Robust Learning and Reasoning
for Complex Event Forecasting

Project Acronym:	EVENFLOW
Grant Agreement number:	101070430 (HORIZON-CL4-2021-HUMAN-01-01 – Research and Innovation Action)
Project Full Title:	Robust Learning and Reasoning for Complex Event Forecasting

DELIVERABLE

D3.3 – Final Use Case Evaluation

Dissemination level:	PU - Public, fully open
Type of deliverable:	R - Document, report
Contractual date of delivery:	31 December 2025
Deliverable leader:	BSC
Status - version, date:	Final – v1.0, 2025-12-30
Keywords:	use cases, evaluation, healthcare, Industry 4.0, infrastructure, neurosymbolic, AI

Funded by the European Union

This document is part of a project that is funded by the European Union under the Horizon Europe agreement No 101070430. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the Commission. The document is the property of the EVENFLOW project and should not be distributed or reproduced without prior approval. Find us at www.evenflow-project.eu.

Executive Summary

Deliverable D3.3 presents the final results of the use cases in the EVENFLOW project, detailing the development and validation of a unified technological framework for complex event forecasting. This framework integrates machine learning, symbolic reasoning, and probabilistic temporal modelling to deliver forecasts that are accurate, interpretable, and deployable.

Its effectiveness is demonstrated across three real-world use cases. In Industry 4.0, EVENFLOW enables proactive deadlock avoidance and smoother multi-robot navigation through forecast-driven, liveness-aware controllers. In personalized oncology, it supports early detection of cancer stage transitions, uncovers interpretable temporal patterns, and generates confidence-calibrated probabilistic forecasts even when patient data is limited. In industrial monitoring, the framework achieves accurate classification of pipe and tap scenarios, reconstructs underlying temporal dynamics, and reliably predicts future events from high-frequency sensor measurements.

Across these domains, EVENFLOW proves both theoretically expressive and operationally robust, highlighting the practical impact of integrating learning, symbolic reasoning, and temporal forecasting in dynamic, high-stakes environments.

Deliverable leader:	Davide Cirillo (BSC)
Contributors:	Davide Cirillo, Guillermo Prol Castelo (BSC) Fatos Gashi, Satyam Dudhagara (DFKI) Karim Ladjeri (EKSO) Nikos Katzouris, Vasilis Manginas, Nikos Manginas, Elina Syrri (NCSR)
Reviewers:	Nikos Katzouris (NCSR) Karim Ladjeri (EKSO) Athanasios Poulakidas (INTRA)
Approved by:	Athanasios Poulakidas, Vasiliki-Eleni Provopoulou (INTRA)

Document History

Version	Date	Contributor(s)	Description
v0.1	2025-11-04	BSC	ToC preparation
v0.2	2025-12-10	NCSR	DFKI UC and BSC UC updates
v0.3	2025-12-12	EKSO	EKSO UC updates
v0.4	2025-12-15	DFKI	DFKI UC updates
v0.5	2025-12-15	NCSR	EKSO UC updates
v0.6	2025-12-20	BSC	Draft ready for review
v0.7	2025-12-26	BSC	Draft addressing review comments
v1.0	2025-12-30	INTRA	Version to be submitted after final QA

Table of Contents

Table of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
AMI	Adjusted Mutual Information
AMR	Autonomous Mobile Robot
ASA	Answer Set Automaton
ASAL	Answer Set Automata Learning
ATC	All taps closed (EKSO dataset scenario)
AUC	Area Under the Curve
BRCA	Breast Invasive Carcinoma
CEF	Complex Event Forecasting
CE	Complex Event
CER	Complex Event Recognition
CNN	Convolutional Neural Network
CSV	Comma-Separated Values (data file format)
DFA	Deterministic Finite Automaton
DSFA	Deterministic Symbolic Finite Automaton
G3 / G4	Medulloblastoma subgroups Group 3 / Group 4
GRU	Gated Recurrent Unit
KIRC	Kidney Renal Clear Cell Carcinoma
LSTM	Long Short-Term Memory network
MAE	Mean Absolute Error
MiMM	Mutual-Information Markov Model
Nav2	Navigation 2 (ROS2 navigation framework)
NAR	Nucleic Acids Research (journal)
NeSy	Neurosymbolic forecasting system
OOD	Out-of-Distribution
POV	Point of View
PST	Prediction Suffix Tree
RGB	Red, Green, Blue (color channels in images)
SAX	Symbolic Aggregate approXimation
SFA	Symbolic Finite Automaton
SHAP	SHapley Additive exPlanations
STFT	Short-Time Fourier Transform
SRE	Symbolic Regular Expression
VAE	Variational Autoencoder
VMM	Variable-order Markov Model
ksps	Kilo-samples per second (sensor sampling rate)

1 Introduction

1.1 Project Information

EVENFLOW develops hybrid learning techniques for complex event forecasting, which combine deep learning with logic-based learning and reasoning into neuro-symbolic forecasting models. This approach combines neural representation learning techniques that construct event-driven features from streams of perception-level data with powerful symbolic learning and reasoning tools, which utilize such features to synthesize high-level, interpretable patterns for forecasting critical events.

To deal with the brittleness of neural predictors and the high volume/velocity of temporal data flows, the EVENFLOW techniques rely on novel, formal verification techniques for machine learning, in addition to a suite of scalability algorithms for training based on data synopsis, federated training and incremental model construction. The learnt forecasters will be interpretable and scalable, allowing for explainable and robust insights, delivered in a timely fashion and enabling proactive decision making.

EVENFLOW is evaluated on three use cases related to (1) oncological forecasting in healthcare, (2) safe and efficient behaviour of autonomous transportation robots in smart factories and (3) reliable life cycle assessment of critical infrastructure.

Table 1: The EVENFLOW consortium.

Number[1]	Name	Country	Short name
1 (CO)	NETCOMPANY-INTRASOFT	Belgium	**INTRA**
1.1 (AE)	NETCOMPANY-INTRASOFT SA	Luxemburg	**INTRA-LU**
2	NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"	Greece	**NCSR**
3	ATHINA-EREVNITIKO KENTRO KAINOTOMIAS STIS TECHNOLOGIES TIS PLIROFORIAS, TON EPIKOINONION KAI TIS GNOSIS	Greece	**ARC**
4	BARCELONA SUPERCOMPUTING CENTER-CENTRO NACIONAL DE SUPERCOMPUTACION	Spain	**BSC**
5	DEUTSCHES FORSCHUNGSZENTRUM FUR KUNSTLICHE INTELLIGENZ GMBH	Germany	**DFKI**
6	EKSO SRL	Italy	**EKSO**
7 (AP)	IMPERIAL COLLEGE OF SCIENCE TECHNOLOGY AND MEDICINE	United Kingdom	**ICL**

1.1 Document scope

The EVENFLOW project tackles the challenge of forecasting rare and complex events across three key domains: Industry 4.0, Personalized Medicine, and Infrastructure Management. By integrating neuro-symbolic reasoning with reproducible, data-driven modelling, EVENFLOW delivers actionable predictive intelligence:

[1] CO: Coordinator. AE: Affiliated Entity. AP: Associated Partner.

- **Industry 4.0:** Enables proactive safety through efficient multi-robot navigation in dynamic industrial environments.
- **Personalized Medicine:** Supports clinical precision by forecasting critical events and adverse outcomes in oncology.
- **Infrastructure Management:** Enhances asset resilience via advanced predictive maintenance and lifecycle assessment of infrastructure.

Together, these applications illustrate EVENFLOW's potential as a scalable, accurate, and interpretable solution for complex, real-world predictive challenges. This document provides a follow-up to the previous deliverable, D3.2, detailing progress and refinements in the project's methodologies and applications.

1.2 Document Structure

This document is structured as follows:

Chapter 1: Introduction – Outlines the scope, objectives, and overall context of the deliverable.

Chapter 2: Use Case Objectives – Defines the goals and expected outcomes for each use case, covering Industry 4.0, Personalized Medicine, and Infrastructure Life Cycle Assessment.

Chapter 3: Use Case Developments – Describes the development of each use case, including experimental platforms, datasets, predictive models, and control or forecasting strategies.

Chapter 4: EVENFLOW Technology Applied to Use Cases – Presents the practical application of the EVENFLOW framework, showcasing neurosymbolic deadlock recognition for Industry 4.0, hybrid approaches for early kidney cancer transition detection in Personalized Medicine, and infrastructure life cycle assessment methodologies.

Chapter 5: Use Case Evaluation – Details the evaluation methodology, experimental setup, quantitative results, and their interpretation for all use cases.

Chapter 6: Conclusions – Summarizes key outcomes, insights, and lessons learned from the project.

2 Use Case Objectives

2.1 Industry 4.0

The main objectives of the Industry 4.0 use case are centred on predictive and proactive robot navigation within an intralogistics system. At the core, the project seeks to develop mechanisms that allow robots to forecast potential conflicts, including deadlocks (a condition in which robots mutually obstruct each other, halting task execution), and adjust their navigation strategies to prevent them. The objectives can be summarized as follows:

- Generate reproducible datasets to support model training, validation, and performance assessment.
- Predict potential deadlocks and collisions between robots using both time-series and neuro-symbolic models.
- Enable proactive navigation that minimizes operational delays and ensures safety.
- Evaluate forecasting and control strategies both at the individual robot level and across the multi-robot system.

These objectives aim to ensure that AMRs can operate efficiently and safely even in complex industrial settings, where multiple robots interact and the environment is highly dynamic.

2.2 Personalized Medicine

This use case leverages EVENFLOW technology to forecast critical tumor evolution and adverse pharmacological reactions within oncological "virtual patient" environments. By synthesizing data-driven deep learning with neuro-symbolic reasoning, the project aims to transform early clinical indicators into proactive, actionable decision support.

The specific objectives are to:

1. Develop reliable and reproducible virtual patient datasets for model benchmarking, utilizing advanced deep generative architectures such as Variational Autoencoders (VAEs).
2. Learn explainable latent representations that map unobserved biological variables to transparent generative factors, with special emphasis on disease progression stages.
3. Infer pseudo-temporal patient trajectories from latent spaces to represent the evolution of tumors throughout stages.
4. Identify temporally correlated event sequences within these trajectories, specifically focusing on the molecular biomarkers.
5. Integrate synthetic data and discovered patterns into the EVENFLOW online neuro-symbolic learning engine to facilitate the early forecasting of high-impact clinical events.

Through these objectives, the use case provides an interpretable and forward-looking framework for managing the dynamic complexities of oncological care and patient evolution.

2.3 Infrastructure Life Cycle Assessment

The primary objective of this use case is to develop a digital twin for pipe networks that can accurately identify and forecast lifecycle assessment (LCA) states and critical incidents. At its core, the initiative seeks to shift infrastructure management from a predominantly reactive approach to a predictive and proactive one, enabling earlier intervention and more informed decision-making. By anticipating key lifecycle events, the digital twin will provide data-driven intelligence to support maintenance planning, service optimization, and timely refurbishment actions.

To achieve this, the use case focuses on creating a continuously updated digital representation of the pipe network that captures both real-time conditions and historical LCA states. It aims to forecast rare yet high-impact events, such as defects, malfunctions, and degradation processes that compromise performance and reliability. In addition, the approach supports the early prediction of End-of-Life conditions, allowing repairs, replacements, or refurbishments to be planned well in advance. The digital twin will also identify lifecycle states associated with inefficient operation or increased CO_2 emissions, enabling targeted interventions that improve both performance and sustainability. By leveraging EVENFLOW's advanced forecasting capabilities, the system is designed to learn from sparse historical data and reliably predict infrequent but critical events. Collectively, these efforts aim to enhance the resilience, sustainability, and cost-effectiveness of pipe network management in the face of highly consequential lifecycle risks.

3 Use Case Developments

3.1 Industry 4.0

3.1.1 Experimental Platform and Dataset Generation

The evaluation relied primarily on a high-fidelity simulation environment, as the physical Festo Robotino robots proved less reliable for repeatable testing due to hardware limitations and navigation inefficiencies with the default Nav2 stack. The simulation environment was enhanced to support multi-robot scenarios, high-level planning, and dynamic task assignments (Figure 1).

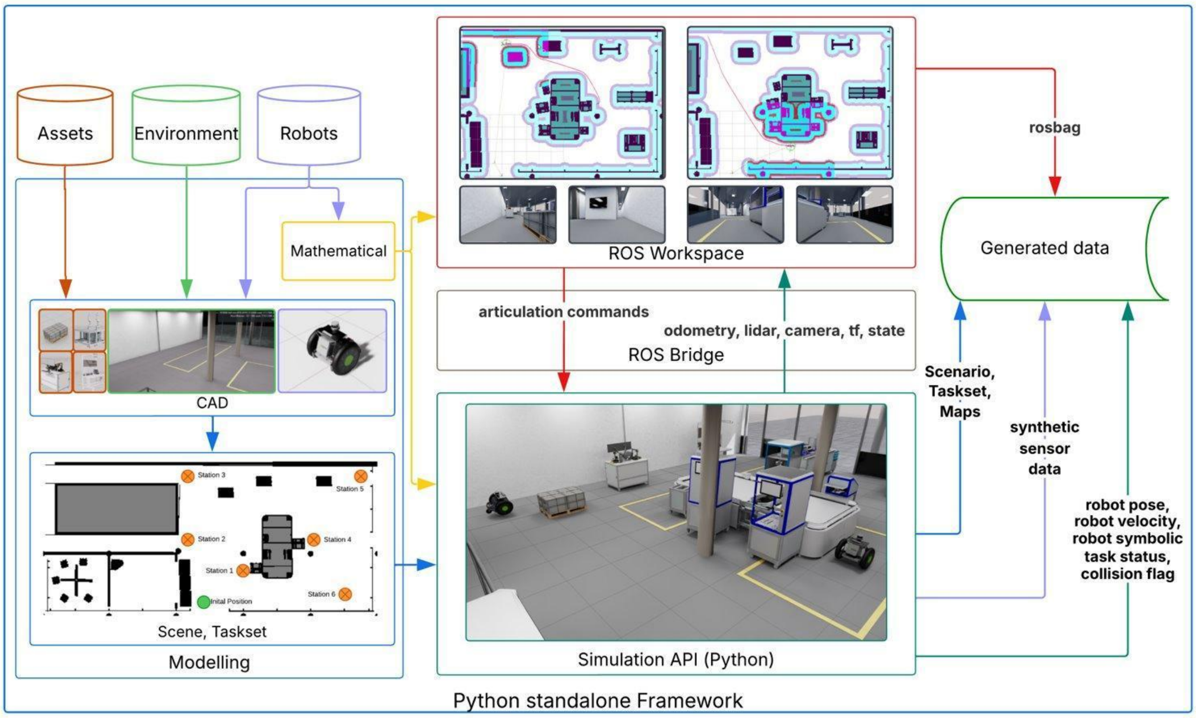


Figure 1: Multi-Robot Simulation Pipeline.

In the simulation, two robots operate simultaneously across a set of six factory stations, representing machine modules or workstations. Each station may provide additional contextual information through RGB images corresponding to the tasks being executed. Robots follow high-level plans that define the sequential order of station visits, ensuring that the trajectories encompass realistic interactions and potential conflict situations.

The simulation generates comprehensive datasets for model development and evaluation. These datasets include robot positions, velocities, goal completion status, deadlock and collision flags, and RGB images from robot-mounted cameras. By providing a reproducible environment and detailed data capture, the simulation allows for rigorous evaluation of both forecasting models and control strategies under diverse operational conditions.

3.1.2 Predictive Models and Forecasting

Two approaches to deadlock forecasting were explored. The first and primary method uses the "Wayeb" time-series forecaster from NCSR, which leverages trajectory data and robot motion dynamics to calculate the probability of entering a deadlock state at a future time step. This model identifies patterns in the relative distances and movement between robots, providing an early warning system for potential conflicts.

The second approach explored neuro-symbolic forecasting, which analyses RGB image streams from robots to infer the intended path of observed robots. This method predicts not only the trajectory of other robots but also their high-level plan, allowing the system to anticipate deadlocks and potential bottlenecks. While this method was investigated, the final evaluation primarily focused on the Wayeb forecaster due to its robustness, reliability, and simplicity for real-time deployment in multi-robot scenarios.

3.1.3 Control and Deadlock Resolution

The control and deadlock resolution component of the EVENFLOW Industry 4.0 use case is designed to operationalize deadlock forecasts produced by the predictive models, translating early warnings into proactive motion adjustments at the robot level.

Unlike traditional reactive navigation approaches, which respond only after a deadlock has materialized, the EVENFLOW control stack assumes that deadlocks can be forecast at the level of high-level task plans. Each robot follows a predefined high-level plan consisting of an ordered sequence of workstation visits. These plans induce characteristic interaction patterns between robots, including potential deadlock configurations. The forecasting models described in Section 3.1.2 are trained on executions of such plans and provide early predictions when a currently executing plan prefix is likely to lead to a deadlock.

The role of the controller is therefore not to detect deadlocks directly, but to maintain liveness once a deadlock risk is forecast, ensuring that robots continue to make progress and do not enter mutually blocking configurations.

3.1.3.1 Liveness-Based Control Concept

Deadlocks in multi-robot navigation typically emerge from symmetric interactions, where robots slow down simultaneously or repeatedly yield to each other. To prevent this, EVENFLOW enforces liveness constraints that guarantee forward progress while preserving efficiency.

Let $v_1(t)$and $v_2(t)$denote the linear velocities of two interacting robots at time t. When the forecaster predicts an upcoming deadlock along the currently executing high-level plan, the controller enforces a velocity asymmetry constraint**:**

$$max(v1(t), v2(t)) \geq \alpha(p_d) \cdot min(v1(t), v2(t))$$

where $\alpha(p_d)$is a risk-adaptive liveness parameter based on the deadlock probability $p_d \in [0,1]$ predicted by the forecaster. In the experiments reported here, $\alpha = 2$ is used as a nominal value. More generally, α can be adjusted according to the strength of the forecast:

$$\alpha(p_d) = 1 + (\alpha_{max} - 1) \cdot p_d$$

- Low forecast probability ($p_d \approx 0$): minimal deviation, $\alpha(p_d) \approx 1$
- High forecast probability ($p_d \approx 1$): stronger asymmetry, $\alpha(p_d) \rightarrow \alpha_{max}$

To minimize control effort, the robot which adjusts its speed (either slows down or speeds up) is chosen to require minimal deviation from its nominal planner velocity $v_i^{nom}(t)$. Formally, the liveness-adjusted target velocities $v_i^{live}(t)$ solve:

$$\sum_{i\in\{1,2\}} ||v_i^{live}(t) - v_i^{nom}(t)||^2$$

subject to:

$$\left(v_1^{live}(t), v_2^{live}(t)\right) \geq \alpha(p_d) \cdot \left(v_1^{live}(t), v_2^{live}(t)\right)$$

This ensures smooth, minimal-effort intervention, breaking symmetry only as much as necessary to maintain forward motion.

3.1.3.2 Integration with Nav2 Controllers (DWB Only)

The liveness mechanism was integrated into the Dynamic Window Approach (DWB) local planner. The liveness constraint acts as a lightweight velocity modulation layer, applied after the planner computes candidate velocity commands. This design preserves compatibility with the standard DWB stack and allows the controller to operate at high frequency (30 Hz) in a fully decentralized manner.

Key features of the DWB integration:

- continuous forward motion,
- symmetry breaking in conflict situations,
- forecast-driven activation based on deadlock probability,
- minimal deviation from nominal path-following behaviour, ensuring smooth and efficient trajectories.

Implementation Concept:

- Each robot evaluates a deadlock potential function based on predicted robot interactions and applies a liveness constraint if forward progress is threatened.
- Velocity commands are modulated in real time to prevent stalling while maintaining goal-directed motion.
- Constraints operate in a decentralized manner, requiring no centralized coordination.

3.2 Personalized Medicine

3.2.1 Use case scope and approach

The EVENFLOW Personalized Medicine use case has focused on advancing computational approaches to interpret temporal omics data in cancer, bridging cutting-edge machine learning techniques with clinically relevant insights. The project explored innovative strategies for both data augmentation and dynamic process reconstruction, leveraging Variational Autoencoders (VAEs) and their extensions to address the complex heterogeneity of cancer progression. The overall strategy combines a solid assessment of the state of the art with two complementary methodological application areas: data augmentation and dynamic process reconstruction.

3.2.2 State of the art survey

As a foundation for the use case, a systematic review on the application of Variational Autoencoders (VAEs) to temporal omics inference in cancer has been conducted and is currently under review in *NAR Genomics & Bioinformatics*. A pre-print is available at biorXiv [REF-01]. The paper provides a comprehensive review of the application of VAEs in cancer research over the past decade, with a particular emphasis on studies leveraging omics data. The work focuses on how VAEs have been employed to model complex, high-dimensional biological data and support cancer-related tasks (Figure 2). The review shows that VAEs have been widely and successfully applied to static analyses, including cancer subtyping, diagnosis, and prognosis. However, it also reveals that the use of VAEs to explicitly model temporal tumor evolution remains limited. Most existing studies rely on cross-sectional datasets, and only a small fraction attempt to capture time-dependent processes such as disease progression or staging. A key limitation identified is the scarcity of longitudinal omics datasets, which constrains the development and validation of models aimed at dynamic inference. As a result, important biological questions related to cancer evolution and temporal trajectories remain underexplored within the current VAE literature. We propose that future research should more fully exploit the generative capabilities of VAEs to model cancer dynamics over time. Such approaches could enable improved reconstruction of disease trajectories, facilitate the study of stage transitions, and ultimately provide deeper insights into tumor evolution.

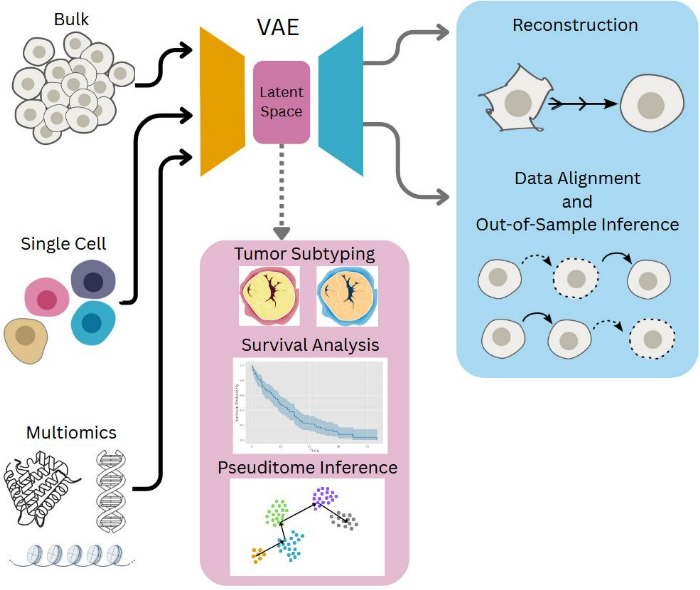


Figure 2: Common representation learning approaches in cancer research. VAEs encode diverse omics data into a latent space mainly used for subtyping, prognosis, and pseudo-time inference. In contrast, decoder-based applications for data reconstruction and temporal cancer progression remain largely underexplored.

3.2.3 Application 1: Data Augmentation

The first application area of the EVENFLOW Personalized Medicine use case focuses on the use of VAEs to generate realistic synthetic omics data for cancer research. This approach addresses data sparsity and imbalance while enabling controlled exploration of biological variability. Special emphasis is placed on model transparency and robustness through the integration of explainable AI (XAI), fairness metrics, and comparisons with alternative modelling paradigms, including multilayer networks and multiscale mechanistic simulations. Results on **medulloblastoma**, a childhood brain tumour, are reported in the pre-print available in biorXiv and bound to be submitted shortly to *Nature Communications* [REF-02].

This work investigates molecular heterogeneity in medulloblastoma using VAEs, refining the canonical subgroup stratification (groups WNT, SHH, Group 3 or G3 and Group 4 or G4) and identifies an intermediate subgroup between G3 and G4. Leveraging the largest available medulloblastoma transcriptomics cohort, we employ a VAE-based pipeline to generate high-quality synthetic data, enabling detailed exploration of the G3–G4 boundary. Explainability methods are integrated to interpret latent representations and uncover gene expression patterns driving subgroup separation. The key contributions include the identification of an intermediate G3–G4 subgroup and the characterization of genes underpinning distinctions among the four canonical medulloblastoma subgroups (WNT, SHH, G3, and G4). Moreover, a three-class classifier that explicitly accounts for the putative G3-G4 subgroup among patients traditionally labelled as G3 or G4 achieves performance comparable to the conventional binary classifier when trained on synthetically balanced data (Figure 3A). Notably, this model also exhibits the lowest equal opportunity gap, indicating fairer performance across groups (Figure 3B). This is particularly relevant because misclassification between G3 and G4 can directly affect treatment decisions, as G3 patients typically receive more aggressive therapy than G4 patients, and the inclusion of a third subgroup may enable the development of more personalized treatment strategies. Overall, the work demonstrates how generative modelling combined with explainable AI can improve interpretability and clinical relevance of generative AI approaches in paediatric brain tumor research.

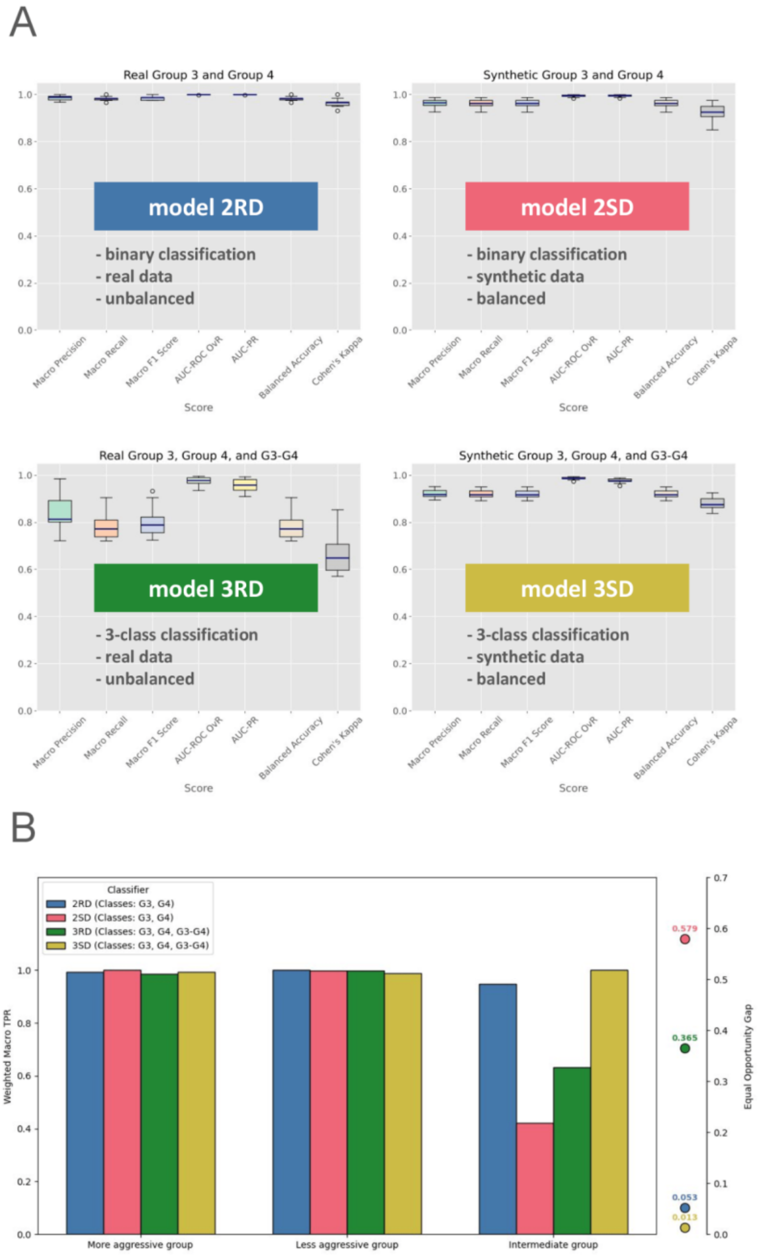


Figure 3: (A) Performance metrics of four classifiers applied to patients traditionally labelled as G3 or G4: binary models trained on unbalanced real data (2RD) and balanced synthetic data (2SD), and three-class models trained on unbalanced real data (3RD) and balanced synthetic data (3SD). (B) Weighted macro true positive rate (TPR) for the four classifiers across three patient groups (more aggressive, less aggressive, and intermediate). Equal opportunity gap values for each classifier are shown on the right.

3.2.4 Application 2: Dynamic Process Reconstruction

The second application area focuses on reconstructing temporal and pseudo-temporal dynamics from cross-sectional or partially aligned omics datasets. By combining VAEs, rule learning, and probabilistic event forecasting, this work aims to infer latent trajectories, identify key transitions in disease progression, and enable predictive modelling of cancer evolution. We investigated these approaches in **breast invasive carcinoma** (BRCA) and **kidney renal clear cell carcinoma** (KIRC). In both cases, the corresponding synthetic datasets are publicly available on Zenodo [REF-03]. For KIRC, a manuscript is currently in preparation for submission to *Nature Machine Intelligence*. Additional details on the KIRC results are provided in Section 4.2.

3.3 Infrastructure Life Cycle Assessment

The Infrastructure Life Cycle Assessment use case is developing the datasets analysis gathered from a pipe section (Figure 4), with the aim of identifying states and incidents ahead of time and with high accuracy. In particular the LAB testing (at UNIPA), and the small scale pilot at EKSO premises (with preliminary AI processing) had led to the definition of the best set of technology (vibration sensors) already deployed in a full scale pilot on a real potable water pipe section, in operations and at present producing data continuously.

3.3.1 Experimental Platform and Dataset Generation

The main features of the Small-scale Pilot are the following:

1. Vibration time series from 1 sensor (event labelled) locally registered;
2. Frequency: 6,6 ksps;
3. Magnitude: 800MB on compressed CSV file;
4. Limited time frame measurements: 1 hour;
5. Different simulated leakage in distance and size.

The main features of the Full scale Pilot are the following:

- Vibration time series from 10 sensors (event labelled) remotely registered;
- Frequency:1,6 ksps (each sensor-BUS main constraint);
- Magnitude: 14MB/10min. (all sensors) in Binary format (Numpy zipped)
- Continuous measurement: 24/7
- Sigle leakage simulation.

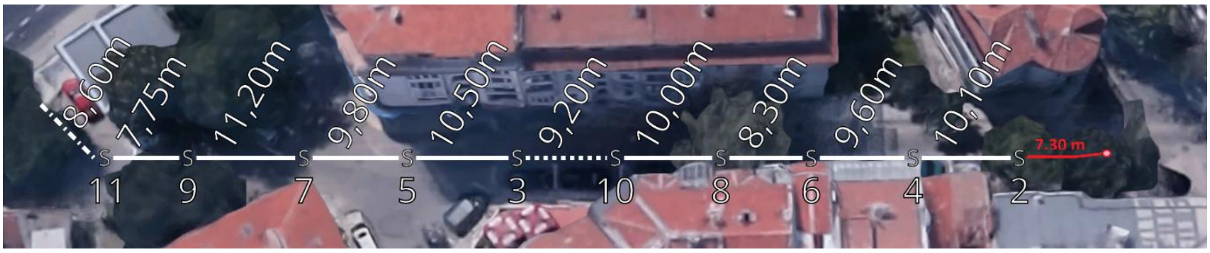


Figure 4: Aerial view of the full-scale pilot potable water pipeline equipped with vibration sensors.

This is enabling EKSO to make educated, data-driven analysis regarding some major relevant phenomenons/defects, operational or structural that could affect the pipe efficiency:

- General Anomaly detection
- leak presence

- sizing leak
- locating leak
- Signal evolution in time
- Ageing phenomenons

These are the main KPIs to base real time evaluation of the efficiency of the pipe status and operations to consequently make decisions about maintenance, service, and repairs of the pipes, including the implementation of refurbishment operations when required sizing and locating them at the best.

First results on the Full Scale Pilot, on the Leak detection and location tell us that one sensor's recordings analysis alone are confirming that it is not enough: Pre-processing one sensor's recordings gives accuracy from 60% to 97%. Progressing on a major complex scenario using the closest sensor, the model increases accuracy to 98.5%; if we use all 10 sensors, we increase accuracy to 99.8 %.

Regarding the short term target related to the leak detection and sizing, data classification issues under consideration are the following:

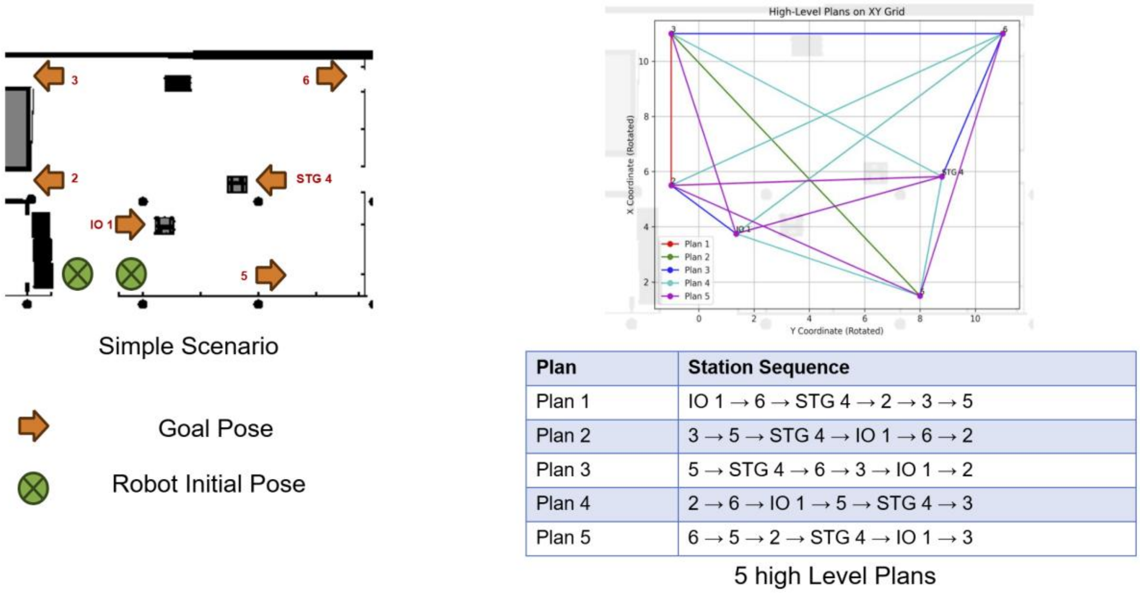
- During training: Create a model for each sensor that answers the relevant question "Is there a leakage X meters to my right/left?"
- During inference: Run each produced model on each sensor's test dataset. Compute leakage location as the average location of the locations given by the 100 models.
- During visualization: When detecting a leakage, if the leakage is not consistently (all the sensors together) detected for the next 1 minute or so, consider it FALSE ALARM.
- Prepare more data to train the models using leakages observed from different locations.

4 Application of the EVENFLOW framework to the Use Cases

4.1 Industry 4.0: Neurosymbolic Deadlock Recognition

A deadlock refers to a situation in which a robot is unable to proceed due to obstacles, including unforeseen interactions with another robot. Such scenarios can lead to unexpected delays within the industrial setting, and it is thus desirable to be able to forecast and consequently avoid them. We would like to achieve the above in a decentralised manner, in which each robot is tasked to forecast its own deadlock given only data streams from its own sensors, such as image streams from a mounted camera and tabular streams from mobility sensors (e.g. position, orientation, velocity, etc.).

Because the deadlock condition intuitively depends on quantities that change over time, such as speed and acceleration, it is natural to express it as a temporal specification. This formulation is also instrumental for the aim of forecasting such situations. In particular, we model deadlock as a deterministic finite automaton (DFA) which operates on the distance between the two robots. Conceptually, we think of this specification as domain knowledge, i.e. it could be constructed by a domain expert. In reality, this automaton is learnt from sequences of mobility data from the two robots, including positive (including deadlock) and negative (not including deadlock) trajectories.



Simple Scenario

Plan	Station Sequence
Plan 1	IO 1 → 6 → STG 4 → 2 → 3 → 5
Plan 2	3 → 5 → STG 4 → IO 1 → 6 → 2
Plan 3	5 → STG 4 → 6 → 3 → IO 1 → 2
Plan 4	2 → 6 → IO 1 → 5 → STG 4 → 3
Plan 5	6 → 5 → 2 → STG 4 → IO 1 → 3

5 high Level Plans

Figure 5: Simple multi-robot scenario and high-level plans illustrating decentralized deadlock forecasting.

We construct an out-of-distribution train/test split, wherein we train using sequences from trajectories following Plans 1-4, and test on sequences taken from Plan 5. This is an 80/20 train/test split. Initially we perform 5-fold cross-validation on the 80 trajectories in order to tune an early stopping parameter. Using this parameter, we then train on all 80 trajectories and test on the held-out set of 20 trajectories.

Table 2: Models, tasks, and performance metrics on training and test datasets.

Model	Task	Performance Metrics			
		train		test	
CNN	Coordinate regression	MAPE*		MAPE	
		3.43%		3.65%	
CNN+LSTM	Sequence classification	Accuracy	F1-Score	Accuracy	F1-Score
		99.1%	98.4%	87.9%	81.3%
CNN+DFA	Sequence classification	Accuracy	F1-Score	Accuracy	F1-Score
		92.3%	80.0%	93.8%	84.0%

* Mean Absolute Percentage Error

We see that while a neural system trained end-to-end on sequence classification for deadlocks outperforms the NeSy system in the training set, these roles flip when tested on a OOD setting. It is worth noting that conceptually the OOD setting is not adversarial, in fact the images are quite similar (the robots are still moving in the same space but are simply following different paths). It is precisely this type of generalisation that is achieved from the introduction of domain knowledge into the system, encoded here as a temporal specification through an automaton.

4.1.1 Early Deadlock Recognition

4.1.1.1 Summary

We evaluate early deadlock recognition, event forecasting, on a shared dataset using three method families: (1) a purely neural LSTM classifier that maps prefixes, early events in the sequence, directly to labels, (2) Forward Recognition, a modular generative + symbolic pipeline that samples suffixes and applies a symbolic DFA to detect deadlocks in the generated sequence, and (3) Wayeb, a symbolic probabilistic forecasting tool. All methods operate on the same trajectories and are compared across a sweep of observation earliness (prefix length k) using metrics, such as precision, recall and F1 score.

4.1.1.2 Introduction

Early classification asks: given an observed prefix $X_{1:k}$ of a discretized trajectory, can we reliably predict whether the unobserved continuation will lead to a deadlock. This task trades off earliness and reliability: smaller k gives earlier warnings but less information. Our evaluation measures performance as a function of k to identify practical operating points for proactive interventions.

4.1.1.3 Data and preprocessing

Experiments use logged robot trajectories whose continuous inter-robot distances are discretized with SAX into 40 symbols. Each trajectory, which contains 20 timepoints, is split into an observed prefix, e.g. 8, and a target suffix, e.g. 12 respectively. For the variable-length generator we sample multiple prefix lengths per example to improve robustness to different observation horizons.

In the figure below we see a sample of 100 trajectories and the values they take at each timepoint. We observe that positive sequences (depicted in orange) have a decreasing trend, which means that the distance between the two robots decreases as time passes in the positive trajectories.

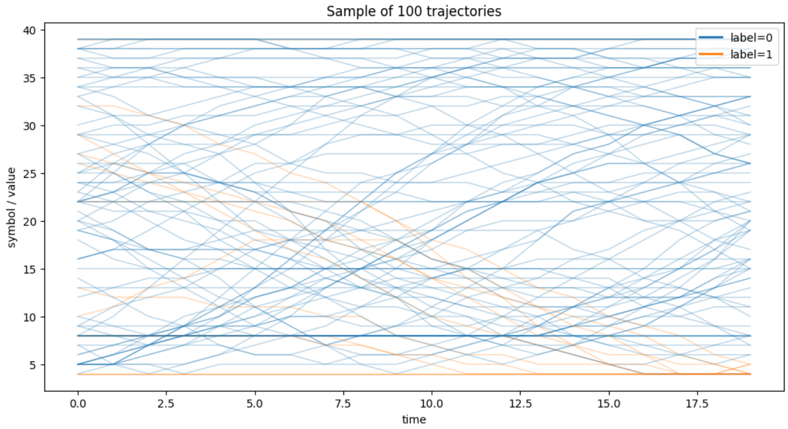


Figure 6: Sample of 100 discretized robot trajectories over time, showing that positive sequences (orange) exhibit a decreasing inter-robot distance trend compared to negative sequences (blue).

The dataset is imbalanced. The training split contains 833 sequences (715 negative / no-deadlock, 118 positive / deadlock), while the test split contains 208 sequences (181 negative, 27 positive). Because the positive (deadlock) class is the minority, accuracy alone can be misleading. Therefore we focus on precision, recall and the F1 score for the positive class as primary evaluation metrics.

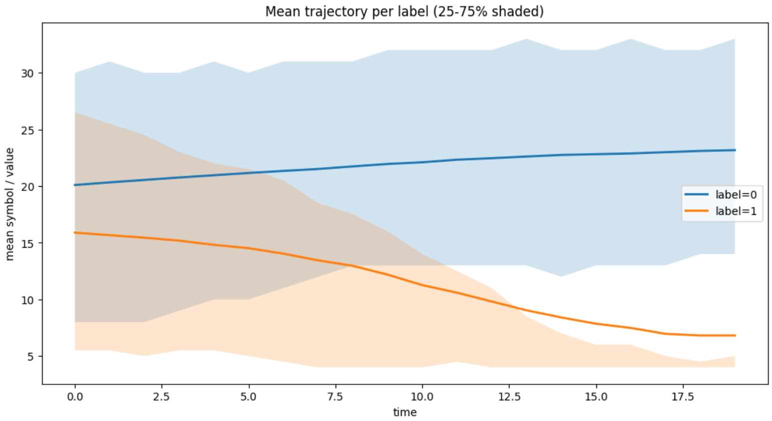


Figure 7: Mean discretized trajectory per label over time, with shaded regions indicating the 25th-75th percentile range, highlighting the decreasing inter-robot distance trend in positive sequences.

The core objective is early and reliable deadlock detection. This means maximizing the F1 (macro) score as early as possible (small k). We report metrics across prefix lengths and compare error modes (precision vs recall) and robustness to prefix variability. To find the best method for this task, we apply the following methods.

4.1.2 Methods

4.1.2.1 Purely Neural

The purely neural method implements[2] an end-to-end LSTM classifier that maps a fixed-length observed prefix directly to a binary deadlock label. Input sequences are encoded over the 40-symbol vocabulary and the model consists of a single LSTM layer (hidden_size=128), whose final hidden state is projected to two logits corresponding to the positive / negative classes. Models are trained with cross-entropy loss using the Adam optimizer (default learning rate 1e-3) and configurable epochs and batch size.

The purely neural design is attractive for its simplicity and for avoiding hand-coded rules: a single model learns discriminative features directly from prefixes. This simplicity comes with trade-offs, such as that the classifier can be sensitive to the class imbalance and that this is a black-box architecture with no information being able to be given to the system manager, in order to do replanning.

4.1.2.2 Forward Recognition Methodology

With this methodology we study[3] early deadlock forecasting for mobile robots by combining generative sequence prediction with symbolic recognition. The pipeline predicts future discrete distance values from partial observations and evaluates whether the completed trajectory satisfies a deadlock pattern encoded as a deterministic finite automaton (DFA). We compare four generators, a second-order Markov chain, a sequence to sequence LSTM, an

[2] *GitHub private repository: https://github.com/EVENFLOW-project-EU/dfki-forward-recognition/blob/main/methods/lstm_classification.py*

[3] *GitHub private repository: https://github.com/EVENFLOW-project-EU/dfki-forward-recognition*

autoregressive LSTM trained on fixed-length prefixes, and a variable-length autoregressive LSTM, and quantify recognition performance (accuracy, precision, recall, F1) as a function of observation earliness. The results show a clear earliness–accuracy trade-off: neural autoregressive approaches attain the highest F1 at early prefixes, while variable-length training improves scores at late prefixes.

4.1.3 Introduction

Being able to forecast deadlocks before they occur enables proactive replanning and safer operation in multi-robot systems. We frame the task as forward recognition: given an observed prefix of discretized inter-robot distances, predict whether the unobserved continuation will lead to a deadlocked configuration. Our methodology separates prediction and recognition: a generative model hypothesizes suffixes, and a symbolic DFA determines whether the hypothesized full trajectory matches the deadlock pattern.

4.1.4 Problem formulation

Let $X_{1:k}$ denote the observed prefix of discrete tokens. The goal is to decide whether the concatenated sequence $X_{1:k}$ || $S_{k+1:T}$, where $S_{k+1:T}$ is a suffix sampled from a generative model conditioned on $X_{1:k}$, is accepted by the deadlock SFA. We study how classification quality varies with k to quantify how early reliable forecasts can be made.

4.1.5 Architecture

4.1.5.1 Generator Component

We evaluated four generators with different inductive biases and computational costs.

- **Second-order Markov chain**: This baseline estimates next-token probabilities by counting observed triplet transitions and normalizing with a small additive smoothing constant. Training is a simple fit of empirical counts. There is no gradient-based loss.
- **Seq2Seq LSTM**: A single-layer encoder processes the prefix and a dense projection produces logits for the whole suffix. Training minimizes cross-entropy aggregated over suffix positions. This model captures if there is a summary/encoding of the prefix useful for multi-step prediction.
- **Autoregressive LSTM (fixed-length)**: The encoder LSTM consumes a fixed-length prefix and an LSTMCell decoder generates the suffix one step at a time. Training uses teacher forcing and token-level cross-entropy, while at inference the model decodes autoregressively with greedy sampling.
- **Variable-length autoregressive LSTM**: Architecturally similar to the fixed-length autoregressive model but trained on many prefix lengths sampled from full sequences. The loss ignores padded positions in the suffix and is therefore robust to variable-target lengths. This training strategy yields better performance when inference prefixes vary in length.

4.1.5.2 Recognition (symbolic) component

Recognition is performed by a deterministic finite automaton that encodes domain knowledge about deadlock configurations. The automaton’s acceptance of a generated full sequence is interpreted as a positive deadlock forecast. This design isolates symbolic decision rules from generative errors and supports interpretable failure analysis.

To generate the automaton we use Answer Set Automata Learning (ASAL). ASAL is a framework for learning and revising complex event patterns represented as symbolic finite automata (SFA) from labelled streams of multivariate event-based data. In ASAL, a symbolic temporal model that accepts or rejects the input event traces is encoded as an answer set automaton (ASA), i.e., an answer set program that combines a generic automata interpreter, a specification of the automaton's structure (states and transitions), background predicates that operate over event tuples (e.g., trends, thresholds, attribute comparisons etc), and transition guard rules, defined as Boolean combinations of the background predicates.

A high-level diagram of the whole pipeline - methodology described above:

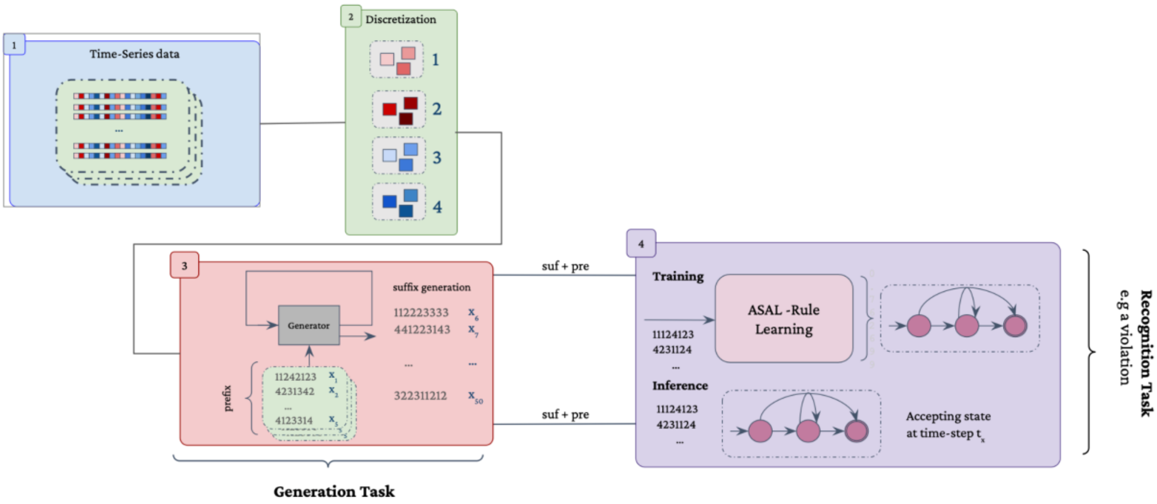


Figure 8: High-level diagram of the ASAL pipeline.

We also compare the above methods with a purely symbolic one. For this symbolic method we just use the automaton for the prefix sequences only. If the automaton does not reach an accepting state by the end of the prefix sequence, we consider the at hand sequence, a sequence with no deadlock and label it as such, otherwise we label it as 1, because the accepting state has been reached.

This method serves as a proof that the deadlock does not happen that early in the sequence and that the automaton alone is not sufficient for forecasting.

4.1.6 Methodology

We sweep prefix lengths k from small (very early observations) to large (near-complete trajectories) and report recognition accuracy, precision, recall and F1 for the positive (deadlock) class at each k. Each generator is trained using standard cross-entropy optimization (except the Markov baseline, which is fitted via counts), with hyperparameters chosen to balance convergence and computational cost.

4.1.7 Results

The experiments reveal consistent patterns across models. At early prefixes (k small, e.g., 2-6), predictions are highly uncertain and recognition F1s are low (roughly 0.3-0.5). As more of the trajectory is observed, performance improves. This is mostly a sanity check, because our primary interest is actually earlyness. Therefore, we focus our analysis on prefix lengths k in the range 5-14.

Within the earliness window the autoregressive LSTM attains the highest average F1 and consistently outperforms seq2seq and the Markov baseline **on average**. The variable-length autoregressive model narrows the gap and provides better robustness across earlier prefixes (it shows smaller drops at smaller k). The Markov chain shows a high single-point peak at k=11,14, likely reflecting a favourable alignment between simple generative predictions and the DFA at that prefix length m but its mean performance across the earliness window is lower than the neural autoregressive approaches.

Table 3: F1 scores show that autoregressive models outperform others in the earliness window.

Type of Generator component / Prefix	**Markov Chain**	**Sequence to Sequence**	**Autoregressive**	**Variable-length Autoregressive**
k=3	0.308	0.42	**0.452**	0.4
k=5	0.414	0.491	**0.518**	0.441
k=8	0.525	0.557	**0.570**	0.557
k=10	0.59	**0.592**	0.588	**0.591**
k=13	0.635	0.623	0.655	**0.669**
k=15	**0.764**	0.664	0.693	0.731
k=17	0.778	0.78	0.881	**0.834**

In the plot below the F1-score vs prefix-length plot shows the earliness-accuracy trade-off directly: each curve corresponds to one generative+recognizer pipeline and the vertical axis reports F1 for the positive (deadlock) class at each observed prefix length.

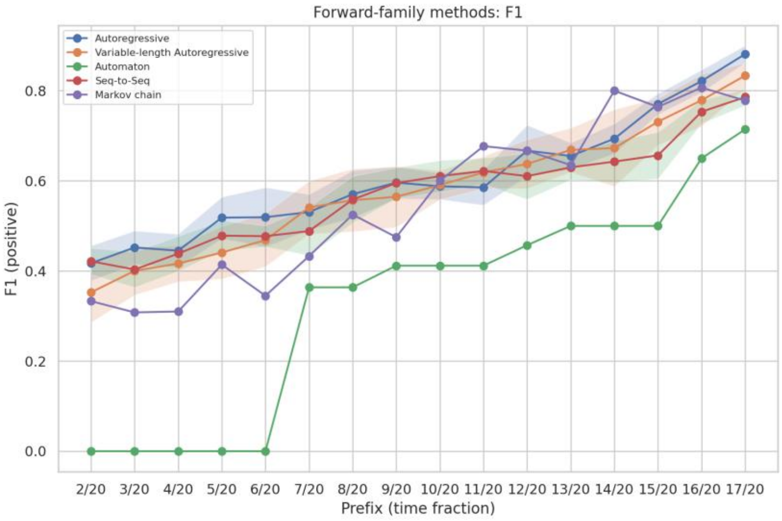


Figure 9: F1 scores for the positive class versus prefix length, illustrating the earliness-accuracy trade-off for each generative+recognizer pipeline.

Precision/recall plots expose differences in error modes: some models trade precision for recall at particular prefix lengths, while others remain more balanced.

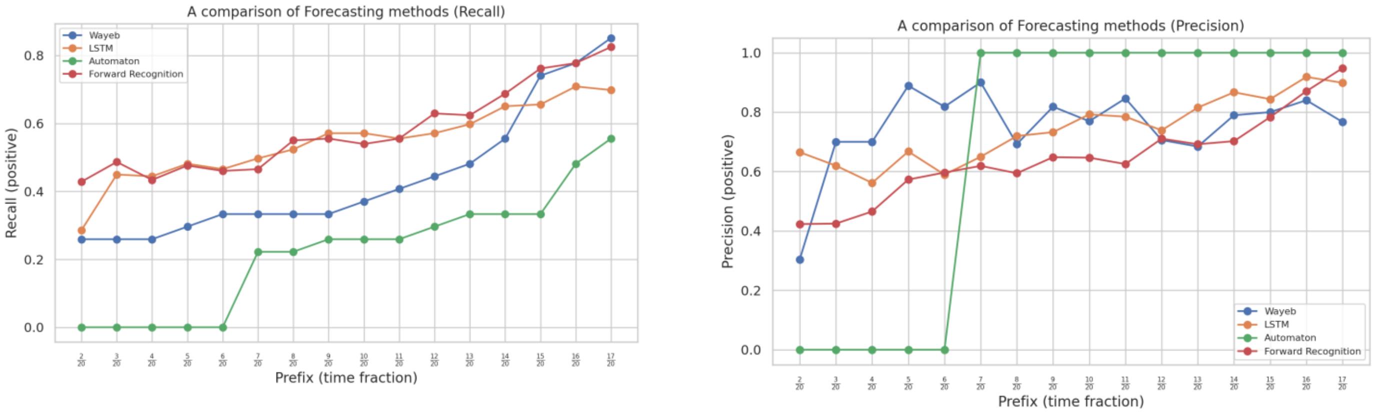


Figure 10: Precision–recall plots highlight differences in error trade-offs across models at varying prefix lengths.

Finally, plotting the generative loss (cross-entropy) across prefixes shows that prediction confidence and sharpness generally increase with prefix length. Inspecting loss together with recognition metrics helps distinguish cases where lower loss does (or does not) translate into better symbolic recognition.

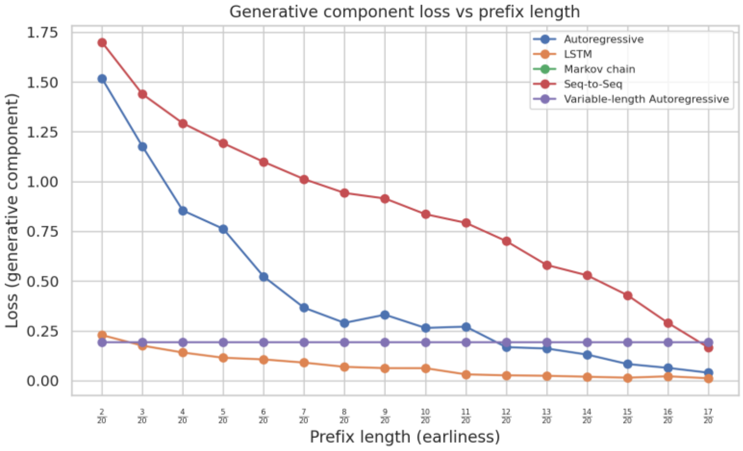


Figure 11: Generative loss (cross-entropy) across prefixes, showing increasing prediction confidence with longer prefixes and its relationship to recognition performance.

4.1.7.1 Method comparison

We compare three approaches for early deadlock recognition: the purely neural LSTM classifier, Forward Recognition with autoregressive generation, and Wayeb (in this setting, Wayeb uses the discretized SAX data and the same ASAL-generated automaton as in the method Forward Recognition). All methods operate on the same discretized SAX trajectories and are evaluated across varying prefix lengths.

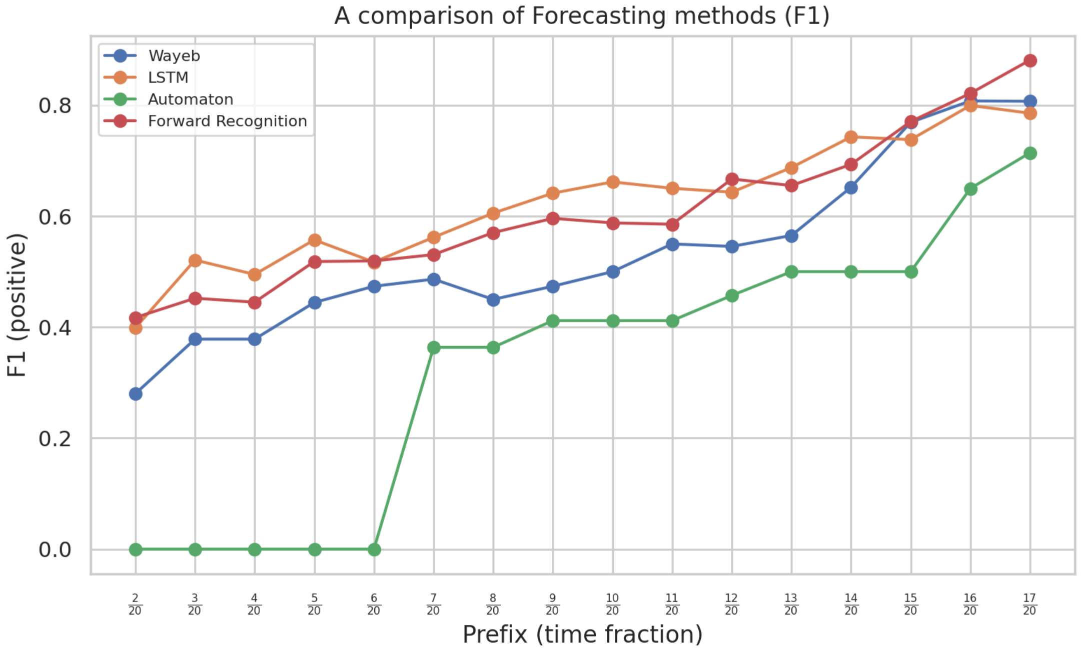


Figure 12: Comparison of early deadlock recognition methods (LSTM classifier, Forward Recognition with autoregressive generation, and Wayeb) across prefix lengths on discretized SAX trajectories.

The purely neural method achieves the highest average F1-score, benefiting from its end-to-end discriminative learning. However, it provides no interpretability or uncertainty quantification, and predictions are opaque binary classifications.

Forward Recognition offers modularity and interpretability: the DFA acceptance provides an explainable decision tied to symbolic domain knowledge. The autoregressive generator achieves competitive performance, particularly at early prefix lengths (k=5-10), though its reliance on generated suffixes introduces error propagation from the generative component.

Wayeb provides probabilistic forecasts with uncertainty estimates via confidence intervals, enabling risk-aware decision-making. Its use of variable-order Markov models captures long-term dependencies efficiently, though performance depends on the quality of the learned PST and the automaton structure.

In summary, we can conclude that the purely neural approach maximizes F1 on average but sacrifices transparency, while Forward Recognition balances performance with interpretability. Wayeb uniquely offers uncertainty quantification alongside symbolic reasoning, but worse average performance overall.

4.2 Personalized Medicine: Hybrid Methods for early KIRC transition recognition

4.2.1 Summary

This work investigates hybrid methods for recognizing and forecasting stage transitions in Kidney Renal Clear Cell Carcinoma (KIRC) from transcriptomic time series. By utilizing discriminative machine learning, neural sequence models, and symbolic event-based forecasting we aim to characterize predictive signals in both patient-derived and synthetic longitudinal data. In collaboration with the Barcelona Supercomputing Center (BSC), we first establish baseline stage classification capabilities on the KIRC dataset. We then evaluate trajectory classification methods on VAE-generated synthetic sequences, demonstrating that purely neural network methods achieve strong full-trajectory discrimination performance. For the critical early-warning task, we compare a purely neural baseline against a neurosymbolic approach that combines ASAL, a framework for learning interpretable finite-state automata from discretized gene expression streams, with Wayeb, a probabilistic complex event forecasting system that produces confidence-calibrated temporal predictions. While the neural method achieves improved performance with rapid early convergence, the ASAL+Wayeb system delivers competitive scoring alongside explicit uncertainty quantification, interpretable symbolic patterns auditable by clinical experts, and probabilistic forecast intervals for transition timing. These capabilities are essential for developing trustworthy, deployable early-warning systems in high-stakes medical contexts.

4.2.2 Goal

The primary objective is to evaluate and compare methods for detecting and forecasting stage transition-related events in patient transcriptomic time series data derived from real KIRC samples. The work focuses on following tasks:

- The development of a baseline classification performance for KIRC stage discrimination and the identification of a minimal, interpretable gene panel that improves this performance
- The implementation of a time-series model that separates trajectories that proceed to late-stage from those that remain early and finally,
- The investigation of symbolic forecasting approaches that provide early-warning predictions with explicit uncertainty for disease progression.

4.2.3 Data and Preprocessing

4.2.3.1 Static TCGA Dataset

The core dataset comprises bulk RNA-sequencing gene expression profiles and corresponding clinical annotations for 530 KIRC patients obtained from The Cancer Genome Atlas (TCGA) via the UCSC Xena browser. The original data encompass expression measurements for 8,516 genes following preprocessing steps that removed genes with zero expression in at least 20% of patients and genes exhibiting both mean and variance below 0.5. Clinical metadata include pathological stage classifications (Stages I, II, III, and IV), which were binarized into "early" (Stages I and II) and "late" (Stages III and IV) categories to reflect clinically meaningful progression thresholds. The dataset exhibits class imbalance, with a higher proportion of early-stage patients, necessitating the use of macro-averaged F1-score as the primary evaluation metric rather than accuracy. The data were partitioned into training (424 patients, 80%) and test (106 patients, 20%) sets with stratified sampling to preserve class proportions, following the same split used in the collaborative VAE training to prevent information leakage.

4.2.3.2 Synthetic Trajectories Dataset

Synthetic longitudinal gene expression trajectories were generated by BSC using a Variational Autoencoder trained on the TCGA KIRC dataset. The VAE architecture learns a low-dimensional latent representation of patient gene expression profiles and employs a decoder to generate synthetic samples. To simulate disease progression, positive trajectories representing early-to-late stage transitions were constructed by interpolating between early-stage and late-stage patient embeddings in the latent space over 50 discrete time points. Multiple negative-trajectory construction strategies were explored to provide contrasting non-progressive patterns: (1) latent-space interpolation between pairs of early-stage patients (early-to-early trajectories), (2) augmentation of early-stage patient expression profiles with additive Gaussian noise in the original gene expression space, and (3) augmentation with Gaussian noise applied in the latent space. These synthetic datasets enable controlled evaluation of classification methods under varying degrees of class separability and noise characteristics. All synthetic trajectory datasets preserved the original train-test patient partition to maintain experimental consistency and avoid contamination between training and evaluation phases.

4.2.3.3 Preprocessing Pipeline

Gene expression values were subjected to standard normalization using MinMaxScaler to ensure all features occupy comparable numeric ranges, mitigating the influence of genes with extreme expression magnitudes. For classification experiments, we initially retained all 8,516 genes, then systematically reduced the feature space through iterative SHAP-based feature importance ranking. Patient samples with missing clinical stage annotations were excluded, reducing the dataset from an initial pool to the final 531 annotated patients. The preprocessing workflow prioritized maintaining the biological signal while removing technical artifacts and uninformative features, as evidenced by subsequent classification performance improvements following dimensionality reduction.

4.2.4 Methods

4.2.4.1 Stage Classification with XGBoost

We employed gradient-boosted decision trees via the XGBoost framework as the primary classification model due to its robust performance on high-dimensional tabular data and built-in handling of feature interactions. The classifier was configured with standard hyperparameters including 50–300 boosting rounds, maximum tree depths of 3–10, learning rates between 0.001 and 0.2, and subsample ratios of 0.5–1.0. Given the class imbalance in the dataset, we explored the scale_pos_weight parameter to adjust the contribution of minority-class samples during training. Model training employed 5-fold stratified cross-validation to ensure robust performance estimation and mitigate overfitting risks. Hyperparameter optimization was conducted using the Optuna framework with a Tree-structured Parzen Estimator to maximize macro-averaged F1-score on validation folds. The final model was trained on the full training set and evaluated on the held-out test set using multiple metrics: F1-score (macro), accuracy, precision (macro), recall (macro), and confusion matrices to assess both overall performance and class-specific discriminative capacity.

4.2.4.2 Feature Selection via SHAP

To identify a minimal gene panel that retains maximal discriminative information for stage classification, we applied SHapley Additive exPlanations (SHAP), a model-agnostic interpretability method grounded in cooperative game theory. SHAP values quantify the marginal contribution of each feature to individual predictions by computing the average change in model output when a feature is included versus excluded across all possible feature coalitions. We employed the TreeExplainer algorithm, which provides exact SHAP values for tree-based models with computational efficiency. The feature importance ranking was derived by computing the mean absolute SHAP value for each gene across all validation samples, capturing both the magnitude and consistency of each gene's contribution to stage discrimination. The feature selection procedure proceeded iteratively: (1) train a baseline XGBoost classifier on the full gene panel, (2) compute SHAP values on an independent validation set (20% of training data), (3) rank genes by mean absolute SHAP values, (4) evaluate classification performance using top-k gene subsets for varying k, and (5) select the k that maximizes validation F1-score. This process identified a panel of 45 genes that achieved improved performance compared to the full 8,516-gene feature space. The selected genes exhibit high discriminative capacity as evidenced by their SHAP importance scores, with the top-ranked genes including OASL (mean |SHAP| = 0.842), HUS1B (0.766), C9orf129 (0.533), HPDL (0.515), and SLC22A1 (0.444). The reduced gene panel seems to improve model generalization.

4.2.4.3 Trajectory classification

Prior to implementing early-warning forecasting, we establish a fundamental capability assessment through full-trajectory classification. This task evaluates whether a model can discriminate complete disease progression sequences, distinguishing trajectories that transition from early to late-stage cancer from those that remain stable, when provided with the entire temporal observation window. Full-trajectory classification serves as an upper-bound performance benchmark, representing the ideal scenario where all temporal information is available before prediction is required.

For this task, we employ a purely neural approach using an LSTM architecture. The network processes complete synthetic gene expression trajectories spanning all 50 discrete time points, with the model's recurrent layers capturing temporal dependencies across the full sequence before producing a final binary classification. Training and evaluation follow the same stratified train-test partitions used throughout the experimental pipeline, with performance measured via macro-averaged F1-score, accuracy, precision, and recall to account for potential class imbalance between progressive and non-progressive trajectories.

The baseline expectation for this task is straightforward: given access to the complete trajectory, the model should minimally achieve performance equivalent to a static classifier operating solely on the terminal time point. That’s true, because the final observed state alone suffices to determine whether a patient has progressed to late-stage disease. Superior performance would indicate that temporal patterns embedded throughout the trajectory, such as rate of change, inflection points, or sequential gene expression dynamics, carry additional discriminative signal beyond the endpoint state. This full-trajectory baseline establishes the performance ceiling against which early-warning methods must be evaluated, quantifying the predictive cost of operating with incomplete temporal information in realistic clinical scenarios.

4.2.4.4 Early Trajectory classification

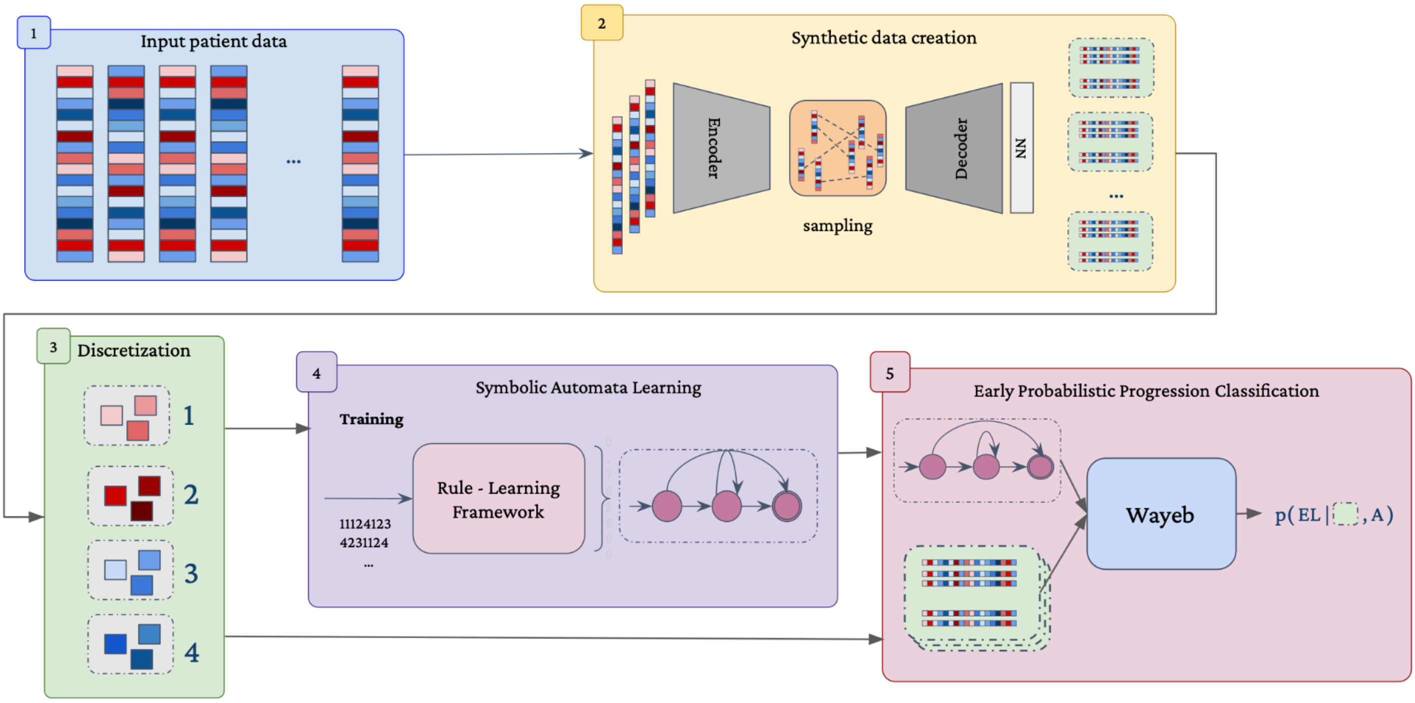


Figure 13: Overview of the hybrid neurosymbolic framework for early trajectory classification, combining ASAL for interpretable pattern discovery with Wayeb for probabilistic forecasting of disease progression from partial longitudinal data.

The early trajectory classification task extends beyond static-stage discrimination to detect impending transitions from early to late-stage disease using only partial temporal observations. Given a prefix of a patient's longitudinal trajectory, the objective is to forecast whether the sequence will ultimately progress to late-stage cancer or remain stable, thereby enabling proactive clinical intervention before irreversible progression occurs. To address this challenge, we employ a hybrid neurosymbolic approach that combines ASAL (Answer Set Automata Learning) for pattern discovery with Wayeb for probabilistic forecasting of transition events.

ASAL (Answer Set Automata Learning) serves as the pattern induction component, learning interpretable finite-state automata from discretized time-series data. Operating on symbolic representations generated with K-bins discretizer, ASAL uses Monte Carlo Tree Search over an Answer Set Programming-encoded hypothesis space to discover temporal state-transition rules that discriminate progressive from non-progressive trajectories. The framework expresses complex event recognition operators, including sequence, iteration, and filtering, through answer set automata that combine a generic automaton interpreter with learnable guard conditions defined as Boolean combinations of background predicates such as trends, thresholds, and attribute comparisons. This formulation enables ASAL to extract clinically interpretable temporal patterns while maintaining predictive accuracy through constraint-driven abductive learning.

Wayeb complements ASAL by providing probabilistic forecasting capabilities for the learned patterns. Wayeb is an online, probabilistic system designed for Complex Event Forecasting (CEF), addressing the challenge of predicting the potential occurrence of a declaratively defined Complex Event (CE) pattern (often formulated as a Symbolic Regular Expression (SRE)) within an event stream before it is actively detected by a Complex Event Recognition (CER) engine. Given a symbolic finite automaton specification and an incoming event stream, Wayeb constructs Variable-order Markov Models, specifically Prediction Suffix Trees, to capture long-term statistical dependencies without the computational burden of exhaustive state enumeration. By modelling waiting-time distributions through the theory of absorbing Markov chains, Wayeb produces forecasts in the form of confidence intervals [start, end] that predict the number of future events until pattern completion occurs with user-defined confidence threshold θ. This integration enables our system to not only detect early warning signals but also quantify uncertainty around predicted transition timing.

The complete framework is shown in Figure 13.

4.2.4.5 Neural Comparison

To establish a purely data-driven baseline for trajectory classification, we implemented a Long Short-Term Memory (LSTM) network trained directly on the synthetic gene expression time series. The LSTM architecture comprises a single recurrent layer with 128 hidden units followed by a fully connected classification head, designed to capture temporal dependencies in the last timepoint trajectories without relying on explicit symbolic abstraction or event detection. The model operates on variable-length trajectory prefixes, enabling assessment of classification performance as a function of observed sequence length-a critical consideration for early-warning scenarios where predictions must be made with limited temporal context.

The LSTM was evaluated using the same train-test splits as the symbolic methods with the discretized data, with performance measured via macro-averaged F1-score, accuracy, precision, and recall. This neural baseline serves as a comparative reference point for assessing whether explicit symbolic pattern extraction and probabilistic forecasting offer advantages over end-to-end learned representations, particularly in terms of interpretability, sample efficiency, uncertainty quantification for transition prediction tasks and performance.

4.2.5 Results

4.2.5.1 Static TCGA Classification

Baseline classification on the static TCGA dataset using the full 8,516-gene panel yielded moderate performance with macro-averaged F1-scores ranging between 0.70–0.75 across cross-validation folds, reflecting the inherent difficulty of stage discrimination from high-dimensional gene expression data. Application of SHAP-based feature selection to identify the top 45 genes resulted in improved performance, with test-set F1-score increasing to approximately 0.79 and AUC rising from 0.787 to 0.873. This improvement demonstrates that dimensionality reduction via explainability-guided feature selection effectively removes noisy, irrelevant features and concentrates the model's capacity on biologically informative transcriptomic markers. Precision and recall metrics exhibited balanced performance across early and late classes, with confusion matrices indicating relatively symmetric error distributions. The classifier achieved accuracy values exceeding 0.80, though this metric is less informative given the class imbalance favouring early-stage patients.

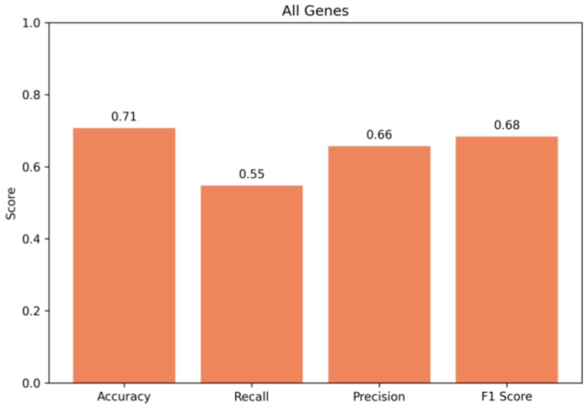


Figure 14: Baseline and SHAP-guided feature selection performance on the TCGA dataset, showing improved F1-score and AUC after reducing to the top 45 informative genes.

4.2.5.2 Feature Importance

The SHAP-based feature ranking revealed a compact set of genes with substantial discriminative capacity. The top 45 genes span diverse biological pathways, including immune response (OASL, CD1C), cell cycle regulation (MCM2, MYCN), metabolic processes (CYP3A4, CYP17A1), and developmental signalling (GATA6, FOXF2, HOXB8). Notably, genes such as OASL and HUS1B exhibited mean absolute SHAP values exceeding 0.75, indicating their consistent and substantial contribution to stage predictions across the patient cohort. The feature importance distribution exhibits a long-tailed pattern, with a small subset of genes contributing disproportionately to model decisions, while the majority of the 8,516 genes provide minimal discriminative signal. This finding validates the feature selection strategy and suggests that KIRC stage transitions are governed by a relatively focused transcriptomic signature rather than diffuse genome-wide alterations.

The full list of the 45 genes is the below:

OASL	HUS1B	C9orf129	HPDL	SLC22A1	C10orf41	LOC100132354	FGF12
TRIM36	CD1C	DNASE1L3	HS3ST1	LOC653113	CYP3A4	KIF17	FOXF2
GJB1	JAKMIP3	NUPR1	GATA6	OTOF	CES8	MCM2	MYCN
HOXB8	MYH7B	EPHB4	CYP17A1	CABYR	MADCAM1	CCDC146	KIAA0802
MASP1	DACT3	CTAGE9	MXRA7	KIAA1024	C4orf6	LRRIQ1	PLEKHH2
KLC3	AFAP1L1	FAM186B	SLC22A16	HSD17B3	-	-	-

In the table below, we also compare with an average of 10 experiments executed each time with a different random reduced gene subset. This shows that feature reduction alone does not suffice to improve performance.

Table 4: Comparison of classifier performance using the full gene panel, random 45-gene subsets, and SHAP-selected 45 genes.

# of genes	F1-macro	Accuracy	Precision	Recall
8.516	0.684	0.708	0.657	0.548
45 (random)	0.6479	0.6736	0.6036	0.5143
45 (SHAP)	**0.799**	**0.811**	**0.789**	**0.714**

4.2.5.3 Synthetic Trajectory Evaluation

As established in the Methodology section, demonstrating full-trajectory classification capability is a prerequisite before implementing early-warning forecasting systems. This evaluation serves dual purposes: first, to confirm that complete temporal sequences contain sufficient discriminative signals for stage progression prediction, and second, to inform dataset selection for subsequent early classification experiments. We present results across three synthetic dataset configurations that vary systematically in their construction of negative trajectories, those representing patients who remain in early-stage disease without progression to late-stage cancer. Positive trajectories, representing early-to-late stage transitions, are constructed identically across all datasets through latent-space interpolation between early-stage and late-stage patient embeddings over 50 discrete time points. Moreover, all datasets include only the 45-genes established in the above experiments.

- **Synthetic Trajectory Data Type A (Latent-to-Latent):** Negative trajectories are generated via latent-space interpolation between pairs of early-stage patients, producing smooth transitions that remain within the early-stage manifold.
- **Synthetic Trajectory Data Type B (Noise-Augmented):** Negative trajectories are constructed by augmenting early-stage patient profiles with additive Gaussian noise, introducing stochastic variability while preserving the fundamental early-stage characteristics. This dataset includes two subtypes based on the space in which noise is applied:
    - **Subtype B1 (Real-Space Noise):** Gaussian noise is applied directly to gene expression values in the original 8,516-dimensional feature space, simulating measurement variability and biological stochasticity at the transcriptomic level.
    - **Subtype B2 (Latent-Space Noise):** Gaussian noise is applied within the VAE's learned latent representation before decoding back to gene expression space, introducing controlled perturbations that respect the statistical structure captured by the generative model.

Table 5: Trajectories classification scores.

Dataset Types	Latent-to-Latent	Real-Space Noise	Latent-Space Noise
F1-macro score	0.5481	0.9382	0.9951

The full-trajectory LSTM classifier exhibits markedly different performance across the three synthetic dataset configurations, revealing fundamental differences in task difficulty that arise from the negative trajectory construction strategies.

The Latent-to-Latent dataset yields near-random performance (F1 = 0.5481), indicating that trajectories constructed through latent-space interpolation between early-stage patients are virtually indistinguishable from those interpolating between early and late-stage patients.

Conversely, the Latent-Space Noise configuration achieves nearly perfect discrimination (F1 = 0.9951), likely because noise perturbations in the learned latent representation create distributional shifts that are easily exploited by the classifier but do not reflect realistic biological variability.

The Real-Space Noise dataset occupies an intermediate difficulty regime (F1 = 0.9382), demonstrating strong but imperfect separability. The classification errors suggest the task remains challenging enough to stress-test early-warning methods without being artificially trivial or impossibly difficult. Therefore, we select the Real-Space Noise dataset for all subsequent early trajectory classification experiments.

4.2.5.4 Early Trajectory classification Evaluation

Having established the feasibility of full-trajectory discrimination, we now address the primary objective: early classification of disease progression from incomplete temporal observations. This task requires predicting whether a patient will transition to late-stage cancer using only a prefix of their longitudinal trajectory, simulating realistic clinical scenarios where intervention decisions must precede observable progression.

The evaluation compares our hybrid approach, combining ASAL for pattern discovery with Wayeb for probabilistic forecasting, against the purely neural LSTM baseline to assess the trade-offs between interpretability, uncertainty quantification, and predictive performance.

4.2.5.5 Learned Symbolic Automaton

A key advantage of the ASAL framework is its production of human-interpretable temporal patterns that can be audited and validated by domain experts. Operating on the top-45 gene panel identified through SHAP-based feature selection and discretized data, ASAL induced the following finite-state automaton from the Real-Space Noise training trajectories.

The learned automaton structure encodes temporal rules that discriminate progressive from non-progressive trajectories through Boolean combinations of gene expression thresholds. For instance, specific state transitions may be guarded by conditions such as "SLC22A1 expression increases above threshold τ_1 for consecutive time steps". This symbolic representation enables clinical experts to evaluate whether the discovered patterns align with known biological mechanisms of KIRC progression and also and most importantly modify the automaton based on expert knowledge.

Below is the automaton generated based on the gene SLC22A1 with ASAL:

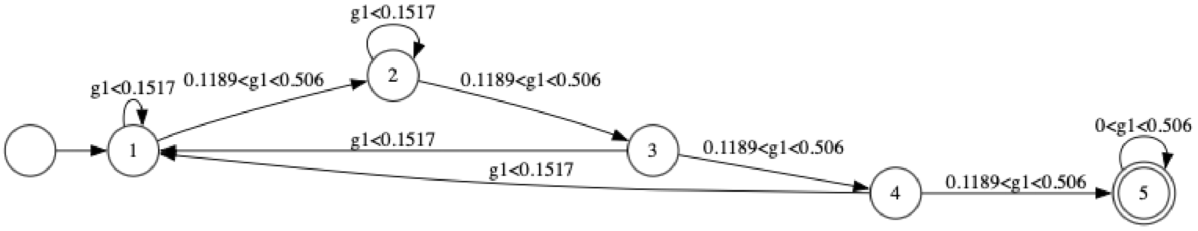


Figure 15: Finite-state automaton induced by ASAL from discretized SLC22A1 expression trajectories.

4.2.5.6 Comparative Performance Results

To evaluate early classification capability, both the ASAL+Wayeb system and the LSTM baseline were assessed across trajectory prefixes of increasing length, simulating progressively later intervention points. The neurosymbolic approach processes symbolic event streams through the learned automaton, with Wayeb computing waiting-time distributions and forecast intervals at user-defined confidence threshold θ = 0.5 to predict pattern completion timing. The LSTM operates also on discretized gene expression prefixes, producing binary classification predictions with associated softmax confidence scores.

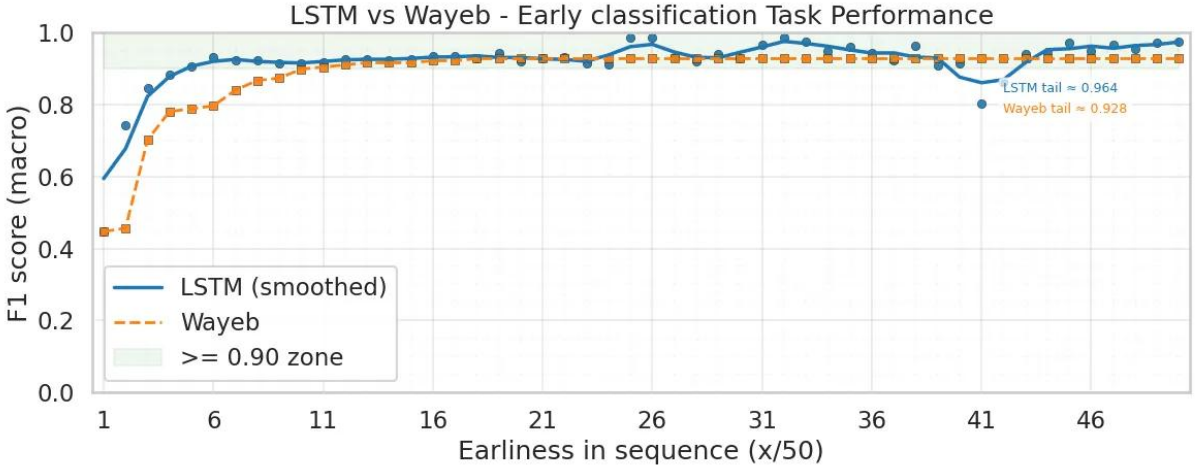


Figure 16: Early classification performance of ASAL+Wayeb versus LSTM across increasing trajectory prefixes.

4.2.5.7 Key Findings

The results reveal distinct performance characteristics and temporal dynamics between the two approaches, as shown in the figure above. The LSTM baseline demonstrates slightly improved trajectory classification performance across different timepoints in the trajectory. Notably, the LSTM reaches the threshold of F1 ≥ 0.90 remarkably early at the timepoint 5-6 (10-12% of the trajectory) and maintains stable, high performance throughout the remaining sequence. This rapid convergence suggests the neural model efficiently extracts early discriminative signals. This indicates that patterns are early detectable.

The ASAL+Wayeb system achieves competitive but slightly lower performance (F1-macro = 0.928 at trajectory completion), with a more gradual performance improvement curve. The neurosymbolic approach requires observing approximately 8-10 timepoints before crossing the F1 ≥ 0.90 threshold.

Despite the LSTM's improved performance, the ASAL+Wayeb system provides critical capabilities absent from the purely neural approach. Wayeb's probabilistic forecasting produces explicit confidence intervals for predicted transition timing. This could potentially allow for risk-stratified clinical interventions. In this setting high-confidence early warnings could trigger immediate action while uncertain forecasts prompt continued monitoring. The learned automaton patterns remain fully interpretable and auditable by domain experts, allowing validation against known KIRC progression mechanisms and identification of novel biomarker dynamics. Furthermore, the symbolic rules can be iteratively refined through expert feedback

The results suggest that hybrid neurosymbolic approaches offer a viable path toward trustworthy early-warning systems that balance predictive accuracy with the transparency and auditability demanded by high-stakes medical decision-making.

4.3 Infrastructure Life Cycle Assessment

4.3.1 Overview

In this section we apply and evaluate neurosymbolic forecasting techniques on a real-world industrial use case provided by EKSO, involving water pipe leakage detection. The raw data consist of high-frequency univariate time series recorded from a pressure sensor on a water pipe under different “scenarios,” such as all taps closed or individual taps opened abruptly.

Our goal is twofold:

- Learn a robust classifier that can map short time windows of the pressure signal to high-level “simple events” (the pipe/tap scenario at that time).
- On top of these learned simple events, build a temporal model that captures how scenarios evolve over time and use it to forecast future events in a neurosymbolic fashion.

The approach proceeds in three stages:

1. Supervised simple-event classification on the original EKSO signal.
2. Construction of a synthetic temporal dataset, where sequences of high-level events follow a controlled Markovian pattern.
3. Semi-supervised learning of a latent Markov model with a mutual-information objective (MiMM - see Chapter 8 of Deliverable D4.2), and use of the learned Markov chain in combination with probabilistic model checking for neurosymbolic forecasting.

4.3.2 Raw EKSO Dataset and Supervised Classification

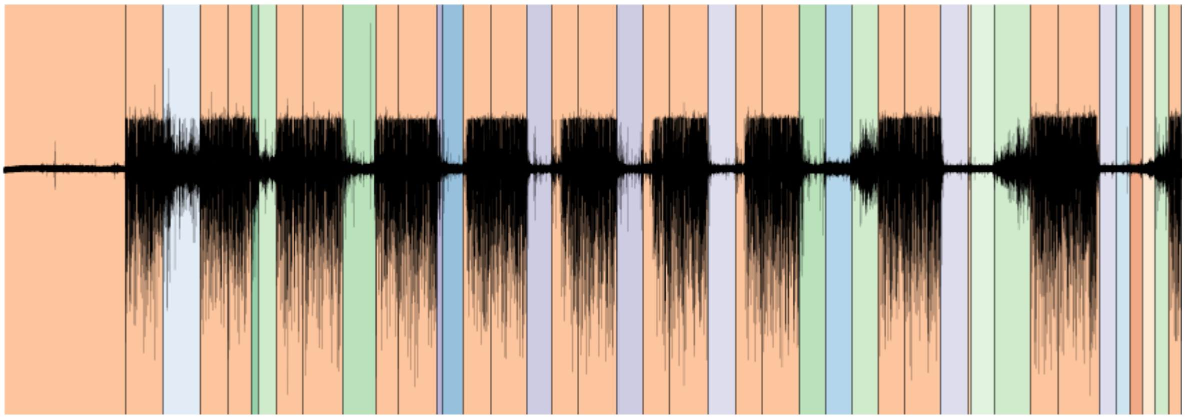


Figure 17: Original time series for the EKSO univariate dataset. Background is set based on the scenario of the segment. The dominant (orange) scenario for example is when all taps are closed.

4.3.2.1 Raw EKSO Dataset and Supervised Classification

We work with a univariate time series comprising pressure measurements sampled at 6.67 kHz from a water pipe. The data are organised into segments, each corresponding to a particular scenario. In the original dataset, many scenarios have very few examples, so we restrict attention to five classes that have sufficient support:

- “All taps closed” (ATC)
- “Tap 1 abrupt”
- “Tap 2 abrupt”
- “Tap 3 abrupt”
- “Tap 4 abrupt”

An illustration of the original dataset is presented in Figure 17. The “all taps closed” class is highly dominant. We split the time series temporally into train/validation/test segments, preserving chronological order to mimic a realistic deployment setting: 50% of the data for training, 25% for validation and 25% for testing.

Within each long segment, we cut the time series into non-overlapping windows of duration 1 second. Since the sampling rate is 6.67 kHz, each window contains 6,670 measurements. We refer to these 1-second windows as our basic *units* for classification. Each window inherits the scenario label of the segment from which it was drawn. Some examples of windows are presented in Figure 18.

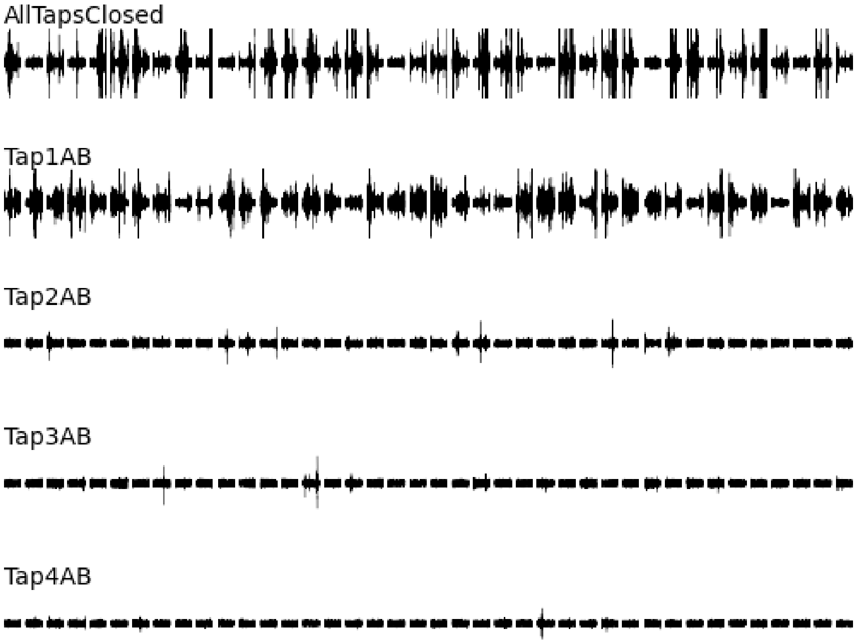


Figure 18: Some units from the validation set for each class in the dataset. Each window (separated by whitespace) is 6670 measurements.

The class supports in the training set (in number of 1-second windows) are:

- All taps closed: 1,518
- Tap 1 abrupt: 92
- Tap 2 abrupt: 147
- Tap 3 abrupt: 124
- Tap 4 abrupt: 175

This distribution highlights the strong imbalance in favour of the “all taps closed” class.

4.3.2.2 Initial attempts with sequential models

A natural first attempt is to treat the raw signal (or down-sampled versions of it) as a sequence and train recurrent neural networks (GRUs, LSTMs) to classify each window. We experimented with:

- Feeding the raw 1-second sequences directly to GRU/LSTM models.
- Down-sampling each 1-second window by averaging over smaller sub-windows (e.g. averaging every 200 samples to produce a sequence of length 34), so that RNNs see a shorter sequence of aggregated values.

Despite these preprocessing steps and hyperparameter tuning, the recurrent models achieved unsatisfactory performance in terms of F1-score. In practice, they struggled to learn discriminative features for the five classes from raw or lightly processed waveforms.

4.3.2.3 Frequency-domain representation and CNN classifier

We obtained substantially better results by moving to a time–frequency representation. Each 1-second window is transformed using a Short-Time Fourier Transform (STFT), yielding a 2D spectrogram. In our setting, the resulting spectrograms have a shape 129 × 209 (frequency bins × time bins). These spectrograms serve as "images" representing the acoustic/pressure footprint of each scenario. Some examples of the STFT spectrograms are presented in Figure 19.

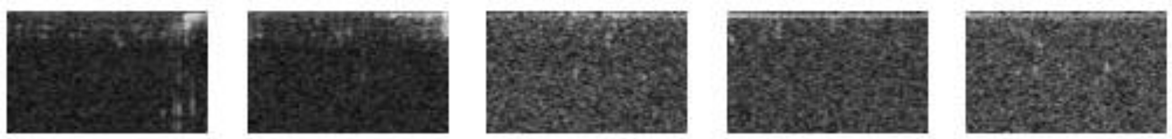


Figure 19: Some units after being processed with STFT. The dimension of the spectrogram is 129x209. Examples from all 5 classes from left to right.

On top of this representation, we train a 2D convolutional neural network (CNN) for window-level classification. The CNN architecture is standard: a stack of convolutional layers with non-linearities and pooling, followed by fully connected layers that output class logits. Training details: Loss: cross-entropy; Class weights: 0.1 for the dominant class ("all taps closed") and 1 for each of the other four, to partially counteract class imbalance; Batch size: 32; Learning rate: 3e-4; Training for 100 epochs, selecting the best model by validation macro F1. The entire CNN architecture used in shown below:

```
CNNClassifier(
  (conv_encoder): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1))
    (1): ReLU()
    (2): InstanceNorm2d(32, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1))
    (5): ReLU()
    (6): InstanceNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=False)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (8): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1))
    (9): ReLU()
    (10): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (11): AdaptiveAvgPool2d(output_size=(1, 1))
    (12): Flatten(start_dim=1, end_dim=-1)
    (13): Linear(in_features=128, out_features=64, bias=True)
    (14): ReLU()
    (15): Dropout(p=0.5, inplace=False)
    (16): Linear(in_features=64, out_features=5, bias=True)

  ))
```

On the test set, averaged over 5 independent runs, we obtain:

- Accuracy: 0.91 ± 0.01
- Macro F1: 0.77 ± 0.04

This confirms that the combination of STFT preprocessing and CNN classification is adequate for mapping raw time series windows to high-level simple events.

4.3.3 Synthetic Temporal Dataset and Markovian Sequencing

4.3.3.1 Motivation

The original EKSO dataset, as collected, does not encode meaningful temporal structure at the level of high-level events: the order in which scenarios occur over time is largely determined by the experimental design and is effectively random. For example, operators may choose arbitrarily when to open or close specific taps, so there is no consistent pattern like “all taps closed is usually followed by tap 1 abrupt” that one could exploit for forecasting.

However, the forecasting methods developed in EVENFLOW, and in particular our mutual-information Markov model (MiMM) framework, require sequential structure in terms of high-level states. We therefore build a synthetic temporal dataset where the high-level events (ATC, Tap 1 abrupt, etc.) evolve according to a Markov chain that we design explicitly. The Markov chain used is illustrated in Figure 20.

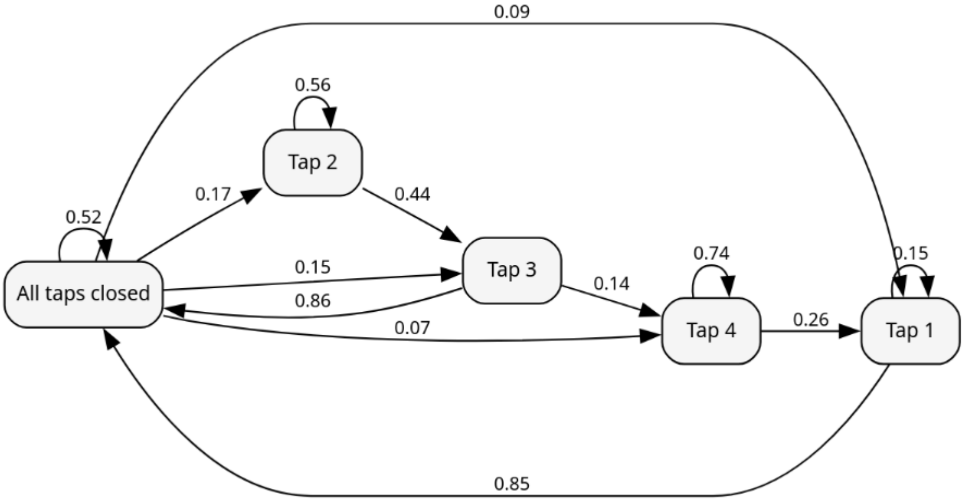


Figure 20: The Markov Chain used to create the synthetic temporal dataset. The chain starts from the left-most All taps closed state and moves between the classes as time goes by.

4.3.3.2 Constructing the Markov chain

We define a Markov chain over the five states:

- ATC (All taps closed)
- T1 (Tap 1 abrupt)
- T2 (Tap 2 abrupt)
- T3 (Tap 3 abrupt)
- T4 (Tap 4 abrupt)

Initially, we set the transition probabilities so that the stationary distribution of the chain roughly matches the empirical class frequencies in the original data. However, the extreme dominance of ATC made the resulting chain too imbalanced for learning. We therefore slightly rebalance the transition matrix to keep ATC as the most frequent state, but not as overwhelming.

The resulting stationary distribution is:

- ATC: 0.55
- T1: 0.10
- T2: 0.08
- T3: 0.06
- T4: 0.15

This means that, when we sample a long trajectory from the chain, roughly 55% of the states will be ATC, and the rest will be distributed among the T1–T4 states as above.

4.3.3.3 Generating temporal sequences of windows

We sample symbolic sequences of length 15 from the Markov chain. A typical sequence might look like:

ATC, T3, ATC, ATC, ATC, T2, T3, ATC, T3, ATC, T1, ATC, T4, T1, ATC

For each symbolic sequence, we generate a corresponding multistep time series by sampling 1-second windows from the original EKSO data: For each symbol in the sequence (e.g. ATC, T3, T1), we pick a 1-second window from the original dataset that has that label. We concatenate these windows in order to form a sequence of 15 seconds; each second annotated with its high-level event. We follow the same temporal splitting strategy as before to construct train, validation and test sets: **Train: 61 sequences**; **Validation: 24 sequences; Test: 23 sequences.**

Each sequence is of length 15, so the training set contains 915 windows in total. Importantly, we assume that only 5% of the training windows are labelled; the remaining 95% are treated as unlabelled and are used for unsupervised representation learning.

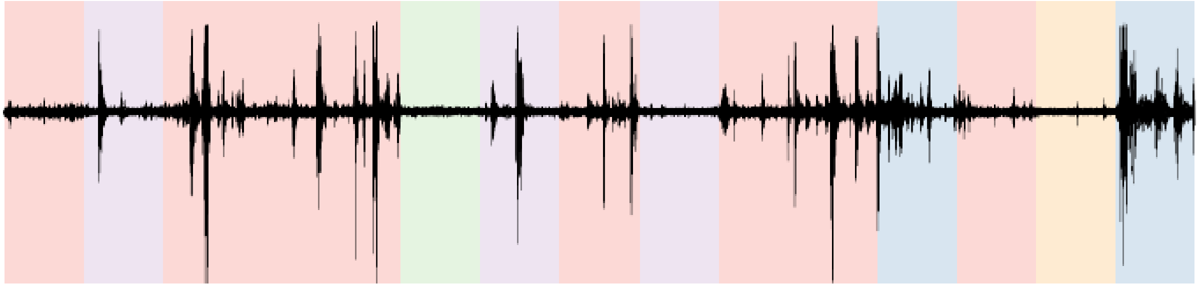


Figure 21: A sample generated sequence. Obtained by: (i) Sampling a symbolic sequence from the Markov Chain; (ii) Choosing windows from the actual data for each element. Each step in the sequence (1s in duration) is color coded based on the label. The sequence length here, as in our experiments, is 15.

4.3.4 Semi-Supervised Learning with Mutual-Information Markov Models (MiMM)

4.3.4.1 Models and training setting

We compare two models, M1 and M2, which share the same CNN architecture and differ only in how they use the available labelled and unlabelled data.

- **M1 (purely supervised baseline):** the CNN is trained only on the small labelled subset of the training windows (5% of 915 = 46 windows). After training, we evaluate M1 directly as a classifier on the test windows.
- **M2 (semi-supervised MiMM model):** we first pretrain the CNN on the same 46 labelled windows as in M1. Then, instead of stopping there, we continue training on the full training set (all 915 windows) using an unsupervised objective derived from mutual-information maximisation between consecutive latent states. The idea is to encourage the CNN to produce discrete latent representations that preserve as much information as possible about the next latent state in the sequence.

In more detail, the MiMM step uses the following intuition: Each 1-second window is mapped by the CNN to logits over the five classes (ATC, T1, T2, T3, T4), which we can interpret as a distribution over latent states. For a pair of consecutive windows (at times t and t+1) in a sequence, we want the latent state at time t to be maximally informative about the latent state at time t+1. We therefore train the network to maximise an estimate of the mutual information I(Z_t; Z_{t+1}), where Z_t is the discrete latent state at time t. We refer to D4.2 for further details on the mutual information estimation.

This mutual-information objective exploits the temporal structure induced by the Markov chain: windows that follow each other are likely to correspond to consistent or predictable changes in the underlying state. By aligning the CNN's latent representations with these temporal regularities, M2 can "pull" the decision boundaries into better positions, even though most of the training windows are unlabelled.

We evaluate the performance of the CNNs on mapping the data to their correct label on the test set (Accuracy). Further we report the induced Markov Chain when passing the whole training set from each model and estimating the probability of each transition, e.g. ATC to T1 for consecutive windows.

4.3.4.2 Evaluation as a classifier

We evaluate both M1 and M2 as simple-event classifiers on the held-out test windows. The results are:

- M1 test accuracy: 0.75
- M2 test accuracy: 0.91

Thus, by leveraging the mutual-information-based unsupervised objective on the unlabelled portion of the data, M2 significantly improves over the purely supervised baseline, effectively closing the gap to the fully supervised CNN trained on the original EKSO dataset. The adjusted mutual information (AMI) is 0.6 and 0.82 respectively. AMI measures the information gained for inferring the label by knowing the prediction of the model compared to a random model.

4.3.4.3 Recovering the underlying Markov chain

Beyond window-level classification, we are interested in how well the learned models capture the underlying temporal dynamics. To this end, we first run the trained CNN (M1 or M2) on all the training sequences. For each pair of consecutive windows, we record the predicted label at time t and at time t+1. Finally, we estimate the empirical transition matrix by counting transitions between predicted labels and normalising. We then compare this estimated transition matrix to the true Markov chain used to generate the symbolic sequences. The comparison is made via the mean absolute error (MAE) over all entries of the transition matrix. The results are as follows:

- M1: MAE per entry ≈ 0.13
- M2: MAE per entry ≈ 0.07

Therefore, M2 not only classifies windows more accurately but also recovers a Markov chain that is substantially closer to the true generative process. In other words, the mutual-information training step allows us to learn a high-quality latent Markov model directly from raw time series data and a very small number of labels. For completeness, we report the different transition matrices.

```
[ 0.52, 0.09, 0.17, 0.15, 0.07 ] [0.43, 0.20, 0.05, 0.22, 0.10] [0.62, 0.06, 0.12, 0.12, 0.08]
[ 0.85, 0.15, 0.00, 0.00, 0.00 ] [0.48, 0.22, 0.07, 0.16, 0.06] [0.84, 0.16, 0.00, 0.00, 0.00]
[ 0.00, 0.00, 0.56, 0.44, 0.00 ] [0.10, 0.10, 0.23, 0.30, 0.27] [0.13, 0.03, 0.58, 0.21, 0.05]
[ 0.86, 0.00, 0.00, 0.00, 0.14 ] [0.36, 0.14, 0.03, 0.26, 0.21] [0.88, 0.00, 0.00, 0.02, 0.10]
[ 0.00, 0.26, 0.00, 0.00, 0.74 ] [0.20, 0.20, 0.06, 0.08, 0.46] [0.05, 0.22, 0.01, 0.00, 0.72]
```

The leftmost matrix is the original transition matrix, the next one is [M1] and the last is [M2].

4.3.5 Neurosymbolic Forecasting

The final step is to use the learned M2 model in a neurosymbolic forecasting pipeline. This involves the following components:

Neural perception layer: Raw EKSO time series windows (1-second segments) are fed to M2, which maps each window to a probability distribution over the five high-level states (ATC, T1, T2, T3, T4). For forecasting purposes, we can use the most likely state or the full distribution.

Latent Markov model: The transition probabilities between states are estimated from the predictions of M2, as described above. This yields a Markov chain that approximates the true dynamics of the system in the space of high-level events.

Symbolic probabilistic reasoning: The learned Markov chain is then passed to a probabilistic model checker (PRISM). There, we express forecasting queries in temporal logic.

An overview of the NeSy forecasting system is presented in Figure 22.

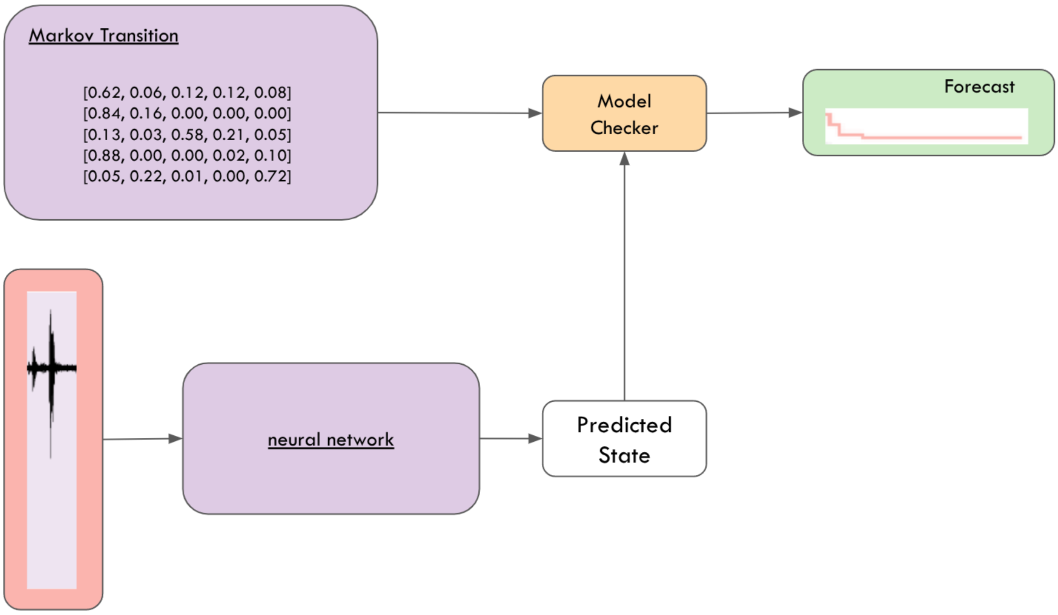


Figure 22: The NeSy forecasting system. Purple components are learnt. Orange (the model checker) is used for reasoning. The input time series is passed through the learnt [M2] model. The transition function induced by [M2] along with the predicted states are passed to a symbolic model checker. The model checker is used to solve for arbitrary queries.

For each sequence in the test dataset, we pick a random timestep between 2 and 6 in the sequence. We extract the window and then apply the neural network to predict the state. Using the induced transition matrix from above (learnt by [M2]) we then call a model checker to predict the probability of certain queries. For this experiment we use simple queries. We randomly select for each sequence a query like:

What is the probability of C for each of the next N timesteps?

where C is a class (randomly selected, e.g. T1 for each sequence) and N is set for 20. We report the forecasting curves in Figure 23.

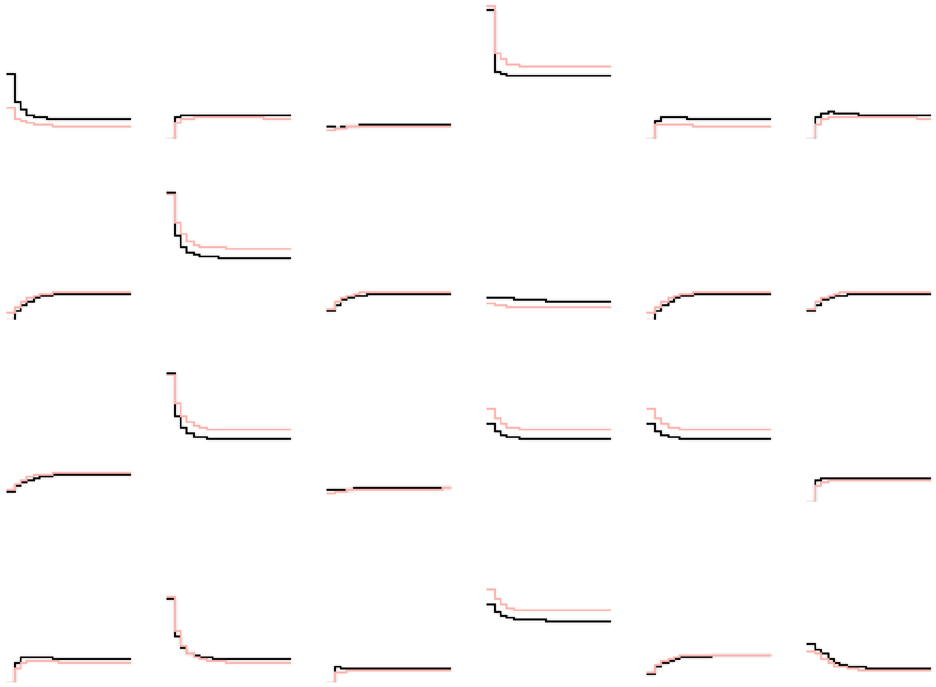


Figure 23: NeSy forecasting results. Black lines show the actual correct forecast if we correctly mapped the input timeseries to the correct class and used the correct transition function for prediction. In the red curve the mapping between timeseries and class are made with the learnt neural network and instead of the true transition function we use the induced transition function.

In Figure 22 the y-axis ranges from 0 to 1 and we have a 20s forecast horizon. As can be seen in the figure, the results are very accurate with the predicted forecast closely matching the actual correct curve.

As a first step to forecasting, this section studies the task of *recognition*, i.e. that a deadlock has occurred within a given trajectory. Specifically, we show that the constructed temporal neurosymbolic system is able to outperform a pure-neural architecture with more parameters in an out-of-distribution (OOD) setting.

The NeSy system consists of a convolutional neural network (CNN) and a deterministic finite automaton (DFA). The CNN operates on the robot's POV camera data and predicts the position (coordinates) of the other robot (regression problem). The CNN is PyTorch's implementation of efficientnet_b0 (4.0M parameters) pretrained on ImageNet. The coordinate prediction is combined with the robot's own position to compute the distance between the robots. This distance is z-normalized and bucketized into 40 bins. It is worth noting that we attempted to directly predict these 40 symbols from the image (a multiclass classification task). However, this approach was significantly less performant than regress → compute → normalize → bucketize. These bins are the symbols that are used as input to the symbolic DFA which specifies the deadlock pattern. The NeSy system thus performs sequence classification, that is, whether the sequence of images in the input contains a deadlock.

4.3.5.1 Experimental setup

100 videos of the two robots executing different trajectories within a factory lab. There exist 5 plans, consisting of different order of visiting workstations. The resulting data distribution is as follows:

#sequences	positive	negative	total
train	104	328	432
test	28	101	129

5 Use Case Evaluation

5.1 Industry 4.0

The evaluation of the Industry 4.0 use case centred on two tightly connected components: the forecasting models, which predict potential deadlocks arising from multi-robot interactions, and the navigation controllers, which determine whether robots can avoid or resolve conflicts once they emerge. Although these elements operate independently in the architecture, they were assessed within the same simulation environment to ensure reproducibility and comparability.

The Controller evaluation measured how well different navigation strategies perform in scenarios specifically designed to induce conflict. This included comparing baseline Nav2 controllers against newly developed liveness-enhanced controllers that enforce continuous progress and prevent stalling behaviour. Such evaluations provide a clear quantitative understanding of how predictive modelling and proactive control support robust multi-robot navigation in dynamic factory environments.

5.1.1 Evaluation Scope and Methodology

The evaluation of the Industry 4.0 use case focuses on the end-to-end interaction between deadlock forecasting and liveness-based control, with the high-level plan as the central unit of analysis.

Robots operate in a simulated factory environment executing predefined high-level plans, each specifying an ordered sequence of workstation visits. The forecasting models are trained on executions of a subset of these plans and evaluated on held-out plans, reflecting realistic deployment conditions where robots repeatedly execute known task structures with varying timing and interactions.

The controller evaluation therefore answers the following question: *Given that a deadlock forecaster predicts an upcoming deadlock along a high-level plan, can liveness-based control prevent the deadlock and improve execution efficiency?*

5.1.2 Experimental Setup

- Simulation environment: Isaac Sim factory/warehouse layout
- Robots: Two differential-drive AMRs (carter1, carter2)
- High-level plans: Five predefined task plans; evaluation performed on held-out plans
- Metrics logged:
    - execution time,
    - distance travelled,
    - deadlock count,
    - recovery actions,
    - task completion status

Each experiment corresponds to a full execution of a high-level plan by both robots, including all induced interactions.

5.1.3 Baseline: Controllers Without Liveness

When executing high-level plans using standard Nav2 controllers without liveness constraints, the following behaviours were observed:

- frequent deadlocks during plan execution,
- repeated triggering of recovery behaviours,
- long execution times despite successful task completion,
- oscillatory velocity profiles caused by symmetric yielding.

These effects were especially pronounced in plan segments that induce close robot interactions, such as shared workstations or overlapping travel paths.

Table 6: Baseline DWB controller without liveness performance.

Experiment ID	Robot	Total Time (s)	Total Distance (m)	Dead-locks	Recov-eries	Goals Completed	Status
1	carter1	694.1	83.798	6956	36	6	Completed
	carter2	661.94	77.084	8937	4	6	Completed
2	carter1	1059.54	93.074	18838	21	6	Completed
	carter2	971.65	76.761	18052	20	6	Completed
3	carter1	877.08	83.919	14300	6	6	Completed
	carter2	846	50.709	13475	124	6	Completed
4	carter1	463.22	73.415	2973	0	6	Completed
	carter2	420.49	66.579	2774	8	6	Completed

5.1.4 Liveness-Enhanced Execution

When the same high-level plans were executed with liveness-based control enabled and activated by deadlock forecasts:

- deadlocks were reduced by orders of magnitude,
- recovery behaviours were largely eliminated,
- execution time decreased significantly,
- robots maintained smooth, continuous motion throughout plan execution.

Crucially, no changes were made to the high-level plans themselves. The improvements stem entirely from proactive, forecast-triggered velocity modulation at the controller level.

Table 7: Liveness Enhanced DWB controller performance.

Experiment ID	Robot	Total Time (s)	Total Distance (m)	Dead-locks	Reco-veries	Goals Completed	Status
1	carter1	489.5	78.43	0	0	6	Completed
	carter2	457.22	70.843	7	7	6	Completed
2	carter1	371.06	70.314	0	0	6	Completed
	carter2	342.95	62.406	0	0	6	Completed
3	carter1	422.32	73.155	0	0	6	Completed
	carter2	402.02	69.423	7	7	6	Completed
4	carter1	340.53	68.912	0	0	6	Completed
	carter2	322.09	61.449	3	0	6	Completed

5.1.5 Quantitative Comparison

The evaluation of high-level plan execution demonstrates that liveness-based control significantly improves multi-robot performance across all measured metrics. Table 8 summarizes the total and average performance for four experimental runs with and without liveness integration.

Table 8: Quantitative Comparison of Reactive and Proactive Deadlock Avoidance.

Metric	Without Liveness	With Liveness	Relative Improvement
Total Execution Time (s)	6252.92	3175.56	49% reduction
Average Execution Time per Robot (s)	781.62	396.95	49% reduction
Total Distance Travelled (m)	655.30	496.00	24% reduction
Average Distance per Robot (m)	81.91	61.99	24% reduction
Total Deadlocks	88,305	17	99.98% reduction
Average Deadlocks per Robot	11,038	2.13	99.98% reduction
Total Recovery Actions	217	21	90% reduction
Average Recovery Actions per Robot	27.13	2.63	90% reduction

Across the four experiments:

- Execution Time: Liveness-based control reduced completion time by approximately half, demonstrating faster, more efficient plan execution.
- Distance Travelled: Robots followed smoother and more direct paths, resulting in ~24% shorter trajectories on average.
- Deadlocks: The number of deadlocks dropped from tens of thousands to near zero, reflecting the success of forecast-triggered liveness interventions.
- Recovery Actions: Recovery behaviours were reduced by 90%, indicating that robots could maintain continuous motion without stalling.

These results highlight the effectiveness of the forecast-driven DWB-Liveness controller, which ensures proactive conflict resolution, smooth trajectories, and high task completion efficiency without modifying the original high-level plans. The improvements confirm that integrating predictive deadlock models with liveness constraints is a robust approach for safe and efficient multi-robot navigation in complex intralogistics environments.

5.1.6 Interpretation

The evaluation demonstrates that deadlock forecasting alone is insufficient unless paired with an appropriate control mechanism. Conversely, liveness-based control is most effective when guided by anticipatory forecasts tied to high-level plans.

Together, the forecasting models and liveness-based controllers form a coherent, decentralized strategy for deadlock avoidance:

- forecasting provides *when* intervention is needed,
- liveness control determines *how* to intervene,
- high-level plans provide the structural context that enables generalization for a factory context.

This alignment confirms the suitability of the EVENFLOW approach for Industry 4.0 intralogistics environments, where robots repeatedly execute structured plans under dynamic interactions.

5.2 Personalized Medicine

The evaluation of the Personalized Medicine use case focused specifically on Kidney Renal Clear Cell Carcinoma (KIRC), reflecting the intensive interdisciplinary effort invested in this cancer type. KIRC was selected because of its clinical significance, well-characterized transcriptomic profiles, and structured stage progression, which together provide a rich substrate for testing hybrid early-warning systems that integrate machine learning, neural sequence modelling, and symbolic reasoning. This cancer-specific focus allowed the team to explore the predictive capacity of temporal gene expression patterns while maintaining a close connection to biological and clinical knowledge.

The evaluation leveraged both real patient data from The Cancer Genome Atlas (TCGA) and synthetic longitudinal trajectories generated using a Variational Autoencoder trained on the KIRC dataset. The TCGA cohort comprised 530 patients with bulk RNA-sequencing profiles and clinically annotated stages, which were binarized into early and late-stage categories to reflect meaningful clinical progression thresholds. Following preprocessing to remove genes with low expression or variance and patients with incomplete annotations, the dataset retained sufficient biological signal for downstream classification. Recognizing the inherent class imbalance favouring early-stage patients, macro-averaged F1-score was adopted as the primary evaluation metric, providing a more informative measure of discriminative performance than accuracy alone. The synthetic trajectories enabled controlled experiments by simulating disease progression over 50 discrete time points. Positive trajectories represented early-to-late stage transitions, while negative trajectories were generated through a combination of latent-space interpolation and Gaussian noise, both in the original gene expression space and in the VAE latent space. Among these, the Real-Space Noise dataset was selected for early-warning experiments because it offered a challenging yet biologically plausible setting in which the classifier had to distinguish progressive from non-progressive trajectories without relying on trivial distributional differences.

Before assessing early prediction capabilities, a baseline classification task was conducted to establish whether transcriptomic features alone could reliably discriminate KIRC stages. Gradient-boosted decision trees (XGBoost) were applied to the full set of 8,516 genes, achieving moderate performance with macro-averaged F1-scores around 0.70–0.75. To enhance interpretability and reduce the dimensionality of the input space, SHAP-based feature selection was performed, resulting in a reduced panel of 45 genes that retained maximal discriminative information. This panel improved the test F1-score to approximately 0.79 and the area under the ROC curve from 0.787 to 0.873. The top-ranked genes, including OASL, HUS1B, and SLC22A1, consistently contributed to stage discrimination, reflecting their biological relevance and supporting the premise that KIRC stage transitions are governed by a focused set of transcriptomic markers. The success of this feature reduction demonstrates the importance of combining domain knowledge with algorithmic interpretability to concentrate predictive power on meaningful molecular signals, while simultaneously simplifying the downstream trajectory modelling task.

With the feature panel established, full-trajectory classification was performed using an LSTM architecture trained on complete synthetic trajectories. The results highlighted substantial differences in classification difficulty depending on the method used to construct negative trajectories. When negative sequences were generated by interpolating between early-stage patients in the latent space, the model achieved near-random performance (F1 ≈ 0.548), indicating that these trajectories closely resembled early-to-late transitions. Conversely, trajectories constructed with latent-space noise achieved near-perfect discrimination (F1 ≈ 0.995), but this configuration introduced distributional shifts unlikely to reflect true biological variability. The Real-Space Noise dataset produced intermediate results (F1 ≈ 0.938), providing a realistic yet challenging benchmark for subsequent early-warning experiments. These findings emphasized that careful construction of synthetic trajectories is critical for testing early prediction models under conditions that simulate clinical uncertainty without artificially simplifying the task.

The primary evaluation focused on early trajectory classification, which simulates the clinical scenario of predicting stage transitions before they are fully observable. Two approaches were compared: a purely neural LSTM baseline and a hybrid neurosymbolic system combining ASAL (Answer Set Automata Learning) with Wayeb, a probabilistic complex event forecasting framework. The LSTM was trained end-to-end on partial trajectory prefixes, capturing temporal dependencies in the gene expression sequences. It demonstrated rapid early convergence, achieving an F1-score of 0.90 within the first 10–12% of the trajectory, suggesting that discriminative patterns are detectable very early in disease progression. In contrast, the ASAL+Wayeb system reached the same threshold after observing approximately 16–20% of the trajectory, reflecting a more gradual accumulation of predictive evidence. Despite this slower initial performance, the hybrid system offers substantial advantages in interpretability and clinical trustworthiness. ASAL induces finite-state automata that encode temporal gene expression dynamics, allowing domain experts to audit, validate, and refine the learned patterns. Wayeb complements this symbolic representation by providing probabilistic forecasts and confidence intervals for the timing of predicted stage transitions, enabling risk-stratified early-warning strategies that purely neural methods cannot supply.

This evaluation underscores the importance of integrating predictive accuracy with interpretability and uncertainty quantification in high-stakes medical contexts. The combination of real and synthetic data, careful feature selection, full-trajectory classification, and early-warning neurosymbolic forecasting collectively demonstrates that KIRC stage transitions are detectable from temporal transcriptomic patterns, and that these predictions can be made actionable while remaining auditable and trustworthy. Moreover, the cancer-specific focus highlights the substantial interdisciplinary effort required to align computational modelling with biological insight and clinical relevance, ensuring that the resulting framework is not only technically robust but also meaningful for patient care and translational research.

5.3 Infrastructure Life Cycle Assessment

The industrial use case provided by EKSO offered a compelling opportunity to apply neurosymbolic forecasting techniques to a real-world system characterized by high-frequency temporal dynamics. The primary objective of this evaluation was to determine whether a hybrid approach, combining supervised neural classification with symbolic temporal modelling, could accurately recognize high-level events in the pressure signal of a water pipe and reliably forecast future scenarios. This task is particularly challenging due to the strong class imbalance inherent in the raw EKSO dataset, as well as the temporal sparsity of meaningful transitions, and thus represents a realistic testbed for neurosymbolic methods in industrial monitoring.

The first stage of the evaluation focused on learning a robust classifier capable of mapping short time windows of the pressure signal to high-level “simple events,” which correspond to the pipe/tap scenario active during the measurement interval. The raw data consist of univariate pressure signals sampled at 6.67 kHz, segmented according to scenario labels such as “All taps closed” or individual taps opened abruptly. Initial experiments with recurrent neural networks, including GRU and LSTM architectures, proved insufficient for capturing discriminative patterns directly from the raw waveform, even after extensive hyperparameter tuning and down-sampling. The limitations of these models highlighted the need for feature representations that better expose the scenario-specific temporal signatures in the signal.

Transitioning to a frequency-domain representation proved decisive. Applying a Short-Time Fourier Transform (STFT) to each 1-second window of the signal produced spectrograms that effectively encoded the temporal and spectral characteristics of each scenario. A convolutional neural network (CNN) trained on these spectrograms achieved substantial performance improvements, with a test-set accuracy of 0.91 and a macro F1-score of 0.77. Importantly, the model successfully handled the extreme class imbalance by weighting the loss function, allowing it to recognize rare abrupt-tap events alongside the dominant “All taps closed” state. These results confirm that the combination of STFT preprocessing and CNN classification is sufficient to extract high-level simple events from raw industrial time series, forming a reliable foundation for subsequent temporal modelling.

However, the original EKSO dataset lacks meaningful temporal structure at the level of high-level events, limiting its utility for sequence-based forecasting. To address this, a synthetic temporal dataset was constructed in which symbolic sequences of events evolve according to a controlled Markovian pattern. This procedure enabled the creation of multi-step sequences that preserve realistic durations for each event while enforcing temporal dependencies suitable for neurosymbolic reasoning. By sampling 1-second windows from the original dataset according to the synthetic symbolic sequences, we generated sequences of 15 seconds in length, reflecting a variety of event transitions. This synthetic dataset allowed the evaluation of semi-supervised sequence modelling techniques under realistic constraints: only 5% of the training windows were labelled, simulating scenarios where limited expert annotation is available in industrial monitoring contexts.

The semi-supervised stage employed the Mutual-Information Markov Model (MiMM) framework to align latent representations with temporal dependencies. By maximizing the mutual information between consecutive latent states, the network learned representations that both capture the semantics of individual windows and encode the dynamics of event transitions. Evaluation of the MiMM-augmented model (M2) against a purely supervised baseline (M1) demonstrates the substantial benefits of this approach. On the test set, M2 achieved a window-level classification accuracy of 0.91, markedly higher than M1’s 0.75, and an adjusted mutual information score of 0.82 versus 0.6. Moreover, M2 recovered the underlying Markovian transition structure with greater fidelity, as evidenced by the lower mean absolute error between the estimated and true transition matrices (0.07 versus 0.13 for M1). These results confirm that the semi-supervised mutual-information objective successfully leverages unlabelled data to improve both classification and sequence modelling performance.

The final evaluation stage integrated the learned CNN and latent Markov model into a full neurosymbolic forecasting pipeline. The neural perception layer produces probability distributions over the five high-level events for each incoming 1-second window. These distributions are used to induce a Markov chain representing the system’s temporal dynamics, which is subsequently passed to a probabilistic model checker (PRISM) to answer temporal queries. Forecasting experiments demonstrated that the system accurately predicts the probability of future scenarios over a 20-second horizon, closely tracking the actual sequences. Importantly, the neurosymbolic approach allows for probabilistic reasoning about future events while remaining interpretable and auditable, a critical requirement for industrial monitoring applications where risk assessment and operational decisions must be justified. The system outperformed pure neural baselines with more parameters in out-of-distribution settings, further underscoring the advantages of combining neural perception with symbolic temporal reasoning.

Overall, the evaluation highlights the effectiveness of the neurosymbolic framework for industrial time-series forecasting. By transforming raw high-frequency signals into high-level symbolic events and modelling their temporal evolution with a semi-supervised latent Markov approach, the system achieves both high predictive accuracy and interpretable probabilistic forecasts. This dual capability (precise recognition of instantaneous events and reliable short-term prediction of system evolution) positions the neurosymbolic methodology as a robust solution for water pipe leakage detection and other analogous industrial monitoring tasks. The results also demonstrate the broader applicability of EVENFLOW techniques, confirming that neurosymbolic forecasting can extract structured temporal knowledge from unstructured sensor streams, even in settings with severe class imbalance and limited labelled data.

6 Conclusions

Across three diverse applications (Industry 4.0 multi-robot navigation, personalized medicine for KIRC stage transition, and EKSO water pipe monitoring) the EVENFLOW framework consistently validates its core hypothesis: robust complex event forecasting achieves practical value when tightly integrated with reasoning-aware, interpretable decision mechanisms.

In the Industry 4.0 use case, training deadlock forecasters on high-level plan executions and feeding their predictions into liveness-enhanced controllers enabled proactive deadlock avoidance, reduced execution times, smoother robot trajectories, and fully decentralized operation. This demonstrates that combining learning, symbolic reasoning, and control is not merely expressive, but operationally effective in real factory environments where task structures are known but execution timing and interactions remain uncertain.

The KIRC use case extends this premise to high-stakes biomedical applications. By uniting discriminative machine learning, neural sequence modelling, and symbolic event-based forecasting (ASAL+Wayeb), EVENFLOW delivers early detection of disease stage transitions from partial longitudinal data, explicit probabilistic forecasts with confidence intervals, and interpretable temporal patterns auditable by clinical experts. Even under conditions of limited patient data and class imbalance, this hybrid neurosymbolic approach achieves robust performance, confirming that the integration of predictive learning and symbolic reasoning is not only theoretically sound but practically deployable for trustworthy early-warning systems in precision oncology.

In the EKSO industrial monitoring scenario, the framework demonstrates similar strengths in a temporal, high-frequency sensor domain. By combining CNN-based perception of pressure signals with semi-supervised latent Markov modelling and symbolic probabilistic reasoning, EVENFLOW accurately classifies high-level pipe/tap scenarios, reconstructs underlying temporal dynamics from sparse labelled data, and produces reliable probabilistic forecasts. The resulting predictions are interpretable, auditable, and operationally actionable, underscoring that learning, symbolic reasoning, and temporal modelling together form a robust and deployable solution for complex industrial event forecasting.

Taken together, these results illustrate that EVENFLOW’s neurosymbolic approach provides a unified paradigm for complex event prediction: it leverages learning to extract patterns from raw data, employs symbolic reasoning to structure and interpret temporal dependencies, and produces actionable forecasts suitable for real-world deployment. Whether in robotics, healthcare, or industrial monitoring, the framework demonstrates that expressivity, interpretability, and operational effectiveness can coexist, enabling practical, high-confidence decision-making in dynamic and uncertain environments.

7 References

[REF-01]	Prol-Castelo G, Cirillo D, Valencia A. 10 years of Variational Autoencoder: Insights from cancer temporal progression studies, a systematic literature review. bioRxiv 2025. https://doi.org/10.1101/2025.05.29.656750.
[REF-02]	Prol-Castelo G, Tejada-Lapuerta A, Urda-García B, Núñez-Carpintero I, Valencia A, Cirillo D. Exploring the boundaries of medulloblastoma subgroups with synthetic data generation. bioRxiv 2024. https://doi.org/10.1101/2024.12.30.630738.
[REF-03]	Search EVENFLOW Project n.d. https://zenodo.org/communities/evenflowproject/ (accessed December 20, 2025).
[REF-04]	Alevizos E, Artikis A, Paliouras G. Wayeb: A tool for Complex Event Forecasting, EasyChair; 2018. https://doi.org/10.29007/2s9t.

Appendix A Wayeb

Wayeb is an online, probabilistic system designed for Complex Event Forecasting (CEF), addressing the challenge of predicting the potential occurrence of a declaratively defined Complex Event (CE) pattern (often formulated as a Symbolic Regular Expression (SRE)) within an event stream before it is actively detected by a Complex Event Recognition (CER) engine. Wayeb converts an SRE into a Deterministic Symbolic Finite Automaton (DSFA), which, when consuming the input stream, is functionally analogous (via isomorphism) to a classical deterministic automaton operating over the minterms of the DSFA predicates.

To model the statistical properties of the stream, Wayeb employs Variable-order Markov Models (VMMs), specifically, Prediction Suffix Trees (PST), which capture long-term dependencies, while avoiding the computational explosion associated with exhaustive enumeration in fixed-order models. The probabilistic model is constructed by learning the PST from the minterms derived from the DSFA, using an approach that either involves creating an embedding of a probabilistic automaton within the DSFA by taking their Cartesian product, or, for superior memory efficiency, by directly estimating waiting-time distributions through recursive traversal of the PST, thereby bypassing the construction of the probabilistic automaton.

These calculated waiting-time distributions, based on the theory of absorbing Markov chains, allow Wayeb to output forecasts, typically in the form of intervals [start, end], representing the predicted number of future events until pattern completion with a user-defined confidence threshold θ. Wayeb has been demonstrated to achieve high throughput and competitive accuracy compared to state-of-the-art solutions, often leveraging its ability to accommodate higher-order models for enhanced performance [REF-04].