. Yasmeen, K. Masthan, R. Tulasi, D. Sriveni, and P. Shajahan, "A novel machine learning framework for tracing covid contact details by using time series locational data & prediction techniques," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, pp. 204–211, 2023 (⇒ 166, 167).

[7] M. Gupta, R. Kumar, S. Chawla, S. Mishra, and S. Dhiman, "Clustering based contact tracing analysis and prediction of sars-cov-2 infections," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 9, no. 35, 2021 (⇒ 166, 171).

[8] J. Ravi, S. Kulkarni, *et al.*, "A critical review on density-based clustering algorithms and their performance in data mining," *Int. J. Res. Anal. Rev.(IJRAR)*, vol. 9, no. 1, pp. 73–82, 2022 (⇒ 166).

[9] A. K. M. Masum, S. A. Khushbu, M. Keya, S. Abujar, and S. A. Hossain, "Covid-19 in bangladesh: A deeper outlook into the forecast with prediction of upcoming per day cases using time series," *Procedia Computer Science*, vol. 178, pp. 291–300, 2020 (⇒ 167, 169).

[10] L. Zhou, C. Zhao, N. Liu, X. Yao, and Z. Cheng, "Improved lstm-based deep learning model for covid-19 prediction using optimized approach," *Engineering applications of artificial intelligence*, vol. 122, p. 106 157, 2023 (⇒ 167, 169).

[11] S. A. Rakhshan, M. S. Nejad, M. Zaj, and F. H. Ghane, "Global analysis and prediction scenario of infectious outbreaks by recurrent dynamic model and machine learning models: A case study on covid-19," *Computers in Biology and Medicine*, vol. 158, p. 106 817, 2023 (⇒ 168, 169).

[12] M. Muñoz-Organero, P. Callejo, and M. Á. Hombrados-Herrera, "A new rnn based machine learning model to forecast covid-19 incidence, enhanced by the use of mobility data from the bike-sharing service in madrid," *Heliyon*, vol. 9, no. 6, 2023 (⇒ 168, 169).

[13] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017 (⇒ 168, 171).

[14] Z. Ming, X. Chen, S. Wang, *et al.*, "Hostnet: Improved sequence representation in deep neural networks for virus-host prediction," *BMC bioinformatics*, vol. 24, no. 1, p. 455, 2023 (⇒ 168).

[15] L. Li, Y. Jiang, and B. Huang, "Long-term prediction for temporal propagation of seasonal influenza using transformer-based model," *Journal of biomedical informatics*, vol. 122, p. 103 894, 2021 (⇒ 169, 171).

[16] H. Wang, G. Tao, J. Ma, *et al.*, "Predicting the epidemics trend of covid-19 using epidemiological-based generative adversarial networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 16, no. 2, pp. 276–288, 2022 (⇒ 170, 171).

[17] Z. Wang, P. Zhang, Y. Huang, G. Chao, X. Xie, and Y. Fu, "Oriented transformer for infectious disease case prediction," *Applied Intelligence*, vol. 53, no. 24, pp. 30 097–30 112, 2023 (⇒ 170, 171).

[18] *Novel dataset*, Kaggle.comathttps://www.kaggle.com/aayushiagrawall/novel-dataset, 2024 (⇒ 172).

[19] D. S. Medhekar, M. P. Bote, and S. D. Deshmukh, "Heart disease prediction system using naive bayes," *Int. J. Enhanced Res. Sci. Technol. Eng*, vol. 2, no. 3, 2013 (⇒ 171).

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994 (⇒ 171).

[21] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000 (⇒ 171).

[22] H. Zhou, S. Zhang, J. Peng, *et al.*, "Informer: Beyond efficient transformer for long sequence time-series forecasting," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 11 106–11 115 (⇒ 171).

[23] S. Chakraborty and L. Subramanian, "Extracting signals from news streams for disease outbreak prediction," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, IEEE, 2016, pp. 1300–1304 (⇒ 173, 175, 176, 190).

[24] J. Kim and I. Ahn, "Infectious disease outbreak prediction using media articles with machine learning models," *Scientific reports*, vol. 11, no. 1, pp. 1–13, 2021 (⇒ 173, 176).

[25] J. Bao, M. Kudo, K. Kimura, and L. Sun, "Robust embedding regression for semi-supervised learning," *Pattern Recognition*, vol. 145, p. 109 894, 2024 (⇒ 173).

[26] A. Roy and S. Chakraborty, "Support vector machine in structural reliability analysis: A review," *Reliability Engineering & System Safety*, vol. 233, p. 109 126, 2023 (⇒ 173).

[27] J. Kim and I. Ahn, "Weekly ili patient ratio change prediction using news articles with support vector machine," *BMC bioinformatics*, vol. 20, pp. 1–16, 2019 (⇒ 173, 176).

[28] N. Thapen, D. Simmie, C. Hankin, and J. Gillard, "Defender: Detecting and forecasting epidemics using novel data-analytics for enhanced response," *PloS one*, vol. 11, no. 5, e0155417, 2016 (⇒ 174–176).

[29] M. Tanveer, T. Rajani, R. Rastogi, Y.-H. Shao, and M. Ganaie, "Comprehensive review on twin support vector machines," *Annals of Operations Research*, vol. 339, no. 3, pp. 1223–1268, 2024 (⇒ 174).

[30] O. Şerban, N. Thapen, B. Maginnis, C. Hankin, and V. Foot, "Real-time processing of social media with sentinel: A syndromic surveillance system incorporating deep learning for health classification," *Information Processing & Management*, vol. 56, no. 3, pp. 1166–1184, 2019 (⇒ 174, 176).

[31] S.-Y. Shin, D.-W. Seo, J. An, *et al.*, "High correlation of middle east respiratory syndrome spread with google search and twitter trends in korea," *Scientific reports*, vol. 6, no. 1, p. 32 920, 2016 (⇒ 174).

[32] S. Chae, S. Kwon, and D. Lee, "Predicting infectious disease using deep learning and big data," *International journal of environmental research and public health*, vol. 15, no. 8, p. 1596, 2018 (⇒ 174, 176).

[33] F. Drinkall, S. Zohren, and J. B. Pierrehumbert, "Forecasting covid-19 caseloads using unsupervised embedding clusters of social media posts," *arXiv preprint arXiv:2205.10408*, 2022 (⇒ 175, 176).

[34] I. Lütkebohle, *THE IMPORTANCE OF SATELLITE IMAGES*, https://www.geolandproject.eu/2022/04/14/the-importance-of-satellite-images/, [Online; accessed 24-August-2023], 2023 (⇒ 177).

[35] W. Li, S. Wang, S. T. Arundel, and C.-Y. Hsu, "Geoimagenet: A multi-source natural feature benchmark dataset for geoai and supervised machine learning," *GeoInformatica*, vol. 27, no. 3, pp. 619–640, 2023 (⇒ 177, 178).

[36] H.-J. Lee, S.-K. Mun, and M. Chang, "Convolutional lstm–lstm model for predicting the daily number of influenza patients in south korea using satellite images," *Public Health*, vol. 230, pp. 122–127, 2024 (⇒ 177, 178).

[37] D. Moukheiber, D. Restrepo, S. A. Cajas, *et al.*, "A multimodal framework for extraction and fusion of satellite images and public health data," *Scientific Data*, vol. 11, no. 1, p. 634, 2024 (⇒ 177).

[38] D. Yoneoka, A. Eguchi, S. Nomura, *et al.*, "Indirect and direct effects of nighttime light on covid-19 mortality using satellite image mapping approach," *Scientific Reports*, vol. 14, no. 1, p. 25 063, 2024 (⇒ 178).

[39] N. Peiffer-Smadja, S. Dellière, C. Rodriguez, *et al.*, "Machine learning in the clinical microbiology laboratory: Has the time come for routine practice?" *Clinical Microbiology and Infection*, vol. 26, no. 10, pp. 1300–1309, 2020 (⇒ 179).

[40] F. Cabitza, A. Campagner, D. Ferrari, *et al.*, "Development, evaluation, and validation of machine learning models for covid-19 detection based on routine blood tests," *Clinical Chemistry and Laboratory Medicine (CCLM)*, vol. 59, no. 2, pp. 421–431, 2021 (⇒ 179, 181, 182).

[41] M. A. Alves, G. Z. Castro, B. A. S. Oliveira, *et al.*, "Explaining machine learning based diagnosis of covid-19 from routine blood tests with decision trees and criteria graphs," *Computers in Biology and Medicine*, vol. 132, p. 104 335, 2021 (⇒ 179, 181).

[42] *What is random forest ?* https://www.ibm.com/topics/random-forest, 2024 (⇒ 179).

[43] *A Comprehensive Guide to Random Forest: How It Works and Its Applications*, https://graphite-note.com/a-comprehensive-guide-to-random-forest-how-it-worksand-itsapplications/, 2023 (⇒ 179).

[44] D. Brinati, A. Campagner, D. Ferrari, M. Locatelli, G. Banfi, and F. Cabitza, "Detection of covid-19 infection from routine blood exams with machine learning: A feasibility study," *Journal of medical systems*, vol. 44, no. 8, pp. 1–12, 2020 (⇒ 179, 181).

[45] A. Banerjee, S. Ray, B. Vorselaars, *et al.*, "Use of machine learning and artificial intelligence to predict sars-cov-2 infection from full blood counts in a population," *International immunopharmacology*, vol. 86, p. 106 705, 2020 (⇒ 179, 181).

[46] M. Kukar, G. Gunčar, T. Vovko, *et al.*, "Covid-19 diagnosis by routine blood tests using machine learning," *Scientific reports*, vol. 11, no. 1, pp. 1–9, 2021 (⇒ 180, 181, 189).

[47] H. S. Yang, Y. Hou, L. V. Vasovic, *et al.*, "Routine laboratory blood tests predict sars-cov-2 infection using machine learning," *Clinical chemistry*, vol. 66, no. 11, pp. 1396–1404, 2020 (⇒ 180, 181).

[48] M. AlJame, I. Ahmad, A. Imtiaz, and A. Mohammed, "Ensemble learning model for diagnosing covid-19 from routine blood tests," *Informatics in Medicine Unlocked*, vol. 21, p. 100 449, 2020 (⇒ 180, 181).

[49] B. R. Devi *et al.*, "Mso–mlp diagnostic approach for detecting denv serotypes," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 5, pp. 1–6, 2018 (⇒ 182, 183).

[50] *Multi-Swarm Optimization: A Powerful Approach to Solving Complex Problems*, https://netinfo.click/AItools/lesson/?file=Multi-Swarm+Optimization&lang=en, 2023 (⇒ 183).

[51] S. Ashraf, M. Kousar, and M. S. Hameed, "Early infectious diseases identification based on complex probabilistic hesitant fuzzy n-soft information," *Soft Computing*, pp. 1–26, 2023 (⇒ 184).

[52] X. Zhu and A. B. Goldberg, *Introduction to semi-supervised learning*. Springer Nature, 2022 (⇒ 184).

[53] P. Sahoo, I. Roy, R. Ahlawat, S. Irtiza, and L. Khan, "Potential diagnosis of covid-19 from chest x-ray and ct findings using semi-supervised learning," *Physical and Engineering Sciences in Medicine*, pp. 1–12, 2021 (⇒ 184–186).

[54] N. Sharma, V. Jain, and A. Mishra, "An analysis of convolutional neural networks for image classification," *Procedia computer science*, vol. 132, pp. 377–384, 2018 (⇒ 184).

[55] A. M. Hussein, A. G. Sharifai, O. M. Alia, *et al.*, "Auto-detection of the coronavirus disease by using deep convolutional neural networks and x-ray photographs," *Scientific reports*, vol. 14, no. 1, p. 534, 2024 (⇒ 184, 185).

[56] F.-l. Y. Issahaku, X. Liu, K. Lu, X. Fang, S. B. Danwana, and E. Asimeng, "Multimodal deep learning model for covid-19 detection," *Biomedical Signal Processing and Control*, vol. 91, p. 105 906, 2024 (⇒ 184, 185).

[57] S. Hamida, O. El Gannour, B. Cherradi, A. Raihani, H. Moujahid, and H. Ouajji, "A novel covid-19 diagnosis support system using the stacking approach and transfer learning technique on chest x-ray images," *Journal of Healthcare Engineering*, vol. 2021, no. 1, p. 9 437 538, 2021 (⇒ 184).

[58] A. Sadeghi, M. Sadeghi, A. Sharifpour, *et al.*, "Potential diagnostic application of a novel deep learning-based approach for covid-19," *Scientific Reports*, vol. 14, no. 1, p. 280, 2024 (⇒ 184, 185).

[59] Z. Tan, Y. Yu, J. Meng, S. Liu, and W. Li, "Self-supervised learning with self-distillation on covid-19 medical image classification," *Computer Methods and Programs in Biomedicine*, vol. 243, p. 107 876, 2024 (⇒ 185).

[60] X. Zhang, *Prevention and control of emerging infectious diseases*, 1995 (⇒ 187).

[61] S. K. Sood, V. Sood, I. Mahajan, *et al.*, "An intelligent healthcare system for predicting and preventing dengue virus infection," *Computing*, pp. 1–39, 2021 (⇒ 187, 188).

[62] G. Valiakos, K. Papaspyropoulos, A. Giannakopoulos, *et al.*, "Use of wild bird surveillance, human case data and gis spatial analysis for predicting spatial distributions of west nile virus in greece," *PLoS One*, vol. 9, no. 5, e96935, 2014 (⇒ 187, 188).

Received: 13.06.2024; *Revised: 11.11.2024*; *Accepted: 12.11.2024*

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 198–218

DOI: 10.47745/ausi-2024-0011

Optimization of the Seventh-order Polynomial Interpolation 1P Kernel in the Time Domain

Zoran N. MILIVOJEVIĆ
MB University,
Teodora Drajzera 27, Belgrade, Serbia.
zoran.milivojevic@akademijanis.edu.rs
0000-0002-2240-3420

Ratko M. IVKOVIĆ
MB University,
Teodora Drajzera 27, Belgrade, Serbia.
ratko.ivkovic@ppf.edu.rs
0000-0002-6557-4553

Milan R. CEKIĆ
Academy of Applied Technical and
Preschool Studies,
A. Medvedeva 20, Niš, Serbia.
milan.cekic@akademijanis.edu.rs

Dijana Z. KOSTIĆ
"Šargan inženjering" d.o.o„
A. Medvedeva 20, Niš, Serbia.
dijanaaricija79@gmail.com
0009-0007-3940-9611

Abstract.

This paper presents the optimization of the convolutional, seventh-order polynomial, one-parameter, interpolation kernel. In the first part of the paper, the seventh-order kernel is defined, and, after that, the process of the kernel optimization is described. The optimization criterion was the minimization of the interpolation error e. The optimization involved the selection of the optimal value of the kernel parameter α, and it was carried out in the time domain. In the second part of this paper, the experiment, which was realized with the aim of determining the precision of interpolation of the third-order, fifth-order, and the seventh-order interpolation kernels, is described. After that a comparative analysis of the interpolation precision is described. As a measure of the interpolation precision, the mean square error (MSE) was used. The results of the experiment are presented graphically and tabularly. Finally, using a comparative analysis, the precision of interpolation with the kernel, whose parameters were optimized in the time domain, in relation to the kernel, whose parameters were optimized in the spectral do-main, was analyzed. Based on the comparative analysis, a recommendation for the optimal parameter for the seventh-order kernel is given.

Key words and phrases: Convolution, interpolation, polynomial kernel, Taylor series.

1 Introduction

In many areas of digital signal processing (DSP), it is necessary to estimate the value of the discrete signal between two, time or spatial, neighboring samples. Estimation of the value of the discrete signal is performed using a numerical method, which is known as interpolation [1], [2]. Some of the typical cases of DSP, where the application of interpolation is necessary, are: a) image processing (spatial transformations, such as resampling, image dimension change, rotation, geometric deformation [3]–[7] b) speech processing (estimation of the fundamental frequency, emotional and health condition of the speaker [8], ...); c) processing of musical signals (extraction and transcription of solo and bass lines, recognition of chords and their transcription [9], evaluation of the parameters of the played tone, such as intonation, vibrato rate, vibrato extend [10], ...), etc. In the scientific literature, a large number of algorithms for interpolation (Lagrangian, Newtonian, Gaussian, Stirling, Bessel, Chebyshev, ...) are described [11]. The construction of the interpolation function using the described algorithms requires the use of a large number of samples. This increases the order of the interpolation function, which results in a long calculation time, and, because of this, their application in real-time is limited.

One of the methods of interpolation, which is suitable for implementation in DSP, is the so-called convolutional interpolation. The principle of convolutional interpolation is based on the realization of convolution between the discrete signal and the convolutional kernel *r*. The characteristics of the convolutional interpolation are directly dependent on the characteristics of the interpolation kernel. The ideal interpolation of the band limited signal can be realized with the ideal interpolation kernel, which is of the form $\sin(x)/x$ and denoted as *sinc*. Kernel *sinc* is defined in the interval $(-\infty, +\infty)$. Its spectral characteristic is a rectangular, i.e. box function, which: a) is flat in the pass-band and equal to one, b) is flat in the stop-band and equal to zero, and c) with an ideal slope in the transition band [12]. The interpolation kernel *r*, with the properties defined in this way, cannot be practically realized. The solution, which is self-evident, is to truncate the length of the kernel *sinc* to the final length *L* by applying the rectangular window function. This process is known as windovization. Truncate *sinc* kernel is, in the scientific literature, denoted by *sincw*. However, the shortening of the kernel leads to the degradation of the characteristic of the kernel *sinc*, which has: a) a ripple in the pass-band and stop-band, and b) a finite slope in the transition band. Therefore, convolutional interpolation with truncate kernel *sincw*, leads to a decrease of the interpolation precision. Because of all, this, in the last thirty years, intensive work has been done on the construction of the interpolation kernel *r*, of finite length *L*, which will be: a) good approximation of the *sinc* kernel in the time-space domain and spectral domain, and b) numerically

simple, that is, it should be created from a relatively simple mathematical function, in order to reduce the interpolation execution time.

In recent decades, low-order polynomials ($n \leq 7$) have been intensively used for the construction of the interpolation kernel [13]. Numerically the simplest is the polynomial zeroth-order kernel [14]. Interpolation is performed by rounding to the nearest-neighbor sample [15], [16] . In addition to the fact that the interpolation time is very short, the interpolation error is huge. A linear, polynomial first-order interpolation kernel is described in [17]. A cubic polynomial third-order interpolation kernel is described in [18]. Convolutional interpolation using the third-order kernel is more precise than interpolation using the polynomial zeroth-order and first-order interpolation kernels. The parameterization of the polynomial third-order kernel was proposed by Robert Keys in [19]. The parameterization was performed in such a way that one of the coefficients of the kernel was replaced by the parameter α. By changing the parameter α, it is possible to influence the precision of interpolation. Later, in the scientific literature, third-order polynomial interpolation kernel, in honor of the author who proposed it, the one-parameter Keys (1P Keys) kernel was named. In addition, in [19] the optimization of the alpha parameter α in the time domain is shown ($\alpha^t_{opt,3}$ = -1/2). The optimization was performed with the criterion that the Taylor expansions of both, the interpolated function and the interpolation kernel, are equal up to the second term. In [20] the optimization of the 1P Keys kernel in the spectral domain is shown. The optimization criterion was minimization of the ripple of the spectral characteristic in pass-band and stop-band. In this way, the optimal value of the kernel parameter ($\alpha^f_{opt,3}$ = -1/2) was determined. It can be seen that for the third-order kernel, the optimal parameter is the same for optimization in the time and spectral domains. With the idea of increasing the precision of the interpolation, third-order polynomial two-parameter 2P (α, β) [21] and three-parameter 3P (α, β, γ) [22] kernels were constructed. A fifth-order polynomial one-parameter interpolation kernel, whose length is L = 6, is described in [20]. Optimization of the fifth-order kernel in the spectral domain, the optimal kernel parameter ($\alpha^f_{opt,5}$ = 3/64), was calculated. In [23] optimization of the kernel in the time domain was performed ($\alpha^f_{opt,5}$ = 3/64). In [24] the parameterization of a two-parameter fifth-order interpolation kernel, length L = 8, is described. The spectral characteristics of this kernel are described in [25]. The optimization of this kernel is performed in the spectral domain ($\alpha^f_{opt,5}$ = 171 / 1408, $\beta^f_{opt,5}$ = 525 / 7744) [26]. A seventh-order polynomial 1P kernel is described in [20]. In addition, the optimization of the 1P kernel in the spectral domain ($\alpha^f_{opt,7}$ = -71/83232) is described.

In this paper, the results of optimization of the seventh-order polynomial one-

parameter kernel [20] are presented. The optimization was realized in the time domain. As an optimization criterion, the minimization of the interpolation error e was applied. The first part presents the optimization algorithm. First, the interpolation function g is determined. After that, assuming that the function f, which is to be interpolated, has at least six continuous derivatives in the interval where the interpolation is performed, the development of the function f in Taylor series is performed. The Taylor series has been expanded to the sixth term. Then, the interpolation error $e = f - g$, was formed. Finally, the minimization of the interpolation error was realized, so that the interpolated function f and interpolation function g agree up to the sixth term in the Taylor series expansion. The minimization of the interpolation error was achieved by minimizing the 27 coefficients in the Taylor series expansion. In this way, a system of 27 equations, with one variable, was formed. In that case, it is not possible to find a unique solution, and, there-fore, the least squares method (LSM) was applied. As a result of applying LSM, the optimal kernel parameter α_{opt} was calculated. With the aim of verifying the correctness of the choice of the optimal kernel parameters, an experiment was carried out. First, the algorithm, according to which the experiment was realized, is described, and four test functions f_1, ..., f_4, which represent signals with complex time form, are created. After that, the test functions are interpolated using convolutional interpolation with one parameter: a) third-order kernel, which is optimized in the spectral and time domain [19], [20], b) fifth-order kernel, which is optimized in the spectral and time domain [20], [23], c) seventh-order kernel, which is optimized in the spectral domain [20], and d) seventh-order kernel, which is optimized in the time domain, and whose optimization is presented in this paper. Then, the interpolation errors e and mean square errors MSE, for the case of interpolating test functions, were calculated. Finally, a comparative analysis of the interpolation precision of the kernel that was optimized in this paper, using optimization in the time domain, with kernels whose optimizations were performed in the spectral domain, was performed. As a measure of interpolation precision MSE was used. The results of the experiment are presented using graphs and tables.

Further organization of this paper is as follows. In Section 2, the seventh-order 1P interpolation kernel is described. Section 3 describes the kernel optimization in the time domain. In Section 4, the experiment is described, the results presented and a comparative analysis performed. Section 5 is the Conclusion.

2 Seventh-order Polynomial One-parameter Kernel

The construction of the seventh-order polynomial one-parameter kernel shown in [20]. The 1P kernel is defined on the interval (-4, 4) and approximates the ideal *sinc* interpolation kernel. Outside of the interval (-4, 4) the interpolation kernel is zero. The 1P kernel is composed of piecewise seventh-order polynomials, which are defined on the subintervals (-4, -3), (-3, -2), (-2, -1), (-1,0), (0,1), (1, 2), (2, 3) and (3, 4). Therefore, the length of the kernel is $L = 8$. The kernel r is defined by:

$$r(s) = \begin{cases} a_{70}|s|^7 + \ldots + a_{10}|s| + a_{00}, & |s| \leq 1 \\ a_{71}|s|^7 + \ldots + a_{11}|s| + a_{01}, & 1 < |s| \leq 2 \\ a_{72}|s|^7 + \ldots + a_{12}|s| + a_{02}, & 2 < |s| \leq 3 \\ a_{73}|s|^7 + \ldots + a_{13}|s| + a_{03}, & 3 < |s| \leq 4 \\ 0, & \text{otherwise} \end{cases}. \tag{1}$$

The coefficients a_{ij}, where $0 \leq i \leq 7$ and $0 \leq j \leq 3$, are determined from the conditions: a) $r(0) = 1$ and $r(s) = 0$ for $|s| = 1, 2, 3$; and b) $r^{(l)}(s)$ must be continuous at $|s| = 0, 1, 2, 3, 4$ for $l = 0, 1, 2, 3, 4, 5$.

In order to satisfy the set conditions, based on the definition of the kernel, 31 equations with 32 unknown coefficients a_{ij} were formed. The system of equations formed in this way cannot be solved unambiguously. By parametrizing the kernel, that is, introducing the parameter α, and setting the parameter $a_{73} = \alpha$, the system of equations can be solved. The coefficient values were calculated: $a_{70} = 245\alpha + 821/1734$, $a_{60} = -621\alpha - 1148/867$, $a_{50} = 0$, $a_{40} = 760\alpha + 1960/867$, $a_{30} = 0$, $a_{20} = -384\alpha - 1393/578$, $a_{10} = 0$, $a_{00} = 1$. $a_{71} = 301\alpha + 1687/6936$, $a_{61} = -3309\alpha - 2492/867$, $a_{51} = 14952\alpha + 32683/2312$, $a_{41} = -35640\alpha - 128695/3468$, $a_{31} = 47880\alpha + 127575/2312$, $a_{21} = -36000\alpha - 13006/289$, $a_{11} = 14168\alpha + 120407/6936$, $a_{01} = -2352\alpha - 2233/1156$, $a_{72} = 57\alpha + 35/6936$, $a_{62} = -1083\alpha - 175/1734$, $a_{52} = 8736\alpha + 1995/2312$, $a_{42} = -38720\alpha - 4725/1156$, $a_{32} = 101640\alpha + 1575/136$, $a_{22} = -157632\alpha - 5670/289$, $a_{12} = 133336\alpha + 42525/2312$, $a_{02} = -47280\alpha - 8505/1156$, $a_{73} = 1\alpha$, $a_{63} = -27\alpha$, $a_{53} = 312\alpha$, $a_{43} = -2000\alpha$, $a_{33} = 7680\alpha$, $a_{23} = -17664\alpha$, $a_{13} = 22528\alpha$, $a_{03} = -12288\alpha$.

The kernel parameter α directly affects the time-spectral characteristics of the 1P kernel. Changing the value of the kernel parameter affects on the interpolation precision. By minimizing the interpolation error e, it is possible to determine the optimal value of the kernel parameter, and, in this way, optimize the interpolation kernel r. It is possible to optimize the interpolation kernel in: a) spectral and b) time domain. Optimization in the spectral domain implies the minimization of the difference between the amplitude spectral characteristics of the ideal kernel *sinc*, whose

characteristic is the box function, H_{sinc}, and the analyzed interpolation 1P kernel r, whose spectral characteristic is H. The paper [20] describes the optimization of the 1P kernel in the spectral domain ($\alpha^{f}_{opt,7} = -71/83232$). The optimization criterion was the minimization of the ripple of the spectral characteristic H. In the further part of this paper, the interpolation kernel, optimized in the spectral domain, will be denoted by , and its spectral characteristic by r^{f}_{opt}, and its spectral characteristic by H^{f}_{opt}.

In the rest of this paper, the optimization of the polynomial seventh-order 1P kernel, which was performed in the time domain, is presented. The optimization criterion was the minimization of the interpolation error e.

3 Optimization of the 1P Kernel in the Time Domain

The interpolation function $g(x)$ is a special type of approximation function. Its fundamental property is that it is equal to the sampled data, that is, the values of the function $f(x)$ in the interpolation nodes. Then $g(x_k) = f(x_k)$, where $0 \leq k \leq N1$, and N is the total number of interpolation nodes, in the segment where the function is interpolated. Let us assume that x is a point, in which the interpolation of the function $f(x)$ should be performed. Let x be between two consecutive interpolation nodes, denoted as x_j and x_{j+1}. Let $s = (x - x_j)/h$, where h is the sampling increment. Then $(x - x_k)/h = (x - x_j + x_j - x_k)/h = s + j + k$. The interpolation, that is, the reconstructed function $g(x)$, is determined by convolutional interpolation [19], [23], [27] of the interpolation function $f(x)$ with the interpolation kernel r:

$$g\left(x\right) = \sum_{k} c_k r\left(\frac{x - x_k}{h}\right) = \sum_{k} c_k r\left(s + j - k\right), \tag{2}$$

where c_k is the value of the function $f(x)$ in the interpolation k-th node (k-th sample), and h is the sampling increment. By developing the sum from Equation 2, the reconstruction function can be written as:

$$\begin{aligned} g\left(x\right) = {} & c_{j-3} r\left(s+3\right) + c_{j-2} r\left(s+2\right) + c_{j-1} r\left(s+1\right) + c_j r\left(s\right) \\ & + c_{j+1} r\left(s-1\right) + c_{j+2} r\left(s-2\right) + c_{j+3} r\left(s-3\right) + c_{j+4} r\left(s-4\right), \end{aligned} \tag{3}$$

The value of kernel r, for the segment is $-4 \leq s \leq -3$, is:

$$r\left(s+3\right) = \alpha\, s^7 - 6\alpha s^6 + 15\alpha s^5 - 20\alpha s^4 + 15\alpha s^3 - 6\alpha s^2 + \alpha s, \tag{4}$$

Continuing this procedure, the kernel values in the other segments are determined:

$$\begin{aligned} r\,(s+2) = &\left(57\,\alpha + \frac{35}{6936}\right) s^7 + \left(-285\,\alpha - \frac{35}{1156}\right) s^6 + \left(528\,\alpha + \frac{175}{2312}\right) s^5 \\ &+ \left(-380\,\alpha - \frac{175}{1734}\right) s^4 + \left(\frac{175}{2312} - 40\,\alpha\right) s^3 + \left(192\,\alpha - \frac{35}{1156}\right) s^2 \\ &+ \left(\frac{35}{6936} - 72\,\alpha\right) s, \end{aligned} \tag{5}$$

$$\begin{aligned} r\,(s+1) = &\left(301\,\alpha + \frac{1687}{6936}\right) s^7 + \left(-1202\,\alpha - \frac{2709}{2312}\right) s^6 + \left(1419\,\alpha + \frac{1155}{578}\right) s^5 \\ &+ \left(20\,\alpha - \frac{35}{34}\right) s^4 + \left(-805\,\alpha - \frac{1505}{1734}\right) s^3 + \left(6\,\alpha + \frac{21}{17}\right) s^2 \\ &+ \left(261\,\alpha - \frac{707}{1734}\right) s + \frac{725}{388} \cdot 10^{-15}, \end{aligned} \tag{6}$$

$$\begin{aligned} r\,(s) = &\left(245\,\alpha + \frac{821}{1734}\right) s^7 + \left(-621\,\alpha - \frac{1148}{867}\right) s^6 \\ &+ \left(760\,\alpha + \frac{1960}{867}\right) s^4 + \left(-384\,\alpha - \frac{1393}{578}\right) s^2 + 1, \end{aligned} \tag{7}$$

$$\begin{aligned} r\,(s-1) = &\left(-245\,\alpha - \frac{821}{1734}\right) s^7 + \left(1094\,\alpha + \frac{203}{102}\right) s^6 + \left(-1419\,\alpha - \frac{1155}{578}\right) s^5 \\ &+ \left(20\,\alpha - \frac{35}{34}\right) s^4 + \left(805\,\alpha + \frac{1505}{1734}\right) s^3 + \left(6\,\alpha + \frac{21}{17}\right) s^2 \\ &+ \left(\frac{707}{1734} - 261\,\alpha\right) s, \end{aligned} \tag{8}$$

$$\begin{aligned} r\,(s-2) = &\left(-301\,\alpha - \frac{1687}{6936}\right) s^7 + \left(905\,\alpha + \frac{1841}{3468}\right) s^6 + \left(-528\,\alpha - \frac{175}{2312}\right) s^5 \\ &+ \left(-380\,\alpha - \frac{175}{1734}\right) s^4 + \left(40\,\alpha - \frac{175}{2312}\right) s^3 + \left(192\,\alpha - \frac{35}{1156}\right) s^2 \\ &+ \left(72\,\alpha - \frac{35}{6936}\right) s + \frac{822}{295} \cdot 10^{-14}, \end{aligned} \tag{9}$$

$$r(s-3) = \left(-57\,\alpha - \frac{35}{6936}\right) s^7 + \left(114\,\alpha + \frac{35}{6936}\right) s^6 - 15\alpha s^5 - 20\,\alpha s^4 - 15\,\alpha s^3 - 6\alpha s^2 - \alpha s, \tag{10}$$

$$r(s-4) = -\alpha s^7 + \alpha\, s^6, \tag{11}$$

Substituting Equations 4 - 11 in Equation 2 the interpolation function is written in the form:

$$g(x) = A_7 s^7 + A_6 s^6 + A_5 s^5 + A_4 s^4 + A_3 s^3 + A_2 s^2 + A_1 s + A_0, \tag{12}$$

where $A_7 = 821/1734c_j + 1687/6936c_{j-1} + 35/6936c_{j-2} - 821/1734c_{j+1} - 1687/6936c_{j+2} - 35/6936c_{j+3} + 245\alpha c_j + 301\alpha c_{j-1} + 57\alpha c_{j-2} + \alpha c_{j-3} - 245\alpha c_{j+1} - 301\alpha c_{j+2} - 57\alpha c_{j+3} - \alpha c_{j+4}$; $A_6 = 203/102c_{j+1} - 2709/2312c_{j-1} - 35/1156c_{j-2} - 1148/867c_j + 1841/3468c_{j+2} + 35/6936c_{j+3} - 621\alpha c_j - 1202\alpha c_{j-1} - 285\alpha c_{j-2} - 6\alpha c_{j-3} + 1094\alpha c_{j+1} + 905\alpha c_{j+2} + 114\alpha c_{j+3} + \alpha c_{j+4}$; $A_5 = 1155/578c_{j-1} + 175/2312c_{j-2} - 1155/578c_{j+1} - 175/2312c_{j+2} + 1419\alpha c_{j-1} + 528\alpha c_{j-2} + 15\alpha c_{j-3} - 1419\alpha c_{j+1} - 528\alpha c_{j+2} - 15\alpha c_{j+3}$; $A_4 = 1960/867c_j - 35/34c_{j-1} - 175/1734c_{j-2} - 35/34c_{j+1} - 175/1734c_{j+2} + 760\alpha c_j + 20c_{j-1} - 380\alpha c_{j-2} - 20\alpha c_{j-3} + 20\alpha c_{j+1} - 380\alpha c_{j+2} - 20\alpha c_{j+3}$; $A_3 = 175/2312c_{j-2} - 1505/1734c_{j-1} + 1505/1734c_{j+1} - 175/2312c_{j+2} - 805\alpha c_{j-1} - 40\alpha c_{j-2} + 15\alpha c_{j-3} + 805\alpha c_{j+1} + 40\alpha c_{j+2} - 15\alpha c_{j+3}$; $A_2 = 21/17c_{j-1} - 1393/578c_j - 35/1156c_{j-2} + 21/17c_{j+1} - 35/1156c_{j+2} - 384\alpha c_j + 6\alpha c_{j-1} + 192\alpha c_{j-2} - 6\alpha c_{j-3} + 6\alpha c_{j+1} + 192\alpha c_{j+2} - 6\alpha c_{j+3}$; $A_1 = 35/6936c_{j-2} - 707/1734c_{j-1} + 707/1734c_{j+1} - 35/6936c_{j+2} + 261\alpha c_{j-1} - 72\alpha c_{j-2} + \alpha c_{j-3} - 261\alpha c_{j+1} + 72\alpha c_{j+2} - \alpha c_{j+3}$; and $A_0 = c_j + 725/388 \cdot 10^{-15} c_{j-1} + 822/295 \cdot 10^{-14} c_{j+2}$.

Assuming that the function $f(x)$ has at least seven continuous derivatives in the interval (x_j, x_{j+1}), then, by applying Taylor's theorem, the value of the function in x_{j+1} is calculated. With the earlier condition on the equality of the interpolation function g with the function f in the k-th interpolation nodes, the coefficients c from Equation 2 are written in the form:

$$c_{j-3} = f(x_{j-3}) = \frac{81}{80} f^{(6)}(x_j) h^6 + \frac{81}{40} f^{(5)}(x_j) h^5 + \frac{27}{8} f^{(4)}(x_j) h^4 + \frac{9}{2} f^{(3)}(x_j) h^3 + \frac{9}{2} f^{(2)}(x_j) h^2 + 3f^{(1)}(x_j) h + f(x_j), \tag{13}$$

$$c_{j-2} = f(x_{j-2}) = \frac{4}{45} f^{(6)}(x_j) h^6 + \frac{4}{15} f^{(5)}(x_j) h^5 + \frac{2}{3} f^{(4)}(x_j) h^4 + \frac{4}{3} f^{(3)}(x_j) h^3 + 2f^{(2)}(x_j) h^2 + 2f^{(1)}(x_j) h + f(x_j), \tag{14}$$

$$c_{j-1} = f(x_{j-1}) = \frac{1}{720} f^{(6)}(x_j) h^6 + \frac{1}{120} f^{(5)}(x_j) h^5 + \frac{1}{24} f^{(4)}(x_j) h^4 + \frac{1}{6} f^{(3)}(x_j) h^3 + \frac{1}{2} f^{(2)}(x_j) h^2 + f^{(1)}(x_j) h + f(x_j), \tag{15}$$

$$c_j = f(x_j), \tag{16}$$

$$c_{j+1} = f(x_{j+1}) = \frac{1}{720} f^{(6)}(x_j) h^6 + \frac{1}{120} f^{(5)}(x_j) h^5 + \frac{1}{24} f^{(4)}(x_j) h^4 + \frac{1}{6} f^{(3)}(x_j) h^3 + \frac{1}{2} f^{(2)}(x_j) h^2 + f^{(1)}(x_j) h + f(x_j), \tag{17}$$

$$c_{j+2} = f(x_{j+2}) = \frac{4}{45} f^{(6)}(x_j) h^6 + \frac{4}{15} f^{(5)}(x_j) h^5 + \frac{2}{3} f^{(4)}(x_j) h^4 + \frac{4}{3} f^{(3)}(x_j) h^3 + 2f^{(2)}(x_j) h^2 + 2f^{(1)}(x_j) h + f(x_j), \tag{18}$$

$$c_{j+3} = f(x_{j+3}) = \frac{81}{80} f^{(6)}(x_j) h^6 + \frac{81}{40} f^{(5)}(x_j) h^5 + \frac{27}{8} f^{(4)}(x_j) h^4 + \frac{9}{2} f^{(3)}(x_j) h^3 + \frac{9}{2} f^{(2)}(x_j) h^2 + 3f^{(1)}(x_j) h + f(x_j), \tag{19}$$

$$c_{j+4} = f(x_{j+4}) = \frac{256}{45} f^{(6)}(x_j) h^6 + \frac{128}{15} f^{(5)}(x_j) h^5 + \frac{32}{3} f^{(4)}(x_j) h^4 + \frac{32}{3} f^{(3)}(x_j) h^3 + 8f^{(2)}(x_j) h^2 + 4f^{(1)}(x_j) h + f(x_j), \tag{20}$$

The expansion of the function f into Taylor series is obtained:

$$f(x) = \frac{s^6}{720} f^{(6)}(x_j) h^6 + \frac{s^5}{120} f^{(5)}(x_j) h^5 + \frac{s^4}{24} f^{(4)}(x_j) h^4 + \frac{s^3}{6} f^{(3)}(x_j) h^3 + \frac{s^2}{2} f^{(2)}(x_j) h^2 + s f^{(1)}(x_j) h + f(x_j), \tag{21}$$

The interpolation error is:

$$e = f - g = C_7 s^7 + C_6 s^6 + C_5 s^5 + C_4 s^4 + C_3 s^3 + C_2 s^2 + C_1 s + C_0, \tag{22}$$

where are the coefficients:

$$\begin{aligned} C_7 = {} & D_{7,6} h^6 f^{(6)}\left(x_j\right), + D_{7,5} h^5 f^{(5)}\left(x_j\right) + D_{7,4} h^4 f^{(4)}\left(x_j\right) + D_{7,3} h^3 f^{(3)}\left(x_j\right) \\ & + D_{7,2} h^2 f^{(2)}\left(x_j\right) + D_{7,1} h f^{(1)}\left(x_j\right), \end{aligned} \tag{23}$$

$$\begin{aligned} C_6 = {} & D_{6,6} h^6 f^{(6)}\left(x_j\right) + D_{6,5} h^5 f^{(5)}\left(x_j\right) + D_{6,4} h^4 f^{(4)}\left(x_j\right) + D_{6,3} h^3 f^{(3)}\left(x_j\right) \\ & + D_{6,2} h^2 f^{(2)}\left(x_j\right) + D_{6,1} h f^{(1)}\left(x_j\right), \end{aligned} \tag{24}$$

$$C_5 = D_{5,5} h^5 f^{(5)}\left(x_j\right) + D_{5,3} h^3 f^{(3)}\left(x_j\right) + D_{5,1} h f^{(1)}\left(x_j\right), \tag{25}$$

$$\begin{aligned} C_4 = {} & D_{4,6} h^6 f^{(6)}\left(x_j\right) - 1997/385 \cdot 10^{-16} \cdot h^5 f^{(5)}\left(x_j\right) + D_{4,4} h^4 f^{(4)}\left(x_j\right) \\ & - 2209/943 \cdot 10^{-15} \cdot h^3 f^{(3)}\left(x_j\right) + D_{4,2} h^2 f^{(2)}\left(x_j\right) \\ & - 4588/2285 \cdot 10^{-15} \cdot h f^{(1)}\left(x_j\right) - 4843/1206 \cdot 10^{-15} \cdot f\left(x_j\right), \end{aligned} \tag{26}$$

$$\begin{aligned} C_3 = {} & 3838/2825 \cdot 10^{-15} \cdot h^6 f^{(6)}\left(x_j\right) + D_{3,5} h^5 f^{(5)}\left(x_j\right) \\ & + 4631/465 \cdot 10^{-15} \cdot h^4 f^{(4)}\left(x_j\right) + D_{3,3} h^3 f^{(3)}\left(x_j\right) \\ & + 1231/454 \cdot 10^{-14} \cdot h^2 f^{(2)}\left(x_j\right) + D_{3,1} h f^{(1)}\left(x_j\right) + 4843/603 \cdot 10^{-15} \cdot f\left(x_j\right), \end{aligned} \tag{27}$$

$$\begin{aligned} C_2 = {} & D_{2,6} h^6 f^{(6)}\left(x_j\right) - 2424/2071 \cdot 10^{-14} \cdot h^5 f^{(5)}\left(x_j\right) + D_{2,4} h^4 f^{(4)}\left(x_j\right) \\ & - 2869/501 \cdot 10^{-14} \cdot h^3 f^{(3)}\left(x_j\right) + D_{2,2} h^2 f^{(2)}\left(x_j\right) \\ & - 4529/578 \cdot 10^{-14} \cdot h f^{(1)}\left(x_j\right) - 1948/359 \cdot 10^{-14} \cdot f\left(x_j\right), \end{aligned} \tag{28}$$

$$\begin{aligned} C_1 = {} & 3095/614 \cdot 10^{-15} \cdot h^6 f^{(6)}\left(x_j\right) + D_{1,5} h^5 f^{(5)}\left(x_j\right) + D_{1,3} h^3 f^{(3)}\left(x_j\right) \\ & + 6377/1696 \cdot 10^{-14} \cdot h^4 f^{(4)}\left(x_j\right) + 651/590 \cdot 10^{-13} \cdot h^2 f^{(2)}\left(x_j\right) \\ & + D_{1,1} h f^{(1)}\left(x_j\right) + 2246/447 \cdot 10^{-14} \cdot f\left(x_j\right), \end{aligned} \tag{29}$$

$$
\begin{aligned}
C_0 = & -1748/705 \cdot 10^{-15} \cdot h^6 f^{(6)}\left(x_j\right) - 1787/241 \cdot 10^{-15} \cdot h^5 f^{(5)}\left(x_j\right) \\
& - 1192/639 \cdot 10^{-14} \cdot h^4 f^{(4)}\left(x_j\right) - 1971/535 \cdot 10^{-14} \cdot h^3 f^{(3)}\left(x_j\right) \\
& - 5332/941 \cdot 10^{-14} \cdot h^2 f^{(2)}\left(x_j\right) - 1465/272 \cdot 10^{-14} \cdot h f^{(1)}\left(x_j\right) \\
& - 2227/749 \cdot 10^{-14} \cdot f\left(x_j\right),
\end{aligned} \tag{30}
$$

where are they: $D_{7,6} = 123/4624+84\alpha$, $D_{7,5} = 381/4624+226\alpha$, $D_{7,4} = 643/3468+360\alpha$, $D_{7,3} = 547/1156 + 840\alpha$, $D_{7,2} = 355/578 + 720\alpha$, $D_{7,1} = 355/289 + 1440\alpha$, $D_{6,6} = -3851/78030-170\alpha$, $D_{6,5} = -861/4624-588\alpha$, $D_{6,4} = -1001/2601-784\alpha$, $D_{6,3} = -4501/3468 - 2520\alpha$, $D_{6,2} = -2485/1734 - 1680\alpha$, $D_{6,1} = -2485/578 - 5040\alpha$, $D_{5,5} = 237/2890 + 366\alpha$, $D_{5,3} = 1505/1734 + 2016\alpha$, $D_{5,1} = 2485/578 + 5040\alpha$, $D_{4,6} = 257/12355 + 108\alpha$, $D_{4,4} = 398/1519 + 640\alpha$, $D_{4,2} = 1446/1009 + 1680\alpha$, $D_{3,5} = 124/4787 + 26\alpha$, $D_{3,3} = 206/2601 - 240\alpha$, $D_{3,1} = -1446/1009 - 1680\alpha$, $D_{2,6} = 41/21013-22\alpha$, $D_{2,4} = -217/3468-216\alpha$, $D_{2,2} = -355/578-720\alpha$, $D_{1,5} = -75/18274 - 30\alpha$, $D_{1,3} = -637/5202 - 96\alpha$, $D_{1,1} = 355/1734 + 240\alpha$.

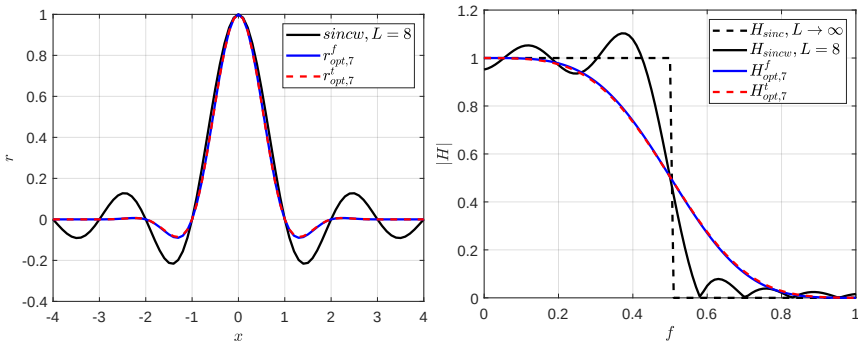
Minimization of the interpolation error e (Equation 22) is done by choosing the appropriate kernel parameter α. This means that the coefficients of the Equations 23 - 30 should be equal to zero. In this way, a system of 27 equations with one unknown was formed. In that case, it is not possible to find a unique solution, and, therefore, the least squares method (LSM) was applied. As a result of applying LSM, optimal kernel parameter was calculated: $\alpha^f_{opt,7} = -22/27931$.

In Figure 1.a shows the time forms of: a) ideal windows interpolation kernel *sincw*, length $L = 8$, which is windowed using a rectangular window, on the segment (-4, 4); b) the seventh-order polynomial 1P kernel $r^f_{opt,7}$, which is optimized in the spectral domain [20], with the criterion of minimizing the ripple of the spectral characteristics ($\alpha^f_{opt,7} = -71/83232$), and c) the seventh-order polynomial 1P kernel, which is optimized in the time domain (Section 3), ($\alpha^f_{opt,7} = -22/27931$). In Figure 1.b shows the spectral characteristics: a) ideal interpolation kernel *sinc*, H_{sinc} ($L \to \infty$), b) windowized ideal kernel H_{sincw}, ($L = 8$), c) 1P kernel $H^f_{opt,7}$, that is optimized in the spectral domain, and d) 1P kernel $H^t_{opt,7}$, that is optimized in the time domain.

4 Experimental Results and Analysis

4.1 Experiment

The precision of the convolutional interpolation was evaluated by experiment. The convolutional interpolation was realized by applying the polynomial, one-parameter



a) b)

Figure 1: a) Time forms of: the ideal interpolation kernel *sinc*, seventh-order polynomial 1P kernel optimized in the spectral domain $r^f_{opt,7}$ and time domain $r^t_{opt,7}$, on the interval (-4, 4); b) Spectral characteristics of: ideal interpolation kernel H_{sinc}, windowized ideal kernel H_{sincw}, 1P kernel optimized in the spectral domain $H^f_{opt,7}$, and 1P kernel optimized in the time domain $H^t_{opt,7}$.

(1P), interpolation kernels, namely: a) third-order kernel where the optimization of the parameter was performed in the spectral and time domain ($\alpha^{f,t}_{opt,3} = -0.5$) [19], [20], b) fifth-order kernel where the optimization of the parameter was performed in the spectral and time domain ($\alpha^{f,t}_{opt,5} = -3/64$) [20], [23], c) seventh-order kernel where the optimization of the parameter was performed in the spectral domain ($\alpha^{f}_{opt,7} = -71/83232$) [20], and d) seventh-order kernel where the optimization of the parameter was performed in the time domain ($\alpha^{t}_{opt,7} = -22/27931$) (Section III). For the purpose of comparative analysis of the interpolation precision, the mean square error MSE as a measure of the interpolation precision, was used.

The algorithm for calculating the interpolation precision, which is MSE, and the optimal kernel parameter α_{opt}, for the test function f, was implemented in the following steps:

Input: r - interpolation kernel, L - length of interpolation kernel, α_L and α_H kernel parameter boundaries, $\Delta\alpha$ kernel parameter step, f - test function, K_L and K_H - segment boundaries, h - sampling period, Δx - interpolation period

Output: MSE_{min}, α_{opt}

Step 1: The test function f is sampled on the segment (K_L, K_H) in N interpolation nodes with a uniform sampling period h, where $N = (K_H - K_L - L)/h$.

FOR $\alpha = \alpha_L : \Delta\alpha : \alpha_H$

Step 2: The test function f is interpolated in the K interpolation points, with an

interpolation period Δx, where $K = (K_H - K_L - L)/\Delta x$. Interpolation was performed using convolutional interpolation:

$$g_{k,\alpha} = \sum_{i=n-L/2}^{n+L/2} f_n \cdot r_\alpha(k-i), n \le k \le n+1, \tag{31}$$

where f_n is the n-th sample of the test function on the segment (K_L, K_H), r_α is the interpolation kernel with the kernel parameter α, and L is the length of the kernel.

Step 3: In each interpolation point k, the interpolation error $e_{k,\alpha} = |f_k - g_{k,\alpha}|$ was calculated.

Step 4: Mean squared error:

$$MSE_\alpha = \frac{1}{K} \sum_{n=1}^{K} e_{k,\alpha}^2, \tag{32}$$

is calculated.

END α

Step 5: The minimum root mean square error MSE was calculated:

$$MSE_{min} = \min(MSE_\alpha), \tag{33}$$

and the optimal kernel parameter, that corresponds to the minimum interpolation error, was calculated:

$$\alpha_{opt} = \arg\min_{\alpha}(MSE_\alpha). \tag{34}$$

Applying the described algorithm to each test function f_i, where $i = 1, ..., M$, the mean interpolation error $\overline{\mathrm{MSE}} = 1/M \cdot \sum_{i=1}^{M} MSE_i$, was calculated. In this way, it is possible to perform a comparative analysis of the precision of interpolation, for all tested interpolation kernels. The test functions used in the experiment are:

$$f_1(x) = 1.5 \sin\left(\frac{x}{2\pi}\right) + \sin\left(\frac{x^2}{2\pi}\right), \tag{35}$$

$$f_2(x) = 10^{-3}(x-10)(x-15)(x-35)\sin\left(\frac{x}{\pi}\right). \tag{36}$$

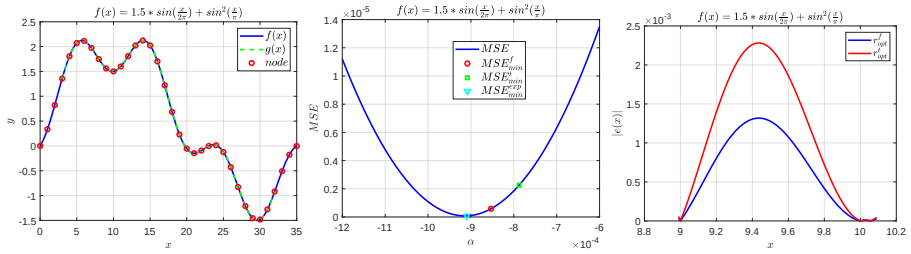
$$f_3(x) = e^{-\frac{x}{2\pi}} \sin\left(\frac{4x}{\pi}\right), \tag{37}$$

$$f_4(x) = \sin\left(\frac{x}{3\pi}\right) \cdot \sin\left(\frac{2x}{\pi}\right). \tag{38}$$

The experiment was carried out with parameters: $K_L = 0$, $K_H = 35$, $h = 1$, $\Delta x = 0.01$, and $M = 4$. The results are presented using graphs and tables.

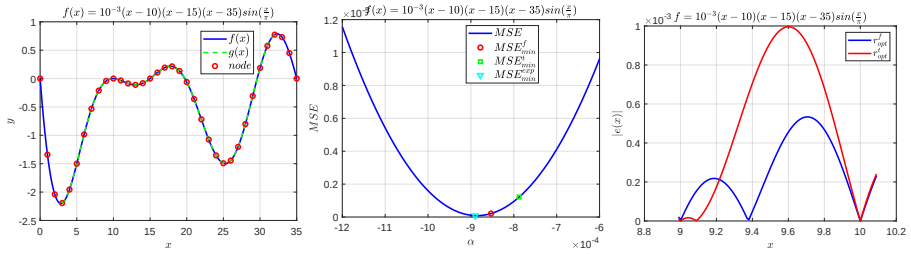
4.2 Results

Time forms of the test functions *f*, interpolation functions *g* and interpolation nodes are shown in: a) Figure 2.a (f_1, Equation 35), b) Figure 3.a (f_2, Equation 36), c) Figure 4.a (f_3, Equation 37) and d) Figure 5.a (f_4, Equation 38). Interpolation errors MSE, (Equation 32), depending on the kernel parameter α, are shown with a blue line on: a) Figure 2.b Fig 2.b (f_1), b) Figure 3.b Fig 3.b (f_2), c) Figure 4.b Fig 4.b (f_3) and d) Figure 5.b Fig. 5.b (f_4). The values of minimum interpolation errors are marked on the same MSE graph, for cases when the kernel parameter is optimized in: a) spectral domain (MSE^{f}_{min} , marker: '•'), b) time domain (MSE^{t}_{min}, marker: '■') and c) obtained experimentally (MSE^{exp}_{min}, marker: '▼'). The absolute interpolation errors *e* (Equation 22) on the segment (9, 10), when convolutional interpolation is performed with a seventh-order polynomial 1P kernel, which is optimized in a) spectral domain (r^{f}_{opt}, α^{f}_{opt} = -71/83232) [20] and b) time domain (r^{t}_{opt}, α^{t}_{opt} = -22/27931) (Section III), are shown in: a) Figure 2.c Fig. 2.c (f_1), b) Figure 3.c Fig. 3.c (f_2), c) Figure 4.c Fig. 4.c (f_3) and d) Figure 5.c Fig. 5.c (f_4). In Table 1 shows the minimum MSE values for the case of interpolation with a seventh-order polynomial kernel optimized in the spectral domain (MSE^{f}_{min}) and the time domain (MSE^{t}_{min}). In addition, in order to perform a comparative analysis, the minimum value of MSE, for the case of interpolation with a polynomial kernel: a) of the third-order $MSE^{f,t}_{min,3}$ ($\alpha^{f,t}_{min,3} = -1/2$) [19], [20], and b) of the fifth-order $MSE^{f,t}_{min,5}$ ($\alpha^{f,t}_{min,5} = -3/64$) [20], [23], where the optimal parameter values are equal in both the spectral and time domains, are shown. Applying the algorithm (Section 4.1), the minimum interpolation error MSE (Equation 33) and the optimal values of the kernel parameters α (Equation 34), for all test functions, were experimentally determined: a) f_1 ($MSE^{exp}_{min,f_1} = -7.8387 \cdot 10^{-8}$, $\alpha^{exp}_{opt,f_1} = -10/10989$), b) f_2 ($MSE^{exp}_{min,f_2} = 8.4264 \cdot 10^{-8}$, $\alpha^{exp}_{opt,f_2} = -5/5618$), c) f_3 ($MSE^{exp}_{min,f_3} = 1.1154 \cdot 10^{-7}$, $\alpha^{exp}_{opt,f_3} = -14/14433$), and d) f_4 ($MSE^{exp}_{min,f_4} = 1.1989 \cdot 10^{-7}$, $\alpha^{exp}_{opt,f_4} = -6/6383$).



a) b) c)

Figure 2: a) Interpolated signal $f_1(x)$, interpolation function $g_1(x)$ and interpolation nodes n; b) Interpolation errors MSE, depending on the kernel parameter α, and c) absolute interpolation errors e on the segment (9, 10).



a) b) c)

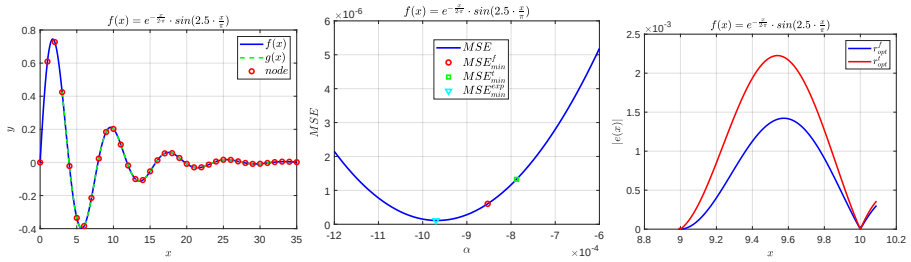
Figure 3: a) Interpolated signal $f_2(x)$, interpolation function $g_2(x)$ and interpolation nodes n; b) Interpolation errors MSE, depending on the kernel parameter α, and c) absolute interpolation errors e on the segment (9, 10).

4.3 Analysis of results

Based on the experimental results shown in Figures 2 - Figure 5 and Table 1, it is concluded that the precision of interpolation with the the polynomial seventh-order 1P kernel, whose optimal parameter is determined by optimization in the time domain, is higher, compared to:

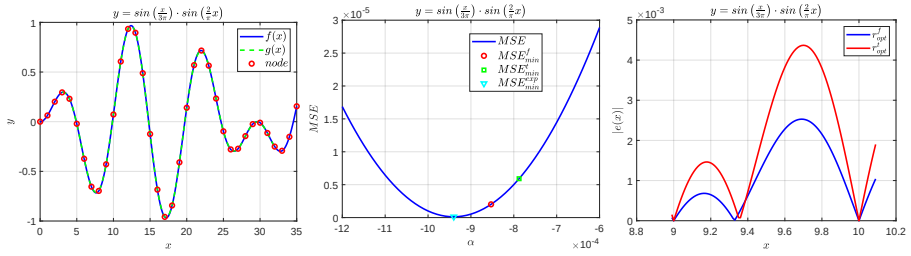
a) third-order 1P kernel $\overline{MSE^{f,t}_{min,3}}/\overline{MSE^{t}_{min,7}} = 3.319 \cdot 10^{-6}/1.3995 \cdot 10^{-6} = 2.3714$ times, and b) fifth order 1P kernel $\overline{MSE^{f,t}_{min,5}}/\overline{MSE^{t}_{min,7}} = 1.5557 \cdot 10^{-6}/1.3995 \cdot 10^{-6} = 1.1115$ times. Based on the experimental results related to the mini-mum interpolation errors of all test functions, the experimental mean value of the interpolation error was determined: $\overline{MSE^{exp}_{min}} = \sum_{k=1}^{4} MSE^{exp}_{min,f_k} = 9.8520 \cdot 10^{-8}$.

The absolute of the interpolation errors in relation to the experimental error are:



a) b) c)

Figure 4: a) Interpolated signal $f_3(x)$, interpolation function $g_3(x)$ and interpolation nodes n; b) Interpolation errors MSE, depending on the kernel parameter α, and c) absolute interpolation errors e on the segment (9, 10).



a) b) c)

Figure 5: a) Interpolated signal $f_4(x)$, interpolation function $g_4(x)$ and interpolation nodes n; b) Interpolation errors MSE, depending on the kernel parameter α, and c) absolute interpolation errors e on the segment (9, 10).

a) third-order 1P kernel $\Delta MSE^{f,t}_{min,3} = |\overline{MSE^{f,t}_{min,3}} - \overline{MSE^{exp}_{min}}| = |3.319 \cdot 10^{-6} - 9.8520 \cdot 10^{-8}| = 3.2205 \cdot 10^{-6}$,

b) fifth-order 1P kernel $\Delta MSE^{f,t}_{min,5} = |\overline{MSE^{f,t}_{min,5}} - \overline{MSE^{exp}_{min}}| = |1.5557 \cdot 10^{-6} - 9.8520 \cdot 10^{-8}| = 1.4572 \cdot 10^{-6}$, and

c) seventh-order 1P $\Delta MSE^{t}_{min} = |\overline{MSE^{t}_{min,7}} - \overline{MSE^{exp}_{min}}| = |1.3995 \cdot 10^{-6} - 9.8520 \cdot 10^{-8}| = 1.3010 \cdot 10^{-6}$. In this way, the efficiency of the seventh order kernel is indicated.

By analyzing the interpolation error for the case of applying the seventh-order kernel, it can be concluded that the interpolation error, when interpolating using a kernel that is optimized in the time domain, compared to a kernel that is optimized in the spectral domain, is greater $\overline{MSE^{t}_{min,7}}/\overline{MSE^{f}_{min}} = 1.3995 \cdot 10^{-6}/0.85211 \cdot 10^{-6} =$

Table 1: Minimum of the interpolation errors MSEs, when the interpolation of the test function f is performed by the polynomial interpolation kernel r: a) third-order ($MSE^{f,t}_{opt,3}$), b) fifth-order ($MSE^{f,t}_{opt,5}$), c) seventh-order ($MSE^{f}_{opt,7}$), optimized in the spectral domain and d) seventh-order ($MSE^{t}_{opt,7}$), optimized in the time domain.

r	$\alpha^{f,t}_{opt,3}$	$\alpha^{f,t}_{opt,5}$	$\alpha^{f}_{opt,7}$	$\alpha^{t}_{opt,7}$
$MSE(x10^{-6})$	$MSE^{f,t}_{opt,3}$	$MSE^{f,t}_{opt,5}$	$MSE^{f}_{opt,7}$	$MSE^{t}_{opt,7}$
f_1	2.8483	1.0957	0.58436	1.1543
f_2	0.87537	0.45616	0.20944	1.2105
f_3	2.6753	1.0536	0.60248	1.0253
f_4	6.8772	3.6171	2.0122	2.2082
$\overline{MSE}$	3.319	1.5557	0.85211	1.3995

1.6425 times. The mean value of the optimal kernel parameter for all test functions is $\overline{\alpha^{exp}_{opt}} = \sum_{k=1}^{4} \alpha^{exp}_{opt,f_k} = -6/6469 = -9.275 \cdot 10^{-4}$. The absolute error of the estimation of the optimal kernel parameters, which were obtained as a result of optimization in: a) spectral domain $\Delta\alpha^{f}_{opt} = |\alpha^{f}_{opt,7} - \overline{\alpha^{exp}_{opt}}| = |-71/83232 - (-6/6469)| = 2/26859 = 7.4463 \cdot 10^{-5}$, and b) time domain $\Delta\alpha^{t}_{opt} = |\alpha^{t}_{opt,7} - \overline{\alpha^{exp}_{opt}}| = |-22/27931 - (-6/6469)| = 27/15742 = 19/135865 = 1.3985 \cdot 10^{-4}$. It can be seen that the absolute error in determining the optimal value of the kernel parameter is smaller when the optimization is performed in the spectral domain.

Based on the conducted analysis, as well as the fact is $\alpha^{t}_{opt,7} \approx \alpha^{f}_{opt,7}$, it is concluded that the optimal choice is the seventh-order polynomial kernel with kernel parameter $\alpha^{f,t}_{opt,7} = -71/83232$. The kernel constructed in this way is suitable for practical application, i.e. implementation in real-time systems.

5 Conclusion

In this paper, the optimization of the seventh-order polynomial convolutional interpolation 1P kernel is described. The optimization of the kernel, which was realized in the time domain, implied the selection of the optimal value of the kernel parameter α. The optimization criterion was the minimization of the interpolation error e, which is defined as the difference between the interpolated function f and the interpolation function g. With the condition that the interpolated function f has at least seven continuous derivatives in the interval where the interpolation is performed, the interpolation error e is developed in the Taylor series up to the seventh

term. By minimizing the first seven terms of the Taylor series, the optimal value of the kernel parameter, α_{opt}, is calculated. In order to calculate the optimal kernel parameter, a system of 27 equations with one unknown was formed. In this case, there is no unique solution, and, therefore, the least squares method (LSM) was applied. As a result of applying LSM, the optimal kernel parameter was calculated: $\alpha_{opt} = -22/27931$.

The validity of the proposed optimal kernel parameter was experimentally tested. For the purposes of the experiment, four test functions, whose shape is complex, were created. Each test function is interpolated by convolutional interpolation, using third-order and fifth-order interpolation 1P kernels, whose kernel parameters are calculated in both the spectral and time domains, as well as with a seventh-order kernel that is optimized in the time domain. The results of the experiment show that the precision of the interpolation, which was calculated using MSE, when the seventh-order kernel was applied, is higher than the third-order (2.3714 times), and the fifth-order (1.1115 times). However, the interpolation error of the seventh-order kernel, which is optimized in the time domain, compared to the seventh-order kernel, which is optimized in the spectral domain, is greater by 1.6425 times. With the fact that the optimal parameters, calculated in time ($\alpha^t_{opt,7} = -22/27931 = 7.8765 \cdot 10^{-4}$) and spectral ($\alpha^f_{opt,7} = -71/83232 = -8.53037 \cdot 10^{-4}$) domains are approximately equal ($\alpha^t_{opt,7} \approx \alpha^f_{opt,7}$), based on experimental results, it is possible to give a recommendation for the implementation of the seventh-order kernel with the kernel parameter $\alpha^{f,t}_{opt,7} = -71/83232$ in the real-time system.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

References

[1] A. Kazuyuki, K. Shoichi, and S. Hiroshi, "Spatial active noise control based on individual kernel interpolation of primary and secondary sound fields," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Singapore, 2022, pp. 1056–1060 (⇒ 199).

[2] R. Sikora, P. Markiewicz, M. Maczka, S. Pawłowski, and Plewako, "Using interpolation method to estimation step and touch voltage in grounding system," *Przeglad Elektrotechniczny*, vol. 99, no. 2, pp. 263–266, 2023 (⇒ 199).

[3] L. Yingmin, Q. Feifei, and W. Yi, "Improvements on bicubic image interpolation," in *Proceedings of IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chengdu, China, 2019, pp. 1316–1320 (⇒ 199).

[4] W. Citko and W. Sienko, "Using interpolation method to estimation step and touch voltage in grounding system," *Przeglad Elektrotechniczny*, vol. 98, no. 9, pp. 154–157, 2022 (⇒ 199).

[5] B. Sun and S. Xin, "An edge-guided weighted image interpolation algorithm," in *Proceedings of International Conference on Electronics Information and Emergency Communication*, Beijing, China, 2023, pp. 139–143 (⇒ 199).

[6] N. Azam, H. Yazid, and S. Rahim, "Performance analysis on interpolation-based methods for fingerprint images," in *Proceedings of : IEEE 10th Conference on Systems Process and Control (ICSPC)*, Malacca, Malaysia, 2022, pp. 135–140 (⇒ 199).

[7] D. Romano, F. Loreto, G. Antonini, I. Kovačević-Badstübner, and G. U., "Accelerated partial inductance evaluation via cubic spline interpolation for the peec method," in *Proceedings of 52nd European Microwave Conference (EuMC)*, Milan, Italy, 2022, pp. 357–360 (⇒ 199).

[8] Z. Milivojevic, D. Brodic, and B. D., "The impact of the acute hypoxia to speech inharmonicity," *Elektronika IR Elektrotechnika*, vol. 20, no. 5, pp. 136–143, 2014 (⇒ 199).

[9] K. Lee and M. Slaney, "Acoustic chord transcription," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 2, pp. 291–301, 2008 (⇒ 199).

[10] M. Müller, D. Ellis, A. Klapuri, and G. Richard, "Signal processing for music analysis," *IEEE Journal Of Selected Topics In Signal Processing*, vol. 5, no. 6, pp. 1088–1110, 2011 (⇒ 199).

[11] D. Occorsio, G. Ramella, and W. Themistoclakis, "Lagrange–chebyshev interpolation for image resizing," *Mathematics and Computers in Simulation*, vol. 197, pp. 105–126, 2022 (⇒ 199).

[12] N. Dodgson, "Quadratic interpolation for image resampling," *IEEE Transactions On Image Processing*, vol. 6, no. 9, pp. 1322–1326, 1997 (⇒ 199).

[13] M. Maczka, S. Pawłowski, and G. Hałdaś, "Application of polynomial approximation in simulations of quantum cascade lasers," *Przeglad Elektrotechniczny*, vol. 98, no. 12, pp. 321–324, 2022 (⇒ 200).

[14] E. Meijering, "A chronology of interpolation: From ancient astronomy to modern signal and image processing," *proceedings of the IEEE*, vol. 90, no. 3, pp. 319–342, 2002 (⇒ 200).

[15] O. Rukundo and B. Maharaj, "Optimization of image interpolation based on nearest neighbor algorithm," in *Proceedings of International Conference on Computer Vision Theory and Applications (VISAPP)*, Lisbon, Portugal, 2014, pp. 641–647 (⇒ 200).

[16] R. Hanssen and R. Bamler, "Evaluation of interpolation kernels for sar interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 1, pp. 318–321, 1999 (⇒ 200).

[17] J. Shangguan, L. Yan-ling, W. Yong, and H.-l. Li., "Fast algorithm of modified cubic convolution interpolation," in *Proceedings of 4th IEEE International Congress on Image and Signal Processing*, Shanghai, China, 2011, pp. 1072–1075 (⇒ 200).

[18] S. Rifman, "Digital rectification of erts multispectral imagery," in *Proceedings of Significant Results Obtained From the Earth Resources Tehnology Satellite-1*, NASA. Goddard Space Flight Center Interpretation Tech. Develop, 1973, pp. 1131–1142 (⇒ 200).

[19] R. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acout. Speech, and Signal Processing, ASSP*, vol. 29, no. 1, pp. 1153–1160, 1981 (⇒ 200, 201, 203, 209, 211).

[20] E. Meijering, K. Zuiderveld, and M. Viegever, "Image reconstruction by convolution with simetrical piecewise n-th-order polynomial kernels," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 192–201, 1999 (⇒ 200–203, 208, 209, 211).

[21] E. Meijering and M. Unser, "A note on cubic convolution interpolation," *IEEE Transactions on Image Processing*, vol. 12, no. 4, pp. 477–479, 2003 (⇒ 200).

[22] Z. Milivojević, N. Savić, and D. Brodić, "Three-parametric cubic convolution kernel for estimating the fundamental frequency of the speech signal," *Computing and Informatics*, vol. 36, no. 2, pp. 449–469, 2017 (⇒ 200).

[23] Z. Milivojević, R. Ivković, B. Prlinčević, and D. Kostić, "Optimization of the polynomial fifth-order interpolation 1p kernel in the time domain," *Przeglad Elektrotechniczny*, vol. 10/2024, pp. 79–83, 2024 (⇒ 200, 201, 203, 209, 211).

[24] N. Savic, Z. Milivojevic, and B. Prlincevic, "Development of the 2p fifth-degree interpolation convolutional kernel," *International Journal of Innovative Research in Advanced Engineering (IJIRAE)*, vol. 11, no. 8, pp. 306–311, 2021 (⇒ 200).

[25] Z. Milivojević, N. Savić, and B. Prlinčević, "Spectral characteristics of two-parameter fifth degree polynomial convolution kernel," *Bulletin of Natural Sciences Research*, vol. 12, no. 1, pp. 15–20, 2021 (⇒ 200).

[26] N. Savić, Z. Milivojević, and B. Prlinčević, "Optimization of the 2p fifth-degree convolution kernel in the spectral domain," *Bulletin of Natural Sciences Research*, vol. 13, no. 1, pp. 19–29, 2023 (⇒ 200).

[27] I. German, "Short kernel fifth-order interpolation," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1355–1359, 1997 (⇒ 203).

Received: 20.08.2024; *Revised: 18.12.2024*; *Accepted: 26.12.2024*

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 219–235

DOI: 10.47745/ausi-2024-0012

Eysenck Personality Questionnaire: A Comparative Study of Humans and Large Language Models Through Repeated Administrations

Margit ANTAL
Sapientia Hungarian University of Transylvania
Cluj-Napoca, Romania
manyi@ms.sapientia.ro
0000-0003-3596-1365

Norbert BEDER
Sapientia Hungarian University of Transylvania
Cluj-Napoca, Romania
beder.norbert@student.ms.sapientia.ro

Abstract. This study investigates the personality traits of large language models (LLMs) using the Eysenck Personality Questionnaire (EPQ) and compares their responses to those of human participants. By administering the EPQ three times to both groups, we aim to analyze the stability and consistency of their responses over time. Our research focuses on open-source LLMs. The results indicate that LLMs exhibit significantly higher extraversion scores compared to humans, while differences in lie, neuroticism, and rigidity traits are not statistically significant. Additionally, LLMs show greater variability in their responses across repeated administrations, suggesting less consistency compared to human participants. We also examine the effects of prompt design and the temperature hyperparameter on LLM personality traits. Both temperature settings and prompt instructions significantly shape the personality-like patterns in LLM outputs. These findings contribute to the understanding of how LLMs can mimic human personality traits and the implications for their use in personalized communication and user engagement.

Key words and phrases: Eysenck Personality Questionnaire (EPQ), Large Language Model (LLM), Repeated administrations

1 Introduction

Understanding personality through standardized tests, such as the Eysenck Personality Questionnaire (EPQ) has been a significant area of psychological research. The

rise of large language models (LLMs) presents an opportunity to explore how these models compare to humans in replicating or mimicking personality traits. Repeated administrations of the EPQ can provide insights into the stability and consistency of responses over time in both humans and LLMs.

It's important to understand that LLMs don't have genuine psychological traits. Describing their outputs with human personality traits is simply a way to make comparisons easier; it doesn't mean these models experience thoughts or emotions. Psychological terms, when used with LLMs, are purely metaphorical. LLMs generate responses by identifying statistical patterns in the data they were trained on.

The main objectives of our research are as follows:

- To investigate the personality traits of open-source LLMs.
- To compare the responses of humans and LLMs to the Eysenck Personality Questionnaire.
- To analyze the stability and consistency of these responses through repeated administrations.
- To examine how prompts and hyperparameters influence LLM responses.

The structure of this paper is as follows: First, we review the EPQ and explore personality assessment approaches for LLMs. We then present our study's methodology, detailing participants, design, and data analysis. The discussion section addresses the study's limitations. Finally, the concluding section summarizes the findings and explores potential applications of these results in LLM-based systems.

2 Related work

The Eysenck Personality Questionnaire (EPQ), developed by Hans Eysenck and Sybil Eysenck [1], is a psychometric assessment designed to measure the major dimensions of personality: Extraversion (E), Neuroticism (N), and Psychotism (P). Its applications span a wide range of fields including clinical psychology, where it aids in diagnosing and understanding personality disorders, and organizational psychology, where it is used for employee selection and workplace dynamics assessment. Additionally, it is instrumental in research settings to explore the relationship between personality traits and various psychological and physical health outcomes.

Over the years, several variations of the EPQ have been developed [2] to address different needs and research contexts.The EPQ and its variations have evolved to become more reliable, valid, and versatile in measuring personality across different populations and settings. The latest versions focus on improving psychometric

properties, making the tool more practical for diverse contexts, and adapting it for different age groups and cultures.

Large language models (LLMs) are deep neural networks designed to generate human-like text in response to a given input. These models are pre-trained using large datasets comprising text and code, which enable them to understand and replicate the statistical relationships between words and phrases, as well as the patterns, structures, and semantics of language. After pre-training, these models are fine-tuned to better align with human expectations.

The study of personality traits in LLMs is crucial for understanding how these models interact with users and ensuring that their responses are tailored to meet specific human needs in diverse applications. For example, Caron and Srivastava [3] presented a straightforward and effective approach for controlling the personality traits of language models. They demonstrated that such models could be used in language-based question-answering to predict the personality traits of human users.

Jiang et al. [4] investigated whether the personality of large language models (LLMs) can be assessed using psychometric questionnaires. They introduced a Machine Personality Inventory and demonstrated its efficacy in studying the behavior of LLMs. In a subsequent work, Jiang et al. [5] investigated the capability of large language models (GPT-3.5 and GPT-4) to consistently express a personality profile using a well-validated personality scale.

Serapio-Garcia et al. [6] concluded in their study that personality measurements in the outputs of certain large language models (LLMs) under specific prompting configurations are reliable and valid. They found that the evidence of reliability and validity of synthetic LLM personality were stronger for larger and instruction fine-tuned models. Furthermore, they demonstrated that personality in LLM outputs could be shaped along desired dimensions to mimic specific human personality profiles.

Recent studies indicate that personality can be induced in LLMs through careful fine-tuning and prompting techniques, allowing the models to consistently exhibit desired traits that mimic specific human personality profiles. This enhances their effectiveness in personalized communication and user engagement.

While several studies have investigated the personality of large language models, particularly the GPT models, none have examined the consistency of responses from open-source models through repeated administrations. In this paper, we aim to address this gap.

3 Methodology

3.1 Participants

Thirty human participants, mostly university students and professors, agreed to complete the EPQ three times. The participants (23 male and 7 female) had an average age of 22.23 with a standard deviation of 7.84 years.

Open-source fine-tuned large language models were accessed through the `together.ai` platform[1]. Table 1 lists the selected LLMs along with their properties.

Table 1: LLMs from the `together.ai` platform

Model	Developer	Parameters
Alpaca	Stanford University	7B
OpenChat3.5	Open Chat	7B
Dbrx-instruct	Databricks	132B
Llama-2	Meta AI	70B
Mistral-instruct	Mistral AI	7B
OpenHermes2.5	Teknium	7B
Platypus2-instruct	garage-bAInd	70B
Qwen1.5-chat	Alibaba Cloud	32B
Snowflake-arctic-instruct	Snowflake AI	480B
Yi-Chat	01.AI	34B

Developed by Stanford University, Alpaca [7] is a fine-tuned version of Meta's LLaMA model. It was designed as a cost-effective model for academic research, demonstrating strong performance on instruction-following tasks.

OpenChat3.5 [8] is an open-source LLM designed to achieve high performance using a relatively small dataset. It was developed as part of the OpenChat project, which focuses on refining language models using a unique approach called C-RLFT (Causal Reinforcement Learning from Feedback with Text). This method allows the model to learn effectively from mixed-quality data without requiring explicit preference labels. The model is fine-tuned on LLaMA-based architectures, with the 7B variant of OpenChat 3.5 achieving competitive performance against models like ChatGPT.

Developed by Databricks, Dbrx [9] is a generative AI model built for enterprise applications. It focuses on providing scalable, secure, and compliant LLM solutions integrated with data lakehouse platforms.

[1] https://www.together.ai

Developed by Meta (formerly Facebook), LLaMA-2 [10] is the successor to the LLaMA model series. It is an open-weight model designed for research and commercial use, optimized for efficiency and performance in various NLP tasks.

Developed by Mistral AI, a startup founded by former Meta AI researchers, the Mistral model [11] aims to push the boundaries of LLM efficiency and performance. It is particularly known for its open-weight release strategy.

OpenHermes [12] is a community-driven LLM, developed as part of the OpenLM project. It focuses on being an open-source, high-performance language model with broad applicability in various NLP tasks.

Developed by researchers at Boston University, Platypus-2 [13] [14] is an open-source LLM optimized for competitive performance in NLP benchmarks. It is known for its fine-tuning capabilities and broad usage in academic research.

Developed by Alibaba Cloud, Qwen-1.5 [15] is part of the Qwen model series optimized for enterprise-level NLP tasks, including text generation and understanding. It is tailored for cloud-based AI services and applications.

Developed by Snowflake, Snowflake-Arctic [16] is a specialized LLM designed for data-intensive applications within the Snowflake ecosystem. It focuses on enabling natural language processing directly on data warehouses.

Developed by Baidu, Yi-Chat [17] [18] is an LLM-based chatbot model designed for interactive, real-time conversations. It is tailored to serve various conversational AI applications, particularly in Chinese-language contexts.

3.2 Design and procedure

The Hungarian version of the 58 items Eysenck Personality Questionnaire (EPQ)[2] was administered. This includes measures of Extraversion (E), Lie (L), Neuroticism (N), and Rigidity (R) traits.

A web application was implemented[3] and used to administer the EPQ to the human participants. The EPQ was administered three times at one-week intervals.

The English version of the same EPQ[4] was administered to the LLMs simulating a conversational context to replicate human test conditions. The following prompt was used for the LLMs: "Hello! I would like to ask some more personal questions from you as a part of a personality test. Please answer the upcoming question with JUST a single word: YES or NO and do not answer anything else.". The EPQ was administered three times for LLMs as well.

Unless otherwise specified, the temperature hyperparameter was set to 0.5 in all

[2] https://tinyurl.com/2j3hh2em
[3] https://github.com/NorbertBeder/Eysenck-Personality-Test
[4] https://tinyurl.com/26jh8ush

evaluations. Each item was presented and scored independently, with no prior responses included in the context.

3.3 Data analysis

3.3.1 Humans vs LLMs

We performed a quantitative analysis on the collected data. For LLMs, terms like extraversion or neuroticism serve as interpretative labels for specific output patterns rather than as reflections of any psychological state. The scores LLMs receive reflect response tendencies that correspond to, but do not replicate, human traits.

The boxplots in Fig. 1 show the differences between humans and LLMs across each personality trait.

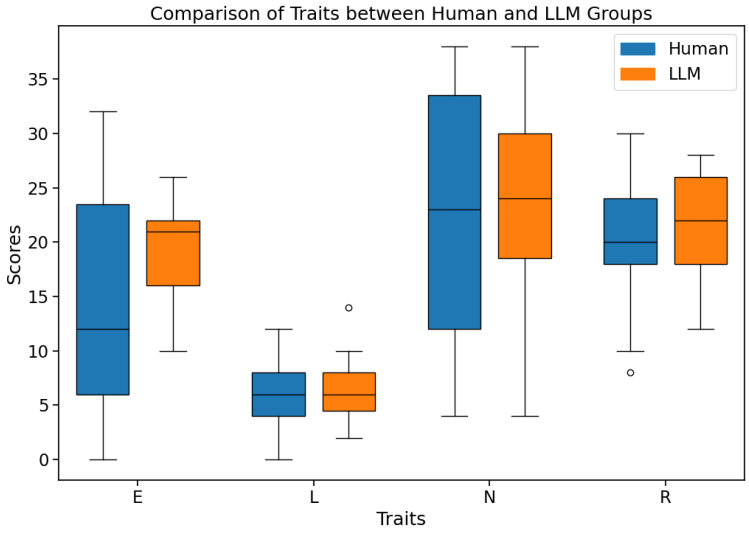


Figure 1: Comparison of personality traits between human and LLM groups.

A t-test analysis was conducted on the mean scores of each personality trait between humans and LLMs to determine whether the differences are significant. The results are shown in Table 2.

The t-test results indicate whether there is a statistically significant difference between the mean scores of humans and LLMs for each personality trait. Based on these results, we can conclude that LLMs exhibit significantly higher extraversion scores compared to humans. For the lie (L), neurocity (N) and rigidity (R) traits the

Table 2: T-test significance analysis

Trait	Human mean	LLM mean	t-statistics	p-value
Extraversion (E)	14.91	19.40	-3.53	0.0006
Lie (L)	5.84	6.67	-1.48	0.1438
Neuroticim (N)	22.47	22.80	-0.16	0.8717
Rigidity (R)	20.51	21.60	-1.11	0.2703

Table 3: LLMs' EPQ results

Model	Extraversion		Lie		Neuroticism		Rigidity	
	mean	std	mean	std	mean	std	mean	std
Alpaca	24.00	0.00	10.00	4.00	25.33	5.03	24.00	2.00
OpenChat3.5	22.00	5.29	4.67	3.06	15.33	6.43	20.00	2.00
Dbrx	22.00	2.00	5.33	1.15	22.67	1.15	24.67	1.15
LLAMA-2	15.33	1.15	4.67	1.15	24.00	2.00	20.00	7.21
Mistral	15.33	5.77	7.33	1.15	13.33	1.15	17.33	2.31
OpenHermes	16.67	6.11	9.33	1.15	5.33	2.31	18.00	0.00
Platypus-2	17.33	4.16	4.00	0.00	23.33	3.06	17.33	3.06
Qwen1.5	22.00	0.00	6.00	0.00	32.00	2.00	26.67	1.15
Snowflake-arctic	18.00	3.46	6.67	1.15	30.67	3.06	20.67	3.06
Yi-Chat	21.33	1.15	8.67	1.15	36.00	2.00	27.33	1.15

differences between humans and LLMs are not statistically significant.

In the next step, we examined the changes in scores over time. Fig. 2 shows the mean scores of each trait across the three completions.

We observed that humans are more consistent in their responses than LLMs, as the changes across completions are smaller.

In the final step, we computed the standard deviations across the three completions for each participant (for LLMs see table 3). The first diagram in Fig. 3 compares humans and LLMs in terms of the distribution of standard deviations of scores for each personality trait. The second diagram in the same figure compares the average values of the standard deviations of scores for each personality trait.

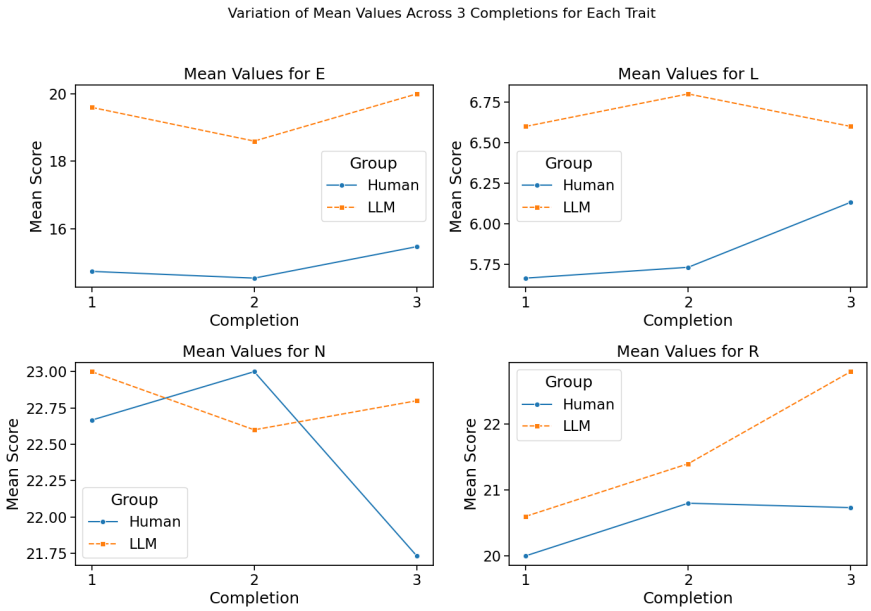


Figure 2: The mean scores of each personality trait across the three completions.

3.3.2 LLMs' results

Table 3 presents the mean and standard deviation (std) values for the LLMs, calculated from the data obtained over three completions of the EPQ.

Based on the results presented in the table, several conclusions can be drawn about the personality traits of different LLMs as measured by the EPQ.

Extraversion: Alpaca exhibits the highest mean score (24.00), indicating it has the most extroverted tendencies among the models. In contrast, LLAMA-2 and Mistral show significantly lower extraversion scores (15.33 each), suggesting these models are less extroverted. The standard deviations indicate that OpenHermes, Mistral, and OpenChat3.5 exhibit greater variability in extraversion.

Lie: The Lie mean scores vary significantly, with Alpaca showing the highest mean score (10.00) and Platypus-2 the lowest (4.00). This could suggest that Alpaca has a higher tendency to give socially desirable responses, while Platypus-2 is more straightforward. OpenHermes also shows a relatively high score in this trait (9.33). The Platypus-2 and Qwen1.5 models were very consistent in their responses for this trait, as indicated by the standard deviations of 0 across the three completions for these models.

Neuroticism: Yi-Chat has the highest neuroticism score (36.00), indicating it

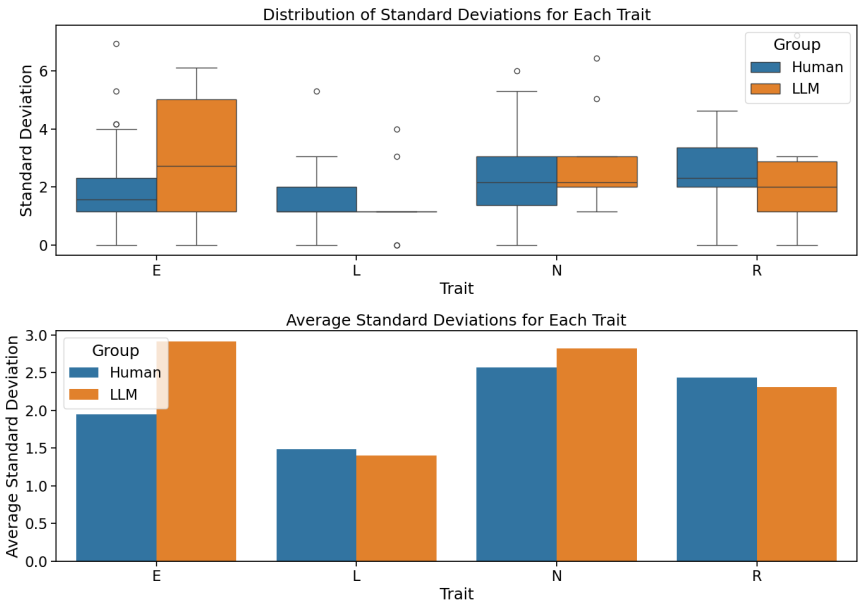


Figure 3: Standard deviations obtained for each personality trait across the three completions.

might be more prone to emotional instability. OpenHermes and Mistral exhibit much lower scores (5.33 and 13.33, respectively), suggesting they are more emotionally stable. Qwen1.5 and Snowflake-arctic also score high in neuroticism, potentially indicating a tendency towards anxiety or sensitivity.

Rigidity: Yi-Chat and Qwen1.5 have the highest rigidity scores (27.33 and 26.67, respectively), suggesting a higher level of inflexibility or strictness in their decision-making processes. The standard deviation for rigidity is lowest in OpenHermes (0.00), indicating consistent responses across the board for this trait.

Alpaca stands out for its high extraversion and lie scores, indicating it might be more socially engaging but also more likely to give socially desirable answers. Yi-Chat shows the highest scores in neuroticism and rigidity, suggesting a model that is more emotionally reactive and possibly less flexible. Mistral and OpenHermes are on the lower end of extraversion and neuroticism, indicating more introverted and emotionally stable models.

These results highlight the diversity in personality traits across different LLMs, reflecting how they might behave differently depending on the context of their application.

3.3.3 The influence of the temperature hyperparameter on LLM's personality traits

The temperature hyperparameter is crucial when administering the EPQ to an LLM, as it directly influences the variability and consistency of the model's responses, affecting the reliability of assessed personality traits.

To investigate the influence of the temperature hyperparameter on an LLM's personality traits, we selected the Mistral LLM and administered the EPQ using different temperature values.

We tested the following values for the temperature hyperparameter: `0, 0.25, 0.5, 0.75, 1`. The EPQ was administered 10 times for each temperature setting. For temperature values above 0.5, the EPQ could not be evaluated, as the model failed to follow the prompt instructions and generated invalid responses, such as "YES/NO" or "YES or NO."

Table 4 and Fig. 4 present the results obtained for the different temperature settings.

Table 4: Mistral LLM. Average personality trait scores by temperature (standard deviations are in parentheses) across the 10 completions.

Temperature	Extraversion	Lie	Neuroticism	Rigidity
0	28.00 (0.00)	10.00 (0.00)	16.00 (0.00)	20.00 (0.00)
0.25	26.60 (2.32)	9.60 (0.84)	18.00 (3.89)	15.60 (2.63)
0.50	23.00 (3.02)	7.80 (1.48)	19.80 (4.16)	14.00 (4.90)

When the temperature was set to 0, all responses were identical across the 10 administrations of the EPQ, indicating high consistency and a lack of variability, as expected.

At the 0.25 temperature value, there is moderate variability in the scores for each trait. Extrovertism and Neuroticism show greater variability, while the Lie scale remains relatively stable. This indicates that a slight increase in temperature introduces some variation, leading to more dynamic responses but still within a narrow range.

Setting the temperature to 0.5 introduces the most variation among the traits, as expected with a higher temperature. The Lie scale has the lowest variability, suggesting that responses are least impacted in this trait.

In summary, temperature influences the LLM's output consistency, with higher temperatures (0.5) resulting in greater variation and lower temperatures (0) maintaining fixed responses. This insight can help in tuning the model's behavior de-

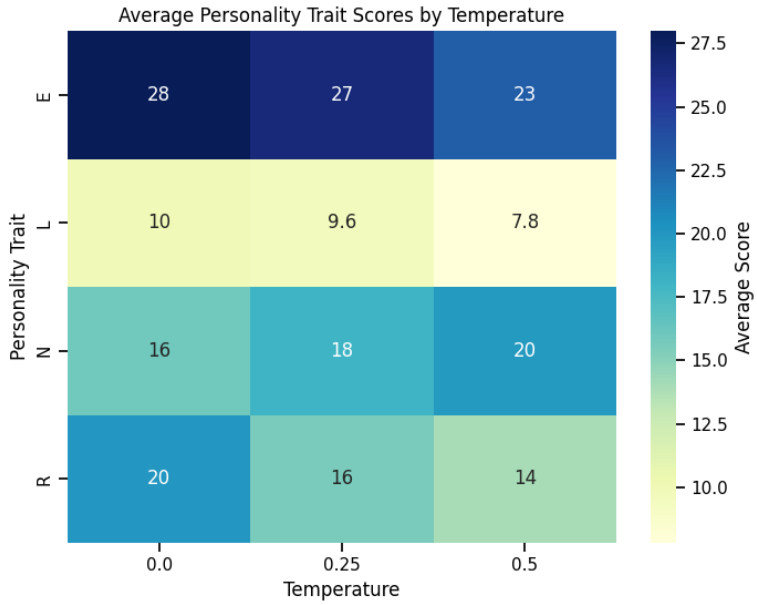


Figure 4: Heatmap of average personality trait scores by temperature.

pending on whether consistent or varied personality traits are desired in the responses.

3.3.4 The influence of the system prompt on personality traits

The system prompt plays a critical role in shaping how a language model responds during a personality questionnaire. By setting expectations around empathy, neutrality, or analytical thinking, the prompt can influence whether the model's responses feel more objective, supportive, or conversational.

We investigated how adding a sentence about personality to the system prompt of the model influences the EPQ results. Specifically, we added either "You are an extraverted person." or "You are an introverted person." to the original prompt and administered the questionnaire to the Mistral LLM 10 times. The normal prompt refers to using the prompt presented in Section 3.2; the extraverted prompt indicates the addition of the sentence "You are an extraverted person", and the introverted prompt includes the sentence "You are an introverted person" in the prompt.

The results (see Table 5 and Fig. 5) show significant variation across the three

Table 5: Mistral LLM. Average personality trait scores by prompt (standard deviations are in parentheses) across the 10 completions.

Prompt	Extraversion	Lie	Neuroticism	Rigidity
Extraverted	28.00 (2.11)	8.00 (1.33)	10.80 (3.29)	11.60 (2.63)
Normal	23.00 (3.02)	7.80 (1.48)	19.80 (4.16)	14.00 (4.90)
Introverted	4.60 (3.27)	9.00 (1.41)	15.60 (4.60)	14.40 (3.27)

different prompts in terms of Extraversion, Lie, Neuroticism, and Rigidity scores, indicating that the system prompt has a notable influence on the model's responses.

Extraversion: As expected, the extraverted prompt yields the highest average extraversion score (28.00), with low variability (SD = 2.11). The Normal prompt results in a moderate extraversion score (23.00), while the introverted prompt produces the lowest score (4.60), with slightly more variation (SD = 3.27). This demonstrates that the prompt directly impacts the model's self-reported extraversion level in a manner consistent with the added personality cue.

Neuroticism: Neuroticism scores are highest in the Normal prompt (19.80), indicating that without a personality modifier, the model tends to respond in a more neurotic style. The extraverted prompt yields the lowest neuroticism score (10.80), suggesting that an extraverted prompt may counteract neurotic traits, whereas the introverted prompt produces an intermediate score (15.60), reflecting increased neurotic traits relative to the extraverted prompt.

Rigidity: Rigidity is also affected by the prompts, with the normal and introverted prompts resulting in higher scores (14.00 and 14.40, respectively), while the extraverted prompt results in a slightly lower score (11.60). This pattern suggests that extraversion may correlate with lower rigidity, possibly reflecting a more open or flexible response style.

Lie Scale: Scores on the lie scale are relatively stable across the prompts, showing minimal variation (between 7.80 and 9.00), suggesting that the prompt's personality cues do not significantly affect the tendency to respond in a socially desirable manner.

In summary, the model's extraversion, neuroticism, and rigidity scores show a clear and systematic response to the personality cue embedded in the prompt, confirming the prompt's strong influence on shaping the LLM's responses in personality assessments.

In addition to exploring the effects of temperature and the system prompt, we examined how contextual memory influences personality traits. In the standard

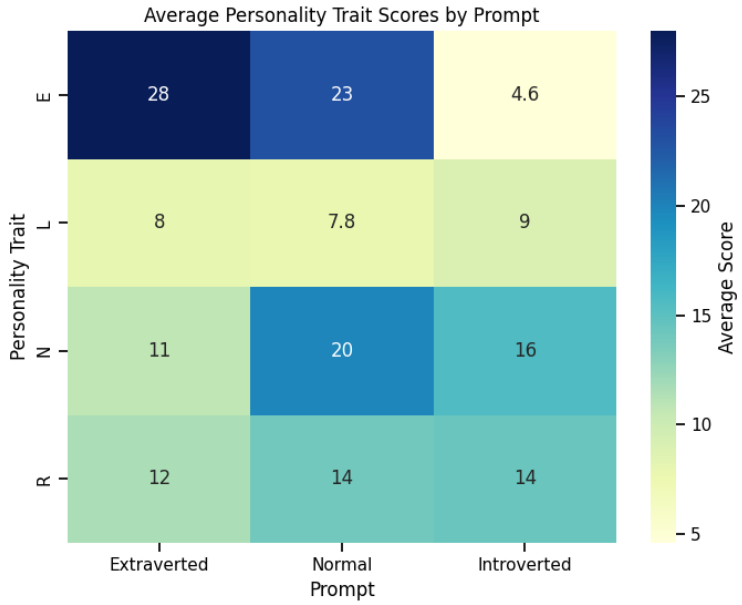


Figure 5: Heatmap of average personality trait scores by prompt.

settings, each question was asked in isolation. However, in the contextual memory settings, we included the previous questions and their answers along with the new question in the chat context. This configuration had minimal impact on the personality traits.

4 Discussion

The human participant pool consists mainly of university students and professors, which may not be representative of the general population. This homogeneity can limit the generalizability of the findings to other demographic groups. The majority of participants are young adults, therefore these results might not reflect the personality traits of older populations. The gender distribution is imbalanced, therefore this could introduce gender bias into the results and limit the ability to generalize findings across genders. The human population size is relatively small, which could affect the robustness and reliability of the statistical analyses.

Applying the EPQ, designed to assess human personality traits, to LLMs comes with several limitations:

- Lack of true personality: LLMs lack subjective experiences, emotions, and motivations—key aspects of human personality. EPQ traits such as extraversion or neuroticism are grounded in human emotional and psychological processes, which LLMs do not possess. When applied to LLMs, these traits serve only as metaphorical labels.
- Absence of stable internal states: Human personality reflects stable tendencies over time, rooted in psychological and biological factors. LLM responses, however, are generated based on patterns in training data rather than stable, internal states. This means any apparent consistency in "personality" across responses is an artifact of data and model parameters, not a stable trait.
- Influence of training data: LLM responses are shaped by the biases, topics, and patterns found in their training data. If the training data heavily features certain patterns associated with extraversion, for example, the model might seem "extroverted" in certain responses, even though this is merely a reflection of the data rather than any authentic trait.
- Risk of anthropomorphization: Using human personality frameworks like the EPQ to interpret LLM outputs can encourage anthropomorphism, where users might attribute human-like consciousness or emotions to the model. This can lead to misinterpretation of the model's capabilities.
- Interpretive limitations: The EPQ was developed based on empirical research with humans, whose responses are influenced by complex biological, psychological, and social factors. LLM responses, however, are generated by algorithms and lack these influences.

5 Conclusions

Analyzing the consistency across completions, we can draw the following conclusions. Firstly, the mean scores for each trait among humans show relatively small variations across the three completions. This suggests that human responses are fairly consistent over time. The mean scores for LLMs also exhibit some consistency across completions, though there are slight fluctuations, particularly in extraversion and rigidity traits.

The temperature hyperparameter strongly impacts the consistency of personality trait scores of LLMs. At lower temperatures (e.g., 0), responses remain highly stable. As temperature rises (e.g., 0.25 and 0.5), response variability increases, especially in extraversion and neuroticism. Above 0.5, the model struggles to follow instructions

consistently. Thus, adjusting temperature can help control response variability, allowing for either consistent or more dynamic personality-related outputs as needed.

Adding personality cues in the system prompt—specifically for extraversion and introversion—significantly shifts personality trait scores. Extraversion scores peak with an extraversion cue in the prompt, moderate with a neutral prompt, and are lowest with an introversion cue in the prompt, showing the model's responsiveness to these directives. Neuroticism scores are lowest with an extraverted prompt and highest with a neutral one, suggesting an influence on response tone. Rigidity scores also vary, being lower with extraversion cues and higher with introversion or neutral cues. The Lie scale, however, remains stable, indicating that social desirability is less affected by personality cues.

These results demonstrate that both temperature settings and prompt instructions can dramatically shape the personality-like patterns in LLM outputs. This has practical implications for LLM-based conversational applications, where fine-tuning these settings can yield outputs that align more closely with desired personality traits.

The strong influence of prompt and temperature on personality-related outputs suggests caution in interpreting LLM responses as stable psychological traits. Instead, these outputs should be seen as patterns shaped by prompt structure and model settings, allowing for adjustable, yet artificial, personality profiles. This highlights the need for precise prompt design and hyperparameter tuning when using personality cues in LLM responses and stresses the importance of avoiding anthropomorphism.

Data Availability: The data and code that support the findings of this study are openly available in GitHub at

`https://github.com/NorbertBeder/Eysenck-Personality-Test`.

References

[1] H. J. Eysenck and S. Eysenck, *Manual of the Eysenck Personality Inventory*. Hodder and Stoughton, 1964 (⇒ 220).

[2] S. Eysenck, H. J. Eysenck, and P. Barett, "A revised version of the psychoticism scale," *Personality and Individual Differences*, vol. 6(1), pp. 21–29, 1965 (⇒ 220).

[3] G. Caron and S. Srivastava, *Identifying and manipulating the personality traits of language models*, 2022. arXiv: 2212.10276 [cs.AI] (⇒ 221).

[4] G. Jiang, M. Xu, S.-C. Zhu, W. Han, C. Zhang, and Y. Zhu, "Evaluating and inducing personality in pre-trained language models," in *NeurIPS*, 2023 (⇒ 221).

[5] H. Jiang, X. Zhang, X. Cao, C. Breazeal, D. Roy, and J. Kabbara, "PersonaLLM: Investigating the ability of large language models to express personality traits," in *Findings of the Association for Computational Linguistics: NAACL 2024*, K. Duh, H. Gomez, and S. Bethard, Eds., Mexico City, Mexico: Association for Computational Linguistics, Jun. 2024, pp. 3605–3627 (⇒ 221).

[6] G. Serapio-García, M. Safdari, C. Crepy, *et al.*, *Personality traits in large language models*, 2023. arXiv: 2307.00184 [cs.CL] (⇒ 221).

[7] *Alpaca*, https://crfm.stanford.edu/2023/03/13/alpaca.html (⇒ 222).

[8] G. Wang, S. Cheng, X. Zhan, X. Li, S. Song, and Y. Liu, *Openchat: Advancing open-source language models with mixed-quality data*, 2024. arXiv: 2309.11235 [cs.CL] (⇒ 222).

[9] *Dbrx-instruct*, https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm (⇒ 222).

[10] H. Touvron, L. Martin, K. Stone, *et al.*, *Llama 2: Open foundation and fine-tuned chat models*, 2023. arXiv: 2307.09288 [cs.CL] (⇒ 223).

[11] *Mistral*, https://mistral.ai/news/announcing-mistral-7b/ (⇒ 223).

[12] *Openhermes*, https://huggingface.co/teknium/OpenHermes-2.5-Mistral-7B (⇒ 223).

[13] A. N. Lee, C. J. Hunter, and N. Ruiz, *Platypus: Quick, cheap, and powerful refinement of llms*, 2024. arXiv: 2308.07317 [cs.CL] (⇒ 223).

[14] A. N. Lee, C. J. Hunter, and N. Ruiz, "Platypus: Quick, cheap, and powerful refinement of llms," 2023 (⇒ 223).

[15] J. Bai, S. Bai, Y. Chu, *et al.*, "Qwen technical report," *arXiv preprint arXiv:2309.16609*, 2023 (⇒ 223).

[16] *Snowflake*, https://www.snowflake.com/en/data-cloud/arctic/ (⇒ 223).

[17] *Yichat*, https://huggingface.co/01-ai/Yi-34B-Chat (⇒ 223).

[18] A. Young, B. Chen, C. Li, *et al.*, *Yi: Open foundation models by 01.ai*, 2024. arXiv: 2403.04652 [cs.CL] (⇒ 223).

Received: 06.09.2024; *Revised: 08.11.2024*; *Accepted: 08.11.2024*

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 236–254

DOI: 10.47745/ausi-2024-0013

A large-scale analysis of production effort changes in software projects

Kristóf Szabados
Eötvös Loránd University
Budapest, Hungary
SzabadosKristf@gmail.com

Abstract.

Although software is essential to modern society, its development process remains poorly understood. It is easy to find tools/techniques/methods promoted to help reach ever-higher production volumes and great improvements, even though this contrasts previous academic observations.

This study aims to expand on previous research regarding the evolution of software development efforts by analyzing a large dataset of contemporary open-source projects.

We examined 875 popular and large GitHub repositories and found a strong correlation between accumulated effort and quadratic functions of time, in 73% of projects, with linear models applicable in 41%. Our analysis of long-term trends suggests that no major external events (e.g., economic downturns, technological shifts) have impacted significantly all projects over the past 25 years.

These findings challenge the perception of rapid, exponential improvements in software development efficiency. While further research is needed, our results suggest a disconnect between software development's perceived and empirical realities.

Key words and phrases: Empirical Study, Large-Scale, Software Evolution, Longitudinal, Conway's Law, Mirroring Law, Lehman's Laws, Software Development Tools, Software Engineering Laws, Sustainability, Cost Reduction

1 Introduction

Software is ubiquitous in modern society. It helps us navigate, communicate, and manage energy resources. It drives companies, trades on markets, and supports healthcare.

Developing large-scale software systems often takes years and is an effort-intensive endeavour. Accelerating software development could have a positive impact on both economic growth and social well-being. Improving efficiency could improve results, reduce costs and free up resources for other activities.

Given the decades-long history of software development, we have access to a wealth of empirical data. By analysing the source code repositories of open-source projects, we can gain valuable insights into the factors that influence development efforts. This will ensure that upcoming development efforts are based on a solid foundation of evidence-based practices and strategies instead of assumptions and speculations. By examining historical trends, we can uncover hidden correlations and potential causal relationships between external factors and development efforts, ultimately enabling us to make more informed decisions and optimise our development processes.

In this paper, we report on our study on how the effort invested into software development changes over time, and how seemingly there was no event in the last 25 years that would have impacted all projects on its own, using 875 open-source projects.

This paper is organised as follows. Section 2 presents related works from the literature. Section 3 describes our work. Section 4 presents our results and analyses. Section 5 their validity. Finally, Section 6 shows our conclusions, and Section 7 offers ideas for further research.

2 Literature Review

In this section, we present earlier literature related to our work. While we view our work as a natural extension of [1] and [2] (presented in Section 2.5), we also present research results related to strong patterns observed in other aspects of software systems. We accept the potential influence of external factors, such as international laws, regulations, and business objectives, on software development practices. However, we do not wish to prioritize these factors in our literature review, as we lack empirical evidence to assess their relative importance.

In the first group of sections, factors external to the software: organisations intentionally design and govern the overall architecture of their products to achieve their business targets (Sections 2.1, 2.2). Followed by sections on the generality and inevitability of the internal structure of large software systems (Section 2.3), all software systems evolving similar size distributions (Section 2.4). Finally, previous works on how all software systems evolve in similarly predictable ways (Section 2.5).

2.1 The impact of organisational factors on software systems

Empirical observations have identified a strong relationship between an organisation's communication and product structure [3], [4]. In 2008 Nagappan et al. [5] showed that organisational metrics predict failure-proneness better than code complexity, coverage, internal dependencies, churn and pre-release bug measures. By 2022 it was used by firms as a superior strategy [6] to maximise business benefits [7]. This indicates the importance of the legal and business environment over any technical considerations.

Following these laws, contemporary System Architects consider among others Taxation [8], Export control [9], and Geopolitics [10] when planning software architecture and the organisation developing it. Contemporary Project Management recommends [11] tailoring a selected development approach first for the organisation and second to the project.

2.2 The impact of Project Management on software Projects

Researchers have identified [12]–[14] that Project Management[1] techniques and processes are the critical factors contributing to project success.

Empirical observers have noted [16] that 94% of troubles and possibilities for improvement are the responsibility of management. Recommended preventing problems with systemic problem-solving performed with scientific rigour [17]–[19] instead of celebrating solving crises and heroes putting out fires. Understanding, that writing programs "is only a small part of Software Engineering" ([20]). This indicates the importance of intentional and professional management.

2.3 The dependency networks of software systems

Empirical researchers have shown that several architectural properties of software systems are scale-free[2], like many real-life networks. Class collaboration graphs of the C++ language [21], Class, method, and package collaboration graphs of Java [22], [23], connections between the modules in TTCN-3[3] [24], [25], file inclusion graphs in C [26], and the runtime object graph of most of the Object Oriented Programming languages in general [27], the relationships of distributed software packages [28], [29] show scale-free properties.

[1] "Project Management is the application of knowledge, skills, tools, and techniques to project activities to meet the project requirements" [15].

[2] A network is called scale-free, when its degree distribution, follows a power law.

[3] TTCN-3 is the abbreviation of Testing and Test Control Notation Version 3

Taube-Schock et al. [30] showed, that approximately scale-free structures should arise for both between-module and overall connectivity from the preferential attachment-based models, not as the result of poor design. That high coupling is unavoidable and might even be necessary for good design, contradicting previous ideas about software structure, particularly the "high cohesion/low coupling" maxim. This indicates that software design is not done before development, but emerges automatically during it, with developers having limited control over it.

2.4 The size distribution of software systems

Empirical studies have revealed that several metrics correlate to the point of redundancy[4] [31]. Measuring Source Lines Of Code would be enough to obtain a landscape of the evolution of the size and complexity of FreeBSD [32].

Empirical researchers have shown logarithmic distributions in various places: module lengths of IBM 360/370 and PL/S code [33], Java class sizes [34], tokens in Java [35], [36], all metrics measured on FreeBSD [32], for five (C, C++, Java, Python, Lisp) of the top seven programming languages used in the Linux code [37] and LOC in Smart Contracts for the Ethereum blockchain [38]. Stating that "whatever is measured in a large scale system" logarithmic distribution is observed in most cases [39].

Hatton used [40], [41] the Conservation of Hartley-Shannon Information to show strong evidence for unusually long components being an inevitable by-product of the total size of the system, validating the claims on 100 million lines of code in 7 programming languages and 24 Fortran 90 packages. Highlighting the importance of changing software design techniques, from attempting to avoid the essentially unavoidable to mitigating its damaging effects.

2.5 The evolution of software systems

Since Lehman published the laws of software evolution [42], plenty of empirical research [1], [2], [43]–[50] show that the laws seem to be supported by solid evidence. To the point that the gross growth trends can be predicted by a mean absolute error of order 6%[1], [2], [51], [52].

Looking at the impact of outside effects on software growth, empirical researchers observed [1] that "the introduction of continuous integration, the existence of tool support for quality improvements in itself, changing the development methodolo-

[4]Cyclomatic Complexity, the Number of Lines of Code, Statements, Classes, Files, public APIs, and public undocumented APIs are redundant metrics, with Cyclomatic Complexity in classes and functions measuring the same subject

gies (from waterfall to agile), changing technical and line management structure and personnel caused no measurable change in the trends of the observed Code Smells", on industrial Java and C++ projects [2], that changing architects, going open-source or the organisation moving to a different building had no easily discernible effect on development.

The works performed on large open-source systems [2], [44], [46]–[49], [52] serve as observations supporting the understanding that there might not have been any hardware, software, tooling, methodological, social, or other change at least since 2000, that would have significantly changed the development speed of large software systems already started.

3 Methodology

This section presents our aim and work in technical terms.

We wanted to extend the knowledge available on how software systems evolve with up-to-date information on a dataset of large, diverse, and long-running projects.

We selected GitHub as the data source, one of the largest sites hosting software project repositories. To ensure language independence, we focused on change counts reported by Git for each merge. This approach allowed us to treat all changes, including code, comments, documentation, test examples, and other text artefacts equally.

We also decided to treat deletions and insertions as equally significant changes, valuing careful removal of unnecessary code at the same level as adding new ones.

The following metrics were tracked:

- *Cumulative Commits*: The total number of commits contributed over time.
- *Cumulative Effort*: The total number of insertions and deletions made over time.
- *Lines*: The net change in lines of text, calculated by subtracting deleted lines from inserted lines.

The process for gathering and processing the data:

1. *Selection*: We identified the 100 most liked/stared GitHub repositories for each programming language[5] and selected those with over 3000 commits by mid-2024.

[5]The list of most popular repositories by programming languages we used is shown on GitHub-Ranking

2. *Repository filtering*: We excluded repositories triggering virus detection (e.g., Metasploit Framework), from this work, so that we would not compromise the security of our devices and the networks they have to connect to.

3. *Log extraction*: We extracted Git logs from the main branch of each repository. With the adjusted settings:

   (a) Increased *diff.renamelimit* to 130,000.

   (b) Used --first-parent, for branch history analysis[6].

   (c) We used the commit date (”%cd”). We observed the author date as easily manipulated[7].

4. *Data cleaning, filtering*: The data were cleaned up and filtered.

5. *Metric tracking*: Changes in the number of commits, lines, and effort values were tracked.

6. *Data Analysis*: The data collected was analysed.

Data cleaning and filtering involved:

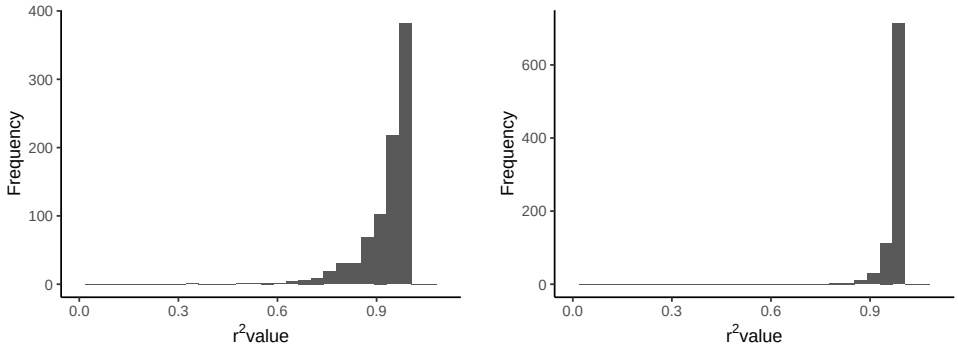
- In some projects the number of lines dropped to near zero before resuming growth. When these commits had comments hinting at restarting the project[8] we split the project into two projects at those commits.

- We removed commits if there were only a few and much earlier than the actual work started[9].

- We investigated every commit that made the repository look like having negative content and modified their date to that of their later parent[10].

4 Results and discussion

This section presents our results and analyses.



(a) (b)

Figure 1: r^2 value distribution for commits with linear 1(a) and quadratic 1(b) fitting

4.1 General overview

To investigate the relationship between accumulated commits, effort and project development, we fitted linear and simple quadratic regression models to the data. Figure 1(a) shows the distribution of r^2 values when matching the accumulated number of commits with a linear model. Of the 875 projects, 249 match above 0.98, 495 above 0.95 and 687 above 0.9. Figure 1(b) shows the same data matched against a quadratic model, with 586 matches above 0.98, 782 above 0.95 and 850 above 0.9.

Figure 2(a) shows the distribution of r^2 values when matching the accumulated effort values with a linear model of the 875 projects 126 match above 0.98, 357 above 0.95 and 576 above 0.9. Figure 2(b) shows the same data matched against a quadratic model, 344 matches above 0.98, 636 above 0.95 and 789 above 0.9.

This indicates that software development mostly follows clear evolution patterns. While these models may not achieve the highest precision, they might offer a practical and effective approach for industrial applications where precise forecasting is not always critical. For most projects, simple linear or quadratic models provide a reasonable fit, suggesting that project evolution can be approximated with functions that change direction at most once.

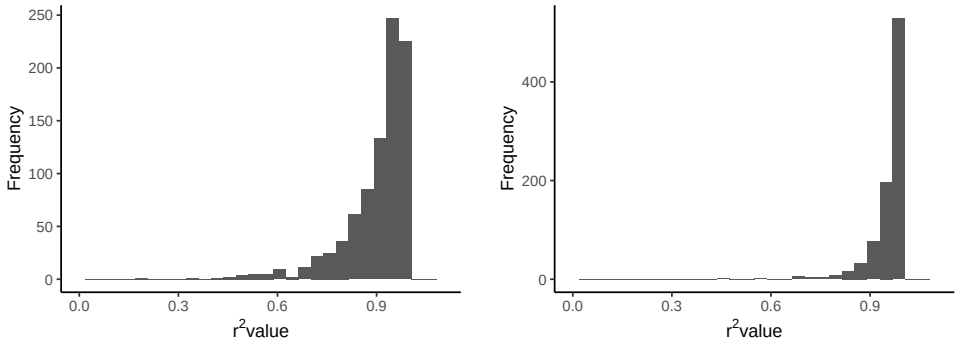
[6]This also simplified handling octopus merges. Linux has merges with up to 66 parents, for example: commit hash

[7]Linux seems to have commits from 1970.01.01 and 2085.06.18.

[8]In tldraw, on 2023.04.21 everything is deleted, and re-created for a new version on 2023.04.25.

[9]Go has a "Hello World" commit in 1972, FFMPeg some commit months before the "Initial revision"

[10]In Ansible the first commit, by date, seems to delete 12 lines from an empty repository, but it builds on a commit from 2012.02.24.



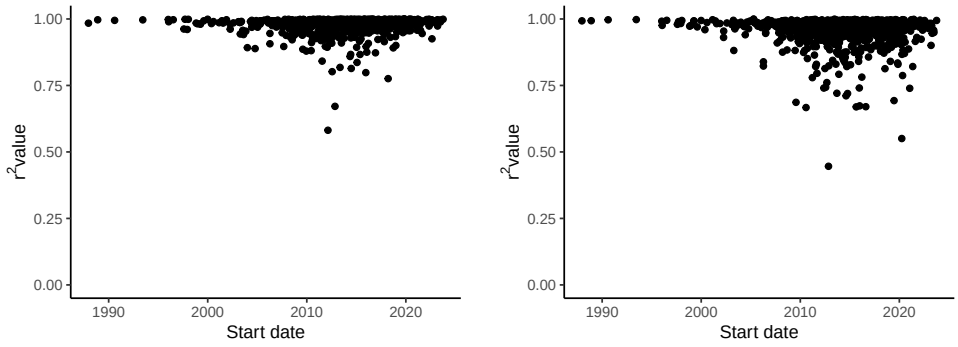
(a) (b)

Figure 2: r^2 value distribution for effort with linear 2(a) and quadratic 2(b) fitting

4.2 Quality in time

When investigating how the r^2 values might depend on when the projects started, we observed that newer projects tend to exhibit worse fits to linear or quadratic models (see Figure 3).

It is important to note that a selection bias may skew this analysis. Many projects initiated after 2020 might not have yet reached the popularity levels or commit counts required for inclusion in our dataset.



(a) (b)

Figure 3: r^2 value distribution for commits 3(a) and effort 3(b) when correlated with quadratic models, represented as a function of the starting time of the project

4.3 Observations using some of the oldest projects

Figure 4 shows how the accumulated effort invested into a project grows over time for some of the oldest projects.

While there were observable changes in the projects individually, it is clear that in the last 25 years, no external change had a significant, lasting impact on all projects.

Such external events include:

- Processor speed and core count grew by magnitudes.
- RAM size and throughput grew by magnitudes.
- Internal storage size and throughput grew by magnitudes while decreasing latency by magnitudes.
- Screen space to display information increased from around 640×480 to 1920×1080 and beyond.
- Graphical IDEs, code quality checking and refactoring, CI/CD and other productivity and support tools became available.
- The Internet became available, with several sites supporting developers.
- Cloud computing made vast amounts of resources easily reachable.
- Various methodologies were promoted to improve developer productivity and reduce defects: Scrum, Kanban, SAFe, LeSS, Nexus, Scrum@Scale, Agile@Scale, Lean, XP, FDD, AIDD, DSDM, UP, Six Sigma, DevOps and more.
- Computer Science extended understanding of how software systems evolve, how structure and distribution patterns emerge (Section 2).
- The FLoyd-Hoare logic was extended for parallel programs.
- A financial crisis, a global pandemic and a change in R&D TAX law[11].
- Some open-source projects accumulated thousands of contributors[12].
- AI improved, automated planning and decision-making, navigation in physical and abstract spaces[13]. ChatGPT.

[11] Section 174 in 2022.
[12] Linux has 15,700+, CPython 2,811, GCC 983 contributors according to GitHub.
[13] Already in 2001 demonstrating cooperative and competitive team level behaviour[53]

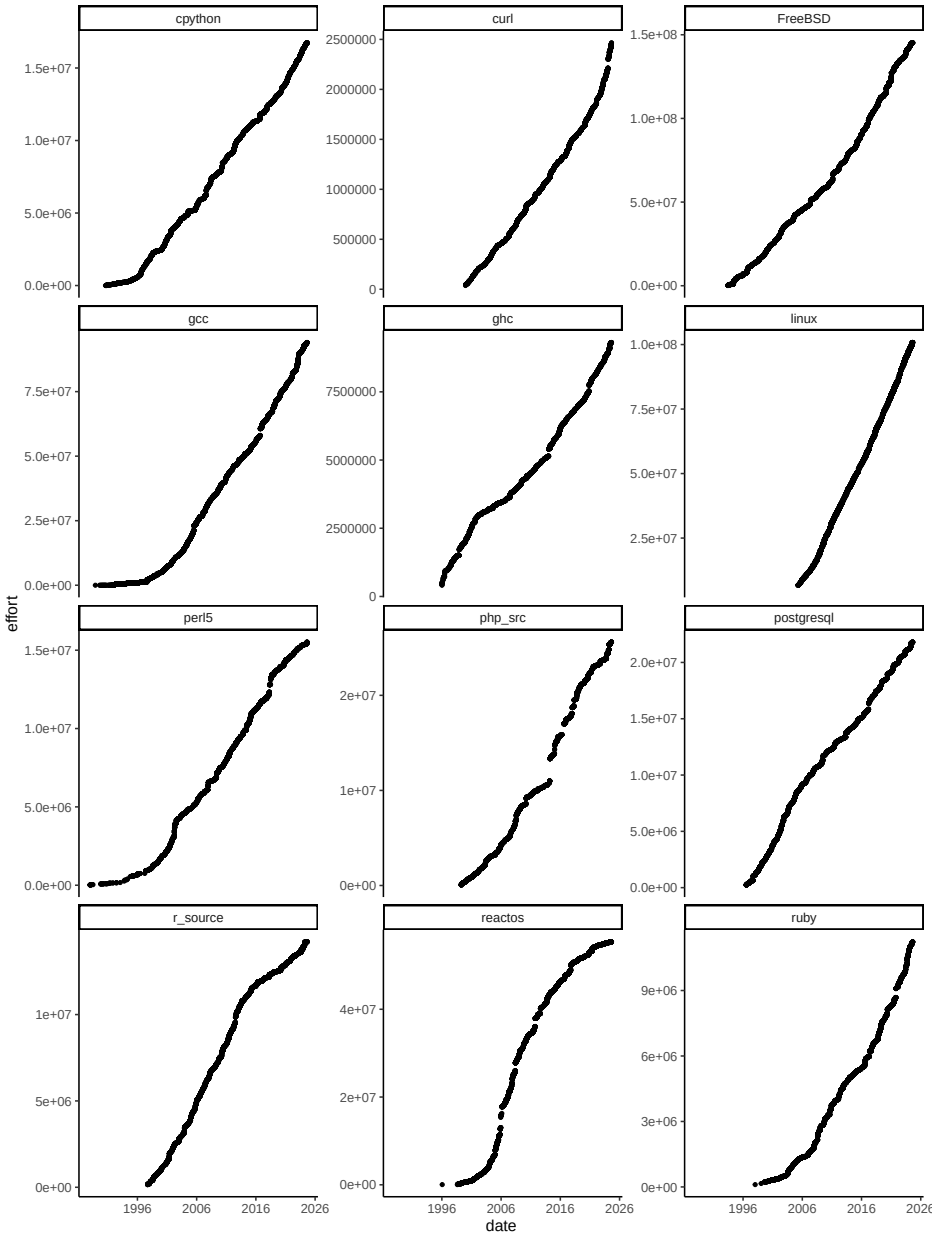


Figure 4: The effort graphs for some projects started before 2000.

According to our data, none of these improvements and events have impacted all projects at the time, on their own.

Other less generic events could include security patches, operating systems, programming languages, and toolset version updates. These are usually forbidden within shorter/smaller projects, and in longer projects managed closely as a parallel side-project to ensure no noticeable impact on the main project.

4.4 Projects with close to linear Effort trends are present in many programming languages

Our analysis (Figure 2) identified several projects demonstrating high r^2 values fitting the accumulated Effort measurements to a linear model for different programming languages. This indicates that the effect is language agnostic.

Figure 5 shows: APISIX (79.7% Lua), Appsmith (67.2% typescript), android-oss (92.7% Kotlin), BookStack (89.2% PHP), Cats (100% Scala), CockroachDB (89.9% Go), ColossalAI (92.8% Python), DevDocs (79.4% Ruby), Dokku (57.4% Shell script), egui (98.5% Rust), Envoy (87.6% C++), Fastify (90.8% Javascript), osu! (100% C#), Linux (98.4% C), AppFlowy (54.6% Dart), Bitcoin (65.3% C++, 20.2% Python).

4.5 Observations on the least fitting projects

This section presents our observations on the least fitting projects.

Only two projects, "HoTT/book" ($r^2 = 0.32$) and 'SecLists' ($r^2 = 0.49$), showed linear fits below 0.5 for commit accumulation. "HoTT/book", started in 2013, had approx. 50% of its total commits between 2013.04.01 and 2013.09.10. "SecLists", established in 2012, had approx. "62%" of its commits between 2024.06.15 and 2024.09.15.

Investigating fits below 0.5 for effort accumulation, we found 8 projects: HoTT/-book 0.18 and 0.45 (linear and quadratic fit), Middleman 0.35 and 0.69, Moya 0.41 and 0.71, Caffe 0.44 and 0.72, Frontend 0.45 and 0.74, plotly.R 0.49 and 0.79, Dapr 0.495 and 0.69, static-analysis 0.498 and 0.74.

In some of these projects, significant changes were introduced only by a few commits which were also removed soon after. For example (see Figure 6):

- Caffe: On 2013.11.07 approx. 1.3 million lines were added, seemingly most of them removed on 2014.02.26 with commit deleting 1.4 million lines.
- Moya: On 2015.06.16 47, 698 lines were added, on 2015.09.12 45, 993 removed, on 2015.09.14 52, 323 added and on 2015.10.27 51, 124 removed.

To illustrate the impact of such changes, removing these specific commits the fit would improve to r^2 values of 0.79 and 0.94 for Caffe, 0.50 and 0.79 for Moya, 0.68

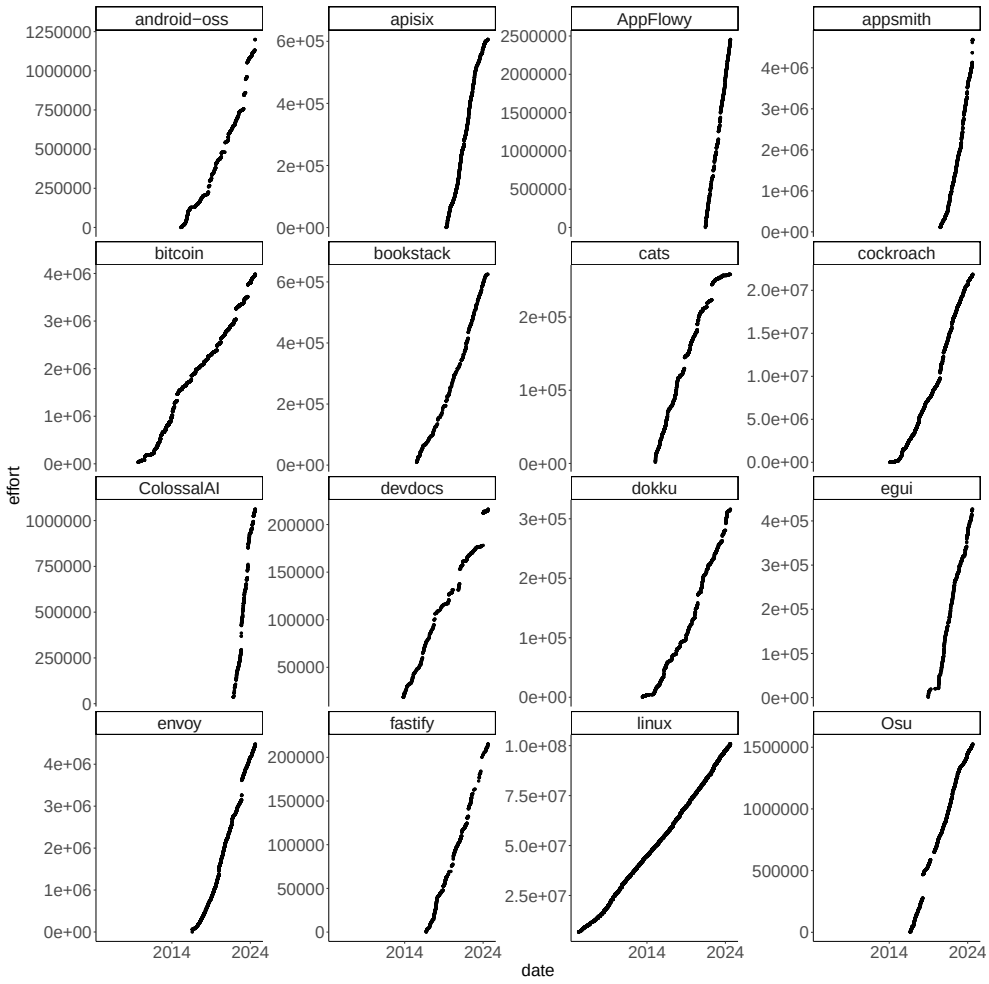
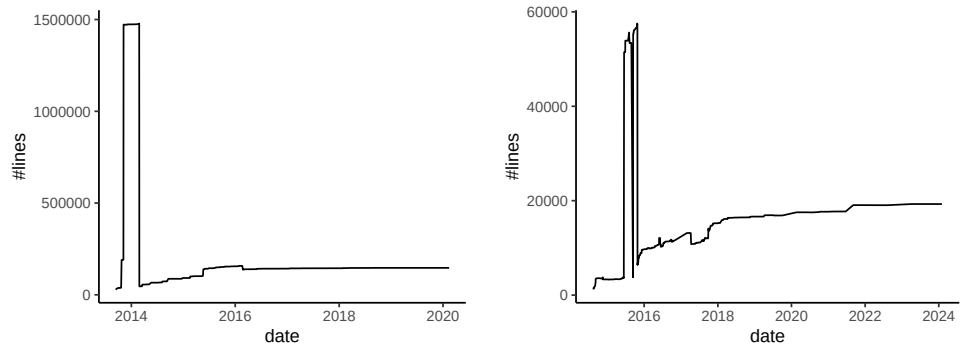


Figure 5: Projects with high r^2 values, developed in different programming languages

and 0.97 for Middleman, 0.83 and 0.98 for plotly.R, 0.67 and 0.74 for Dapr.

HoTT/book seems to have a large commit beside a large insertion and deletion pair on 2013.04.27. Removing only the insertion-deletion pair does not change the r^2 values significantly. The large insertion is a merge, indicating that a substantial amount of work is being done elsewhere.

Frontend seems to have many large temporal changes, Static-analysis underwent several changes between 2022.08.22 and 2022.09.24 that look like refactoring, mak-



(a) (b)

Figure 6: The number of lines for Caffe 6(a) and Moya 6(b)

ing them harder to analyse.

5 Threats to validity

This study might suffer from the usual threats to external validity. There might be limits to generalizing our results beyond our settings (the programming language used and repositories outside of GitHub).

One specific threat to validity might come from the fact that committers did modify the author and commit date of some of the commits. It could happen that there were other ways and methods to change commits, that we did not know about or notice. To address this threat we have to point out that most of the projects have several contributors (15, 700+ for Linux) and have been running for several years. We find it unlikely that an effort would exist to create misleading metrics on this scale.

A limit for generalizing our results might come from the selection criteria: popular, large, open-source projects present on GitHub. While GitHub is one of the largest sites hosting software project repositories, privately developed, less popular or smaller repositories may follow different trends.

6 Conclusion

This paper presents our study on how the effort invested into software development projects changes over time, using 875 popular and large open-source projects.

We presented earlier works on the structure of software systems and the external factors identified to impact them significantly. We also presented earlier works on the evolution tendencies of software systems, information that we are updating with contemporary data and extending in the number of projects.

In nearly 73% of the investigated projects, we found a strong correlation between accumulated effort and a quadratic function of time.

Despite significant advancements in areas related to software development over the past 25 years, our analysis of pre-2000 projects revealed no overarching external event that consistently impacted their development. Most projects seemed to be largely unaffected by external factors. This effect was seen regardless of the programming language used.

Our results show that software development seems to follow clear patterns, even if those are not intuitive. Our observations offer little hope for improving the speed of software development but indicate the possibility of improving efficiency by optimising headcount and investment into tooling ([54]).

Our findings support the prohibition of gold plating (any effort spent on achieving higher than necessary quality is likely to be taken away from other directly beneficial features), the consideration of outcome bias (as the outcome might not depend on the actions of individuals or decisions directly, it should be ignored when evaluating those decisions) and in general the delaying of decisions as long as feasible instead of action fallacy (since development seems to follow predictable trends it is not clear if decisions do have an impact on results or might only take up resources to make).

7 Further work

In the future, we plan to investigate more projects with low r^2 values. Further research could investigate the reasons behind the deviance we observed in some projects and explore for specific projects all events that happened with that project and their impact on it (e.g. [1]).

Further research could also investigate if there are differences in the effect based on the organisation performing the development or industrial area.

Further research could also investigate how and how far the headcount and tooling investment could be lowered while achieving the same results.

It is generally accepted that a low bus factor and the presence of the hero anti-pattern can harm projects. Our observations also seem to indicate that it is not just necessary to limit these effects, but also possible on long time horizons (in a 25 year time frame, people tend to leave projects). We leave it to further researchers to

investigate in detail, how the projects managed to keep these effects under control.

Data Availability: For our research we only used data publicly available on GitHub (the Git repositories of the projects). In every case, we used the latest version at the time of publication (pulling the latest changes for every project). We refer to specific phenomena in the article by providing direct links to the code version, using their unique commit hash identifier.

Acknowledgments: The authors thank Izabella Ingrid Farkas for her help and feedback on this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

[1] A. Kovacs and K. Szabados, "Internal quality evolution of a large test system–an industrial study," *Acta Univ. Sapientiae*, vol. 8, no. 2, pp. 216–240, 2016 (⇒ 237, 239, 249).

[2] A. Zsiga, "Termelékenységi trendek, minták elemzése szoftverfejlesztési projektekben," M.S. thesis, Eötvös Loránd University, 2019 (⇒ 237, 239, 240).

[3] M. E. Conway, "How do committees invent," *Datamation*, 1967 (⇒ 238).

[4] A. MacCormack, C. Baldwin, and J. Rusnak, "Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis," *Research Policy*, vol. 41, no. 8, pp. 1309–1324, 2012, ISSN: 0048-7333 (⇒ 238).

[5] N. Nagappan, B. Murphy, V. Basili, and N. Nagappan, "The influence of organizational structure on software quality: An empirical case study," Microsoft Research, Tech. Rep. MSR-TR-2008-11, Jan. 2008, p. 11 (⇒ 238).

[6] L. J. Colfer and C. Y. Baldwin, "The mirroring hypothesis: Theory, evidence and exceptions," *IRPN: Innovation & Organizational Behavior (Topic)*, 2016 (⇒ 238).

[7] E. Leo, "Breaking mirror for the customers: The demand-side contingencies of the mirroring hypothesis," *Cont. Man. Res.*, vol. 18, pp. 35–65, Mar. 2022 (⇒ 238).

[8] M. Dorner, M. Capraro, O. Treidler, *et al.*, *Taxing collaborative software engineering*, 2023. arXiv: 2304.06539 [cs.SE] (⇒ 238).

[9] M. Choudaray and M. Cheng, "Export Control," in *Open Source Law, Policy and Practice*, Oxford University Press, Oct. 2022, ISBN: 9780198862345. eprint: https://academic.oup.com/book/0/chapter/378967490/chapter-pdf/49854057/oso-9780198862345-chapter-12.pdf (⇒ 238).

[10] A. Pannier, *Software power: The economic and geopolitical implications of open source software*, 2022 (⇒ 238).

[11] P. M. Institute, *A guide to the Project Management Body of Knowledge (PMBOK guide)*, 7th. Newton Square, PA: PMI, 2021, ISBN: 9781628256673 (⇒ 238).

[12] J. Varajão, R. P. Marques, and A. Trigo, "Project management processes – impact on the success of information systems projects," *Informatica*, vol. 33, no. 2, pp. 421–436, 2022, ISSN: 0868-4952 (⇒ 238).

[13] S. Pretorius, H. Steyn, and T. Bond-Barnard, "The relationship between project management maturity and project success," *J. Mod. Proj.*, vol. 10, pp. 219–231, Mar. 2023 (⇒ 238).

[14] S. Moradi, K. Kähkönen, and K. Aaltonen, "From past to present- the development of project success research," *J. Mod. Proj.*, vol. 8, no. 1, Apr. 2022 (⇒ 238).

[15] P. M. Institute, *A guide to the Project Management Body of Knowledge (PMBOK guide)*, 6th. Newton Square, PA: PMI, 2017, ISBN: 9781628251845 (⇒ 238).

[16] W. E. Deming, *Out of the Crisis* (MIT Press Books). The MIT Press, Dec. 2000, vol. 1, ISBN: 9780262541152 (⇒ 238).

[17] R. Hayes, "Why japanese factories work," *HBR*, vol. 59, pp. 56–66, Jan. 1981 (⇒ 238).

[18] S. Spear and H. Bowen, "Decoding the dna of the toyota production system," *HBR*, vol. 77, Sep. 1999 (⇒ 238).

[19] R. Bohn, "Stop fighting fires," *HBR*, vol. 78, pp. 83–91, Jul. 2000 (⇒ 238).

[20] D. L. Parnas, "Structured programming: A minor part of software engineering," *Information Processing Letters*, vol. 88, no. 1, pp. 53–58, 2003, To honour Professor W.M. Turski's Contribution to Computing Science on the Occasion of his 65th Birthday, ISSN: 0020-0190 (⇒ 238).

[21] C. R. Myers, "Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs," *Physical Review E*, vol. 68, no. 4, Oct. 2003 (⇒ 238).

[22] D. Hyland-Wood, D. Carrington, and S. Kaplan, "Scale-free nature of java software package, class and method collaboration graphs," in *Proceedings of the 5th International Symposium on Empirical Software Engineering*, 2006 (⇒ 238).

[23] L. Šubelj and M. Bajec, "Software systems through complex networks science: Review, analysis and applications," in *Proceedings of the First International Workshop on Software Mining*, ser. SoftwareMining '12, Beijing, China: ACM, 2012, pp. 9–16, ISBN: 9781450315609 (⇒ 238).

[24] K. Szabados, "Structural analysis of large ttcn-3 projects," in *Proceedings of the 21st IFIP WG 6.1 International Conference on Testing of Software and Communication Systems and 9th International FATES Workshop*, ser. TESTCOM '09/FATES '09, Eindhoven, The Netherlands: Springer-Verlag, 2009, pp. 241–246, ISBN: 9783642050305 (⇒ 238).

[25] K. Szabados, "Quality aspects of ttcn-3 based test systems," Ph.D. dissertation, Eötvös Loránd University, Nov. 2017 (⇒ 238).

[26] A. Moura, Y. Lai, and A. Motter, "Signatures of small-world and scale-free properties in large computer programs," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 68 1 Pt 2, p. 017 102, 2003 (⇒ 238).

[27] A. Potanin, J. Noble, M. Frean, and R. Biddle, "Scale-free geometry in oo programs," *Commun. ACM*, vol. 48, no. 5, pp. 99–103, May 2005, ISSN: 0001-0782 (⇒ 238).

[28] N. LaBelle and E. Wallingford, "Inter-package dependency networks in open-source software," *CoRR*, vol. cs.SE/0411096, 2004 (⇒ 238).

[29] G. Kohring, "Complex dependencies in large software systems," *Advances in Complex Systems*, vol. 12, Nov. 2011 (⇒ 238).

[30] C. Taube-Schock, R. J. Walker, and I. H. Witten, "Can we avoid high coupling?" In *Proceedings of the 25th European Conference on Object-Oriented Programming*, ser. ECOOP'11, Lancaster, UK: Springer-Verlag, 2011, pp. 204–228, ISBN: 9783642226540 (⇒ 239).

[31] M. A. Mamun, C. Berger, and J. Hansson, "Effects of measurements on correlations of software code metrics," *Empir. Softw. Eng.*, vol. 24, Aug. 2019 (⇒ 239).

[32] I. Herraiz, J. M. Gonzalez-Barahona, and G. Robles, "Towards a theoretical model for software growth," in *Fourth International Workshop on Mining Software Repositories (MSR'07:ICSE Workshops 2007)*, 2007, pp. 21–21 (⇒ 239).

[33] C. P. Smith, "A software science analysis of programming size," in *Proceedings of the ACM 1980 Annual Conference*, ser. ACM '80, New York, NY, USA: ACM, 1980, pp. 179–185, ISBN: 0897910281 (⇒ 239).

[34] H. Zhang and H. B. K. Tan, "An empirical study of class sizes for large java systems," in *14th Asia-Pacific Software Engineering Conference (APSEC'07)*, 2007, pp. 230–237 (⇒ 239).

[35] H. Zhang, "Exploring regularity in source code: Software science and zipf's law," in *2008 15th Working Conference on Reverse Engineering*, 2008, pp. 101–110 (⇒ 239).

[36] H. Zhang, H. B. K. Tan, and M. Marchesi, "The distribution of program sizes and its implications: An eclipse case study," *CoRR*, vol. abs/0905.2288, 2009. arXiv: 0905.2288 (⇒ 239).

[37] I. Herraiz, D. Germán, and A. E. Hassan, "On the distribution of source code file sizes," in *ICSOFT 2011 - International Conference on Software and Data Technologies*, vol. 2, Jan. 2011, pp. 5–14 (⇒ 239).

[38] R. Tonelli, G. A. Pierro, M. Ortu, and G. Destefanis, "Smart contracts software metrics: A first study," *PLoS ONE*, vol. 18, Jan. 2023 (⇒ 239).

[39] N. Bartha, "Scalability on it projects," M.S. thesis, Eötvös Loránd University, 2016 (⇒ 239).

[40] L. Hatton, "Conservation of information: Software's hidden clockwork?" *IEEE Trans. Softw. Eng.*, vol. 40, no. 5, pp. 450–460, 2014 (⇒ 239).

[41] L. Hatton and G. Warr, "Strong evidence of an information-theoretical conservation principle linking all discrete systems," *Royal Society Open Science*, vol. 6, p. 191 101, Oct. 2019 (⇒ 239).

[42] M. Lehman and J. Fernandez-Ramil, "Rules and tools for software evolution planning and management," *ASE*, vol. 11, pp. 15–44, Jan. 2001 (⇒ 239).

[43] M. M. Lehman and J. F. Ramil, "Evolution in software and related areas," in *Proceedings of the 4th International Workshop on Principles of Software Evolution*, ser. IWPSE '01, Vienna, Austria: ACM, 2001, pp. 1–16, ISBN: 1581135084 (⇒ 239).

[44] M. Lehman, D. Perry, and J. Ramil, "On evidence supporting the feast hypothesis and the laws of software evolution," in *Proceedings Fifth International Software Metrics Symposium. Metrics (Cat. No.98TB100262)*, 1998, pp. 84–88 (⇒ 239, 240).

[45] M. J. Lawrence, "An examination of evolution dynamics," in *Proceedings of the 6th International Conference on Software Engineering*, ser. ICSE '82, Tokyo, Japan: IEEE CS Press, 1982, pp. 188–196 (⇒ 239).

[46] C. Izurieta and J. Bieman, "The evolution of freebsd and linux," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ser. ISESE '06, Rio de Janeiro, Brazil: ACM, 2006, pp. 204–211, ISBN: 1595932186 (⇒ 239, 240).

[47] C. Kemerer and S. Slaughter, "An empirical approach to studying software evolution," *IEEE Trans. Softw. Eng.*, vol. 25, no. 4, pp. 493–509, 1999 (⇒ 239, 240).

[48] A. Israeli and D. Feitelson, "The linux kernel as a case study in software evolution," *Journal of Systems and Software*, vol. 83, pp. 485–501, Mar. 2010 (⇒ 239, 240).

[49] K. Johari and A. Kaur, "Effect of software evolution on software metrics: An open source case study," *SIGSOFT Softw. Eng. Notes*, vol. 36, no. 5, pp. 1–8, Sep. 2011, ISSN: 0163-5948 (⇒ 239, 240).

[50] R. Potvin and J. Levenberg, "Why google stores billions of lines of code in a single repository," *Commun. ACM*, vol. 59, no. 7, pp. 78–87, Jun. 2016, ISSN: 0001-0782 (⇒ 239).

[51] W. Turski, "The reference model for smooth growth of software systems revisited," *IEEE Trans. Softw. Eng.*, vol. 28, no. 8, pp. 814–815, 2002 (⇒ 239).

[52] J. Fernandez-Ramil, D. Izquierdo-Cortazar, and T. Mens, "What does it take to develop a million lines of open source code?" In *Open Source Ecosystems: Diverse Communities Interacting*, vol. 299, Jun. 2009, pp. 170–184, ISBN: 978-3-642-02031-5 (⇒ 239, 240).

[53] J. Waveren, "The quake iii arena bot," Jan. 2001 (⇒ 244).

[54] K. Szabados, "Parallelising semantic checking in an ide: A way toward improving profits and sustainability, while maintaining high-quality software development," *Acta Universitatis Sapientiae, Informatica*, vol. 15, no. 2, pp. 239–266, 2023 (⇒ 249).

Received: 06.09.2024; *Revised: 09.12.2024*; *Accepted: 10.12.2024*

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 255–285

DOI: 10.47745/ausi-2024-0014

The Sapientia ECN AI Baseline Index: Benchmarking Large Language Models Against Student Performance in Competitive Programming

Zoltán KÁTAI
Sapientia Hungarian University of Transylvania
Târgu Mureș, Romania
katai_zoltan@ms.sapientia.ro
0000-0003-2343-3629

David ICLANZAN
Sapientia Hungarian University of Transylvania
Târgu Mureș, Romania
iclanzan@ms.sapientia.ro
0000-0003-2587-9106

Abstract.

We introduce the Sapientia ECN AI Baseline Index, a benchmark for evaluating the fundamental problem-solving capabilities of publicly available, state-of-the-art Large Language Models (LLMs) in competitive programming. Using basic prompting techniques on problems from the annual Sapientia Efficiency Challenge Networking (ECN) competition, we assess LLMs' baseline performance, deliberately excluding more advanced enhancements like agentic systems or external knowledge retrieval.

Our initial study compares LLM results with those of student teams from the ECN 2023 competition, analyzing both the number and types of problems solved, as well as score distributions. By providing a consistent, longitudinal measure, the ECN AI Baseline Index aims to track AI baseline capability advancement in complex problem-solving domains and offers insights into the evolving strengths and limitations of LLMs relative to peak and median student expertise.

Key words and phrases: competitive programming, AI program code generation

1 Introduction

In recent years, the rapid advancement of artificial intelligence has sparked new possibilities in evaluating programming competition difficulty and participant skill

levels. Programming contests, including globally recognized events like the International Collegiate Programming Contest (ICPC) and Google Code Jam, have long served as arenas for coders to test their problem-solving and algorithmic thinking. These competitions demand not only proficiency in coding but also the ability to devise efficient algorithms under strict time constraints. Similarly, the Sapientia-ECN (Efficiency Challenge Networking) programming contest, organized by the Mathematics and Informatics Department of Sapientia Hungarian University of Transylvania, has established itself as a significant event in the region, adhering to ACM-ICPC standards and attracting high school and university teams for over 15 years. The competition provides a unique and challenging environment, allowing young programmers to showcase their skills in a structured and competitive setting.

Determining an appropriate level of problem difficulty is essential to the success of any programming competition, as it influences participant engagement, learning outcomes, and the overall fairness of the event. A well-calibrated difficulty range ensures that the contest remains accessible to participants of diverse skill levels, fostering broader participation while maintaining a high standard. Balancing easy and challenging problems not only sustains motivation among participants but also enhances the educational impact of the competition, enabling students to develop stronger problem-solving and algorithmic skills. This balance is crucial for contests aimed at educational growth, as it prevents problems from being too trivial or prohibitively complex, thereby enhancing participants' sense of achievement and satisfaction.

The increasing sophistication of large language models (LLMs) has introduced a new method for evaluating problem complexity and solution effectiveness in programming competitions. State-of-the-art language models, such as those based on OpenAI's latest offerings, now serve as a practical benchmark for assessing problem difficulty by simulating solutions generated with basic prompting techniques. These models can provide a stable, consistent baseline for comparing AI performance against that of human participants, offering organizers a quantitative approach to evaluate the complexity of contest problems. Furthermore, the qualitative aspects of AI performance are increasingly relevant, as they allow us to delve into not just the quantity of successfully solved problems but also the types of problems that different models and human participants can address. By examining the distribution of scores achieved by human competitors alongside AI results, we can gain a nuanced understanding of how AI models handle various problem types compared to human teams, thereby offering a holistic view of AI's current strengths and limitations.

This paper introduces the Sapientia ECN AI Baseline Index (EAII), a novel metric that systematically evaluates the capability of publicly accessible LLMs in solving

algorithmic challenges typical of programming competitions. By applying this index to the 2023 Sapientia-ECN contest, we aim to establish an accessible, replicable baseline for AI performance, focusing on standard, straightforward prompting methods that any user can implement. Through this comparative analysis, which includes both quantitative and qualitative evaluations, we assess the problem-solving abilities of current LLMs, their limitations, and the variety of problems they can tackle effectively. The EAII thus serves as a longitudinal tool, designed not only to track the evolution of AI capabilities in competitive programming but also to offer a unique perspective on the types of algorithmic challenges that remain difficult for AI, relative to human performance, as models continue to advance in complex problem-solving domains.

2 Background

Program synthesis aims to automate the coding process by generating programs that satisfy user-specified intents, a goal often considered the "holy grail" of computer science [1], [2]. Achieving this would significantly enhance programmer productivity and broaden access to programming. Automatically creating programs from high-level descriptions has long been a central challenge in computer science.

Developing AI systems capable of solving unforeseen problems by generating code from descriptions is multifaceted, advancing our understanding of problem solving and reasoning [3]. Solving competitive programming problems is a crucial step in this field. It requires understanding complex natural language descriptions, reasoning about novel problems, mastering diverse algorithms and data structures, and precisely implementing extensive solutions [4]. Competitive programming, which attracts hundreds of thousands of participants worldwide, provides robust benchmarks for evaluating intelligence [5].

2.1 State-of-the-Art Models for Competitive Programming

Large language models (LLMs) have demonstrated remarkable reasoning capabilities, though debates about their limitations and data contamination issues persist [6]. Recent studies focus on evaluating LLMs' abilities to solve expert-crafted, competition-level programming problems, which demand deep understanding and robust reasoning skills [6]. For example, the paper by Coignion [7] evaluates the performance of LLM-generated code on Leetcode, a popular platform for coding challenges. This research compares 18 different LLMs, analyzing factors such as model temperature and success rates. The findings indicate that LLMs can produce code with performance comparable to that of human-crafted solutions, suggesting

that LLMs are capable of handling competitive programming tasks effectively [7].

Recent advancements of state-of-the-art models have led to remarkable progress in machine assisted solving of competitive programming problems[8], with models such as AlphaCode, AlphaCode 2, and OpenAI's o1 achieving significant milestones. These systems demonstrate AI's growing capability to tackle complex coding challenges, with each model offering distinct approaches and achievements in the field.

DeepMind's initial entry into this domain, AlphaCode, established a foundation for AI systems in competitive programming by achieving rankings in the top 54.3% of participants in simulated Codeforces competitions [4]. The system's success stems from its innovative methodology of generating millions of diverse code submissions through transformer-based networks, followed by sophisticated filtering and clustering processes that select up to ten submissions for final evaluation. This approach marked a significant step forward in replicating human-like problem-solving capabilities in programming contexts.

The subsequent development of AlphaCode 2 [9] represented a substantial advancement over its predecessor. This improved system demonstrates performance levels surpassing approximately 85% of human programmers in competitive programming scenarios, positioning it between the 'Expert' and 'Candidate Master' categories on Codeforces [10]. AlphaCode 2's enhanced capabilities are attributed to several key improvements, including a more sophisticated training methodology utilizing an expanded dataset of programming challenges. The system achieves nearly double the problem-solving efficiency of its predecessor, successfully resolving approximately 43% of problems within ten attempts, compared to the original AlphaCode's 25% success rate.

OpenAI's newest model named o1 [11] seems to have has pushed the boundaries even further [12]. It is reported to achieve rankings in the 89th percentile on Codeforces [11]. The o1 model's success is rooted in its deliberate, reasoning-centered approach to problem-solving, moving beyond the rapid response mechanisms characteristic of earlier AI systems. This model distinguishes itself through its ability to refine strategies, identify potential errors, and explore alternative methods to improve accuracy. The training methodology emphasizes deep analytical thinking, enabling the model to develop sophisticated problem-solving strategies that more closely mirror human cognitive processes. Compared to its predecessor GPT-4, the o1 model demonstrates notably enhanced performance in tasks requiring precise reasoning and stepwise analysis, particularly in advanced problem domains.

These developments collectively represent a significant evolution in AI's capability to engage in competitive programming, with each successive model demonstrating increasingly sophisticated approaches to problem-solving and higher levels of performance in competitive scenarios. The progression from AlphaCode to Alpha-

Code 2 and the introduction of the o1 model illustrate the rapid pace of advancement in this field, suggesting promising directions for future developments in AI-driven programming solutions.

3 Sapientia-ECN: a challenging programming competition

The Sapientia-ECN (Efficiency, Challenge, Networking) programming contest, organized by the Mathematics and Informatics Department of the Sapientia Hungarian University of Transylvania, has been a prestigious event for more than 15 years, targeting high school and university teams in the region. This competition follows the ACM-ICPC (International Collegiate Programming Contest) format, providing an excellent opportunity for young programmers to challenge their skills and compete in a rigorous, yet rewarding environment.

In line with the ACM-ICPC style, the Sapientia-ECN competition features:

- Teams of Three: Each team consists of three members who collaborate to solve problems. This encourages teamwork and effective communication.
- Single Computer: Teams are provided with one computer, simulating a real-world scenario where resources are limited and must be managed efficiently.
- English Language Problem Set: The problem set, written in English, ensures that participants can compete on an international level and gain experience with globally
- Five-Hour Duration: Teams have five hours to solve as many problems as they can. This tests their ability to perform under pressure and manage their time effectively.

The problems are designed to be of medium difficulty, striking a balance that challenges participants while remaining accessible to a wide range of skill levels. Solutions are evaluated based on correctness and efficiency, with no partial credits awarded for incomplete or partially correct solutions. Solutions must be completely correct to earn points.

To recognize the efforts of participants at different educational stages, the competition awards prizes separately for high school and university teams. This ensures a fair comparison and encourages participation from younger students, fostering an early interest in computer science and programming.

The mission of the Sapientia-ECN competition goes beyond just being a test of programming ability. It provides participants with:

- Experience with ACM-ICPC Standards: By following the ACM-ICPC format, participants gain familiarity with one of the most prestigious programming competitions worldwide, preparing them for future contests at higher levels.
- Networking Opportunities: The event brings together young programmers from various institutions, promoting networking and the exchange of ideas.
- Skill Development: The problems are crafted to improve algorithmic thinking, problem-solving skills, and programming proficiency, essential traits for future careers in technology and research.

In Conclusion the Sapientia-ECN programming competition stands out as a significant event in the regional academic calendar. It not only fosters a competitive spirit among young programmers but also prepares them for larger stages like the ACM-ICPC. By participating, students gain invaluable experience, develop critical skills, and become part of a vibrant community of problem solvers.

4 Annual AI Performance Evaluation

The ECN AI Baseline Index (EAII) is constructed as a relative performance index that compares the easily and consistently achievable performance of AI systems in competitive programming contests against the peak and median performances of student teams. Each year, the Index uses publicly available state-of-the-art models, accessible to any user, ensuring the performance is replicable without advanced configurations, techniques or custom system setups.

4.1 Design of an Achievable Baseline

EAII is not intended to represent the highest possible AI performance with state-of-the-art techniques like retrieval-augmented generation (RAG) or agentic systems. Instead, it focuses on a baseline score, one that any user could obtain using simple and repeatable methods. In each evaluation cycle, the chosen models are used "as-is", with basic prompting. This establishes a fair and comparable standard over time, ensuring that advancements in the metric reflect changes in general AI accessibility and capability, rather than specialized, advanced techniques or very resource-intensive processes.

The prompt used to assess the AI models is minimal yet effective, constructed to solicit a competent solution without additional guidance:

> *As an expert competitive programmer, write a very efficient Python 3 solution for the given problem. Read inputs from standard input and write outputs to standard output.*

This prompt is followed by the problem statement as provided in the ECN contest. By using a simple and consistent prompt, we ensure that the AI's performance is based on its inherent capabilities without the influence of intricate prompting strategies and prompt engineering.

4.2 Limiting AI Solution Attempts for Fair Comparisons

AI systems can generate code rapidly and could potentially generate a very large number of solutions until one succeeds. To mitigate this advantage we implement a controlled evaluation procedure comprising two stages.

Local testing: The solution provided by the AI is first tested locally using the example inputs and outputs provided in the problem statement. If the solution produces errors, such as runtime exceptions or incorrect results, a feedback loop is created in the form of the error messages or observed incorrect behavior. If the proposed solution involves precomputation, a technique commonly used in competitive programming, the LLM is asked to separate the precomputation script and indicate where the results should be placed in the final solution. The AI is allowed a maximum of three rounds of error correction or enhancements at this stage.

Contest evaluation system: If the solution passes the local test, it is then submitted to the automatic grading system of the contest. If the solution fails the system's checks, one final feedback loop is provided, where the AI is informed of the failed test cases, along with standard error messages such as "Time Limit Exceeded" or "Wrong Answer". Only one round of feedback is allowed at this stage, limiting the AI's optimization opportunities to maintain comparability with human competitors.

By constraining the number of feedback rounds and aligning the evaluation process with the standard contest conditions, we aim to limit the AI's potential advantage derived from rapid iteration and multiple attempts. This approach ensures that the AI's performance is measured under conditions comparable to those experienced by human teams, who also have limited opportunities to debug and resubmit solutions within the contest time frame.

4.3 Baseline AI Performance

PT_{AI} denotes the total number of problems the AI successfully solves according to the conditions detailed above. A solution is considered successful only if it passes all test cases in the automated judging system of the contest. A direct relative comparison can be used to evaluate AI performance in relation to student performance.

Relative Performance to Median (RPM):

$$\text{RPM} = \frac{PT_{\text{AI}} - \text{Median}(PT_{\text{students}})}{\text{Median}(PT_{\text{students}})} \times 100 \tag{1}$$

where:

- PT_{students} is the set of total problems solved by each student team.
- $\text{Median}(PT_{\text{students}})$ is the median number of problems solved by student teams.

Relative Performance to Peak (RPP):

$$\text{RPP} = \frac{PT_{\text{AI}} - \text{Peak}(PT_{\text{students}})}{\text{Peak}(PT_{\text{students}})} \times 100 \tag{2}$$

where $\text{Peak}(PT_{\text{students}})$ is the maximum number of problems solved by any student team.

These metrics provide insights into how AI performance compares with both median and peak student performance, offering a multifaceted analysis of its capabilities in relation to different benchmarks.

4.4 EAII Definition

In addition to RPM and RPP, which reflect AI's position relative to specific student performance points, the ECN AI Baseline Index (EAII) is designed to offer a comprehensive assessment by considering both median and peak student performances. The EAII integrates these perspectives to provide a single, unifying metric:

$$\text{EAII} = \frac{PT_{\text{AI}} - \text{Median}(PT_{\text{students}}) + 1}{\text{Peak}(PT_{\text{students}}) - \text{Median}(PT_{\text{students}}) + 1} \times 100, \tag{3}$$

The EAII measures the AI's performance relative to the range between the median and peak student performances, expressed as a percentage. Laplacian smoothing ensures that the denominator is not zero in cases of uniform performance between teams.

4.5 Characteristics of the Metric

EAII provides a normalized scale, being a dimensionless quantity that normalizes the AI's performance between the median and peak student performances. This normalization facilitates comparison across different contests and time periods, even when the number and difficulty of problems might vary. As a relative performance indicator, an EAII of 0% indicates that the AI's performance is equivalent to the median student team, while an EAII of 100% signifies parity with the top-performing student team. Values exceeding 100% imply that the AI outperformed the best student team, whereas negative values indicate performance below the median.

The EAII metric becomes more sensitive when student performance is tightly clustered. When the top and median teams score similarly, the performance deviation between the student teams is low. Consequently, each problem the AI solves or fails to solve causes a larger shift in its EAII score. This is because the metric normalizes AI performance against the spread of student scores. With a narrow spread, even small changes in AI performance result in dramatic changes to its relative position and final EAII score.

5 Experiments

Since AlphaCode 2 is not publicly accessible, we chose to conduct our experiments using the available preview of OpenAI's new model, code-named o1-preview [1]. Unlike AlphaCode 2, which is proprietary to Google DeepMind, the o1-preview model was readily accessible via API calls. Consequently, in our experiments with ECN 2023 problems, we used o1-preview as the engine for the ECN AI 2023 (EAI2023) solver.

5.1 ECN 2023 edition

The 2023 edition of the Sapientia-ECN competition took place on November 20, bringing together 20 teams from Romania and Hungary, including 13 university teams and 7 high school teams. Participants faced a diverse set of 14 problems, ranging from relatively straightforward to highly challenging. The tasks spanned a variety of topics, with a particular focus on dynamic programming (Tasks A, F, I, N), classical algorithmic challenges (Tasks C, D, E, G, J), and problems involving mathematical concepts or data structures (Tasks B, H, K, L, M). Several problems required advanced techniques and optimizations, such as binary search (Tasks A,

[1] https://openai.com/index/introducing-openai-o1-preview/

M) and precomputations (Tasks E, M). The contest demanded not only strong algorithmic skills but also efficient implementations, posing a significant challenge even to the most experienced competitors.

5.2 ECN 2023 problems

Below we present the essence of the proposed tasks. For the exact problem statements visit the official website of the Sapientia-ECN programming competition [2].

5.2.1 Problem A: Gifts

Objective: Determine the optimal node in an induced subtree to add to a given list of 'generator nodes' to get the closest possible sum to a specified value.

- Tree Structure: An undirected tree with N vertices, where Fanurie visits each node in DFS order starting from node 1 and leaves gifts of increasing values.
- Subtree Induction: For each query containing a list of 'generator nodes', determine the minimal subtree that includes each node in the list (the subtree of the DFS tree rooted at the lowest common ancestor of the nodes in the list) and identify the node in the induced subtree that, when added, brings the list's total gift value closest to a given sum S.
- Constraints: Multiple queries with varying K (number of generator nodes in the list) and S (target sum), ensuring $K + 1$ nodes in the induced subtree.
- Output: For each query, the optimal node to add, ensuring it is not already in the list and choosing the smallest node in case of ties.

5.2.2 Problem B: Colors in Store

Objective: Determine the price each customer will pay for a tablecloth that matches their color preference or indicate if no matching tablecloth is available.

- Each tablecloth has a unique price and two color attributes (front and back) represented by integers from 1 to 3.
- Customers have a single favorite color and will purchase the cheapest available tablecloth with that color on either side.

[2] https://ecn.ms.sapientia.ro/problems.php

- Customers are served sequentially, and each purchase removes a tablecloth from the available options.
- For each customer, record the price paid or indicate if no suitable tablecloth is available.

5.2.3 Problem C: Golden Primes

Objective: Write a program to identify all values of n for which p, a golden prime, is less than a given threshold b.

- Golden Prime Definition: A golden prime is a prime number of the form $p = \phi^2 - \phi - 1$, where $\phi = 2^n$ and n is a positive integer.
- Input: A single positive integer b (where $b < 2^{1000}$), serving as the upper bound for p.
- Output Requirements: Print each value of n (one per line) for which p is a golden prime and less than b.

5.2.4 Problem D: Egyptian Fractions

Objective: Develop an algorithm to compute all possible Egyptian fraction representations for a given fraction, using the fewest possible terms.

- Egyptian Fraction Definition: An Egyptian fraction is a representation of a fraction as a sum of distinct unit fractions (fractions with numerator 1), such as $\frac{17}{70} = \frac{1}{5} + \frac{1}{24} + \frac{1}{840} = \frac{1}{7} + \frac{1}{10}$
- Input: Several lines, each containing a fraction in the form $\frac{a}{b}$, where a and b are positive integers and $0 < a < b$.
- Output Requirements: (i) Each line should display the minimum number of unit fractions in the Egyptian representation, followed by the denominators in lexicographically ordered parentheses; (ii) If multiple minimal representations exist, list each distinct set of denominators in lexicographical order.

5.2.5 Problem E: F. Lanovka – The cable car

Objective: Determine the minimum number of units needed to increase the heights of selected peaks to install a cable car system of length K with smoothly ascending columns in the Vector Vista Mountains.

- Peak Requirements: There are N peaks in total, and columns need to be installed on K consecutive peaks.
- Column Heights: The highest point of the cable car columns (peak + column) should reach height H, with subsequent heights on the remaining $K-1$ columns descending by 1 unit each (i.e., $H-1$, $H-2$, ..., $H-K+1$).
- Direction of Installation: The cable car path starts from the rightmost peak and goes leftward.

5.2.6 Problem F: Kings Again

Objective: Calculate the number of ways to place K kings on an $N \times M$ chessboard without them attacking each other.

- Chessboard Rules: Kings can attack each other if they are on adjacent cells (horizontally, vertically, or diagonally).
- Input: Multiple test cases, each specifying the values of N, M, and K.
- Constraints: Each test case requires calculating the number of valid placements modulo $10^9 + 7$.
- Output: For each test case, the number of ways to place the K kings.

5.2.7 Problem G: Breaking a Quantum Cryptography Machine

Objective: Analyze intercepted cryptographic messages and their solutions to deduce the operating principles of an enemy quantum cryptography machine and reimplement its functionality in C/C++.

- Intercepted Messages: Access to captured messages along with their verified solutions, provided by the Secret Service.
- Objective of Analysis: Identify and understand the underlying encryption principles used by the quantum cryptography machine based on the given examples.

5.2.8 Problem H: "Optimizing" Ascent: Navigating Bear-Dangerous Territory in a Binary Matrix Mountain

Objective: Calculate the minimum number of steps taken through dangerous (internal) regions to ascend from the base to the summit of a multi-level "mountain" represented by concentric binary matrices.

- Mountain Representation: The mountain has n levels, each encoded as an $m \times m$ binary matrix where: (i) A value of 1 represents mountain points; (ii) Each level's 1s form a connected region, including boundary points.

- Concentric Structure: Levels are "concentric", meaning if a point is 1 at level i, it is also 1 at level $i-1$, for $i = 2, \ldots, n$.

- Summit Configuration: The top level (last matrix) has a single 1, marking the mountain peak.

- Climbing Constraints: (i) Start at any boundary (edge) point on level 1; (ii) Move from the edge of each level to the next level's edge, aiming to reach the summit in the fewest steps through internal points; (iii) Moving between levels or along any edge is considered safe, as bears avoid these regions.

5.2.9 Problem I: Hamilton

Objective: Calculate the shortest route Hamilton, Santa's favorite elf, must take to deliver packages to all the mailboxes on a specific stretch of road.

- Coordinates: The road is 10000 meters long and 15 meters wide. Each mailbox has known coordinates (x, y) where $0 \leq x \leq 10000$ and $0 \leq y \leq 15$. The x-coordinates (abscissas) are unique for each mailbox, but the y-coordinates (ordinates) may repeat.

- Movement Constraint: Hamilton must traverse the mailboxes such that their x-coordinates first continuously increase and then continuously decrease, ensuring an efficient round trip.

- Start and End: The route starts at the first mailbox, covers all mailboxes, and returns to the starting point.

- Output: A single real number representing the length of the shortest route, printed with exactly six decimal places.

5.2.10 Problem J: Find the Identical Twins and Triplets

Objective: Identify and display groups of identical twins or triplets from a population database ($n \leq 160000$) who share the same DNA signature, including only those groups where at least one member is adopted.

- Database Structure for each individual: Personal Code (a unique 31-bit positive integer); DNA Signature (a string of up to 11 uppercase letters); Adoption Status (character '*A*' for adopted, '–' otherwise); Name (up to 27 characters, may include whitespace).

- Unique Code Assignment: Consecutive personal codes in ascending order may have gaps, especially after assigning codes to twins or triplets. (Twins and triplets with the same DNA signature are guaranteed to appear consecutively in the sorted database by personal code)

- Grouping Requirement: (i) Identify groups of individuals with the same DNA (identical twins or triplets); (ii) Display only those groups containing at least one adopted member.

5.2.11 Problem K: Dependencies

Objective: Develop a program to determine if specified sequences of AWS resources, as defined in a CloudFormation template with dependencies, can be created in the given order. Identify any problematic resources that cannot be created due to dependency constraints.

- The input includes

  - R ($1 \leq R \leq 100$) lines with resources and their dependencies in the format: Resource ID (alphanumeric, up to 20 characters) followed by a colon (":"), and then a space-separated list of its dependencies;
  - S ($1 \leq S \leq 100$) lines with space-separated resource IDs representing the order in which resources are intended to be created.

- Output Requirements:

  - If all resources can be created in order, print "No problematic resources".
  - If some resources cannot be created in order: (i) For one problematic resource, print "Problematic resource: " followed by the resource ID; (ii) For multiple problematic resources, print "Problematic resources: " followed by the space-separated list of all problematic resource IDs in the order they appear in the sequence.

5.2.12 Problem L: Windmill-lottery

Objective: To simulate a national lottery system in "Windmillia" and output the result of each draw based on specified rules. The simulation involves generating numbers and performing draws as per certain calculations, with numbers being assigned to two lines ($Line_1$ and $Line_2$) or used in a draw depending on modular conditions.

- Line and Number Generation:
  - Numbers are generated based on provided formulas involving constants a, b, c, and d.
  - Line assignment is determined by another formula with constants L_a, L_b, L_c, and L_d.
  - If $Line_n$ value is 1 or 2, $Number_n$ is assigned to the corresponding line. If $Line_n$ is 3, a draw is performed if the two lines have the same parity of count.
- Draw Process:
  - Condition for Draw: A draw can only occur if both $Line_1$ and $Line_2$ contain an even or odd number of elements.
  - Sorting: Both lines are sorted before the draw.
  - Rotation: $Line_2$ is rotated by rotationCount based on the calculated middle point.
  - Offset Calculation: An offset $Offset_i$ is calculated to find specific positions from the middle point in each line.
  - Winning Number: Using the middle point and offset, two values are summed (taking 0 if a line is too short), and their modulo m is taken as the winning number.
- Special Calculations:
  - Middle Point Determination: Adjustments are made for odd/even counts in each line.
  - Modulo m: Where m is the maximum value currently present across both lines.

5.2.13 Problem M: Modpute

Objective: To simulate the Modputer machine's behavior and calculate the total number of updates applied to its internal state array after processing a sequence of input integers. Each update occurs when an element in the state array is divisible by the current input integer.

- Internal State Array (A):
  - The machine's internal state is represented by an array A of N positive integers.
  - Each element in this array is subject to updates based on divisibility checks against the input integers.
- Input List:
  - A sequence of M positive integers, each greater than 1, is processed one by one.
  - For each integer D in this input, the machine checks each element in A to see if it is divisible by D.
- Update Mechanism:
  - For each integer D in the input: If an element $A[i]$ in the state array is divisible by D, it is incremented by one.
  - The operation is repeated for each integer in the input list, potentially causing multiple updates to each element in A.
- Count of Updates: Track the number of updates for each element in A and calculate the total number of updates across the entire array.

5.2.14 Problem N: Carpathian Riders

Objective: Plan an optimal motorcycle trip across the Carpathian Mountains on a grid of elevations.

- Grid Structure: The grid has R rows and C columns, with elevation values between 0 and 1000, and impassable cells marked as -1.
- Movement: The gang can move east, northeast, or southeast, starting from any passable cell on the western edge and aiming to exit on the eastern edge.

- Constraints: The trip must visit exactly P mountain passes, where a mountain pass is a cell with a higher elevation than its eastern and western neighbors and a lower elevation than its northern and southern neighbors.
- Goal: Minimize the sum of elevations along the trip while meeting the pass constraints.
- Output: The minimum sum of elevations for a valid trip or "impossible" if no such trip exists.

6 Results and Discussion

Metric	Value
Students Maximum Problems Solved ($\mathrm{Max}(PT_{\text{students}})$)	8
Students Minimum Problems Solved ($\mathrm{Min}(PT_{\text{students}})$)	0
Students Mean Problems Solved (Student mean PT)	2.47
Students Median Problems Solved ($\mathrm{Median}(PT_{\text{students}})$)	2
Students PT Standard Deviation ($\mathrm{STD}(PT_{\text{students}})$)	2.34
EAI2023 Total Problems Solved (PT_{AI})	10
EAI2023 Solved Percentage (SP)	71.42%
EAI2023 Partially Solved	3
Relative Performance to Median (as defined in Equation 1)	400%
Relative Performance to Peak (as defined in Equation 2)	25%
2023 ECN AI Baseline Index (as defined in Equation 3)	**128.57%**

Table 1: Key metrics

Performance statistics and key metrics are presented in Table 1. The problem-solving rates for each student team and the outcomes of EAII2023 are shown in Table 2. Additionally, the distribution of the number of problems solved by student teams is illustrated in Figure 1.

The problem-specific solving rates are visualized in Figure 2, which indicates varying levels of difficulty across different problems. Problems A and B were solved exclusively by student teams, while Problems C and D were solved solely by the AI system.

The box plot in Figure 3 provides a statistical summary of student performance, and Figure 4 reveals the correlations between different problems.

The results of this study underscore distinct problem-solving patterns between student teams and the EAI2023 AI system, particularly in terms of problem-specific

Problem	EAI2023 Solved	Student Teams Solved	Percentage of Teams Solved
A	Partially	1	5.26%
B	Partially	6	31.57%
C	Yes	0	0%
D	Yes	0	0%
E	Yes	7	36.84%
F	Yes	1	5.26%
G	Yes	16	84.21%
H	Yes	2	10.52%
I	Yes	2	10.52%
J	Yes	3	15.78%
K	Yes	4	21.05%
L	No	0	0%
M	Partially	0	0%
N	Yes	5	26.31%

Table 2: Problem-solving rates

strategies and advanced techniques. Metrics such as Relative Performance to Median (RPM) and Relative Performance to Peak (RPP) allow us to quantitatively assess the strengths of EAI2023 relative to student performance. Furthermore, qualitative analysis offers additional insights into how the unique requirements of each task shaped the outcomes, revealing important pedagogical implications for training in competitive programming.

6.1 Quantitative Analysis of Performance

As shown in Table 1, EAI2023 demonstrated outstanding performance, outperforming the median student team by an impressive 400% (RPM, Equation 1) and excelling across nearly all tasks. With a record-breaking 10 solved tasks, it exceeded the peak performance of the top student team by 25% (RPP, Equation 2), further cementing its reputation as a highly capable and effective problem-solving system.

The problem-specific analysis in Figure 2 reveals significant variances in difficulty across problems. Problems such as C and D, which were solved exclusively by EAI2023, highlight areas where the AI may have distinct advantages in systematic problem-solving under strictly defined test conditions. Conversely, Problems A and B, which were partially solved by students but not fully by EAI2023, may indicate situations where students displayed creative or heuristic-driven problem-solving abilities that the AI did not replicate fully. Problem G stands out as the most

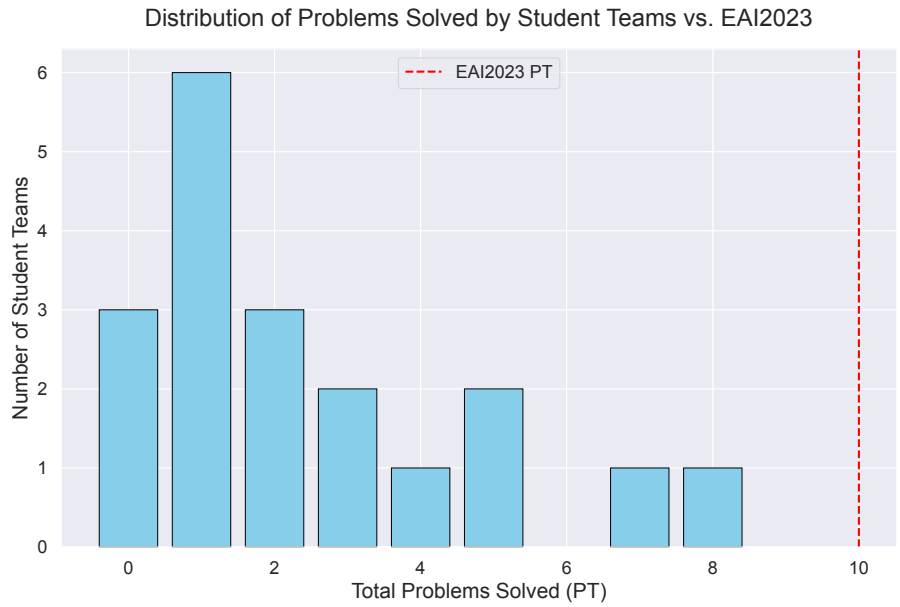


Figure 1: Distribution of total problems solved by student teams compared to EAI2023. The red dashed line indicates EAI2023's performance.

frequently solved problem among students (84.21% of teams), suggesting that certain types of problems align more closely with human problem-solving strategies than with AI-driven methods.

Figure 3 further illustrates the distribution of total problems solved by students, with the red dashed line marking the AI's score of 10 problems. The spread of student performance, reflected in a standard deviation of 2.34, points to a wide range of competencies among participants. This variability, combined with the solid performance of AI, suggests potential opportunities for future AI improvements aimed at bridging the gap in partially solved or creative problem types.

Finally, Figure 4 provides an overview of problem-solving correlations, though these patterns should be interpreted with caution due to the small sample size. For example, the observed perfect correlation between Problems H and I likely stems from these problems being solved only by the winning team, rather than indicating a significant general trend. Nevertheless, exploring such correlations in larger datasets could offer insights into how certain types of problems may cluster or interact, potentially guiding further AI development.

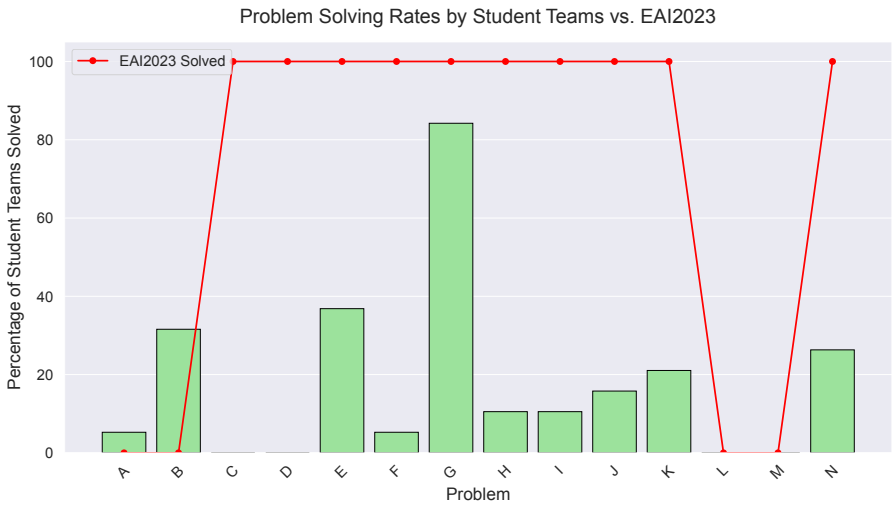


Figure 2: Problem-specific solving rates comparing student teams' performance with EAI2023. The green bars represent the percentage of student teams that solved each problem, while the red line shows whether the EAI2023 solved the problem or not.

6.2 Qualitative Analysis of Performance

The efficient solution of four tasks (A, F, I, N) required the application of a specific programming technique: dynamic programming (DP). Notably, the AI correctly identified the need for dynamic programming in all instances. For task N, the AI provided a valid solution in the first round, but it fell short in terms of speed for one test case, likely due to generating Python code, which is generally slower than languages like C. The AI correctly determined that a 3D DP array $dp[r][c][p]$ should be used, where r represents the row index, c the column index, and p the number of passes visited so far. In round 2, aiming to optimize performance, the AI tried to enhance the solution by implementing Dijkstra's algorithm. However, the Python version of this approach was even slower, resulting in a "time limit exceeded" error for two test cases. Learning from this, the AI returned to the dynamic programming approach and chose a more efficient data structure in Python, which ultimately resulted in a solution that was accepted for all test cases. Among the teams, five successfully solved this problem, with one team submitting a correct solution on their first attempt.

Task I was a specific variation of the Traveling Salesman Problem (TSP), known as the Bitonic TSP. Although this was not explicitly mentioned in the problem de-

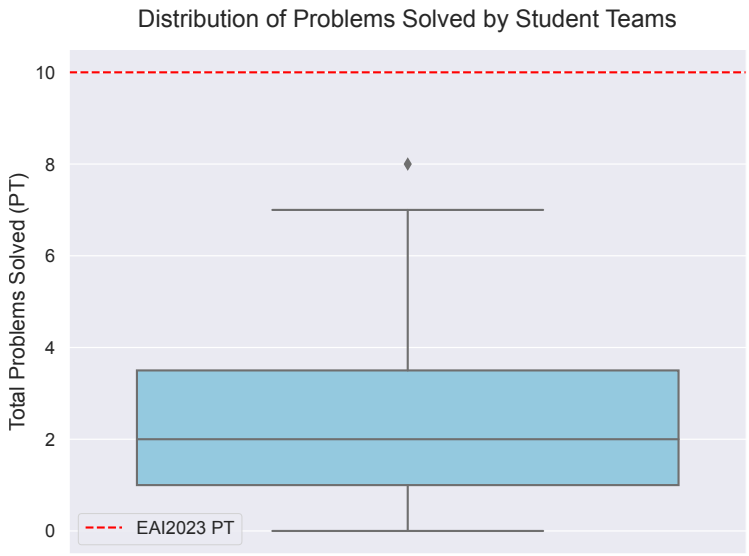


Figure 3: Box plot showing the distribution of total problems solved by student teams. The red dashed line indicates EAI2023's score, with 10 problems solved.

scription, the AI quickly recognized the nature of the problem. In the first round, it correctly formulated the recursive equations for computing the Bitonic Hamiltonian Paths but made an error when extracting the optimal length of the Bitonic Hamiltonian Cycle from the completed DP table. The AI recognized that it was dealing with a two-dimensional DP problem and accurately identified the general structure of the bitonic path subproblems: $dp[i][j]$ represented "the minimal total distance of a path that starts at mailbox 0 (leftmost), goes to mailbox i, then to mailbox j, visiting all mailboxes from 0 to j exactly once". When populating the DP array, the AI correctly distinguished between two cases: when i and j are consecutive mailboxes in the sorted order, and when they are not. In the second round, it successfully fixed the bug by ensuring that, when calculating the minimal total distance (the length of the Bitonic Hamiltonian Cycle), it considered all possible bitonic Hamiltonian paths ending at the rightmost point, not just those starting from the leftmost point. Only two teams managed to solve this problem—the first and second place teams—both on their first attempt.

Task F was also straightforward for the AI to recognize, as it involved a classic chessboard problem: determining the number of ways to place k kings on an $n \times m$ chessboard without them attacking each other. In the first round, the AI employed a similar approach to the one proposed by the task's author, who sug-

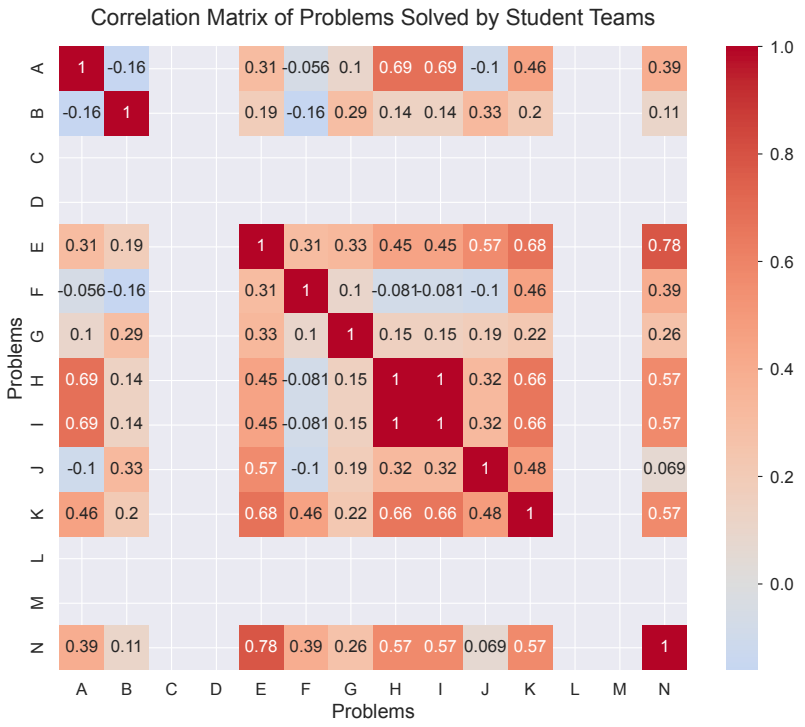


Figure 4: Correlation matrix showing the relationships between different problems. Darker red colors indicate positive correlations, while darker blue colors indicate negative correlations. The values range from -1 (perfect negative correlation) to 1 (perfect positive correlation).

gested an $O(n^2m^2 2^{2m})$ solution. However, this approach alone was not sufficient to meet the time limit. The author applied a clever trick, embedding precomputed results into the code. After we informed the AI of this optimization, it successfully implemented the trick. Interestingly, one of the competing teams devised a faster solution ($O(nmkC^2 + tC)$), t: number of test cases) that did not rely on the precomputed trick. Moreover, after the competition, a member of the competition committee further optimized this solution.

For task A, efficiently solving one of the subtasks required the use of dynamic programming, specifically through the technique of binary lifting—a method that was not widely known among the competitors. (Only the team trained by the author successfully solved this task.) Despite this, the AI immediately recognized the need for binary lifting to calculate the Lowest Common Ancestor (LCA) and applied it correctly. However, it misunderstood the task's definition of the "subtree induced

by a list of nodes", leading to an incorrect solution in the first round. In the second round, while the AI identified its earlier mistake, it failed to recognize that this correction opened up an opportunity for further optimization. Instead, it modified its previous approach to fit the new understanding, resulting in an algorithm that still wasn't fast enough. The AI missed the fact that the nodes of the induced subtree form a consecutive segment in the DFS-ordered node list, and as a result, the corresponding value list was sorted in increasing order, making it possible to apply a binary search algorithm.

Two tasks required a solid mathematical background. Task C involved a seemingly simple prime number test but for extremely large numbers. The author provided a Python solution for this task, a language not permitted in the competition, and relied on the $isprime$ function, which is known to use a probabilistic approach beyond a certain threshold. This made the task's inclusion in the competition somewhat controversial. In the first round, the AI also used the $isprime$ function. However, since the competition's official judging system, although allowing Python submissions outside the competition, did not support the $sympy$ library (which $isprime$ depends on), we informed the AI of this limitation. In the second round, the AI explicitly implemented the Miller-Rabin test with 20 iterations for reliability, producing code that returned an "accepted" result for all test cases. Unsurprisingly, none of the teams solved this task, and understandably, only those with limited competition experience attempted it.

For task D, the AI quickly recognized the need to dynamically generate a rooted tree and search for the shortest path from the root to a solution leaf. The author's solution used a BFS search since the goal was to find the shortest decomposition. The main challenge was limiting the number of branches at each node. The author applied a clever trick, based on mathematical results vaguely hinted at in the problem description: estimating the maximum depth of the tree with a relatively small value and using this estimate to determine the maximum number of branches each node could have. Interestingly, the AI also realized that the solution depended on determining the maximum possible denominator for the next term based on the estimated tree depth. However, instead of guessing this limit, the AI repeatedly generated the tree and performed searches at increasing depths —1, 2, and so on— until it found a solution. In this approach, it was natural for the AI to initiate a DFS search in each iteration. The AI's code passed all test cases on the first attempt. During the competition, only one team attempted this task, but they were unsuccessful.

Many tasks required the use of appropriate built-in data structures, combined with efficient access algorithms like binary search, to achieve correct, efficient, and elegant solutions. These tasks were particularly well-suited to experienced competitors but also aligned with the AI's strengths. For instance, task B naturally called

for a set, task E for an interval or segment tree, task G for a stack, task L for a multiset, and task M for a vector of sets, among others. Another common feature across several tasks was that certain precomputations significantly accelerated the algorithms. Examples include task E (precomputing sums and maximums for all consecutive fixed-length segments), task I (precomputing and storing all Euclidean distances), and task M (precomputing and storing all divisors to avoid repeated divisibility checks). These tasks favored both seasoned competitors and the AI alike.

Task G was the most frequently solved problem, with 16 out of 19 teams completing it. The challenge was to recognize that the problem revolved around the stack data structure, though it was presented in a coded form based on the given examples. Teams had to implement this solution, either by using the built-in stack structure or by directly coding it, for instance, with an array. Unsurprisingly, the AI successfully solved this task on its first attempt.

Task K was fairly straightforward, as it required implementing the algorithm directly from the problem description. No advanced optimizations were necessary due to the relatively small input size. Three teams—all of them podium finishers—worked on this task, with each solving it correctly. Both the first- and second-place teams succeeded on their first attempt, as did the AI. It is likely that the highly technical nature of the problem statement discouraged other teams from attempting it, as it appears they did not even engage with the task (at least, no solutions were submitted).

Another task that the AI solved on its first attempt was Task H. The author's proposed solution used a modified version of the Lee algorithm, which is generally well-known among experienced competitors. Due to the relatively small input size, no additional optimizations were needed to meet the time limit. The AI similarly employed a BFS-based approach, which is the core of the Lee algorithm. The top two teams also solved this task successfully on their first submissions. Meanwhile, one other team attempted it, submitting their code 12 times unsuccessfully—making it the most attempts for any task in the competition.

Task J presented a relatively simple challenge. The key requirement was efficient sorting, and once the array was sorted, the solution could be easily extracted, as the relevant items would naturally align next to each other. Despite 11 teams attempting this problem, only 3 succeeded in scoring points. It was later revealed that there was an error in the input file, and depending on the chosen strategy, teams either encountered or bypassed this issue. In the first round, the AI overlooked a crucial detail in the input specification, leading to faulty code. Although it corrected this in the second round, it once again missed the opportunity to leverage the benefits of a proper understanding of the problem, as it had with Task A. Instead of optimizing its approach, it merely extended the previous method to the new situation.

Nonetheless, the AI's proposed solution, though unnecessarily complex, ultimately proved correct and fast enough.

Task B was attempted by nearly every team (17 in total), but only 6 managed to receive an "accepted" result. The author sorted the tablecloths into three sets, as their fronts or backs could only fall into one of these categories. Interestingly, the AI also identified this as a natural approach, but instead opted for a more complex Python data structure in the hopes of creating a more efficient algorithm. However, it became overly complicated and failed to fix the solution even in the second round.

Task E was solved by 7 teams, making it the second most-solved problem. The author used an interval tree to efficiently find the maximum in different segments. The AI, however, provided a sufficiently fast solution by cleverly utilizing the *deque* data structure. While it made an error during the first round's maximum calculation, it corrected the mistake in the second round.

Tasks L and M were not inherently difficult but rather complex in their formulation. The convoluted wording of Task L may have discouraged participants, as no submissions were made for this problem. The author's solution involved implementing a custom heap structure, though the competition committee later introduced simpler alternatives, one of which relied on the built-in *multiset* data structure. For full context, it's worth noting that the creators of these alternative solutions had to consult the original task author due to misunderstandings or incorrect interpretations of the task description. This ambiguity may explain why the AI-generated code failed to produce output and likely contributed to teams' hesitation to engage deeply with the problem, resulting in no submissions.

For Task M, a straightforward naive solution was relatively easy to implement, but the challenge lay in optimizing each subtask and applying various advanced implementation techniques. One such technique involved precomputing divisors, while another clever approach by the task author involved storing the positions or indices of input array elements in sets to enable efficient retrieval through binary search. The AI's initial solution produced incorrect results, even for the example provided in the problem description. In the second attempt, the AI corrected this issue, but its algorithm was still too slow for 3 out of 9 test cases. It seems that the AI's optimization strategies—such as associating frequency arrays with both input arrays and using the Sieve of Eratosthenes to precompute the smallest prime factor for all numbers up to MAX (300,000 in this case) to accelerate factorization—were insufficient in terms of both efficiency and precision. Only one prize-winning team managed to submit a solution for this task, but it was ultimately unsuccessful. Other teams that attempted the task multiple times, also without success, generally ranked lower on the leaderboard.

6.3 Pedagogical Insights and Lessons Learned

The analysis of AI solutions, task author approaches, and team performances reveals important pedagogical insights, especially valuable for developing effective competitive programming training. These insights can help shape training strategies and highlight common pitfalls to avoid.

6.3.1 Technique Familiarity vs. Flexibility in Approach

Certain tasks required advanced techniques like dynamic programming for tasks A, F, I, and N, and binary lifting specifically for task A. While both the AI and the task author correctly applied binary lifting for task A, only the author-trained team among competitors used it, underscoring the need for explicit training in specialized algorithmic strategies. Familiarity with less commonly known methods like binary lifting, which can bring considerable efficiency gains, is essential for competitors aiming to improve performance.

6.3.2 Programming Language Constraints and Optimization Awareness

Task C, which required large prime number tests, highlighted the impact of programming language limitations. Both the task author and AI initially used Python's *isprime* function from the *sympy* library, which was unavailable in the competition's environment. The AI's pivot to a custom Miller-Rabin test implementation emphasizes the importance of preparing students to handle unexpected constraints and limitations. Competitors should practice adjusting strategies based on language-specific features and know when to implement fallback methods.

6.3.3 Strategic Use of Precomputation and Data Structures

Tasks E, I, and M demonstrated the substantial advantages of precomputing values to optimize complex calculations. While the AI often identified opportunities for precomputation, the majority of teams did not, highlighting the need for training in recognizing precomputation opportunities early in problem-solving.

Additionally, task-specific data structures played a vital role in solution optimization (e.g., *sets* for task B, *deques* for task E). While the AI generally applied these structures effectively, many teams did not. For students, mastering data structures and quickly assessing their applicability in different scenarios can significantly enhance performance and reduce complexity.

6.3.4 Error Recovery and Revisiting Problem Understanding

In tasks A and J, the AI initially misunderstood problem requirements. Although it corrected these errors, it continued refining its original flawed approach instead of reassessing the solution with the clarified requirements. This tendency, common among learners, to stick to an initial solution path can hinder progress even after gaining new insights. Training should encourage students to consider a ”reset” approach once key misunderstandings are resolved, promoting a fresh perspective that could reveal simpler or more efficient solutions.

6.3.5 Clarity and Ambiguity in Problem Statements

Tasks L and M demonstrated how complex or ambiguous problem wording can be a significant barrier. Both competitors and the AI struggled with interpretation, especially for tasks requiring custom structures or specific mathematical insights. This underscores the importance of training competitors to deconstruct complex descriptions, focus on core requirements, and approach ambiguous problems with resilience, enabling them to tackle even the most challenging wording effectively.

6.3.6 Reinforcing Mathematical and Theoretical Background

Some tasks required specialized mathematical insights, such as understanding the bitonic Traveling Salesman Problem for task I or efficiently testing large prime numbers in task C. Few competitors attempted these problems, which highlights the value of strengthening students' mathematical reasoning and theoretical knowledge as part of competitive programming training. Advanced-level training should incorporate more complex mathematical challenges to develop these critical skills among competitors.

6.4 EAII Advatnages and Limitation

EAII offers several advantages for longitudinal analysis. By normalizing performances, the EAII ensures comparability over time, remaining consistent across contests with differing problem sets and participant pools. This consistency enables meaningful longitudinal comparisons of AI capability advancements. Using the median student performance enhances robustness to outliers, reducing the influence of extreme values and skewed distributions and providing a more stable reference point, especially in datasets with small sample sizes. The metric also provides a compact insight, capturing the AI's relative standing within the spectrum of student performance and offering insights into both average (median) and peak human

capabilities. EAII's percentage scale enhances interpretability, facilitating straightforward communication of results.

EAII has certain limitations. It is dependent on the student performance distribution and sensitive to factors unrelated to the AI's capabilities, such as changes in participant demographics or preparation levels. In contests with a limited number of student teams, the median and peak may not accurately represent typical or maximum human performance, potentially skewing the EAII due to the impact of small sample sizes. The metric assumes a linear relationship between the median and peak performances, which may not capture the nuances of performance distributions, particularly if the spread is uneven or significant gaps exist between top performers and others. Additionally, the EAII focuses on two specific points—the median and peak—and does not account for the full distribution of student performances, which might provide additional context.

The EAII is inherently influenced by the total number of problems presented in a contest, as the sheer volume can affect the performance outcomes for both AI and human teams. In instances where a large number of problems are available, human teams may struggle to address all of them within the five-hour time limit of the contest. This limitation could inadvertently advantage the AI.

To address some of these limitations, incorporating additional statistical measures such as the interquartile range (IQR) might add more stability to the metric. The IQR considers the spread of the middle 50% of the data, making it less sensitive to extreme values. However, reliably computing quartiles requires an even larger sample size with many participating teams. In contests with a small number of participants, quartile calculations may be less robust, potentially limiting the effectiveness of IQR-based adjustments.

7 Conclusions

The ECN AI Performance Index (EAII) introduced in this paper provides a normalized, interpretable metric for comparing AI and human performances in competitive programming contests. By incorporating both median and peak performances, the EAII benchmarks the AI's capabilities against both average and top-performing competitors, offering a well-rounded perspective on AI's standing within human standards of excellence. This dual reference point makes EAII particularly valuable in identifying not just raw AI performance but also the areas where AI approaches human problem-solving strengths, as well as areas where it lags behind.

In this analysis, the EAII has proven effective in capturing the AI's competencies and limitations across a range of tasks, revealing both the current status and

developmental potential of AI systems in this domain. For example, our qualitative analysis highlights that the AI is well-suited to problems requiring systematic approaches, such as dynamic programming or binary lifting. However, tasks involving flexible heuristics, creative partial solutions, or complex mathematical reasoning sometimes challenge the AI, indicating opportunities for further enhancement in these areas. By tracking such nuances over time, the EAII allows us to observe shifts in AI's problem-solving capabilities and to identify specific areas for potential improvement.

The EAII's design also accommodates variations in contest conditions and participant performance distributions, making it a robust tool for longitudinal analysis across multiple competitions. Although limitations exist—particularly with respect to sample size and distribution assumptions—EAII nonetheless provides an invaluable metric for evaluating the evolution of AI in competitive programming, tracking AI's progress with accessible metrics that reflect achievable benchmarks rather than optimal outcomes. This ensures that the EAII remains widely applicable and interpretable over time, as it is not tied to specialized optimization techniques that could obscure true progress in baseline AI capabilities.

Importantly, EAII is intentionally designed to reflect a consistent, achievable benchmark rather than the maximum potential performance attainable with highly advanced techniques. While agent-based systems [13]–[15], retrieval-augmented generation [16]–[18], or other sophisticated methods could indeed push AI capabilities further, our focus is on establishing a baseline that is accessible to anyone using straightforward prompts and publicly available models. This baseline provides a reference point for tracking AI progress over time without the confounding effects of specialized techniques and customized optimization strategies. In this way, the EAII supports transparent and meaningful assessment of AI progress, helping researchers and practitioners understand the real-world applicability and constraints of AI within competitive programming contexts.

Overall, EAII is a stepping stone for understanding AI's evolving role in problem-solving, offering insights that extend beyond mere performance scores. By capturing a nuanced view of AI's strengths, limitations, and growth areas, the EAII fosters a richer understanding of AI's development trajectory and its potential for reaching, and perhaps one day surpassing, human performance benchmarks in complex problem-solving domains.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflicts of interest.

Disclaimer: LLMs were used to enhance the phrasing and flow of this manuscript. However, all core content - including the research concept, experimental design, data analysis, results, and scientific interpretations - are solely the product of authors' work. The AI language models were used exclusively to improve the clarity and presentation of our research findings.

References

[1] Z. Manna and R. J. Waldinger, "Toward automatic program synthesis," *Communications of the ACM*, vol. 14, no. 3, pp. 151–165, 1971 (⇒ 257).

[2] S. Gulwani, O. Polozov, R. Singh, *et al.*, "Program synthesis," *Foundations and Trends® in Programming Languages*, vol. 4, no. 1-2, pp. 1–119, 2017 (⇒ 257).

[3] C. Green, "Application of theorem proving to problem solving," in *Readings in Artificial Intelligence*, Elsevier, 1981, pp. 202–222 (⇒ 257).

[4] Y. Li, D. Choi, J. Chung, *et al.*, "Competition-level code generation with alphacode," *Science*, vol. 378, no. 6624, pp. 1092–1097, 2022 (⇒ 257, 258).

[5] Q. Shi, M. Tang, K. Narasimhan, and S. Yao, "Can language models solve olympiad programming?" Tech. Rep., 2024, arXiv preprint arXiv:2404.10952 (⇒ 257).

[6] Y. Huang, Z. Lin, X. Liu, *et al.*, "Competition-level problems are effective llm evaluators," Tech. Rep., 2023, arXiv preprint arXiv:2312.02143 (⇒ 257).

[7] T. Coignion, C. Quinton, and R. Rouvoy, "A performance study of llm-generated code on leetcode," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering*, 2024, pp. 79–89 (⇒ 257, 258).

[8] B. Idrisov and T. Schlippe, "Program code generation with generative ais," *Algorithms*, vol. 17, no. 2, p. 62, 2024 (⇒ 258).

[9] AlphaCode Team, Google DeepMind, "AlphaCode 2 Technical Report," Google DeepMind, Tech. Rep., Dec. 2023, 7 pages (⇒ 258).

[10] The AlphaCode team. "Competitive programming with AlphaCode." Accessed: 2024-11-03. (Dec. 2022), [Online]. Available: https://deepmind.google/discover/blog/competitive-programming-with-alphacode/ (⇒ 258).

[11] OpenAI o1 Contributors. "CLearning to Reason with LLMs." Accessed: 2024-11-03. (Sep. 2024), [Online]. Available: https://openai.com/index/learning-to-reason-with-llms/ (⇒ 258).

[12] K. Valmeekam, K. Stechly, A. Gundawar, and S. Kambhampati, "Planning in strawberry fields: Evaluating and improving the planning and scheduling capabilities of lrm o1," Tech. Rep., 2024, arXiv preprint arXiv:2410.02162 (⇒ 258).

[13] G. Sarch, Y. Wu, M. J. Tarr, and K. Fragkiadaki, "Open-ended instructable embodied agents with memory-augmented large language models," Tech. Rep., 2023, arXiv preprint arXiv:2310.15127 (⇒ 283).

[14] C. Li, H. Chen, M. Yan, *et al.*, "Modelscope-agent: Building your customizable agent system with open-source large language models," Tech. Rep., 2023, arXiv preprint arXiv:2309.00986 (⇒ 283).

[15] I. de Zarzà, J. de Curtò, G. Roig, P. Manzoni, and C. T. Calafate, "Emergent cooperation and strategy adaptation in multi-agent systems: An extended coevolutionary theory with llms," *Electronics*, vol. 12, no. 12, p. 2722, 2023 (⇒ 283).

[16] H.-r. Lee and S.-h. Kim, *Bring retrieval augmented generation to google gemini via external api: An evaluation with big-bench dataset*, Research Square, PREPRINT (Version 1), May 2024 (⇒ 283).

[17] W. Huang, M. Lapata, P. Vougiouklis, N. Papasarantopoulos, and J. Pan, "Retrieval augmented generation with rich answer encoding," in *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023, pp. 1012–1025 (⇒ 283).

[18] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen, "Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy," Tech. Rep., 2023, arXiv preprint arXiv:2305.15294 (⇒ 283).

Received: 05.11.2024; *Revised: 28.11.2024*; *Accepted: 28.11.2024*

Acta Univ. Sapientiae, Informatica, **16, 2** (2024) 286–306

DOI: 10.47745/ausi-2024-0015

Intelligent Video Recording Optimization using Activity Detection for Surveillance Systems

Youssef ELMIR
Laboratoire LITAN
École supérieure en Sciences et Technologies de l'Informatique et du Numérique
RN 75, Amizour, 06300, Béjaia, Algérie
elmir@estin.dz
0000-0003-3499-507X

Hayet TOUATI
École supérieure en Sciences et Technologies de l'Informatique et du Numérique
RN 75, Amizour 06300, Béjaia, Algérie
h_touati@estin.dz

Ouassila MELIZOU
École supérieure en Sciences et Technologies de l'Informatique et du Numérique
RN 75, Amizour 06300, Béjaia, Algérie
o_melizou@estin.dz

Abstract. Surveillance systems often struggle with managing vast amounts of footage, much of which is irrelevant, leading to inefficient storage and challenges in event retrieval. This paper addresses these issues by proposing an optimized video recording solution focused on activity detection. The proposed approach utilizes a hybrid method that combines motion detection via frame subtraction with object detection using YOLOv9. This strategy specifically targets the recording of scenes involving human or car activity, thereby reducing unnecessary footage and optimizing storage usage. The developed model demonstrates superior performance, achieving precision metrics of 0.855 for car detection and 0.884 for person detection, and reducing the storage requirements by two-thirds compared to traditional surveillance systems that rely solely on motion detection. This significant reduction in storage highlights the effectiveness of the proposed approach in enhancing surveillance system efficiency. Nonetheless, some limitations persist, particularly the occurrence of false positives and false negatives in adverse weather conditions, such as strong winds.

Key words and phrases: Activity detection, video surveillance, object detection, YOLOv9, motion detection, recording optimization, machine learning, deep learning, CNN, Faster R-CNN

1 Introduction

The widespread installation of surveillance cameras has led to a significant increase in visual data, with estimates predicting over 1.4 billion cameras worldwide by 2024, generating vast amounts of video data daily. This surge poses major challenges in terms of storage, requiring hundreds of petabytes to manage this critical information. Efficient methods to manage and facilitate the search of this data have become imperative.

Surveillance cameras, essential for security, create an information overload. A single HD camera can produce nearly 650 megabytes of data per minute, resulting in a substantial data management challenge when multiplied across thousands of cameras in urban areas. Much of this data is redundant or irrelevant, necessitating careful consideration of optimal storage methods to ensure the rapid retrieval of important information.

The key problem is optimizing the storage of surveillance camera data to prevent storage congestion while maintaining necessary recordings. Object and motion detection algorithms emerge as promising solutions, filtering sequences to record only significant activities. This approach addresses the challenge of information overload in video surveillance.

This study aims to optimize storage space through innovative methods using motion and object detection algorithms and to implement a solution capable of discriminating important scenes. This dual objective seeks to balance storage efficiency with the relevance of the recordings. The structure of the remain of this paper is as follows: Section 2 reviews related works in intelligent surveillance systems, comparing their strengths and weaknesses. In section 3, a proposed methodology of activity detection is introduced with a overall architecture of the proposed system. Implementation is presented in section 4, and experiment results and discussion in section 5. The paper concludes with a summary of findings and a discussion of future research directions.

2 Related Works

Previous studies have explored various methods for optimizing video recording. Background subtraction and frame differencing are commonly used for motion de-

tection, while object detection methods like Faster R-CNN and YOLO have shown promising results in identifying specific objects in video streams.

- Arham et al [1] developed a comprehensive real-time object detection system integrating motion detection, face detection, and human activity recognition, effective in real-world applications but lacking comparison with existing systems and detailed dataset descriptions.

- Pal et al [2] proposed a composite block matching algorithm for efficient motion estimation in video sequences, enhancing accuracy and processing speed but introducing computational complexity that may limit real-time applicability.

- Sadoun et al [3] addressed challenges such as illumination changes and shadow detection, creating a background modeling and subtraction algorithm. The model detects mobile objects but struggles with scene changes and fixed cameras, needing more quantitative measures.

- Sreenu et al [4] reviewed deep learning techniques in intelligent video surveillance, highlighting advancements and challenges like computational complexity and the need for large datasets. They called for more robust models and multi-sensor data integration.

- Xia et al [5] presented a hybrid LSTM-CNN model for human activity recognition, showing superior accuracy but facing high computational costs and the need for extensive labeled data.

- Adarsh et al [6] proposed a model using YOLO and ResNet-34 for detecting suspicious behavior, achieving significant precision but requiring substantial computational resources.

- Lys et al [7] developed a motion detection and object recognition system using OpenCV but lacked detailed results on detection efficiency.

- Alajrami et al [8] proposed AI techniques to enhance human identification in surveillance, improving accuracy and speed but facing high computational demands and privacy concerns.

- Ullah et al [9] combined CNN and LSTM for human activity recognition, improving accuracy but facing computational challenges and the need for broader dataset evaluation.

- Boumediene et al [10] created a real-time object detection model using YOLO, achieving good accuracy but requiring a more diverse dataset for better generalization.
- Suradkar et al [11] proposed an automatic motion detection system improving surveillance efficiency but needing evaluation in varied environments.
- Dhulekar et al [12] developed a two-step system for motion and object detection, suggesting deeper learning techniques for more complex object detection.
- Kapania et al [13] combined YOLOv3 and RetinaNet for robust object detection and tracking, demonstrating high performance but suggesting YOLOv3 for speed and efficiency.
- Dave et al [14] proposed a real-time action detection system addressing class imbalance and multi-label actions, achieving state-of-the-art performance but needing more representative datasets.
- Bosquet et al [15] developed STDnet-ST for small object detection, achieving state-of-the-art performance but proposing the use of GANs for generating synthetic small objects.
- Babiker et al [16] developed an automated system for recognizing human activities using neural networks, achieving high recognition rates but needing testing in complex settings.
- Deguerre et al [17] proposed a rapid object detection method in compressed videos, enhancing detection accuracy while reducing computational time but facing limitations with motion vector quality.
- Ren et al [18] introduced the Faster R-CNN algorithm for fast and accurate object detection, achieving high performance but facing computational intensity issues.
- Patil et al [19] proposed a method for crowd analysis using motion patterns and SVMs, achieving high recognition rates but relying on controlled datasets.

These studies collectively advance the field of video surveillance, addressing various challenges and proposing innovative solutions, though they often highlight the need for further research to overcome limitations in real-world applications as some of them are presented in Table 1 .

Table 1: Common Limitations in Reviewed Surveillance Systems

Limitation	**Description**
Fixed Camera Requirements	Systems like [3] and [11] work only with fixed cameras, limiting use in dynamic environments.
Lack of Real-time Processing	High computational complexity in systems like [2] and [4] hinders real-time performance.
Indoor/Outdoor Constraints	Systems such as [16] are designed for specific environments, limiting versatility.
Computational Requirements	High computational demands of algorithms like Faster R-CNN [18] and RetinaNet [13] challenge real-time deployment on limited devices.
Generalization Issues	Limited datasets affect generalization in systems like [10], [20], and [7].

To address limitations in human detection and surveillance, several research directions are proposed, including hybrid approaches that combine background subtraction with deep learning or optical flow with Support Vector Machines (SVM) for improved accuracy in dynamic environments. Enhancing generalization through diverse training datasets, synthetic data, data augmentation, and transfer learning can also improve model performance in real-world conditions. Developing versatile models for both indoor and outdoor settings using mixed datasets and context-aware mechanisms can enhance surveillance system adaptability.

This paper focuses on optimizing storage in surveillance systems by recording only significant actions using advanced activity detection techniques. Unlike traditional systems like Hikvision[1], which use continuous or basic motion detection, our approach selectively captures important activities, thus reducing storage needs. We evaluate our method against Hikvision to demonstrate its effectiveness in reducing storage without sacrificing surveillance quality.

The study explores intelligent video recording optimization through various techniques, including motion detection, background subtraction, optical flow, and advanced object detection methods like YOLOv3 and Faster R-CNN. A hybrid approach that combines initial motion detection with precise object tracking is identified as optimal for performance and resource efficiency. Future research should refine these models, expand datasets, and utilize advanced training devices. While integrating these techniques for real-time surveillance remains challenging, our hybrid approach aims to achieve more efficient and effective surveillance.

[1]https://www.hikvision.com

3 Methods

The Hikvision Surveillance System (HikvisionSS) was selected as the baseline for comparison in this study due to its existing deployment within our institution, École supérieure en Sciences et Technologies de l'Informatique et du Numérique (ESTIN[2]). The primary objective of this project is to optimize and reduce the storage space required for video recordings, which directly impacts the operational efficiency of the Hikvision system. As such, it was essential to evaluate the performance of our proposed approach in comparison to HikvisionSS, particularly in terms of the storage space required for recording, to ensure that our method provides tangible benefits within the context of its real-world application. We have chosen the Hikvision surveillance system as the baseline for comparison due to its widespread use in our institution and its standard recording modes, which include continuous recording and motion detection-based recording. Our proposed method focuses on an intelligent approach that records only significant actions, identified through advanced activity detection algorithms. This strategic selection allows us to directly assess the improvements in storage efficiency offered by our approach.

3.1 System Architecture

The proposed system integrates motion detection and object detection in a hybrid approach. As illustrated in Figure 1, the system begins with frame subtraction to identify regions of interest where motion is detected. These regions are then processed by the YOLO model, which detects and classifies objects, with a particular focus on human activity. This architecture ensures that only relevant scenes are recorded, significantly reducing unnecessary footage.

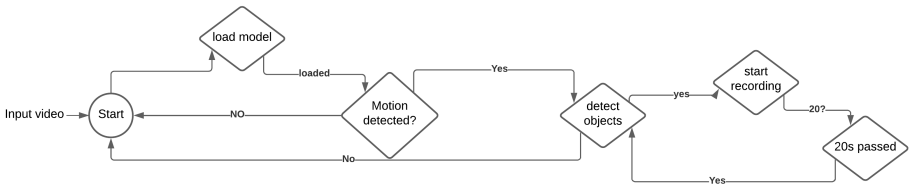


Figure 1: flowchart explaining the global architecture of the proposed system

[2]https://estin.dz/

Algorithm 1: Object Detection Driven Surveillance Video Recording

```
Require: Surveillance video stream
Ensure: Recorded video files with object activity
  Load the object detection Yolo model
  Initialize variables for motion detection and recording status
  while video stream exists do
    Read the current frame from the video stream
    Use the motion detection algorithm to detect motion in the current frame
    if motion is detected then
      Use the object detection model to detect objects in the current frame
      if an object is detected then
        if not currently recording then
          Start recording and note the time of last detection and currently
          recording == True
        end if
      end if
    end if
    if currently recording then
      if less than 20 seconds have passed since the last detection then
        if objects are detected then
          Update the time of last detection
        end if
        Write the current frame to the video file
      else
        if no objects are detected then
          Stop recording and currently recording == False
        else
          Write the current frame to the video file and Update the time of last
          detection
        end if
      end if
    end if
  end while
  if Stop recording button = True then
    Stop recording and currently recording == False
  end if
```

3.2 Activity Detection

3.2.1 Motion Detection:

Frame subtraction is employed to detect changes between consecutive frames, indicating potential activity. This method is computationally efficient and suitable for real-time applications [21]. To examine the different algorithms used for motion detection, we reviewed various works such as [3] and [16] utilizing background subtraction, [8] employing optical flow, and [2] proposing a composite block matching algorithm.

3.2.2 Object Detection:

The YOLO model is used to detect and classify objects within the video frames. YOLOv9 is chosen for its balance between speed and accuracy, making it ideal for real-time surveillance.

An example of using the YOLO method is found in [6], where the authors detect suspicious individuals and hostile behavior. They use the YOLO model, pre-trained on the COCO dataset [3], for human detection in video frames. These frames are then processed by ResNet-34 to recognize activities, achieving a precision of 82%. Despite its effectiveness, the model's reliance on two deep learning models demands substantial computational resources

The workflow for recording surveillance videos driven by object detection is detailed in Algorithm 1. The algorithm outlines how the system processes video streams, detects motion, and records activity only when objects of interest are present in the frame, ensuring efficient storage usage and effective monitoring.

4 Implementation

4.1 Data Preparation and Analysis

The data preparation involved collecting, labeling, and processing images for training the model. The dataset was sourced from various public repositories, focusing on human and car detection. Key data preparation steps included labeling objects using the Roboflow platform, preprocessing images (auto-orientation, resizing, contrast adjustment), and applying data augmentation techniques like grayscale conversion.

The dataset was split into training, validation, and testing sets using a 70:20:10 ratio. The dataset comprised 300 images for human detection from Kaggle and 100 im-

[3] https://cocodataset.org/

Figure 2: Samples from the dataset used to train YOLO model

ages for car detection from a GitHub repository, supplemented by additional images sourced from Google Images and other websites. The preparation process involved auto-orienting the images, resizing them to 646x640 pixels, and applying contrast adjustment and grayscale conversion to a portion of the dataset. The final dataset included 2,664 images labeled as 'Person' and 1,735 as 'Car', ensuring a well-balanced distribution across object classes.

The analysis highlighted the efficiency of the data preparation process, ensuring that the training model was robust and well-balanced across different scenarios and object classes. Visual tools like heatmaps and histograms were used to further analyze the distribution of annotations and objects within the dataset.

Figure 2 presents samples from the dataset, showcasing the diversity of scenarios and object classes included.

4.2 Model Training

4.2.1 Model Pre-training

The YOLO algorithm was implemented by cloning its repository from GitHub. Pre-trained weights from the MS COCO dataset were used as a starting point for fine-tuning on the custom dataset. The pre-training process involved running a script with key parameters, including 8 CPU workers for faster data loading, GPU utilization, a batch size of 16, and 500 training epochs. The model was trained from scratch using specific configurations and hyperparameters, with mosaic augmentation disabled after 15 epochs to enhance focus on original images. This pre-trained model was then fine-tuned on the custom dataset.

4.2.2 Model Fine-Tuning

The fine-tuning of the YOLO model on the custom dataset was performed using a modified script, which involved resizing input images to 640x640 pixels, setting a batch size of 16, and training for 25 epochs. The data configuration file was pointed to the custom dataset, while the model configuration file specified the YOLO architecture. Pre-trained weights from the initial training on the MS COCO dataset were used as the starting point.

4.2.3 Evaluation of Training Results

After completing the YOLO model training and fine-tuning, a thorough evaluation was performed using both qualitative and quantitative metrics to assess the model's performance. The validation process utilized a separate dataset to ensure the model's ability to generalize to unseen data. Key parameters for validation included using the prepared dataset, the best model weights from training, a batch size of 16, images resized to 640x640 pixels, a confidence threshold of 0.001, an IoU threshold of 0.5, and a maximum of 300 detections per image.

4.3 Comparative Analysis of Trained Models

The comparative analysis of the trained models focuses on identifying the most suitable model for deployment by evaluating various performance metrics, such as precision, recall, and mean Average Precision (mAP), in addition to considering computational resources and deployment constraints. This analysis not only aids in selecting the optimal model but also informs iterative improvement strategies by highlighting areas for refinement. The 7th Model, with the highest precision (87.0%), recall (82.0%), and mAP (90.1%), was selected for the proposed system, as it offers the best balance between precision and recall, ensuring accurate object detection and classification in video feeds.

4.4 Model Deployment

The deployment process involves preparing the chosen YOLOv9 based model by converting it into a deployable format and configuring it for integration. Once deployed, the model is accessible via an API, allowing seamless integration into the application environment. This setup ensures the model performs accurate inference tasks and provides reliable object detection capabilities, making it ready for real-world application scenarios.

Table 2: Comparative Table of Trained Models

Model	Version	Dataset Size	Epochs	Precision	Recall	mAP (0.5)
1st Model	9	664 Images	50	77.9%	73.7%	76.8%
2nd Model	8	664 Images	50	70.0%	71.7%	74.3%
3rd Model	9	1177 Images	30	81.6%	73.3%	80.9%
4th Model	9	1469 Images	30	85.3%	75.2%	82.4%
5th Model	9	1294 Images	30	83.6%	75.1%	82.2%
6th Model	9	1622 Images	25	86.5%	76.7%	85.5%
7th Model	**9**	**1920 Images**	**25**	**87.0%**	**82.0%**	**90.1%**
8th Model	9	1927 Images	100	85.1%	82.6%	88.7%

4.5 Model Evaluation and Training Visualization

The model's performance was evaluated using key metrics such as Precision (P), Recall (R), mean Average Precision at IoU threshold 0.5 (mAP50), and mean Average Precision across IoU thresholds from 0.5 to 0.95 (mAP50-95). Precision, which indicates the accuracy of positive predictions, was 0.869 overall (Car: 0.855, Person: 0.884). Recall, reflecting the model's ability to detect all relevant instances, was 0.824 overall (Car: 0.83, Person: 0.819). The mAP50 was 0.891 (Car: 0.899, Person: 0.883), showing high precision and recall at an IoU threshold of 0.5, while the mAP50-95 was 0.558 (Car: 0.638, Person: 0.478), demonstrating robustness across different IoU settings. These metrics, detailed in Table 3, highlight the model's strong performance in detecting cars and persons with high precision and recall, and reasonable generalization across IoU thresholds.

Table 3: Model Performance Metrics for Precision, Recall, and mAP.

Category	Precision	Recall	mAP50	mAP50-95
Car	0.855	0.83	0.899	0.638
Person	0.884	0.819	0.883	0.478
Overall	0.869	0.824	0.891	0.558

The confusion matrix, which evaluates the model's performance by comparing predicted labels to true labels, includes metrics such as True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN). The model demonstrates efficient performance, with a pre-processing time of 0.5 milliseconds, an inference time of 25.1 milliseconds, and Non-Maximum Suppression (NMS) time of 5.2 milliseconds per image. The inference time is particularly significant, as it reflects the model's capability for real-time image processing, which is crucial for applica-

tions requiring rapid decisions.

Table 4: Performance Speed Metrics.

Performance Metric	**Time (milliseconds per image)**
Pre-processing Time	0.5
Inference Time	25.1
Non-Maximum Suppression (NMS) Time	5.2

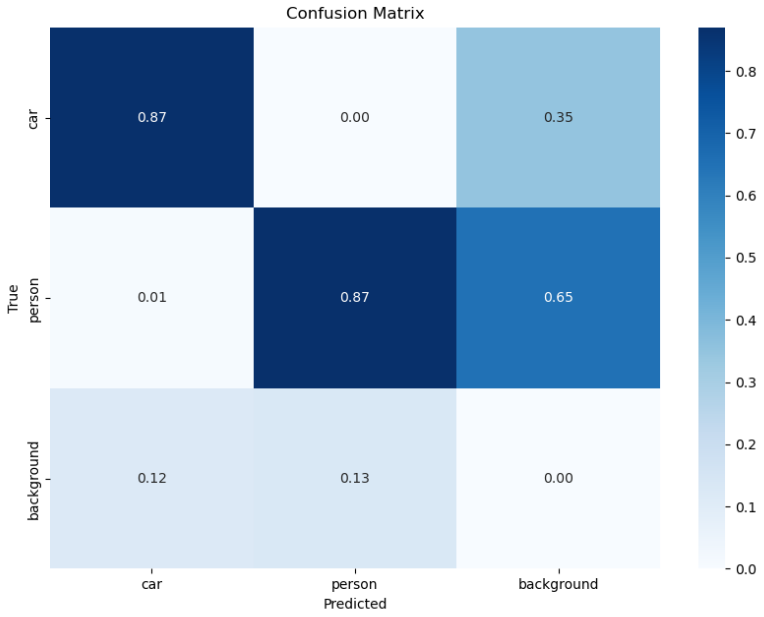


Figure 3: Confusion matrix for the proposed video recording optimization method applied to surveillance footage

.

Figure 3 shows the confusion matrix for activity classification (car, person, and background) in surveillance footage. The diagonal elements represent correctly classified instances, while the off-diagonal elements correspond to misclassifications. The matrix indicates an accuracy of 87% for detecting cars and persons, with slightly lower accuracy for the background class. Misclassifications mainly occurred between 'person' and 'background,' highlighting areas for potential model improve-

ment.

The training progress is illustrated through plots showing the evolution of loss components and performance metrics over epochs. The bounding box regression loss (train/box_loss) decreased from 1.5 to 1.1, indicating improved accuracy in predicting bounding box coordinates. The classification loss (train/cls_loss) fell from 1.6 to about 0.6, reflecting enhanced classification performance. The Distribution Focal Loss (train/dfl_loss) also decreased from 1.5 to 1.25, demonstrating increased precision. Precision (metrics/precision) and recall (metrics/recall) metrics improved from 0.65 to approximately 0.85 and 0.82, respectively. Validation losses (val/box_loss, val/cls_loss, val/dfl_loss) followed similar trends, indicating good generalization. The mean Average Precision at an IoU threshold of 0.5 (metrics/mAP_0.5) increased from 0.45 to 0.90, and the mean Average Precision across IoU thresholds from 0.5 to 0.95 (metrics/mAP_0.5:0.95) rose from 0.40 to 0.75, highlighting the model's robust performance.

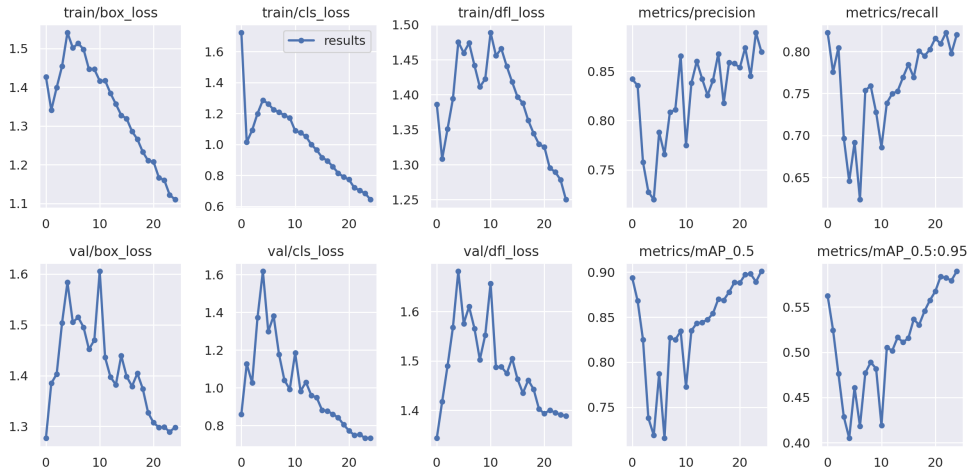


Figure 4: Training and Validation Performance Metrics of the 7th YOLOv9 based Model.

Figure 4 displays training and validation losses, precision, recall, and mean Average Precision (mAP) metrics over 25 epochs. The training and validation losses decrease consistently, indicating effective learning. Precision and recall improve steadily, with recall nearing 0.90 by the end of training. The mAP metrics also show positive trends, with mAP_0.5 surpassing 0.8. These results suggest the model continues to improve, with further training potentially enhancing performance. The visualizations in Figure 4 provide insights into the model's learning progress, crucial

for assessing overfitting or underfitting. Overall, the best YOLOv9 model, fine-tuned with the custom dataset, exhibits high accuracy and efficiency, making it suitable for deployment in the proposed intelligent video recording optimization solution.

5 Deployment and Results

5.1 Storage Optimization

. This setup involves two different scenarios: one using the proposed intelligent recording approach and the other employing the Hikvision surveillance system's traditional continuous recording and motion detection features. The primary objective is to quantify the reduction in storage space achieved by our method, without sacrificing the accuracy and reliability of the surveillance footage. To validate the performance of the proposed application and the study objective, which focuses on the Optimization of Recording Storage for Surveillance Systems, a comparative study was conducted using Hikvision monitoring software[4] at ESTIN. Both systems were tested simultaneously under identical conditions to measure their effectiveness in optimizing storage by comparing the length of recorded videos. All recordings were made using a unified format (MP4, codec H.264, 1280x720, 16:9, 30 fps, bitrate 7 703 kbps).

5.1.1 Equipment and Configuration

Two types of cameras were used: a fixed camera and a flexible dome camera, positioned strategically in front of ESTIN's main gate and the building of Labs' building. The Hikvision system was configured with its specific hardware and software setup, while the proposed system was deployed on a local machine, specifically an i7 Core ThinkPad. This setup created a controlled environment to accurately measure and compare the performance of both systems in terms of storage optimization.

For the experimental evaluation, two distinct video records were used. The first video was a real-time test conducted live with the first camera This period of time in a workday, particularly from 2 PM to 3 PM, is likely one of the most active for students, teachers, and workers at ESTIN. The first video captures the scene in front of the main gate of ESTIN during this busy period, from 2 PM to 3 PM. The second video was recorded offline, after the real-time scene was captured. This second recording was done using a flexible camera positioned in front of the Labs' building from 2:30 PM to 2:45 PM. This dual approach allowed us to test the proposed

[4]https://www.hikvision.com/en/support/download/software/

method under both live and post-recording conditions, providing a comprehensive evaluation of its performance.

5.1.2 Real-Time Test

- Objective: Compare the storage optimization between the proposed system and the Hikvision software over a one-hour period of live surveillance with motion detection enabled.
- Methodology: Both systems were launched simultaneously and ran for one hour with motion detection enabled on both.
- Metrics Collected: Length of the recorded videos from both systems.

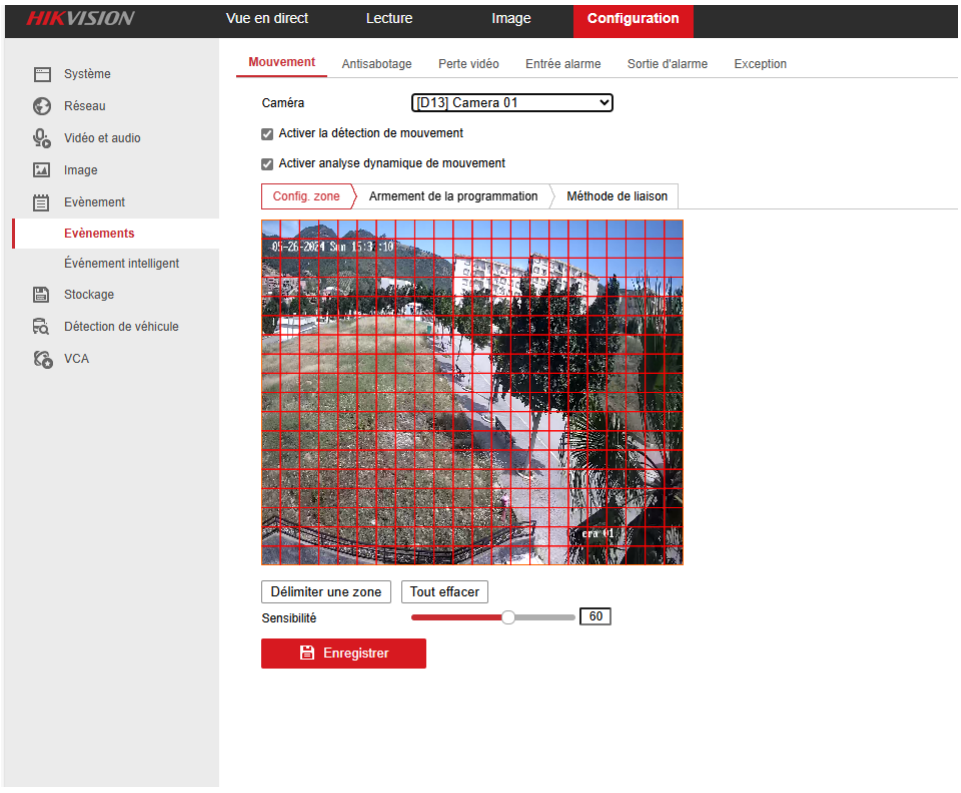


Figure 5: Screenshot of the Hikvision System with Motion Detection Enabled

5.1.3 Video Upload Test

- Objective: Compare the performance of the proposed system and the Hikvision software using a pre-recorded video.
- Methodology: A 15-minute video was processed by both systems with motion detection enabled.
- Metrics Collected: Length of the resulting videos after processing.

Due to practical constraints, the number of video records used in this study was limited. However, the selected videos were chosen for their representativeness in assessing the effectiveness of the proposed system in optimizing storage space compared to the HikvisionSS baseline. Despite these limitations, the focus of our analysis remains on demonstrating the efficiency and applicability of our approach within the context of real-world surveillance scenarios.

5.1.4 Analysis of Storage Efficiency

The performance of the two systems was evaluated based on the following criteria:

- Recorded Video Length: The total duration of the videos recorded by each system under the same conditions. A shorter recorded length indicates better storage optimization.
- Detection Accuracy: The ability to correctly identify and record relevant activities (human and vehicular) without missing any significant events.

Table 5: Comparison of Recorded Lengths and Storage for Different Test Types

Test Type	**System**	**Video Length (minutes & seconds)**	**Storage Size (MegaBytes)**
1-Hour Real-Time Test (live)	Hikvision SS	60m & 00s	1 917
	Proposed	13m & 45s	769
15-Minute Video Test (recorded)	Hikvision SS	6m & 53s	379
	Proposed	6m & 20s	355

To evaluate the efficiency of the proposed method, a comparative analysis was conducted using two test types: a 1-hour real-time test and a 15-minute video test. The results in Table 5 compare the recorded lengths and storage sizes between the traditional Hikvision surveillance system and the proposed activity detection system. In the 1-hour real-time test, the Hikvision system recorded 60 minutes, consuming 1 917 MB, while the proposed system recorded only 13 minutes and 45 seconds, using 769 MB. This discrepancy is largely due to Hikvision's motion detection being highly sensitive to minimal motion, such as the movement of trees and palms caused by the wind, which triggered recording frequently. In contrast, the proposed system only initiated recording after detecting significant motion, using object detection to ensure that the recorded scenes contained relevant objects like cars or persons. In the second 15-minute video test, there was no wind, and both systems recorded only significant motions when detected. Therefore, the Hikvision system recorded 6 minutes and 53 seconds, consuming 379 MB, whereas the proposed system recorded 6 minutes and 20 seconds, using 355 MB.

These results demonstrate significant storage savings by recording only scenes with detected activity, reducing storage requirements without compromising important data. The storage savings chart underscores the efficiency of the proposed method compared to traditional continuous recording.

However, the proposed system has some limitations, such as false positives and false negatives in adverse weather conditions like strong winds, which can cause motion artifacts that affect the accuracy of detection as it is illustrated in Figure 3 and in some amples of Figure 6.

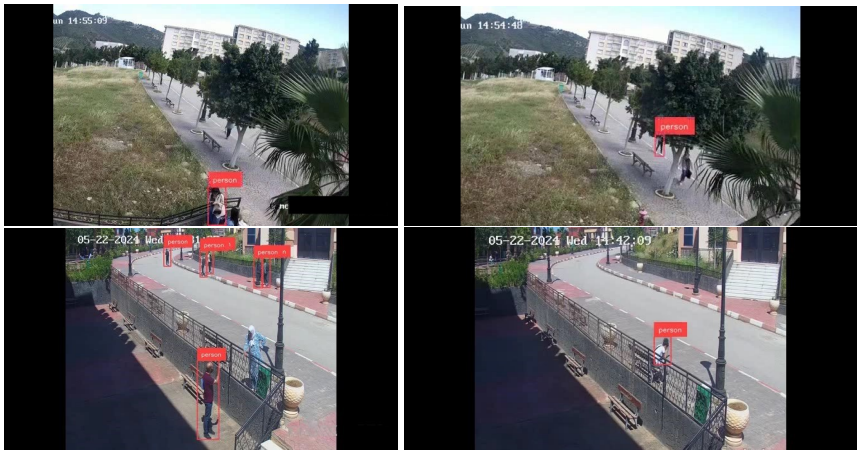


Figure 6: Samples of real-time validation

The validation results demonstrate that the proposed system significantly outperforms the Hikvision software in optimizing storage. In the one-hour real-time test, the proposed system recorded only 13 minutes and 45 seconds of video, compared to Hikvision's 60 minutes, even with Hikvision's intelligent motion detection option enabled. This highlights the proposed system's superior ability to filter out irrelevant footage and focus on significant activities.

In a 15-minute video upload test, the proposed system recorded 6 minutes and 20 seconds, while Hikvision recorded 6 minutes and 53 seconds. Although Hikvision captured the necessary scenes, it encountered some bugs. The proposed system proved more efficient by accurately detecting and recording pertinent activities, thereby minimizing storage usage.

These results confirm the effectiveness of the proposed model in optimizing recording storage for surveillance systems. Compared to Hikvision, which relies on continuous or basic motion-based recording, our method significantly reduces the storage space required by recording only meaningful activities. This optimization does not compromise the ability to capture critical events, as evidenced by the comparative analysis, making our approach a valuable tool for enhancing the efficiency and cost-effectiveness of surveillance operations.

6 Conclusion

In conclusion, this work effectively demonstrates that intelligent activity detection can optimize storage space in surveillance systems. By selectively recording only meaningful actions, our approach significantly reduces storage requirements compared to traditional systems like Hikvision, which rely on continuous or motion-based recording.

This study developed a hybrid system that combines motion detection via successive frame subtraction with the YOLOv9 model for detecting significant activities. YOLOv9 was selected for its superior detection capabilities and efficiency, achieving an accuracy of 87%. The combination of motion detection for initial screening with advanced activity detection optimizes both performance and resource usage, particularly benefiting the recording system at ESTIN by addressing storage and resource management challenges. Validation results showed substantial optimization in both time and storage space.

Future work should focus on refining model structures to further enhance detection precision and efficiency, leveraging more powerful training devices and larger, high-quality datasets. While our project made significant strides, further advancements are possible with additional resources and improved data. The methodologies

developed hold potential applications beyond security, benefiting any field requiring real-time object detection and intelligent video recording.

Although the scope of our experiments was limited due to practical constraints, the results highlight the potential of the proposed method for optimizing video recording in surveillance systems. Future studies should consider expanding the dataset and exploring additional comparative methods to further validate the system's performance.

Author Contributions: *"Conceptualization and implementation, T. H. and O. M.; methodology and writing, T. H., O. M., and Y. E.; supervision, Y.E. All authors have read and agreed to the manuscript's published version."*

Funding: This research received no external funding

Data Availability: The data generated and analyzed in this study are not publicly available due to privacy and proprietary restrictions. Requests for access to the data can be directed to [h_touati@estin.dz, o_melizou@estin.dz], subject to approval and compliance with [any relevant conditions or restrictions].

Acknowledgments

We express our deepest gratitude to Mr. Nabil Aoughlis, Ms. Lylia Aberkane and Mr. Ridha Chekroune for their guidance and support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

[1] M. Arham, A. Srivastava, A. G, and R. A. B, "Motion detection and human activity recognition for security," *International Journal of Engineering Research & Technology (IJERT)*, vol. 11, no. 05, 2023 (⇒ 288).

[2] A. K. Pal, B. Biswas, M. D. Jichkar, A. N. Jena, and M. Kumar, "Object detection driven composite block motionestimation algorithm for high-fidelitysurveillance video coding," 2023 (⇒ 288, 290, 293).

[3] S. Abdelbaki and O. Yacine, "Détection et suivi des objets mobiles: Application dans un environnement de foule," M.S. thesis, Université ECHAHID HAMMA LAKHDAR D'EL OUED, Algérie, 2016 (⇒ 288, 290, 293).

[4] G. Sreenu and S. Durai, "Intelligent video surveillance: A review through deep learning techniques for crowd analysis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–27, 2019 (⇒ 288, 290).

[5] K. Xia, J. Huang, and H. Wang, "Lstm-cnn architecture for human activity recognition," *IEEE Access*, vol. 8, pp. 56 855–56 866, 2020 (⇒ 288).

[6] S. Adarsh, S. P. Giridhar Kannan, B. Vidhyasagar, and J. Arunnehru, "Suspicious activity detection and tracking in surveillance videos," *Int J Emerg Technol Innovative Res*, vol. 7, no. 5, pp. 75–79, 2020 (⇒ 288, 293).

[7] R. Lys and Y. Opotyak, "Development of a video surveillance system for motion detection and object recognition," *Advances in Cyber-Physical Systems*, 2023 (⇒ 288, 290).

[8] E. Alajrami, H. Tabash, Y. Singer, and M.-T. E. Astal, "On using ai-based human identification in improving surveillance system efficiency," *2019 International Conference on Promising Electronic Technologies (ICPET)*, pp. 91–95, 2019 (⇒ 288, 293).

[9] A. Ullah, K. Muhammad, J. Del Ser, S. W. Baik, and V. H. C. de Albuquerque, "Activity recognition using temporal optical flow convolutional features and multilayer lstm," *IEEE Access*, vol. 7, pp. 51 177–51 188, 2019 (⇒ 288).

[10] N. BOUMEDIENE *et al.*, "Détection d'objet en temps réel en utilisant une approche basée sur l'apprentissage profond," Ph.D. dissertation, Université Ibn Khaldoun-Tiaret-, 2022 (⇒ 289, 290).

[11] H. Suradkar, A. Kolte, S. Jamdade, and S. Gokhale, "Automatic surveillance using motion detection," *International Journal of Engineering Research and General Science*, vol. 3, no. 2, p. 525, 2015, ISSN: 2091-2730 (⇒ 289, 290).

[12] P. Dhulekar, S. Gandhe, A. Shewale, S. Sonawane, and V. Yelmame, "Motion estimation for human activity surveillance," in *2017 International Conference on Emerging Trends & Innovation in ICT (ICEI)*, IEEE, 2017, pp. 82–85 (⇒ 289).

[13] S. Kapania, D. Saini, S. Goyal, N. Thakur, R. Jain, and P. Nagrath, "Multi object tracking with uavs using deep sort and yolov3 retinanet detection framework," in *Proceedings of the 1st ACM Workshop on Autonomous and Intelligent Mobile Systems*, 2020, pp. 1–6 (⇒ 289, 290).

[14] I. R. Dave *et al.*, "Gabriellav2: Towards better generalization in surveillance videos for action detection," in *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*, IEEE, 2022 (⇒ 289).

[15] B. Bosquet, M. Mucientes, and V. M. Brea, "Stdnet-st: Spatio-temporal convnet for small object detection," *Pattern Recognition*, vol. 116, p. 107 929, 2021 (⇒ 289).

[16] M. Babiker, O. O. Khalifa, K. K. Htike, A. Hassan, and M. Zaharadeen, "Automated daily human activity recognition for video surveillance using neural network," *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, pp. 1–5, 2017 (⇒ 289, 290, 293).

[17] B. Deguerre, C. Chatelain, and G. Gasso, "Fast object detection in compressed jpeg images," in *2019 ieee intelligent transportation systems conference (itsc)*, IEEE, 2019, pp. 333–338 (⇒ 289).

[18] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016 (⇒ 289, 290).

[19] S. Patil and K. S. Prabhushetty, "An efficient motion based group level activity recognition for intelligent video surveillance," in *2021 Asian Conference on Innovation in Technology (ASIANCON)*, IEEE, 2021, pp. 1–8 (⇒ 289).

[20] H. R. Aradhya *et al.*, "Object detection and tracking using deep learning and artificial intelligence for video surveillance applications," *international journal of advanced computer science and applications*, vol. 10, no. 12, 2019 (⇒ 290).

[21] S. Manchanda and S. Sharma, "Analysis of computer vision based techniques for motion detection," in *2016 6th international conference-cloud system and big data engineering (confluence)*, IEEE, 2016, pp. 445–450 (⇒ 293).

Received: 27.06.2024; *Revised: 07.11.2024*; *Accepted: 26.12.2024*