Explainable Machine Learning for Predictive Modeling and Abstraction in Industrial-Scale Systems

Licentiate Thesis Proposal

Student: Edin Jelačić

Supervisors: Tiberiu Seceleanu, Cristina Seceleanu, Ning Xiong, Peter Backeman

2.10.2025.

Mälardalen University

Color coding

- Methods

- Problems

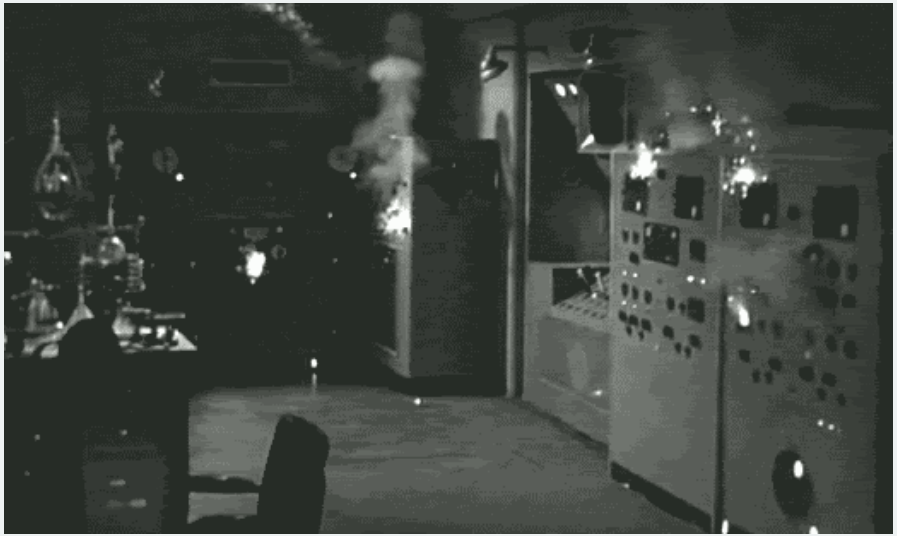
- Contributions

Background

- Picture a **power station measurement device** tasked with monitoring critical parameters.
- The device runs flawlessly when shipped, but engineers later **add new monitoring tasks** to adapt to changing specifications.

The device keeps on working…

- The measurement device keeps working, seemingly flawlessly, for months on end, so engineers keep on adding tasks
- Over time, the CPU load creeps upward… until one day, the device **overloads**, misses critical measurements, and fails to signal an anomaly.



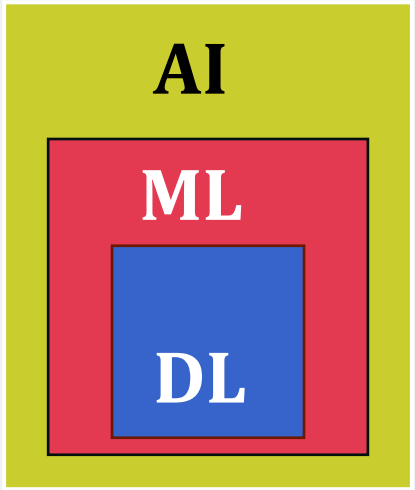
Potential risks

- **System blackout or safety hazard** due to missed signals

- **Current tools** (e.g., Task Manager, simulators) either:
  - show only aggregated usage (too coarse), or
  - require **deep CPU expertise,** run too slowly for engineers or are maladapted to use in practice

How can industrial-scale forecasting be solved before issues arise?

A **fast, transparent, uncertainty-aware method** is required to forecast industrial device safety margins **before deployment.**

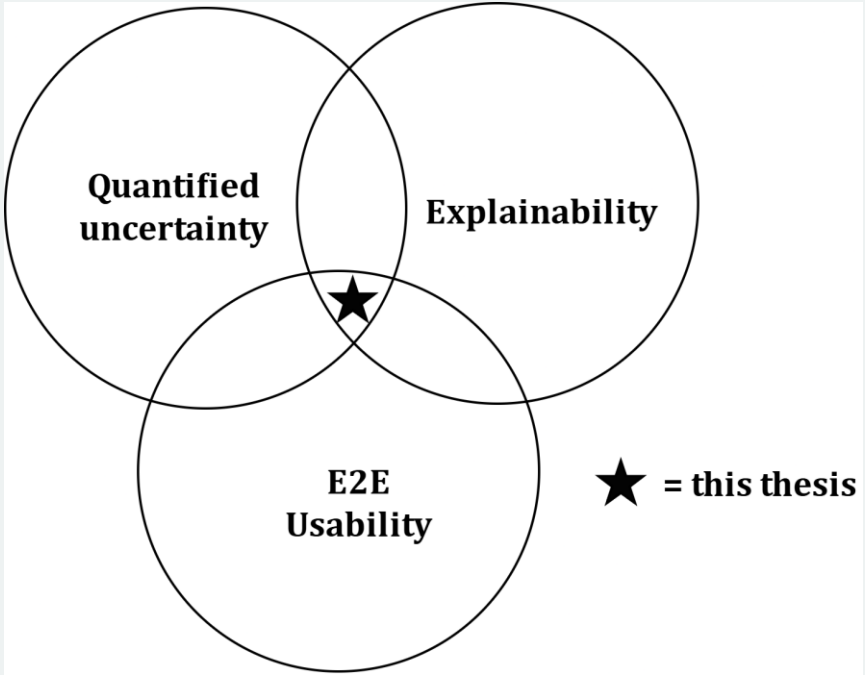


- Industrial devices = **complex, high-dimensional data**
- Manual or rule-based models cannot* scale to capture all interactions, thus we need an intrinsically automated technique
- *Machine learning* can uncover **hidden patterns** and *predict CPU load/cache performance* faster than simulators

* = in practice

Trustworthiness

That said, traditional ML is not enough → we need **trustworthy ML**



The blackout risk shows why accuracy alone is not enough.

Problem: Industrial AI is often accurate but not usable or trustworthy

Lack of trust slows adoption in safety-critical domains.
Consequences include unsafe ML systems, wasted engineering effort throughout both setup and usage.
Engineers need models they can **understand, trust, and apply**.

Traditional machine learning approaches
✅ High Accuracy
❌ No Guarantees
❌ Opaque
❌ Difficult for engineers to use

From accuracy to trustworthiness: key requirements
✅ High Accuracy
✅ Uncertainty Guarantees
✅ Explainability
✅ Practical Usability

Identification of three core research gaps

- **Gap_A**: Guarantees (and explainability) under-addressed in ML for industrial-scale systems
- **Gap_B**: Generalizable, fast low-level performance microarchitectural modeling is sparse; simulators are accurate but slow
- **Gap_C**: End-to-end, usable integration (narrative/data to runnable artifacts and decisions) is scarcely treated

Overarching research goal (**RG**)

Design and evaluate an automation-oriented, trustworthy ML approach for industrial-scale systems, combining explainability, guarantees, performance awareness, and end-to-end integration into usable workflows.

Research Questions (**RQs**)

- **RQ_1**: How can input features be ranked or discarded in a way that provides both interpretability and formal/statistical guarantees on their influence?
- **RQ_2**: How can forecasting methods be designed to provide valid uncertainty guarantees and interpretable explanations, while remaining lightweight enough for practical deployment in real systems?
- **RQ_3**: How can data-driven models of system behavior (e.g., cache performance) be designed to generalize across both workloads and microarchitectures?
- **RQ_4**: How should data-driven performance models be evaluated to ensure fidelity?
- **RQ_5**: How can natural-language narratives be transformed into runnable simulation scenarios with quantifiable and minimized expert repair effort?

Thesis contributions

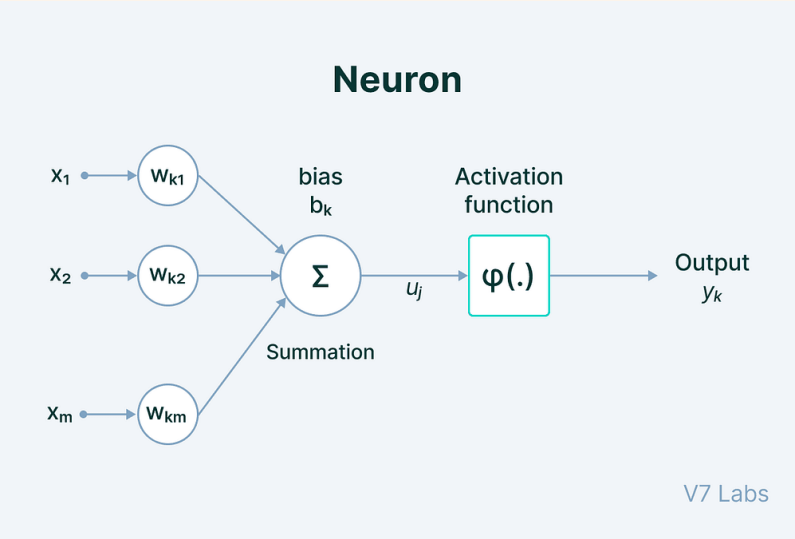
Contribution answers **RQ_1** and **RQ_5**

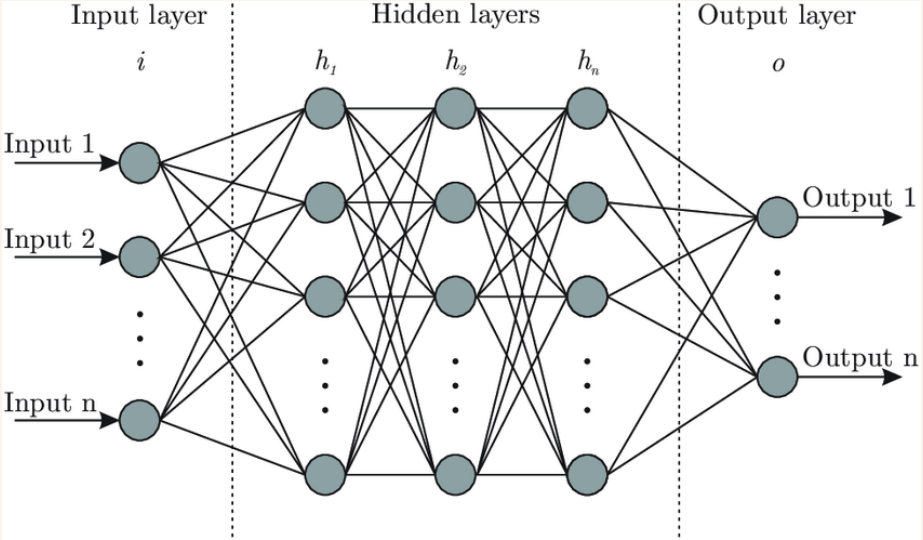
Contribution 1: Abstraction-based reduction of neural networks

We develop a principled mechanism for simplifying neural networks while retaining provable guarantees.

Neural Networks (NNs)

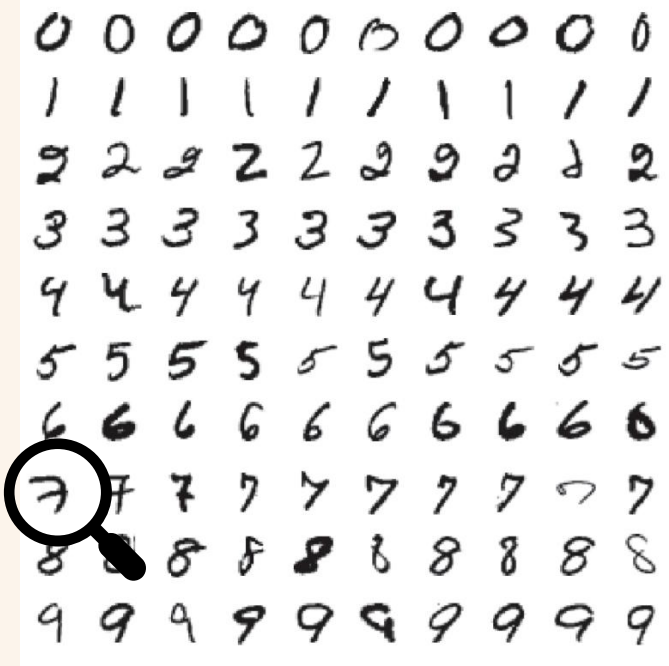
- Inspired by the brain, but mathematical: input → hidden layers → output
- **Universal approximators**: can model almost any function through induced nonlinearities (various existence theorems proving this throughout history, beginning in 1989 and onwards)
- Useful for CPU load/cache because they handle complex nonlinearities


Neuron
x1
xm
wk1
wk2
wkm
bias
bk
Σ
Summation
uj
Activation
function
φ(.)
Output
yk
V7 Labs

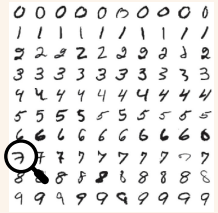

Input layer
Hidden layers
Output layer
i
h1
h2
hn
o
Input 1
Input 2
Input n
Output 1
Output n

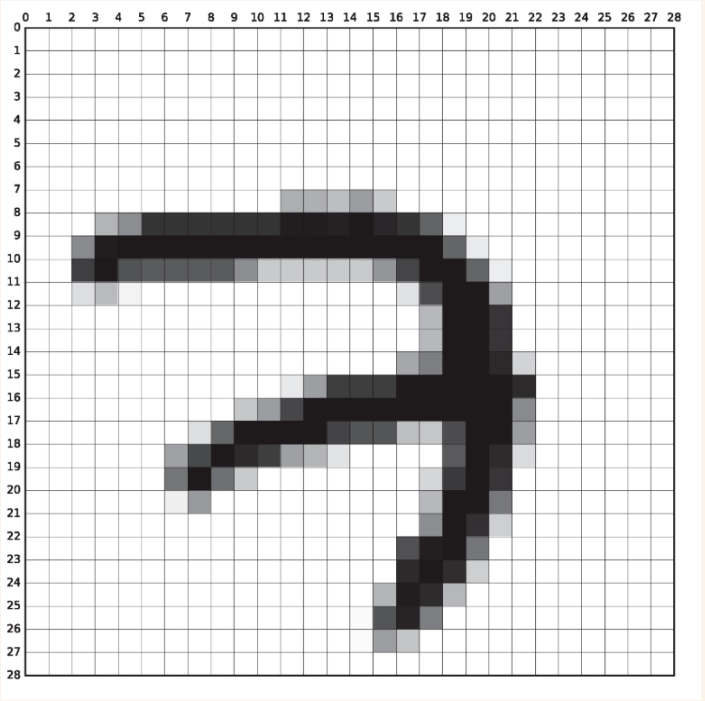
Neural Network example

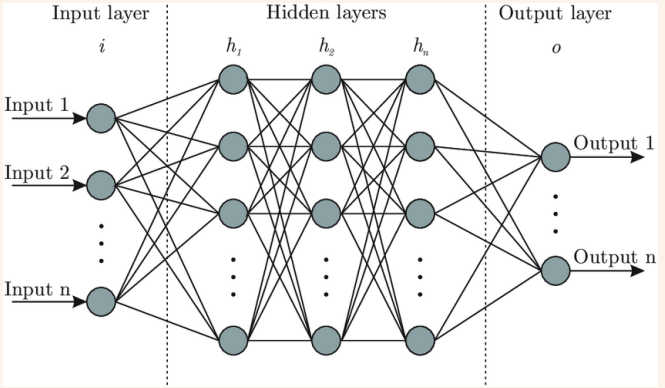
Neural Network example



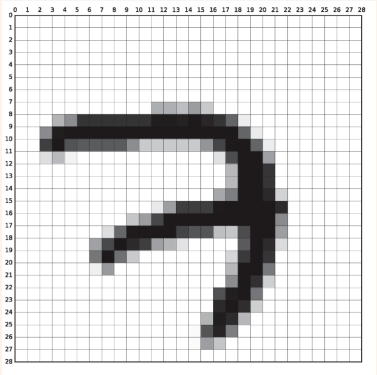
Neural Network example

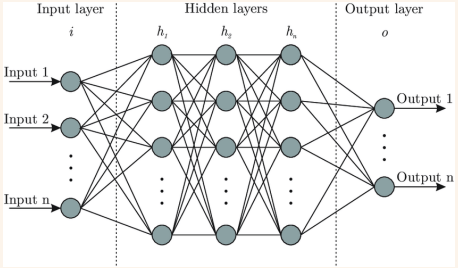


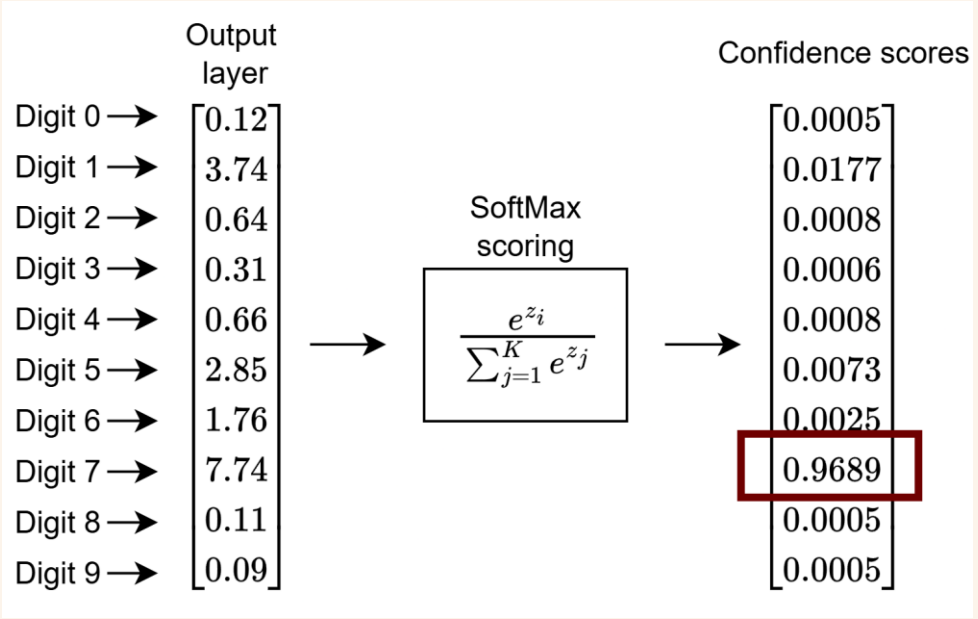




Neural Network example

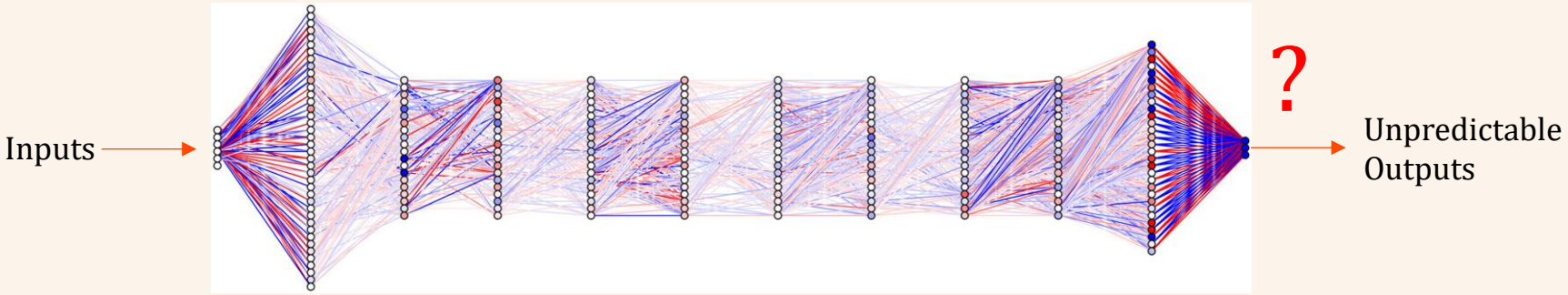






Verification

- **Problem:** Neural networks behave like black boxes
- In **safety-critical or industrial systems**, “probably correct” is not enough.

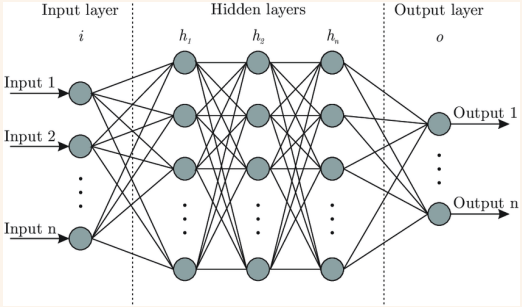


Source: https://tbenthompson.com/project/visconn/

Verification

Verification = proving that for **all** possible inputs, the system respects a property.

Inputs →



→ Property check $f(x) \leq \tau$

Verification

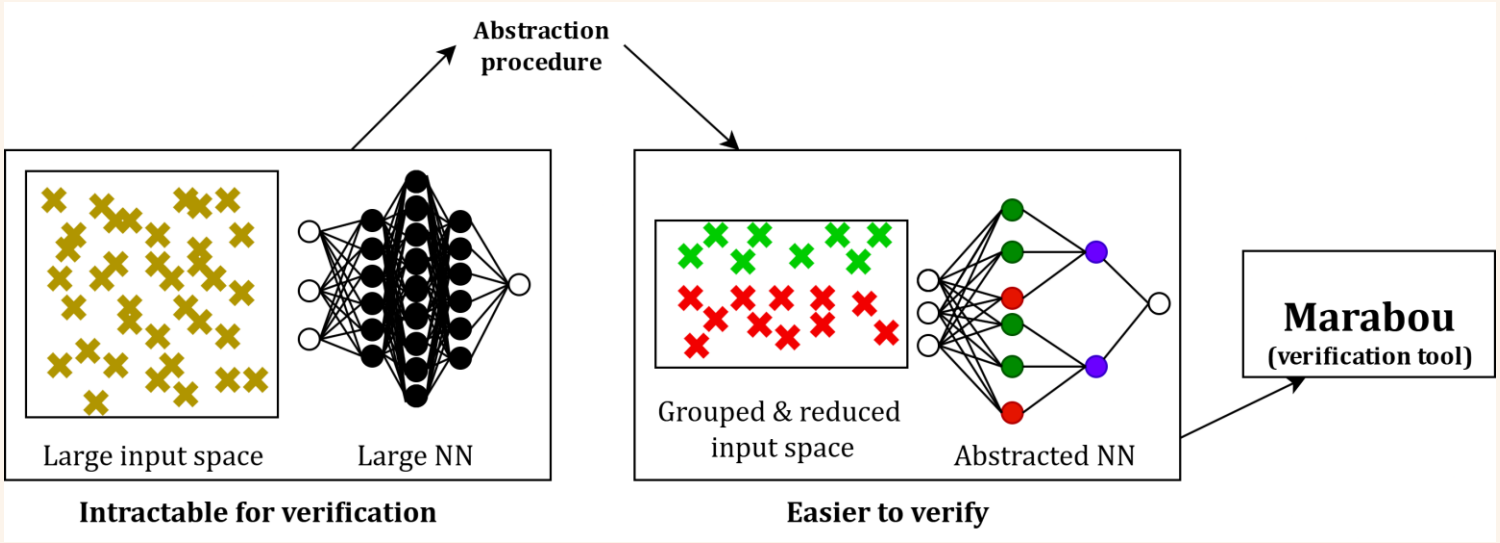
- Verification gives **formal guarantees** that increase **trustworthiness**.
- Without it, we risk **hidden failures** in rare scenarios.

Testing: checks some cases ✗

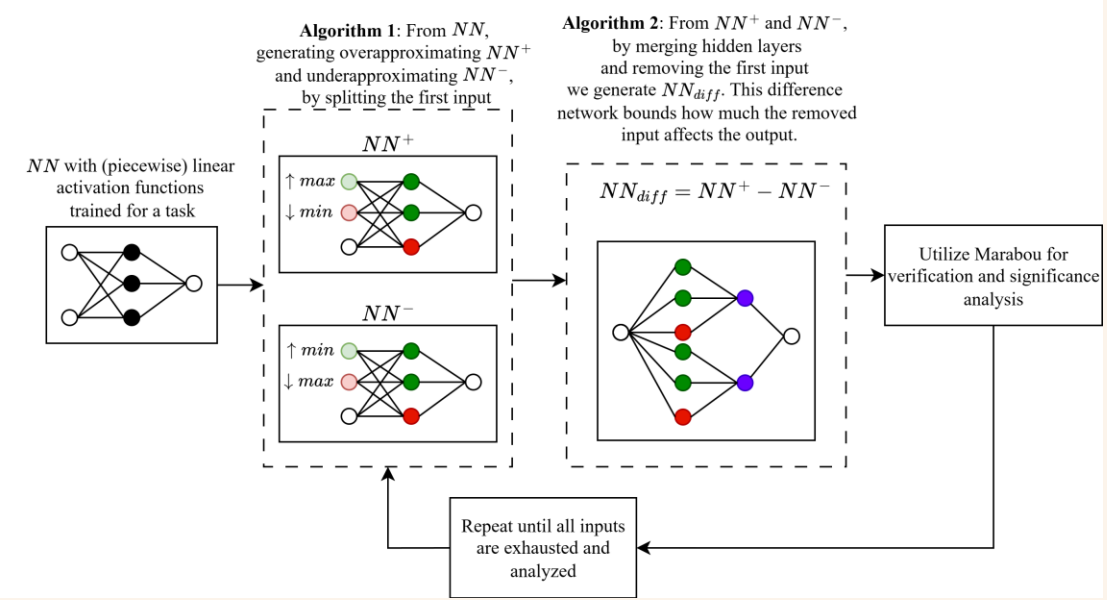
Verification: covers all cases ☑

Neural Network Abstraction

- **Challenge:** NNs = huge, hard to verify properties
- **Idea:** Abstraction shrinks/compartmentalizes search space
- **Why:** Enables formal tools (e.g., Marabou)



Contribution 1 - core



- **Approach:** Abstract inputs → build smaller "difference network"
- **Result:** Enables Marabou usage to safely remove features with negligible influence while preserving output behavior within known bounds

Contribution to RG: Advances *trustworthiness* by making NN feature significance identification feasible in practice.

Contribution answers **RQ_1**, **RQ_2** and **RQ_3**

Contribution 2: Forecasting with uncertainty and explanations

We develop an uncertainty-aware CPU load forecasting framework designed to support practical safety assessment in industrial computing platforms.

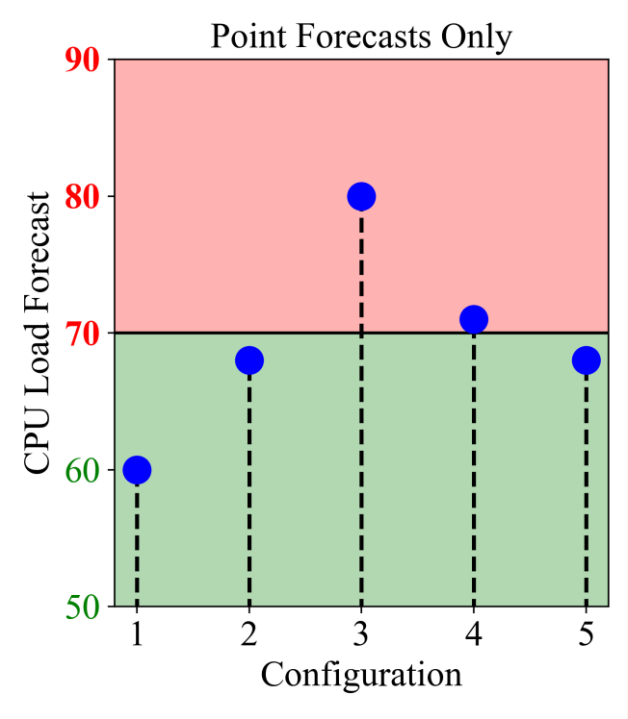
Conformal Prediction (CP)

•Problem: Point predictions hide uncertainty.

• Other methods such as Bayesian NNs, ensemble & dropout are ad-hoc & heuristic.

•**CP adds prediction intervals** with statistical guarantees over the dataset (e.g., 90% confidence).

•Ensures engineers know not only "what" the model predicts but also **"how certain" it is.**



Without uncertainty quantification

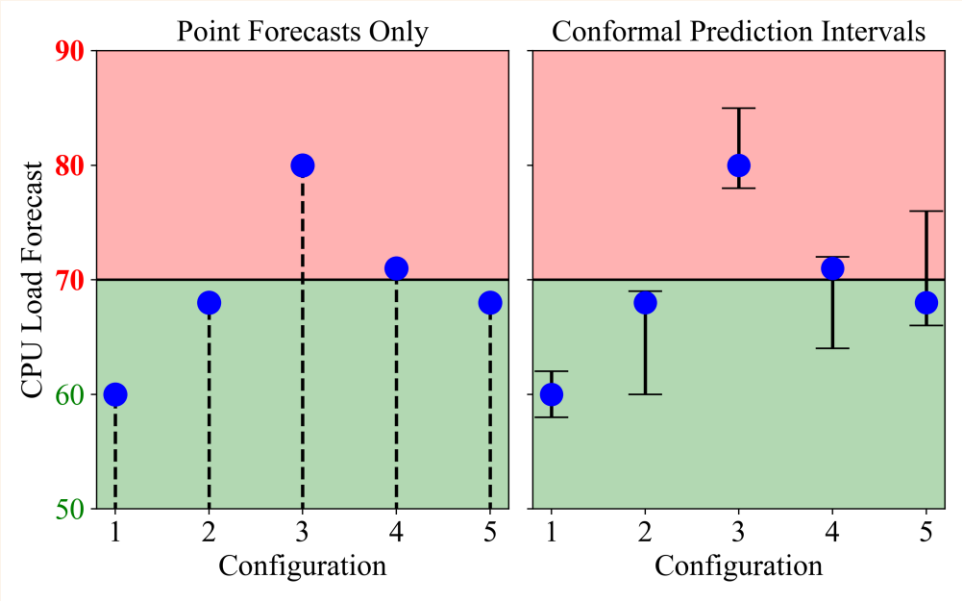
Conformal Prediction (CP)

- Problem: Point predictions hide uncertainty.

- Other methods such as Bayesian NNs, ensemble & dropout are ad-hoc & heuristic.

- **CP adds prediction intervals** with statistical guarantees over the dataset (e.g., 90% confidence).

- Ensures engineers know not only "what" the model predicts but also **"how certain" it is.**



With uncertainty quantification

On conformal prediction - visualization

Without CP

On conformal prediction - visualization

Explainability

- **Idea:** Fairly assign each feature a share of the prediction (from game theory).
- **Why Shapley:** Guarantees additivity & fairness, every feature's contribution sums to the output.
- **Value:** Tells us *which inputs drive predictions most*.

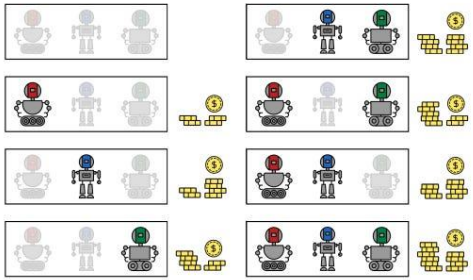
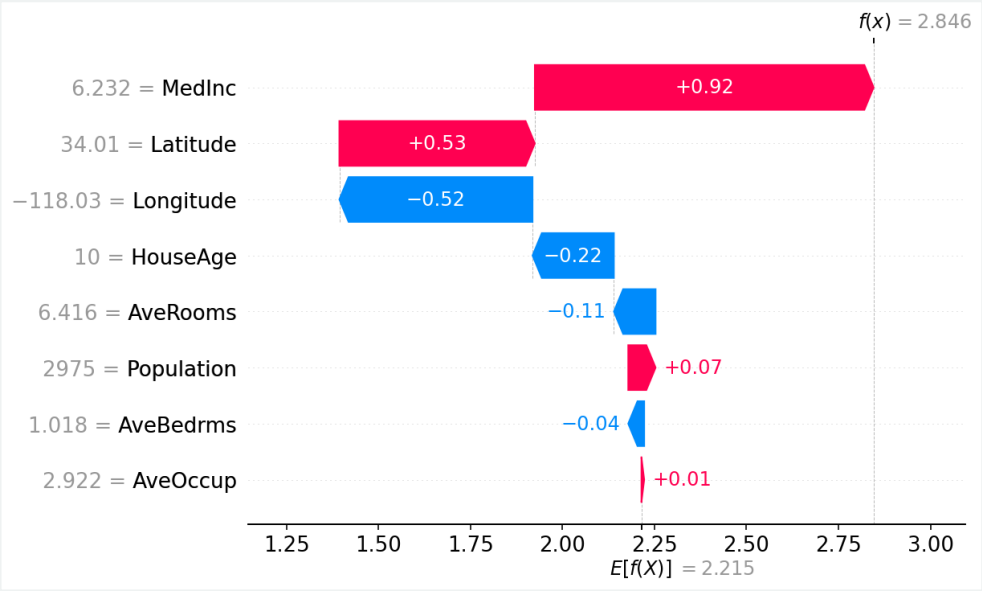


Figure 1: The Shapley value can be used to solve cooperative games An ensemble game is a machine learning application for it – models in an ensemble are players (red, blue, and green robots) and the financial gain of the predictions is the payoff (coins) for each possible coalition (rectangles). The Shapley value can distribute the gain of the grand coalition (right bottom corner) among models.

$$Load_{prediction} = Load_{base} + \sum task_i$$

Source:https://arxiv.org/abs/2202.05594

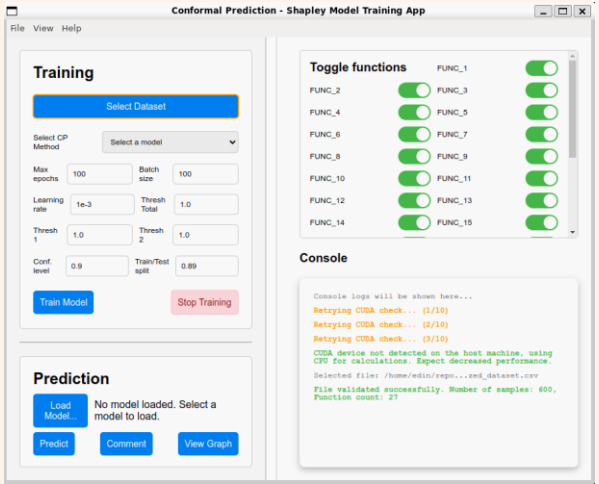
Example of Shapley values proving their mettle

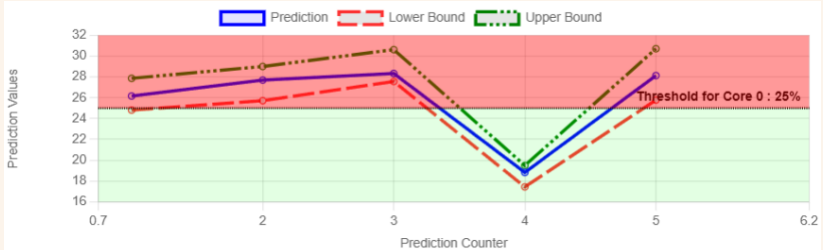


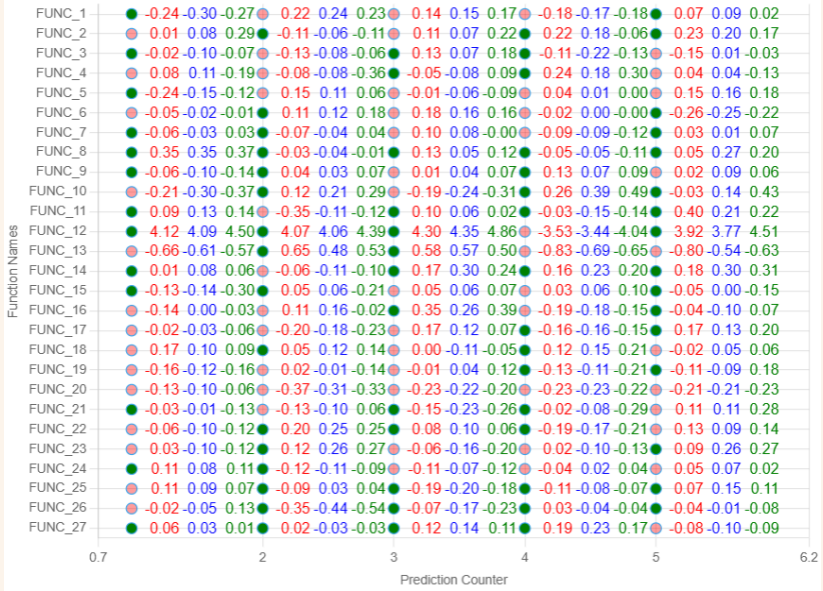
$$f(X) = E[f(X)] + \sum_i feature_i$$

Contribution 2 - core

- **Result:** Provides **statistically valid prediction intervals** and **clear explanations** of which tasks drive CPU overload.
- **Contribution to RG:** Strengthens *trustworthiness* of ML forecasts in safety-critical industrial systems.







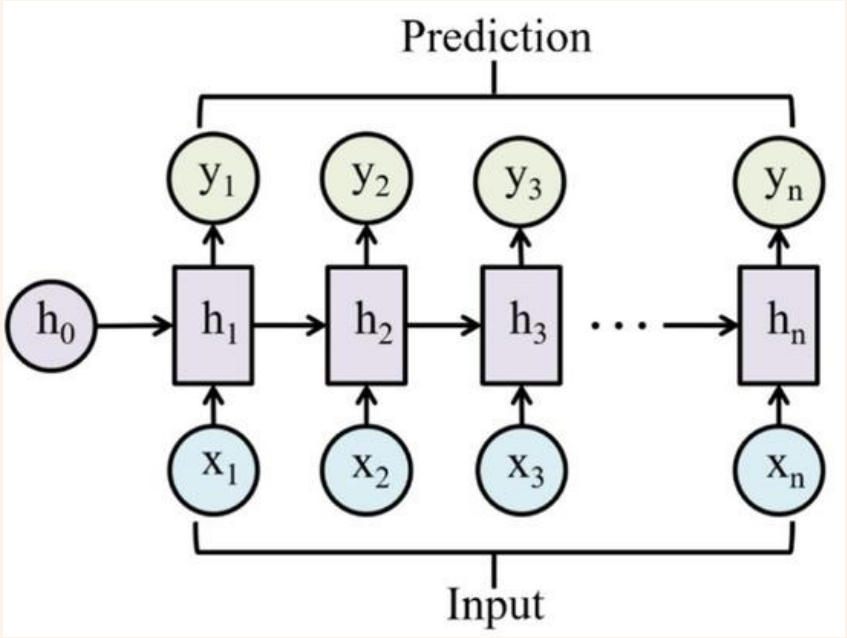
Contribution answers **RQ_2**, **RQ_3**, **RQ_4** and **RQ_5**

Contribution 3: Data-driven cache miss prediction

We develop a data-driven model framework for cache miss prediction at the microarchitectural level.

Recurrent Neural Networks (RNNs)

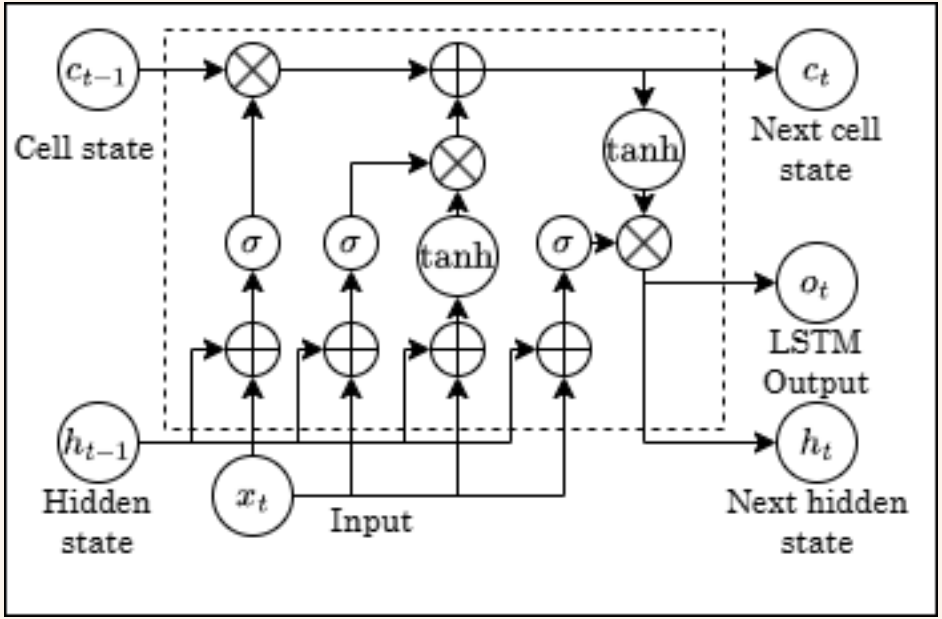
- Unlike standard NNs, RNNs handle **sequences**.
- Feedback loops → remember past inputs.
- Great for time-dependent data
- Downside: gradient vanishing or explosion



Long Short-Term Memory (LSTM) networks

- Fixes problems with long-term memory in standard RNNs (vanishing/exploding gradients).
- Uses **“gates”** to decide what to remember/forget.

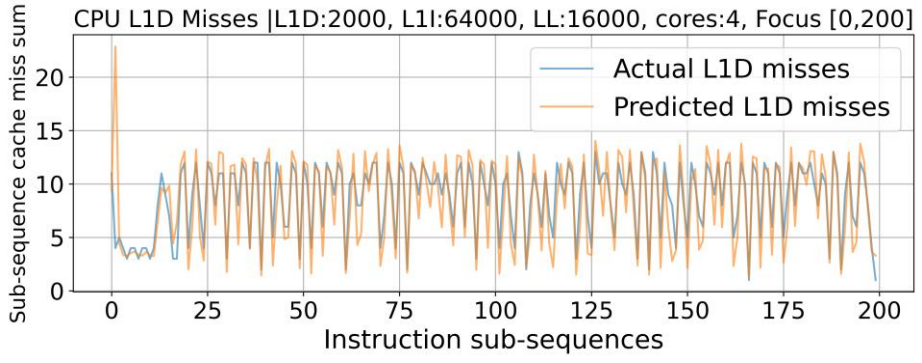
- Perfect for time-series CPU data (e.g. “cache miss sequences ”)

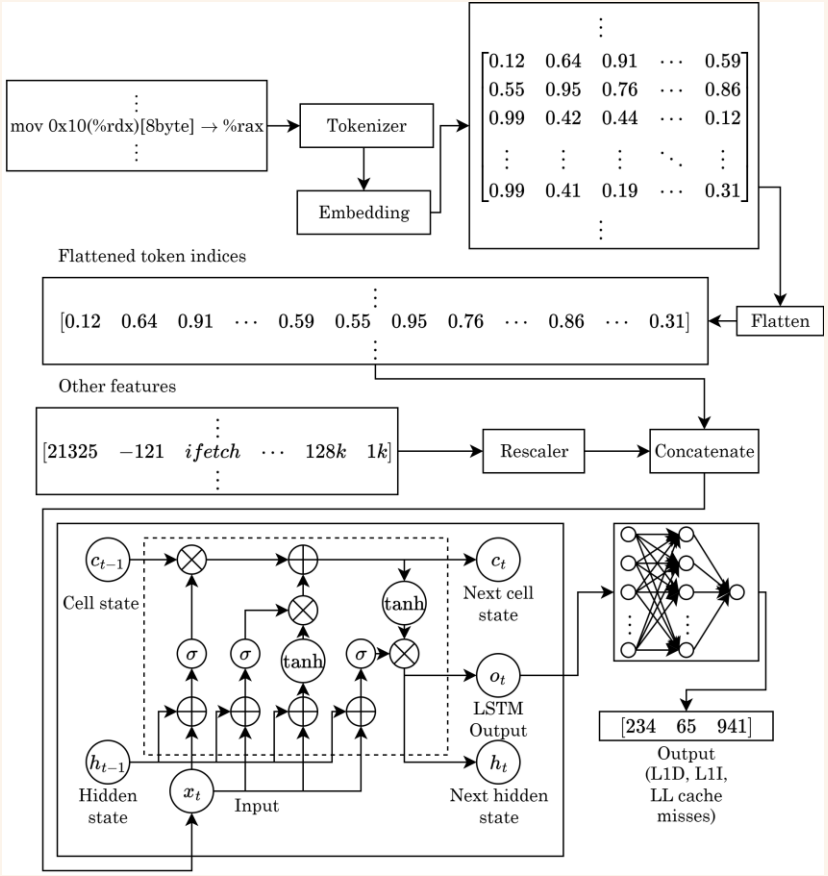


Contribution 3 - core

- **Accuracy:** Predicted cache miss distributions closely match simulator outputs.
- **Novelty:** First ML-based approach to approximate cache simulation *without full architecture knowledge*.
- **Contribution to RG:** Improves *scalability and efficiency* of forecasting in industrial systems.

21% time reduction as opposed to running simulation in DynamoRIO





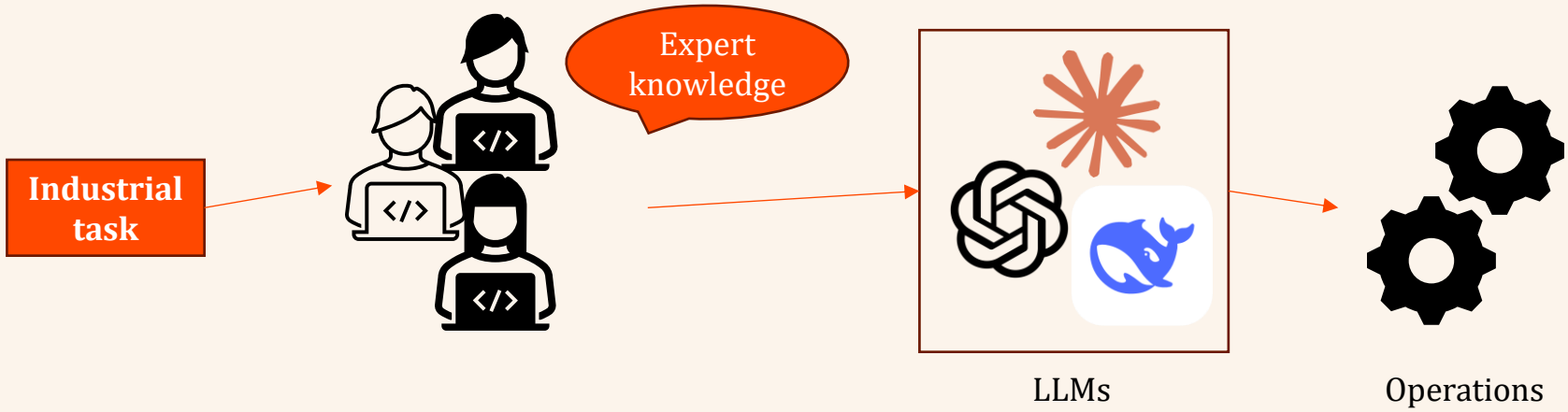
Contribution answers **RQ_4** and **RQ_5**

Contribution 4: Hybrid AI Simulation Compiler (HASCo)

- Finally, **HASCo**, a **H**ybrid **A**I **S**imulation **Co**mpiler that transforms natural-language reports into runnable OpenSCENARIO/OpenDRIVE simulations.
- HASCo integrates retrieval over standards, execution-based validation, and self-repair loops on top of large language models.

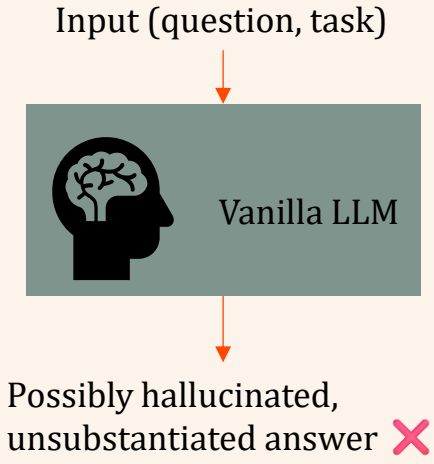
Large Language Models (LLMs)

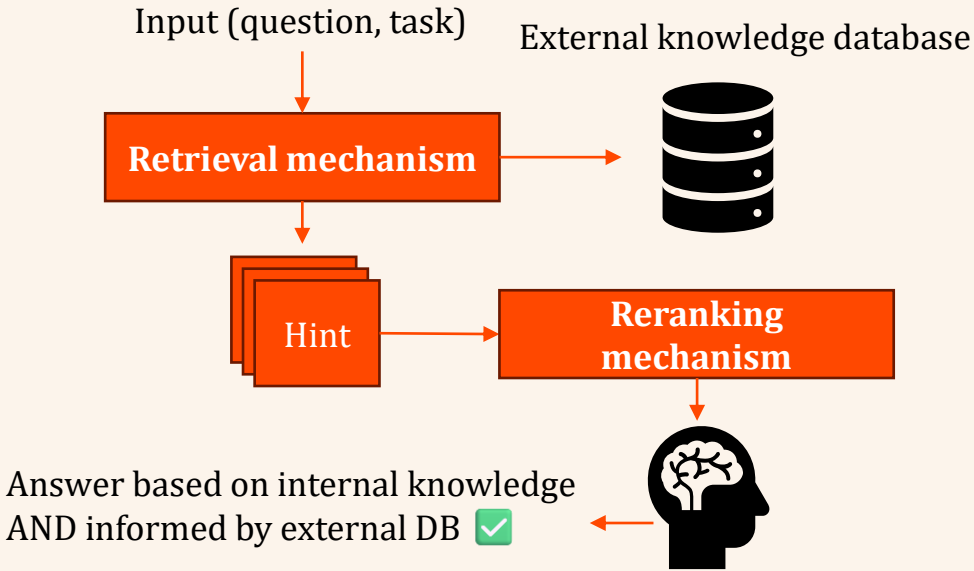
- **What:** LLMs (like ChatGPT) = trained on massive text corpora.
- **How:** Can interpret/generate language → natural interaction.
- **In our work:** Engineers query technical systems directly.



Retrieval-Augmented Generation (RAG) & Reranking

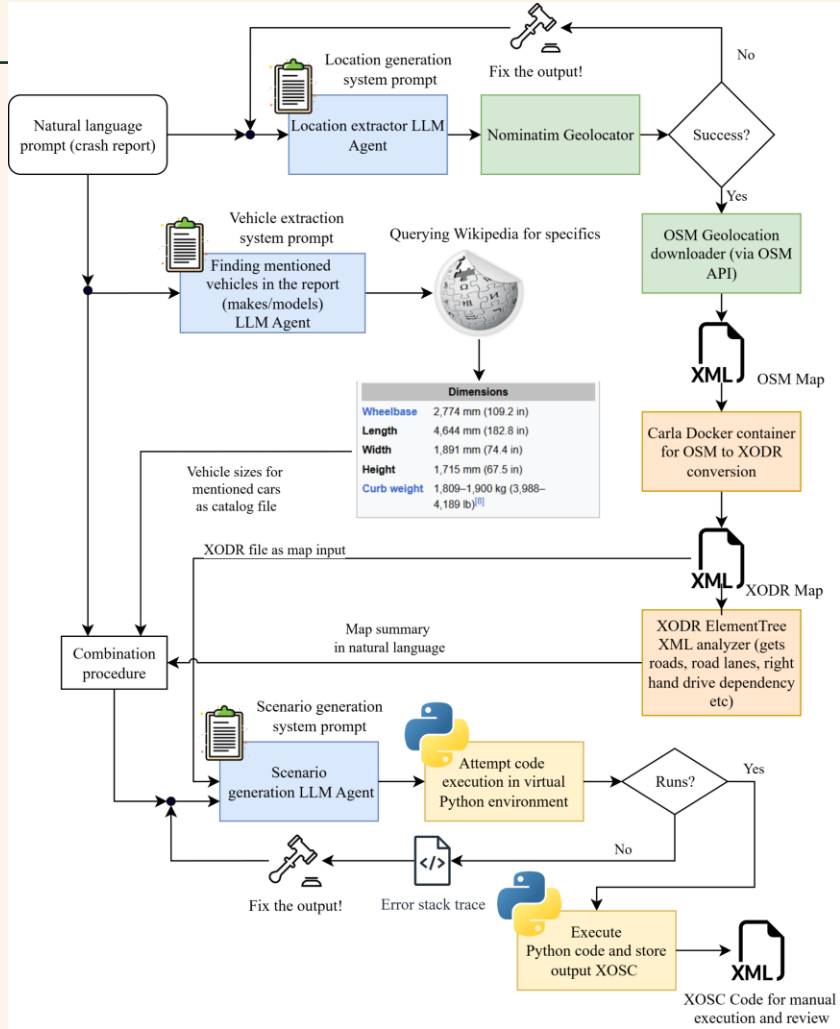
- **Problem:** LLMs may “hallucinate”.
- **RAG:** Retrieves facts from trusted docs/specs.
- **Reranking:** Picks most reliable information chunk→ improves answers.





Contribution 4 - core

- **Problem:** Scenario generation from natural language - error-prone and unreliable.
- Use **LLMs + retrieval + reranking** to translate text reports into simulation-ready scenarios.
- **Result:** Produces **executable OpenSCENARIO code** with higher precision and reduced hallucination.
- **Contribution to RG:** Brings *trustworthiness* to AI-assisted E2E workflows.



Future Work

- **Scaling up verification:** Extend abstraction framework to larger, non-linear networks with industrial-size datasets.

- **Sharper uncertainty:** Adaptive conformal prediction to tighten intervals while preserving guarantees.

- **Real-time integration:** Embed forecasting tools into live industrial systems (beyond offline use).

- **Trustworthy AI assistants:** Advance RAG + reranking pipelines so engineers can query systems naturally *with well informed systems*.

Conclusion

- **Problem:** Industrial ML = powerful but fragile (black-box, slow, no guarantees).
- **Solution:** Four contributions spanning abstraction, uncertainty, forecasting, and natural-language interaction.
- **Impact:** Toward *trustworthy, explainable, efficient ML* for industrial-scale systems.