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

D6.3 – Architecture Design and Integrated System Specification (PU version)

Dissemination level:	PU - Public, fully open
Type of deliverable:	R - Document, report
Contractual date of delivery:	30 September 2023
Deliverable leader:	INTRA
Status - version, date:	Final – v1.1, 2023-09-29
Keywords:	architecture, requirements, platform, AI, neuro-symbolic, scalability, verification

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

EVENFLOW is conducting world-class research in neuro-symbolic learning models and applications. In this direction, the project is researching neuro-symbolic learning methods, while integrating them into use cases that demonstrate the added-value of the neuro-symbolic AI paradigm. Moreover, the project aims to develop a platform that will meet various types of robustness and scalability requirements for the development, deployment and operation of neuro-symbolic learning applications end-to-end i.e., from data acquisition and pre-processing to the creation of pipelines involving integrated formal learning and deep learning components. The EVENFLOW platform will promote a structured approach to the development and deployment of neuro-symbolic learning applications. Instead of relying on an ad-hoc integration of neuro-symbolic learning components that are different for each use case, the platform will provide generalized infrastructures that ease the structuring, development, integration and deployment of end-to-to-end neuro-symbolic learning and prediction pipelines.

This deliverable details the architectural design of the EVENFLOW platform and its specifications regarding the development, deployment, and operation of integrated neuro-symbolic learning workflows and applications. The deliverable provides an integrated and completed description of the platform architecture, based on the 4+1 views methodologies for describing software architectures i.e., based on multiple views that include a logical, a process, an implementation, a physical deployment and a use cases viewpoint of the integrated platform.

This first platform deliverable provides a comprehensive description of the logical view of the platform's architecture, which focuses on the functionalities of the various components that comprise neuro-symbolic learning applications. It provides the logical structure of a neuro-symbolic learning application, along with concrete logical architectures for the three EVENFLOW use cases. Moreover, the deliverable provides early insights on the information flows between the different components of the EVENFLOW platform (i.e., process view consideration), along with information about the main implementation technologies that will be used to implement the platform. Also, the deliverable introduces a high-level reference model for the EVENFLOW functionalities, which are clustered into explainable AI, hybrid learning and reasoning, as well as verification and scalability functionalities. This high-level reference model aims at facilitating stakeholders' communications regarding architectural issues, with an emphasis on communications about the functionalities of the logical modules of the platform.

This deliverable, D6.3, has a sibling sensitive one, D6.1, but as D6.1 contains no sensitive information, the contents of the two deliverables are the same.

Deliverable leader:	Ioannis Christou (INTRA)
Contributors:	Ioannis Soldatos, Konstantinos Tziotis (INTRA) Nikos Katzouris (NCSR) Nikos Giatrakos (ARC) Davide Cirillo, Guillermo Prol Castelo (BSC) Benjamin Blumhofer (DFKI) Karim Ladjeri (EKSO) Alessandro De Palma ICL)
Reviewers:	Nikos Giatrakos (ARC) Davide Cirillo (BSC)
Approved by:	Athanasios Poulakidas, Dimitrios Liparas (INTRA)

Document History			
Version	**Date**	**Contributor(s)**	**Description**
0.1	2023-09-29	I. Christou, A. Poulakidas	Version derived from D6.1 with same content (see D6.1 for its version history)
1.0	2023-09-29	A. Poulakidas, D. Liparas	QA and final version for submission
1.1	2023-09-29	A. Poulakidas	Added missing contributors

Table of Contents

Table of Figures

List of Tables

Definitions, Acronyms and Abbreviations

Acronym/ Abbreviation	Title
AI	Artificial Intelligence
CEF	Complex Event Forecasting
GDPR	General Data Protection Regulation
LSTM	Long short-term memory
ML	Machine Learning
RNN	Recurrent Neural Network
XAI	Explainable AI

Term	Definition
FAIR data	FAIR data are data which meet principles of findability, accessibility, interoperability, and reusability (FAIR)

1 Introduction

1.1 Project Information

EVENFLOW is developing hybrid learning techniques for complex event forecasting, which combine deep learning with logic-based learning and reasoning into neuro-symbolic forecasting models. The envisioned methods combine (i) neural representation learning techniques, capable of constructing event-based features from streams of perception-level data with (ii) powerful symbolic learning and reasoning tools, that utilize such features to synthesize high-level, interpretable patterns of critical situations to be forecast.

Crucial in the EVENFLOW approach is the online nature of the learning methods, which makes them applicable to evolving data flows and allows to utilize rich domain knowledge that is becoming available progressively. 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 federated training and incremental model construction. The learnt forecasters will be interpretable and scalable, allowing for fully explainable insights, delivered in a timely fashion and enabling proactive decision making.

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

Expected impact:

- New scientific horizons in integrating machine learning and machine reasoning, neural, statistical and symbolic AI
- Breakthroughs in verification, interpretability and scalability of neuro-symbolic learning systems
- Interpretable, verifiable and scalable ML-based proactive analytics and decision-making for humans-in-the-loop and autonomous systems alike
- Robust, resilient solutions in critical sectors of science and industry
- Accurate and timely forecasting in vertical sectors (healthcare, Industry 4.0, critical infrastructure monitoring)
- Novel FAIR datasets for scientific research
- Novel resources and approaches for verifiable, interpretable, scalable and knowledge-aware machine learning

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**

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

Number[1]	Name	Country	Short name
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**

1.2 Document Scope

D6.1 and D6.3 provide the sensitive (SEN) and the public (PU) versions of the same deliverable. However, as no sensitive information was included in D6.1, **the contents of D6.1 and D6.3 are the same**.

This first platform deliverable provides a comprehensive description of the logical view of the platform's architecture, which focuses on the functionalities of the various components that comprise neuro-symbolic learning applications. It provides the logical structure of a neuro-symbolic learning application, along with concrete logical architectures for the three EVENFLOW use cases. Moreover, the deliverable provides early insights on the information flows between the different components of the EVENFLOW platform (i.e., process view consideration), along with information about the main implementation technologies that will be used to implement the platform. Also, the deliverable introduces a high-level reference model for the EVENFLOW functionalities, which are clustered into explainable AI, hybrid learning and reasoning, as well as verification and scalability functionalities. This high-level reference model is destined to facilitate stakeholders' communications regarding architectural issues, with an emphasis on communications about the functionalities of the logical modules of the platform.

1.3 Document Structure

This document is comprised of the following chapters:

Chapter 1 presents an introduction to the project and the document.

Chapter 2 presents a summary of the background and requirements driving the architecture.

Chapter 3 presents the high-level EVENFLOW architecture, together with the major important views of it (both static and dynamic views included)

Chapter 4 presents the connections between architecture, scenarios and use-cases, for each of the three major use-cases of the project.

Chapter 5 is the concluding chapter that outlines the conclusions of this deliverable.

2 Background and Driving Requirements

2.1 Reference Architecture Models for Artificial Intelligence

Reference architectures in AI are invaluable blueprints that guide the development of intricate AI systems, aligning them strategically with business needs while outlining critical components and their interplays. They encompass data processing, AI models, orchestration, and the core infrastructure components. Famous models such as the IBM Cloud Pak for Data, Google Cloud's AI Hub, and Microsoft's Azure AI employ these principles. The data layer generally comprises data sources, storage, and data management methodologies. On the AI modelling side, machine learning algorithms, deep learning networks, or reinforcement learning models can be considered. For orchestration, the components often included are those for model training, validation, deployment, and monitoring. The infrastructure layer provides the backbone, with hardware, cloud services, and networking, offering the needed computational capabilities. In sum, reference architectures, though they possess a degree of generality, are crucial as they provide a roadmap for the efficient design of AI systems, minimize design errors, ensure scalability and interoperability, and encourage the reuse of architectural elements.

2.2 Requirements for Integrated Systems

Requirements for integrated systems usually specify the APIs of the various sub-systems that are to be connected to the entire integrated system, so that other sub-system integrators know what to expect from each sub-system, and how to use it. API specifications can be given in the form of REST endpoints (in loosely connected web-service fashion), or they can be as tight as full specifications of the data structures stored in a centralized data repository that can be an RDBMS or a distributed message bus, or a distributed NoSQL document database, etc.

The requirements gathered for this project in particular, are collected in a single Excel file stored in the project's common share space. See requirements R001-R012 in the exact URL for this requirements Excel file: EVENFLOW integrated system requirements.xlsx.

2.3 Regulatory Requirements

The development, deployment and operation of neuro-symbolic learning systems must comply with applicable European laws and regulations. Specifically, the relevant requirements must be met as follows:

- **GDPR Compliance**: EVENFLOW systems are data-intensive and as such, must comply with the mandatory GDPR (General Data Protection Regulation). Hence, they must adhere to the GDPR principles and provide support for its mandates such as the purpose limitations and the right to be forgotten. Note that several EVENFLOW systems may use neuro-symbolic learning in scenarios with the human in the loop and use cases involving sensitive data. GDPR requirements must be accounted for during the systems' development and operation. However, the fulfilment of these requirements is only indirectly linked with the core research topics of the project, which concern AI development.

- **AI Act**: EVENFLOW undertakes AI research, which must be in-line with the mandates of the European AI regulation proposal i.e., the AI Act [REF-03]. The AI Act takes a risk based approach to the classification and compliance of AI systems. Depending on the risk classification of an AI system (i.e., of a neuro symbolic learning system in the case of EVENFLOW) different requirements must be met. For instance, in the case of minimal-risk systems (e.g., systems that display information without any essential safety of abuse concerns) deployers do not have any essential restrictions. Compliance to AI code of conduct for them may be recommended, yet it is not mandatory. On the other hand, high risk systems must comply with several requirements that are spelled out in the AI Act. Some of these requirements are depicted in Figure 1 and include robustness, accuracy, human oversight, explainability/transparency, logging and traceability, the deployment of strong cybersecurity measures and more. The EVENFLOW research is directly related to several of these requirements. Specifically, the project's systems can be used to boost AI Act compliance for high-risk AI deployments. This is for example the case with the following regulatory requirements:
    - **Transparency:** EVENFLOW Explainable AI systems can boost compliance to transparency requirements.
    - **Robustness and Accuracy:** EVENFOW Hybrid Learning and Reasoning systems can boost the robustness and accuracy of AI systems especially in selected scenarios (e.g., hybrid use cases where adequate quality data are lacking).
    - **Logging and Traceability:** The EVENFLOW platform shall provide audit trail management and generation functionalities to support data logging and traceability.

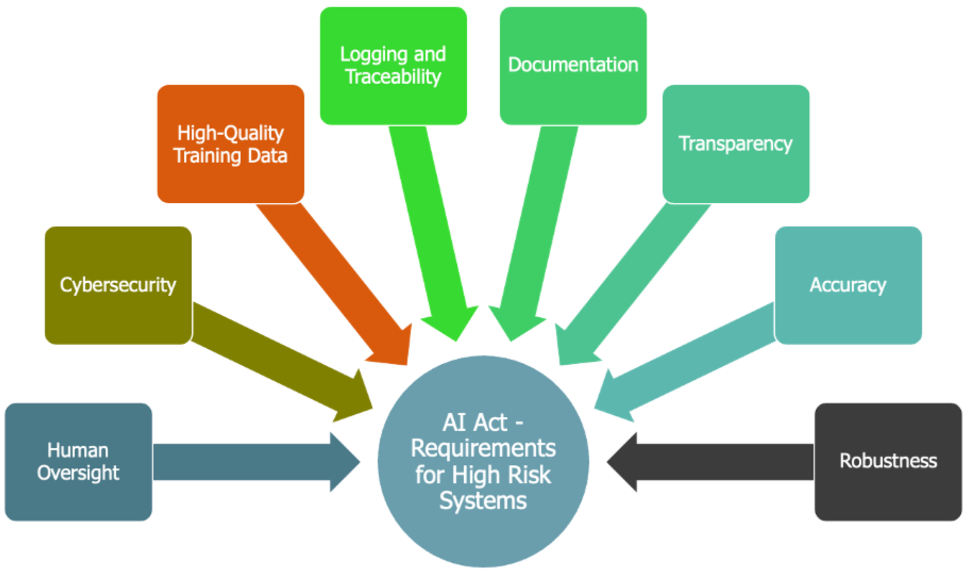

Figure 1: AI Act for High-Risk Systems.

In practice, the EVENFLOW use cases will fulfil applicable regulatory requirements. The latter will be fulfilled based on core functionalities of the EVENFLOW systems (e.g., Explainable AI functionalities) and based on use case specific functionalities that will be implemented in the scope of the use cases implementation.

3 EVENFLOW Architecture

3.1 High level Reference Model

Figure 2 illustrates a high-level reference model for Neuro-Symbolic AI. It is provided as a high-level reference model that classifies the main functionalities that are offered by the project's platform. Primarily, this high-level reference model aims at facilitating communications between various stakeholders of the EVENFLOW compliant neuro-symbolic learning system. It presents the main functions offered by the project's platform in terms of neuro-symbolic learning, along with their interactions. Most importantly, it classifies the EVENFLOW functionalities in their main categories. This enables stakeholders to position EVENFLOW functionalities within a proper cluster of neuro-symbolic AI related functions. Specifically, EVENFLOW offers three different types of functionalities, including:

- **Explainable AI**: This category comprises the project's XAI functionalities, including forecasting explainability and explainability of neural networks. Explainability may be also supported through Glass Box models that provide inherent transparency regarding how they produce their AI outcomes. The XAI techniques may be used by methods and systems of the other two categories of the reference architecture.
- **Hybrid Learning and Reasoning**: This includes the project's hybrid learning and reasoning functionalities, including neural learning, on-line symbolic learning, complex event forecasting and reasoning assisted programming functionalities. These functionalities reflect the project's neuro-symbolic AI methods, which may interact with the XAI as well as with the verification and scalability functionalities as depicted in the figure.
- **Verification and Scalability**: EVENFLOW provides a set of verification and scalability functionalities, which are mostly developed in WP5 of the project. They are clustered in this category of the reference architecture model and include exact verification methods, abstract-based verification, as well as techniques for attributing various types of scalability on neural learning.

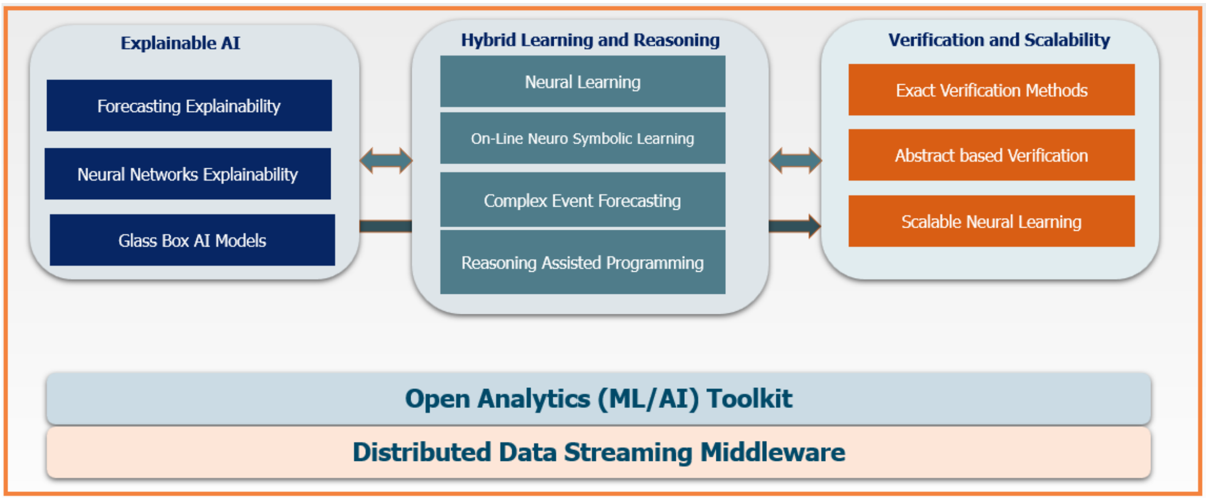
Explainable AI
Forecasting Explainability
Neural Networks Explainability
Glass Box AI Models
Hybrid Learning and Reasoning
Neural Learning
On-Line Neuro Symbolic Learning
Complex Event Forecasting
Reasoning Assisted Programming
Verification and Scalability
Exact Verification Methods
Abstract based Verification
Scalable Neural Learning
Open Analytics (ML/AI) Toolkit
Distributed Data Streaming Middleware

Figure 2: EVENFLOW's High-level Reference Architecture for Neuro-symbolic AI.

The execution of the above-listed functionalities is powered by an AI platform and a distributed streaming middleware platform. The former enables the execution of AI techniques (including XAI functions), while the latter facilitates the collection, ingestion and processing of data to the various functions.

The high-level functionalities of the reference architecture models are further detailed in the logical view of the EVENFLOW platform in the following sections. In particular, the logical view of the architecture elaborates on the functional modules of the architecture that enable the collection and management of the data for the execution of the EVENFLOW neuro-symbolic learning functionalities.

3.2 Logical View

The overall architecture is shown in the following logical view below in Figure 3.

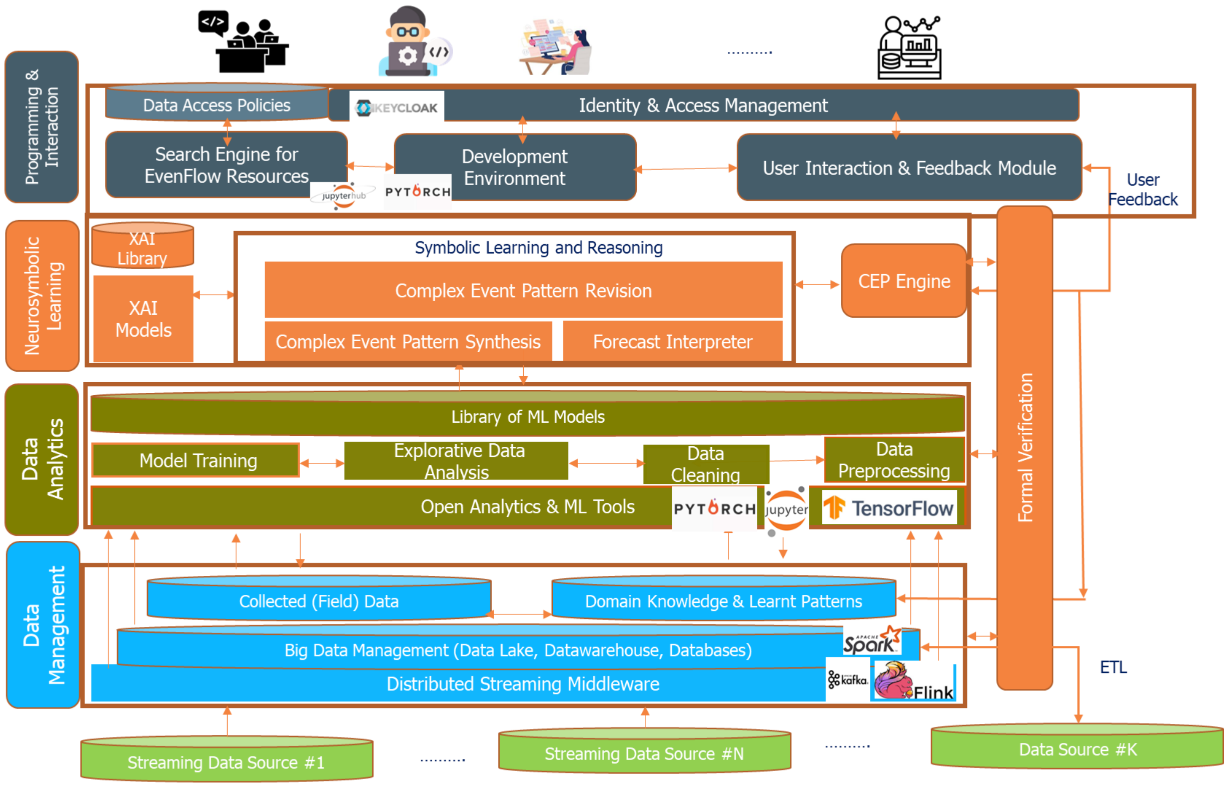

Figure 3: EVENFLOW Architecture Logical View.

In the following, we discuss the major components of the architecture, as depicted in Figure 3.

3.2.1 Component 1: Data Management

This component consists essentially of a high-performance set of distributed streaming middleware, together with accompanying high-performing data bases.

The streaming middleware is required to include the following components:

- Apache Kafka as distributed message bus

- Appropriate ETL scripts for interacting with various data sources

The streaming middleware may also include the following components:

- Apache Spark distributed in-memory processing engine
- Apache Flink distributed in-memory processing engine

The databases layer is required to include the following component:

- InfluxDB server v.2.0 time-series database

3.2.2 Component 2: Data Analytics

The data analytics component comprises a set of Machine Learning tools (some with Explainability capabilities), and a set of dashboards (analytics, business intelligence features). It is expected that some of the tools will be Open-Source algorithm implementations (e.g. the wittgenstein python package that implements the Ripper algorithm) while other tools will be custom-written Neural Network implementations using TensorFlow and/or the Pytorch framework for Deep Learning extended with scalability, among other, characteristics. Even though it is not a strict requirement, it is expected that a good number of these implementations will be in Jupyter Notebook scripts to ease adoption of project outcomes as individual components besides, as an integrated platform.

3.2.3 Component 3: Neuro-symbolic Learning

The main toolkit in this component will be the "Symbolic Learning & Reasoning" component, that is also expected to be the most time-consuming, and therefore performance-demanding component in the entire project. This toolkit will contain the following major sub-components:

- Neural Learning, including convolutional, LSTM, RNN, Deep and other types of layers, utilizing back-propagation for automatic differentiation of the loss function, and first-order gradient descent algorithms such as Stochastic Gradient Descent or Adam/Groundhog etc. for loss function optimization. Moreover, this also includes (a) online neural learning over a number of workers/learners following a distributed/federated learning paradigm and (b) approximate, synopses-based neural learning for scalability purposes.
- Online Neuro-symbolic learning component that matches neural network learning capabilities with classical AI symbolic reasoning approaches; in this process, a major computational component is the "grounding" of world-variables, or in other words, figuring out all combinations of variable value assignments that are feasible.
- Complex Event Forecasting - CEF component that uses the previous two components for complex event detection, and later prognosis and forecasting).
    - Complex Event Pattern Synthesis sub-component of CEF
- Reasoning Assisted Programming component that allows complex logic programming involving rules to be enhanced by deep learning high-performing components.
- eXplainable AI components that can reason about their own decision-making, albeit in an underlying statistical framework.

3.2.4 Component 4: Interaction Layer

The interaction layer is concerned with the APIs and GUIs that will be made available to the end-user of the project results (in our cases, AI researchers and domain experts in general, seeking better tools to complete highly complex scientific/engineering tasks.) The main sub-components of this layer as depicted in Figure 3 are the following:

- Identity & Access Management that will be based on the Keycloak IDM Open-Source solution.
- Development environment that will be based on PyTorch, TensorFlow and JupyterHub (without excluding other IDEs for different developers).
- User Interaction and Feedback module that will provide some degree of Human-induced Reinforcement Learning capabilities to the system.
- Search Engine for EVENFLOW resources, regulated by some rules stored on any repository for data access policies.

3.2.5 Component 5: Formal Verification Techniques

Formal verification techniques leading to feasible NP-hard optimization problems (verifiable, on average, in less than exponential time) are the last cross-cutting layer and concern within the project. Its output will be stored in the data management layer's domain knowledge and learnt patterns data-store.

3.3 Process View

In Figure 4, we show the Data Flow Diagram corresponding to the view depicting the neuro-symbolic process of complex event pattern recognition. The whole process consists of 3 major processes (level-1 processes), labelled "Complex Event Pattern Authoring", "Parameter Learning", and finally, "Forecasting". The first process that kick-starts the entire logic is that of authoring complex event patterns (a manual process), which provides as output patterns that need to have their parameters optimized, which is done in process 2, "Parameter Learning" in a classical neural computation. The probabilistic model that results is then fed to the standard forecasting process for real-time inference. Results are stored in the "Forecasting Results" data store shown at the bottom of the figure.

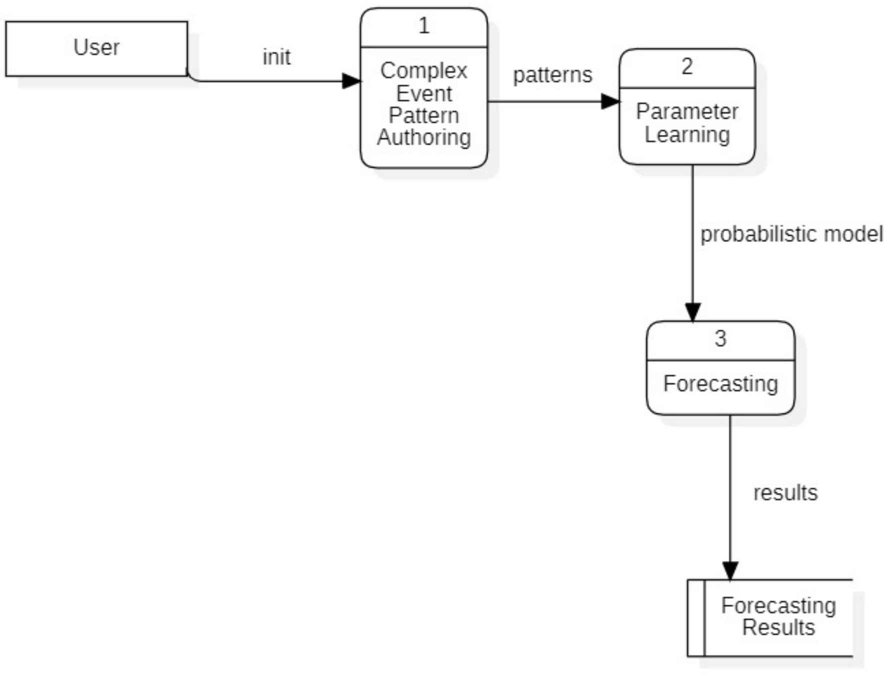

Figure 4: Data Flow Diagram showing the Complex Event Pattern Recognition General Process.

Figure 5 shows the Data Flow Diagram for the verification module. The main processes are labelled "Verification" and "Robust Training": these inter-operating models receive as input from the user the property to be verified, and either a pre-trained model or the model architecture to train for verifiability according to the input specification. The outputs of the module include a decision on the property (satisfied, not satisfied, or undecided), a counterexample if the property is not satisfied and, if required, a trained model. Further details are provided in Section 4.5.

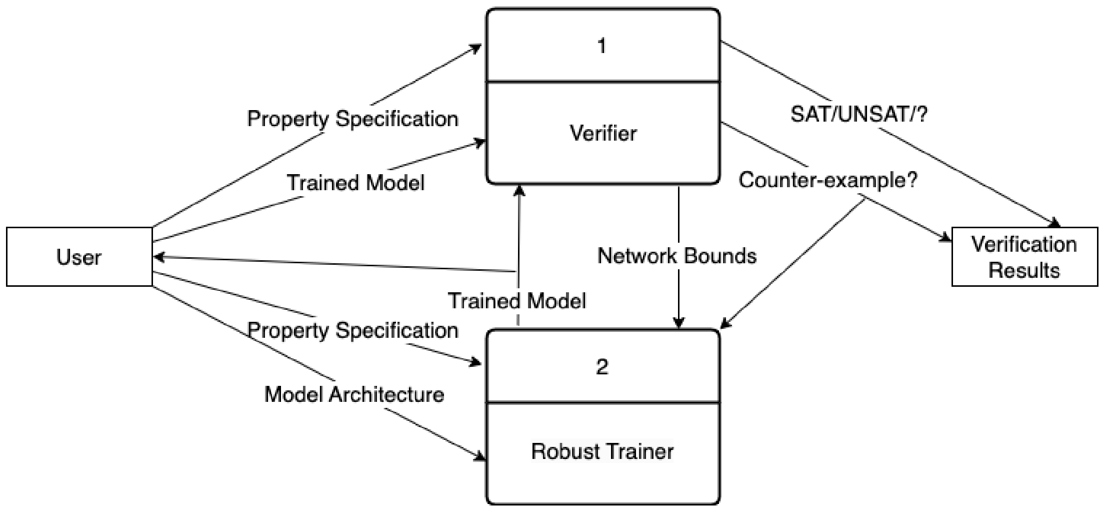

Figure 5: Data Flow Diagram showing the Verification Process.

3.4 Implementation View

We propose the following implementation technologies for each component:

- **Component 1 Data Management**: As already mentioned, the standard data management platforms to be used are:
  - The Apache Kafka distributed message bus (together with the Apache ZooKeeper management tool)
  - An instance of the InfluxDB v.2 time-series database
  - If RDBMS technology is needed, we will also install and use a PostgresQL relational database system.
  - If computational requirements must be met on the EVENFLOW cluster, at least one (or both) of the following middleware will also be installed and supported:
    - Apache Spark
    - Apache Flink
- **Component 2 Data Analytics:** As already mentioned, new algorithms will be developed, tested and tried using the Google TensorFlow, PyTorch, and Jupyter notebook libraries and platforms. In addition, analytics can be "brought on" to the project from IDE tools such as Orange3 and/or KNIME.
- **Component 3 Neuro-symbolic Learning:** In this major component of the project, standard Python libraries including numpy, scipy, sklearn, tslearn, pandas, pandas.ai, PyTorch, TensorFlow etc. will be used for algorithm development and implementation. Further Python packages including GurobiPy or PySCIP can/may be used for interfacing to high-performance optimization codes. Most other toolkits and libraries for combinatorial optimization such as OR-Tools (Google) etc. also feature user-friendly Python API bindings.
- **Component 4 Presentation and Interaction Layer:** Traditionally, JavaScript or its variants is used for front-end development. Implementing various REST APIs and endpoints can be done using the Django or Flask frameworks for Python, and Spring Boot for Java developers.
- **Component 5 Formal Verification Techniques:** Interfacing with state-of-the-art optimizers is best done through the API libraries provided by each vendor/provider, for each language they support. For example, GUROBI (being the top commercial solver today) has bindings to every major programming language, including Basic.NET, C#.NET, C/C++, Java, FORTRAN, Python, Julia etc. It also interfaces directly with optimization/computer algebra systems (CAS) such as GAMS, AIML, etc. SCIP on the other hand, being free and Open-Source, provides interfaces for C/C++ (the language it's written in) and Python. It also provides its own modelling language, called ZIMPL for specifying mathematical programs at a level comparable to GAMS. It is expected that the main implementation language will be Python or C/C++. More accurately, state-of-the-art (SOTA) neural network verifiers are typically written in Python, exploiting efficient backends such as PyTorch for hardware acceleration. Furthermore, they typically avoid dependencies on black-box optimizers.

3.5 Deployment View

The current state of the deployment view, as far as the distributed data management component is concerned, is shown in Figure 6.

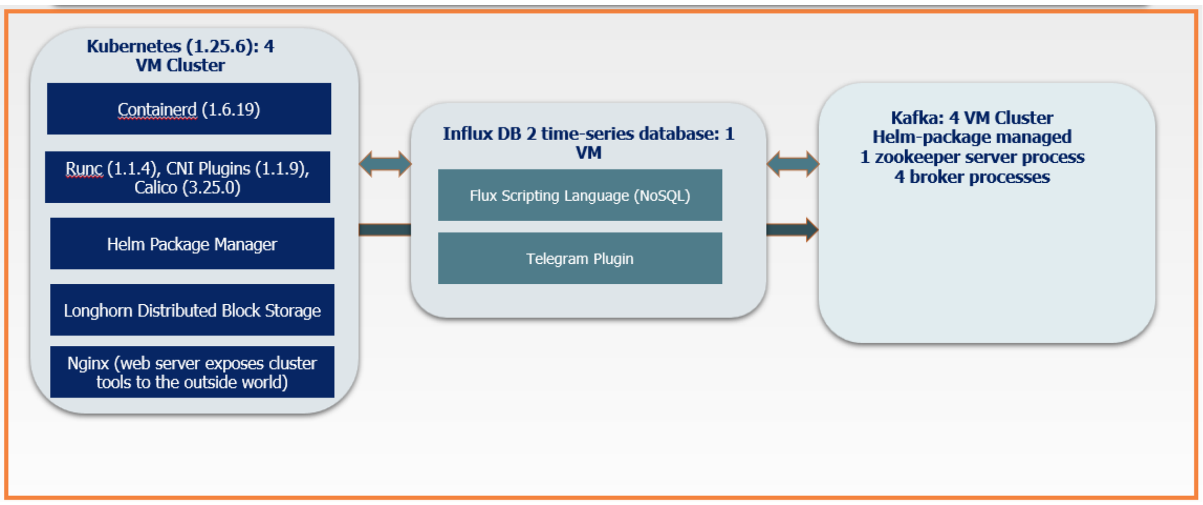
Kubernetes (1.25.6): 4 VM Cluster
Containerd (1.6.19)
Runc (1.1.4), CNI Plugins (1.1.9), Calico (3.25.0)
Helm Package Manager
Longhorn Distributed Block Storage
Nginx (web server exposes cluster tools to the outside world)
Influx DB 2 time-series database: 1 VM
Flux Scripting Language (NoSQL)
Telegram Plugin
Kafka: 4 VM Cluster
Helm-package managed
1 zookeeper server process
4 broker processes

Figure 6: Distributed Data Management Deployment View.

The distributed data management layer currently consists of the following major core servers/functionalities:

- Kubernetes (1.25.6) 4 VM cluster
- Influx DB server single process
- Kafka 4VM cluster including ZooKeeper management software

4 EVENFLOW Scenarios and Use Cases

4.1 Logical Architecture of Industry 4.0 Use Case

4.1.1 Simulation “environment modelling for prediction”

The NVIDIA Isaac Sim simulation tool was used to create a meaningful prediction data set. This allowed a realistic replication of the physical environment in different scenarios.

First, a dataset was created for complex event prediction of motion patterns for other robots. In the further course, prediction for other moving objects and humans will also be done.

4.1.2 MPC for trajectory generation

Model Predictive Control (MPC) is deployed to generate optimal velocity commands within a given prediction horizon, along the three Euclidian axis. The solution is dependent on the actual robot measurements (i.e., actual position, velocity...) and information provided by Costmap, which essentially contains feasible paths. The optimal velocity commands are translated into desired robot wheel velocities, a task for which the low-level PID (Proportional-Integral-Derivative) controller is responsible. In a nutshell, MPC block takes into consideration all constraints, such as physical constraints of the Robotino or the ones coming from Costmap, and then finds the optimal trajectories.

The overall nonlinear programming problem is modelled using CasADi framework, available in Python, Matlab and C++. Solvers used are IPOPT (Interior Point Optimization) and occasionally some open-source sequential quadratic programming solvers (e.g., QRQP). Moreover, the open-source linear solver used are the ones from HSL (Harwell Subroutine Library) such as MA27. For a portion of the experimental results, the free-academic linear solver MA57 is used.

4.1.3 Costmap "Dynamic Path Generation"

The costmap is an essential component for navigation in mobile robotics. It allows the robot to take sensor data from the environment and convert it into a 2D or 3D raster. In this grid, the cost is inflated and determined based on occupancy and the inflation radius set by the user. This enables the robot to generate motion paths within the occupancy grid.

For our implementation, we use the ROS2 navigation stack, which is considered state of the art in robotics. This stack provides a robust and proven basis for navigating the robot in different environments. The stack consists of different layers that process information about the environment in different ways. One of the layers we developed is the "Dynamic Obstacle Layer with Semantic Recognition". This layer detects dynamic obstacles in real time and uses semantic information to classify the obstacles according to their type. This allows specific safety distances for different obstacles to be taken into account. In addition, the current speed vector is calculated and the actual lane is blocked, which serves as a basis for the integration of the Forecasting Trajectories of dynamic obstacles.

Another layer is added to take into account the predicted trajectories of other participants, such as robots and humans. This information helps the robot to navigate predictively and plan an optimal, cost-effective and time-saving route to the goal (see Figure 7).

The goal is not only to avoid dynamic obstacles, but also to incorporate the movement patterns of other people and objects into the navigation planning, thus increasing the safety, efficiency, and security of the robot.

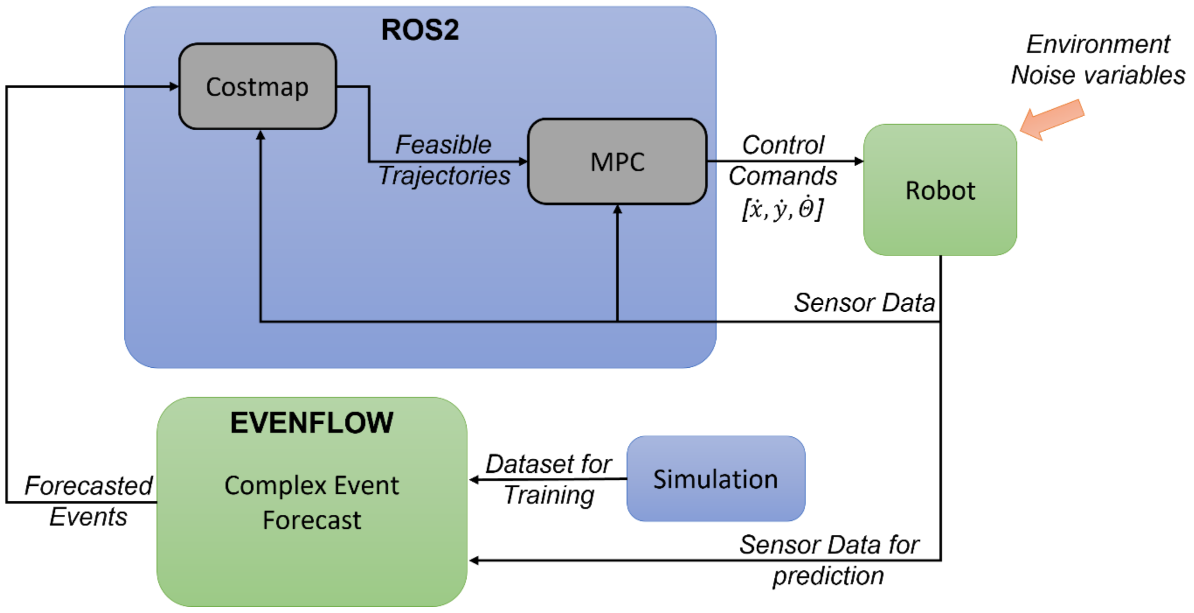

Figure 7: Logical Architecture of the Industry 4.0 Use Case.

4.2 Logical Architecture of Personalized Medicine Use Case

4.2.1 Generation of synthetic molecular trajectories between cancer stages

The first component of the use case architecture involves a generative model known as the beta-Variational AutoEncoder (beta-VAE) [REF-04]. This model is trained on molecular data from different stages of a specific cancer of interest, such as breast cancer stages I to III, obtained from the TCGA database[2]. The purpose of this component is to generate synthetic instances or data points that are interpolated between the endpoints of interest. The interpolation is performed in the latent space using various approaches. The resulting synthetic dataset consists of molecular profiles, specifically gene expression values, corresponding to the pseudo-time points between the given cancer stages.

4.2.2 Mapping gene to pathways

The second component of the use case architecture involves mapping the genes associated with the synthetic molecular profiles to cellular pathways. This operation is essential to reduce the high cardinality of the original dataset and identify key elements representing

[2] https://www.cancer.gov/ccg/research/genome-sequencing/tcga

general cellular processes in which the genes are known to be active. The mapping between genes and pathways is accomplished using the Reactome database[3], which provides a comprehensive collection of biological pathways and their associated gene participants.

4.2.3 Forecasting molecular trajectories

The third component of the use case architecture is the forecasting machinery. This component adopts a neuro-symbolic approach to predict the stage of a new patient based on their molecular profiles and the most probable molecular events that are likely to occur before progressing to the next stage of the cancer. The forecasting machinery combines neural network-based algorithms with symbolic reasoning techniques to make accurate predictions and provide valuable insights for clinical decision-making (see Figure 8).

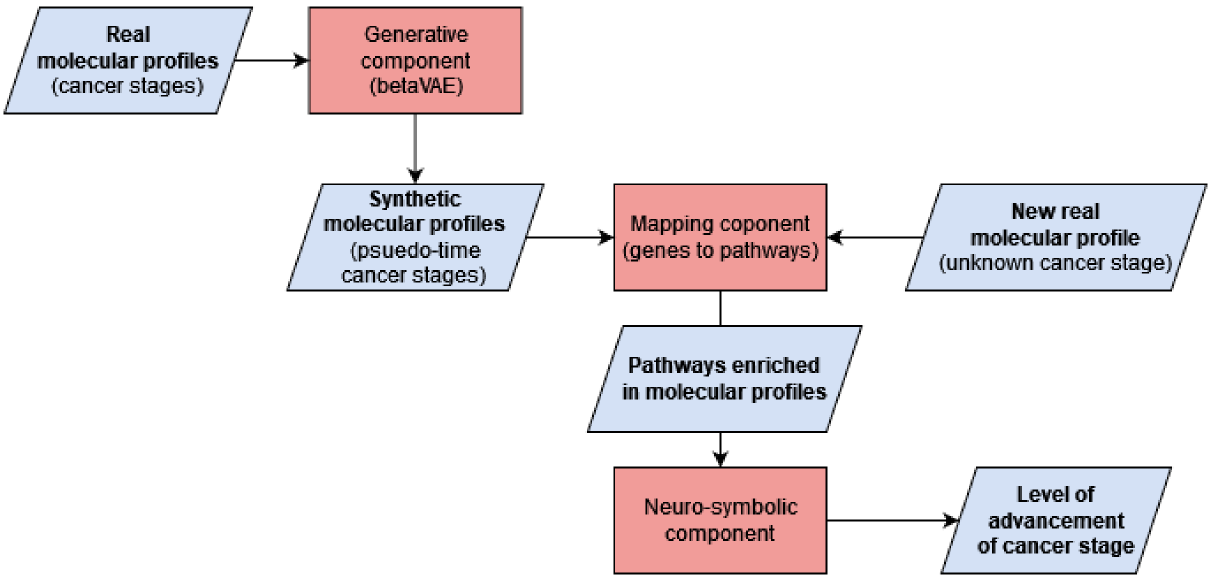
Real molecular profiles (cancer stages)
Generative component (betaVAE)
Synthetic molecular profiles (psuedo-time cancer stages)
Mapping coponent (genes to pathways)
New real molecular profile (unknown cancer stage)
Pathways enriched in molecular profiles
Neuro-symbolic component
Level of advancement of cancer stage

Figure 8: Logical Architecture of the Personalized Medicine Use-Case.

4.3 Logical Architecture of Infrastructure Life Cycle Assessment Use Case

The Infrastructure Life Cycle Assessment Use Case comprises a testbed of water leakage in pipes simulations done not on computer hardware but rather on specialized pipes networks that aims to mimic the behaviour of leakages in real underground water pipes transferring water to all buildings in a city. A number of vibration sensors are installed in strategic locations along the main pipe, and a number of scenarios are actually carried out, attempting to emulate different leakage conditions. The leakages themselves are implemented as taps with certain diameter that can be open or closed at any time. In any given scenario, a period of "warming up" (incoming water to the pipe) is followed by the opening and/or closing of a certain tap(s) along the main pipe; during the whole scenario, the installed vibration sensors record vibrations at a very high frequency (around 6000 measurements per second.) The

[3] https://reactome.org/

objective is to devise a leakage detector that can detect within seconds the existence of a leakage in the real water pipe network, and in addition can accurately estimate the location of the leakage for maintenance crews to know where to look.

Each sensor transmits wirelessly its data to a centralized PC where the collection and storing of the data takes place. This data is then to be transferred in (near) real time to a classifier/regressor that EVENFLOW will develop that will raise an alert with the location of an estimated rupture in the water pipe, if such a decision is made (see Figure 9).

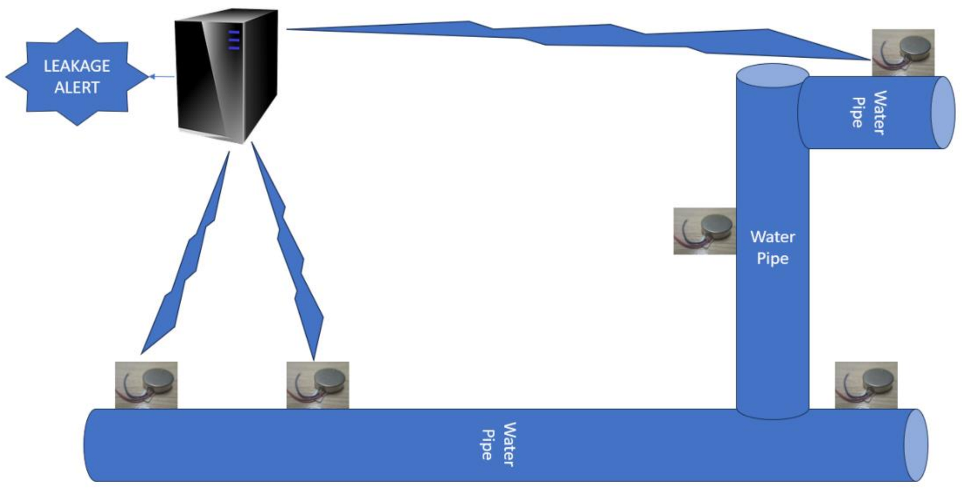

Figure 9: Logical Architecture of the (EKSO) Public Utilities Maintenance Use Case.

The major components for this Use-Case therefore involve the following:

1. **Data Collection & Storage**: this component is likely to be implemented in terms of the Kafka message bus or the Influx DB, as data are clearly multi-variate time-series.
2. **Pre-processing component**: the high frequency of the data is likely to require some kind of aggregation functionality. This can be achieved either with special-purpose custom-built scripts (ETL scripts) or directly in the neural network architecture as a series of convolutional layers followed by some pooling operator (max-pooling, min-pooling, average-pooling or any combination.)
3. **Training Component**: Both XAI as well as scalable distributed Neural Networks (including direct or memory-based or attention-based) can both be used for this task.
4. **Decision making Component**: the component will be responsible for using the results of the Training Component and turning them into an estimation of the location of a leakage, together with the probability of there being a significant enough leakage.

It is important to note that the developed mechanism must be robust and accurate enough so that it never confuses the opening of a residential tap (e.g., someone taking a shower, or watering their garden) with the creation of an actual leakage.

4.4 Logical Architecture of Synopsis/Evolving ML Solutions

4.4.1 Synopses Data Engine – as – a – Service (SDEaaS) Component

SDEaaS[4] [REF-01] is built on top of Apache Flink and implements a novel synopses-as-a-service paradigm. That is, the SDE runs as a constantly running job (Figure 10) in one or more, potentially geo-dispersed, computer clusters accepting on the fly requests for maintaining data stream summaries (approximately 20 different data stream summaries are supported). In that, it achieves (i) concurrently maintaining thousands of synopses, of various kinds, for thousands of streams, on demand, (ii) reusing synopses that are common across various concurrent training pipelines, (iii) providing data summarization facilities even for cross-neural learning platforms, (iv) pluggability of new synopses on-the-fly, (v) increased potential for training optimization (discussed shortly). SDEaaS provides scalable training of neural models by enabling 3 types of scalability: (i) enhanced horizontal scalability, i.e., not only scaling out the computation to a number of processing units available in a computer cluster, but also harnessing the processing load assigned to each by operating on carefully-crafted data summaries, (ii) vertical scalability, i.e., scaling the computation to very high numbers of processed streams and (iii) federated scalability i.e., scaling across geo-distributed clusters and clouds by controlling the communication required to develop global training models.

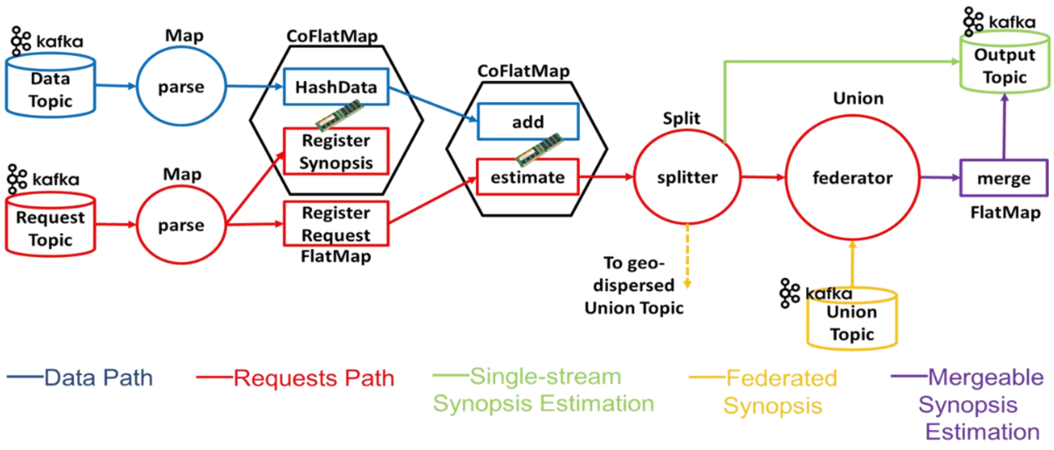

Figure 10: Preliminary Version of SDEaaS Internal Architecture over Apache Flink.

4.4.2 Synopses-based Training Optimization Component

This component uses the SDEaaS and the various types of scalability it provides, to tune and optimize scalable, synopsis-based learning on any given neural training task. More precisely, it aims at determining (i) the kind of synopsis that should be used, (ii) the amount of data reduction that should be imposed on the raw data streams, (iii) the number of epochs that should be used during the training process, using data stream summaries instead of the raw streams. The outcome of this component is the optimal setup for (i) - (iii) to properly balance the training time vs accuracy trade-off for a scalable neural model training procedure.

4 https://sdeaas.github.io/

4.4.3 Data-driven Distributed Training Component

This component boosts the scalability of neural training by acting complementarily to the Synopses-based Training Optimization Component. It follows a Parameter Server (PS) training paradigm 28[REF-02], distributing the training process of a given neural model over a number of worker machines available in corporate data centres or the cloud. More precisely, each worker trains an identical copy of the neural model but does so on a separate (disjoint) portion of the incoming streams (Figure 11). The local models developed at each worker are synchronized, from time to time, to a global model. Besides the traditional synchronous/asynchronous synchronization mechanisms that are supported by the PS paradigm, in EVENFLOW we develop advanced, data-driven synchronization protocols that require a synchronization step only when a concept drift is likely to have occurred. Therefore, the accuracy vs training time vs communication/latency trade-offs of the training process can be controllably tuned for scalability and model evolution (drift) detection purposes.

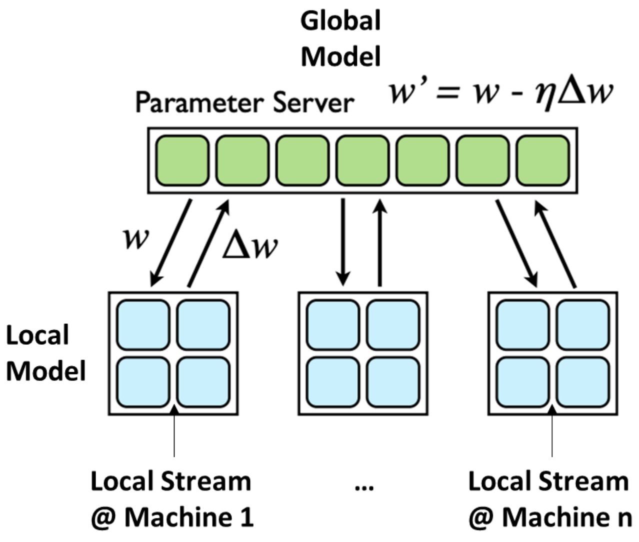

Figure 11: Basic Representation of the Parameter Server – based Training Paradigm.

4.4.4 Interplay of Components in EVENFLOW Scalability Toolkit

Figure 12 illustrates the interplay of the various components of EVENFLOW Scalability Toolkit. The SDEaaS use is initially internal to the Synopses-based Training Optimization Component as it provides and maintains that supported data stream summarization techniques to be tested for accuracy vs training time performance. Based on the result of the optimization, which dictates the best possible triplet of <types of synopses, data reduction ratio, number of training epochs>, SDEaaS is tuned and employed in the production pipeline. On the right-hand side of the figure, each worker machine maintains the determined synopses on the local streams it receives and locally trains a copy of the neural model devised by the application. The local models built by each worker are synchronized based on either traditional [REF-02]or novel synchronization protocols, developed in the scope of EVENFLOW, into a global model. The trained global model is employed to label simple derived events used by the neuro-symbolic component of EVENFLOW. The described pipeline can be executed continuously or in predefined intervals.

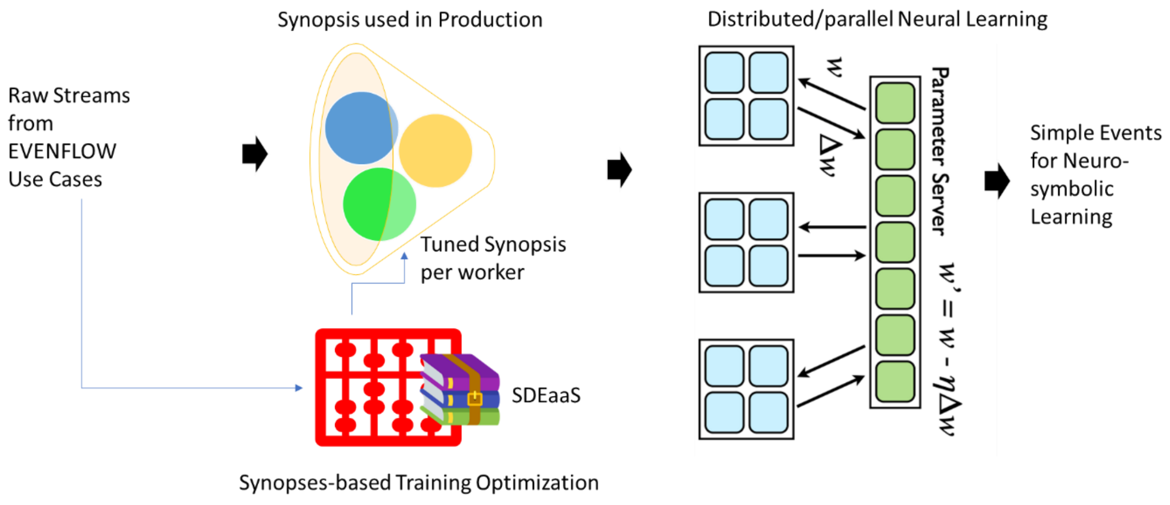
Synopsis used in Production
Distributed/parallel Neural Learning
Raw Streams from EVENFLOW Use Cases
Tuned Synopsis per worker
SDEaaS
Synopses-based Training Optimization
Parameter Server
w
Δw
$w' = w - \eta\Delta w$
Simple Events for Neuro-symbolic Learning

Figure 12: Architectural Viewpoint of EVENFLOW Scalability Toolkit.

4.5 Logical Architecture of Verification Solutions

Verification solutions include two main modules: (i) a formal verification toolbox that checks whether a property provably holds on a pre-trained network—henceforth, the "*verifier*" (ii) a scheme that trains the network so as for the required property to provably hold, post-training—henceforth, "*robust training*".

4.5.1 Required inputs

Both modules will require the following inputs:

- A *property specification*, expressed as an *input domain* (the set of allowed inputs: for instance, for adversarial examples, the set of allowed perturbations) and an *output condition* (all the outputs from the input domain should satisfy the condition: for instance, they should all be correctly classified). In order to maximise support from toolboxes, input domains should be convex sets that allow for efficient optimization, and output conditions should be Boolean formulas over inequalities.
- The *model*, which can be pre-trained if input to the verifier, or randomly initialized, if input to the verified training module. Ideally, the architecture itself is designed to maximize verifiability: as small as possible, with the least number of non-linearities.

4.5.2 Verifier module

Given the inputs above, *the verifier outputs either "satisfied", "violated" or "undecided"*. If the answer is "violated", the verifier provides a concrete counterexample violating the property.

The verifier first attempts to violate the property by searching for counterexamples (typically, through local optimization methods). If no counterexample is found, the verifier creates an abstraction of the model (typically, through a convex relaxation) and attempts verification on it. If verification of the abstraction fails, the verifier refines the abstraction by iteratively dividing the original problem into a series of sub-problems, on which it attempts verification through abstractions and looks for counterexamples, until it is timed out (case in which it outputs "undecided"), or a definite answer is provided. Sub-problems can be solved in parallel. The verifier implementation will be tailored to the specifics of the EVENFLOW-specific inputs (the model types and the property specifications).

4.5.3 Robust training module

Given the required inputs as defined above, *the robust training module outputs a version of the model that satisfies the specified property on a relatively large share of the training examples and generalizes to unseen examples*.

Depending on the property specification, standard-trained networks may either display easy-to-find counterexamples to the property or be hard to verify (inducing the verified to hit the exponential worst-case). This module internally employs abstractions and potentially counterexamples from the verifier module at training time. Adequate verification is also employed to ensure the property holds on unseen examples. As for the verifier component, the robust training model will be specifically designed for the EVENFLOW use-cases.

5 Conclusions and Future Outlook

EVENFLOW is carrying out cutting edge research in different areas of neuro-symbolic learning, including explainable AI and formal verification methods. The project's research output includes novel neuro-symbolic learning techniques, along with their applications in real-life applications. Along with these research outcomes, the project is also researching and developing a platform that will integrate the different research outcomes of the project towards easing access to integrate neuro-symbolic learning models and applications. The project's platform is destined to facilitate the development and integration of end-to-end neuro-symbolic AI applications, including data access, data pre-processing, data analytics, machine learning, formal verification, and visualization modules, among other elements of EVENFLOW applications. As such, the EVENFLOW platform will facilitate the integration of the project's use cases as part of WP6 of the project.

This deliverable has provided an initial outline of the architecture and logical design of the platform, in-line with the 4+1 view methodology for describing architectures of software systems. The 4+1 views methodology allows EVENFLOW to deliver the architecture design in a modular fashion. In the scope of the present deliverable, the logical view of the EVENFLOW has been introduced, which focuses on the functionalities of the platform and includes the structuring principles that will drive the integration of the different modules of the platform. At the logical level, the EVENFLOW platform comprises conventional modules that typically used to development ML pipelines, along with novel modules that enable the development and integration of neuro-symbolic learning modules. Apart from detailing the logical view of the architecture, the deliverable has provided early insights on other views of the EVENFLOW platform architecture such as the implementation and deployment views that will be driving the development and operation of the platform at later stages of the project. Moreover, the deliverable presents an initial set of interactions and information flows between the different components of the platform, as part of early versions of the process view(s) of the platform.

Overall, the present deliverable provides a sound basis for advancing the platform development and use case integration activities of the project. The latter will boost the development of additional views of the architecture, as well as the improvement of already presented views. The next versions of the deliverable will gradually lead to a complete and detailed description of the architecture of the EVENFLOW platform, as well as to the final specifications of the integration EVENFLOW systems and applications.

6 References

[REF-01]	Antonios Kontaxakis, Nikos Giatrakos, Dimitris Sacharidis, Antonios Deligiannakis, And synopses for all: A synopses data engine for extreme scale analytics-as-a-service, Information Systems, Volume 116, 102221, 2023
[REF-02]	Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. 2014. Scaling Distributed Machine Learning with the Parameter Server. In 11th USENIX Symposium on Operating Systems Design and Implementation, OSDI '14, Broomfield, CO, USA, October 6-8, 2014, Jason Flinn and Hank Levy (Eds.). USENIX Association, 583–598.
[REF-03]	EC, Regulation of the European Parliament and of the Council – Laying Down Harmonized Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative, https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A52021PC0206.
[REF-04]	Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, Alexander Lerchner, beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, ICLR 2017 Poster, https://openreview.net/forum?id=Sy2fzU9gl.