International Journal of Intelligent Information Systems
2025, Vol. 14, No. 5, pp. 88-95
https://doi.org/10.11648/j.ijiis.20251405.11

Research Article

Neural Network Axiomatic AGI Method for Solving Scientific Problems

Evgeny Bryndin*

Scientific Department, Research Center Nature Informatic, Novosibirsk, Russia

Abstract

Modern neural network methods combine work with an axiomatic mathematical description (laws, equations, invariants, logical rules) and the power of neural networks for learning from data, pattern recognition and differentiation through complex spaces. This combination produces systems that can learn from data, observe given laws and, as a result, make predictions, solve problems and even discover new hypotheses. Quality depends on the formulation of axioms and the presence of correct formulations, the complexity of scaling to very large axiomatic bases, trade-offs between the accuracy of fitting to data and compliance with laws, interpretation and verification of results. Modern neural network methods with an axiomatic mathematical description have better generalization and physical interpretability due to compliance with axioms, the ability to work with small data due to built-in laws and the ability to discover new dependencies within the framework of formalized rules. Theoretical principles and formal axioms set requirements for neural networks and their training so that solutions to scientific problems correspond to the laws of nature, invariances, data characteristics and other desired properties. Power: an axiomatic neural network tends to be accurately modeled given its sufficient complexity and large scientific data and knowledge. The author proposes a neural network axiomatic AGI method for solving scientific problems according to their formulations and developed systems of axioms.

Keywords

Neural Network Axiomatic Method, Solving Scientific Problems, Axiom Systems

1. Introduction

Neural network methods for solving scientific problems are currently becoming relevant [1-14]. Neural networks are capable of working with large and complex data: images, text, sound, multimodal data, time series, and others. Neural network models themselves extract useful representations from the data. One architecture can be applied to different tasks (transfer learning, fine-tuning). Rapid prototyping and process acceleration. Rapid iterations, automation of individual processes (generative, classification, forecasting tasks). Automation of complex tasks, improved personalization, new services. Development of infrastructure and ecosystem. Ready-made models, frameworks, pre-trained weights, data management, monitoring, and operation tools. Increased availability of data and computing power. Cloud, optimization for training. Accuracy and quality of solutions on complex patterns that are difficult to describe with explicit rules. Automation of intellectual tasks: text understanding, image recognition, content generation, recommendations. Personalization and adaptation to the user based on large volumes

*Corresponding author: bryndin15@yandex.ru (Evgeny Bryndin)

Received: 27 August 2025; **Accepted:** 9 September 2025; **Published:** 9 October 2025

of data. Acceleration of decision-making and reduction of manual labor costs.

Neural networks require a high-quality data set, the data must be representative and ethically correct to obtain transparency and explainability of decisions. There are risks of validation and error generation: biases, malicious content, unpredictable behavior, privacy issues and user data regulation, confidentiality. The need to monitor and update models after deployment: data drift, changing conditions.

The task of solving scientific problems is really suitable for neural networks. There are data and patterns that recognize them better than traditional methods. Let's consider examples by domain:

1) NLP and text understanding: chatbots, summarization, translation, search in large corpora of documents.
2) Visual tasks: object recognition and clustering, medical imaging, autonomous systems.
3) Bridge and multimodal tasks: combining text and images, video processing, generative models. - Time series and forecasting: financial markets, energy consumption, predictive maintenance.
4) Recommender systems: personalization of content and products.
5) Science and engineering: molecular design, materials modeling, acceleration of discoveries.

A brief description of the neural network method is as follows. Start with a clear description of the problem and goals of the project. Collect and clean the data, evaluate the quality of the annotations. Choose the architecture based on the problem and available resources: transformers for text/multimodal data, for images, temporal networks for series. Try pre-trained models: fine-tuning on your problem is often more effective and less expensive. Think through the infrastructure: data processing, training, version control of models, inference monitoring. Plan for assessment and security: test for overlapping risks, implement protection against malicious content. Ensure ethics and transparency: documentation, explainability where critical. Develop a support plan: data updates, retraining, how to cope with drift.

Let's review the main directions and characteristic approaches that are used for axiomatic-oriented application of neural networks to science.

1) Physics-informed and differentiable approaches (working with axioms of physics and mathematics):
    1) Physics-informed neural networks: neural networks are trained so that their output satisfies the equations and laws of physics (Putin's, mass-energy balances, etc.). This allows solving direct and inverse problems, working within the framework of given axioms. Particularly useful for problems that cannot be solved analytically or with limited experimental data.
    2) Hamiltonian and Lagrangian neural networks: network models that preserve the structural properties of mechanics (energy, symplectic structure, conservation of momentum). This ensures better reproduction of physics and stability on long integrations.
    3) Invariant and equivariant neural networks: embed symmetries (e.g. spatial/temporal symmetries) as axioms of the model. This reduces the need for data and ensures that the underlying laws are respected when generalizing.

2) Discovery of laws and equations (where axioms are the basis of physics and mathematics):
    1) List and regularized equation search techniques: a combination of neural networks and symbolic or numerical recovery of laws. Examples of problems: identifying active dynamics equations from data, checking consistency with dimensionality and symmetries, finding hidden variables.
    2) Symbolic regression + physical constraints: finding expressions that are consistent with the data and known physical laws. Often used in combination with partitioning by invariants (e.g. dimensionality) to narrow the solution space.
    3) There are approaches that allow first training a neural model on data, and then by restricting the axioms they try to bring the resulting formula to a clear mathematical form.

3) Neuro-logical and neuro-symbolic methods (neuro-symbolic robotization of inferences from axioms):
    1) Neural theorem provers and neuro-symbolic reasoning engines: a neural network helps to select applicable rules/lemmas and the direction of proof within a given theory. Then the symbolic mechanism checks the correctness of the inference steps. Suitable for the tasks of automating theorems, formalizing scientific hypotheses and checking evidence against axioms.
    2) DeepProbLog approaches: combining neural networks with logic and probabilistic logic to handle uncertainty and ambiguity in data and axiomatic reasoning. Good for tasks that require partially informal knowledge and probabilistic assessment of inferences.

4) Differentiable Simulation and Programming (Differentiable Modeling):
    1) Differentiable physics engines and differentiable programming: create models that can be trained via gradients to produce predictions consistent with the laws of nature. Applicable to material, climate, and biophysical modeling problems.
    2) Differentiable solutions to modeling problems: integrate ML and numerical simulations, where axioms specify the formal context of the solution (e.g., existence of solutions, stability).

5) Discrete Identification of Dynamic Laws and Problems in the Natural Sciences:
    1) extensions: sparse representation of plausible dynamics through a combination of functions and their

coefficients. Often used in combination with physical assumptions (invariants, symmetries) to derive the equations of motion.

2) Autoframeworks and intelligent methods for discovering dependencies where data is limited: AI Feynman and similar approaches explore physically reasonable forms of dependencies and can suggest conservation laws, dimensions, and simple expressions.

6) In what problems are such methods used:

1) Physics and engineering: solving and identifying equations for plasma, fluid, materials, heat transfer, acoustics, aerodynamics.
2) Climatology and geoscience: climate models, wave processes, substance transfer, conservation of energy and mass in models.
3) Chemistry and materials science: reaction modeling, quantum-mormon approximation, finding effective descriptions of chemical processes.
4) Biology and medicine: biochemical networks, reaction kinetics, pathogen propagation, energy conservation in biomechanical systems.
5) Mathematics and computational science: automation of proofs, hypothesis testing and search for formal patterns.

How axiomatic neural network systems are built in practice:

1) Step 1. Formalize axioms and laws: determine which equations, conservations, symmetries or logical rules must be observed.
2) Step 2. Choose an approach that fits the problem: Hamiltonian/Lagrangian for mechanics, neural logic for reasoning, or for equation inference.
3) Step 3. Prepare data: simulations, experimental observations, or a combination of both; with limited data, focus on axioms and invariants.
4) Step 4. Define the learning objective: data approximation + penalties for deviation from axioms/laws + regularization.
5) Step 5. Verification and interpretability: check for energy conservation, symmetries, stability; analyze the obtained patterns on independent data.
6) Step 6. Incremental evolution: supplement the axioms as needed or introduce new hypotheses that can be tested by experiments.
7) Step 7. Tools and infrastructure: PyTorch for neural networks, libraries for differentiable modeling, tools for automatic differentiation of equations, as well as frameworks for neuro logic (DeepProbLog, neuro symbolic approaches).

Neural network axiomatic approaches to problem solving are considered in published works [15-18]. In the article, the author proposes a neural network axiomatic AGI method for solving scientific problems according to their formulations and developed systems of axioms.

2. Training Neural Networks to Generate Axiom Systems

You can train a neural network to propose axiom systems as a set of formulas, then use automatic provers to check the correctness and necessity of these axioms. This requires a hybrid approach: a neural network generates hypotheses-axioms, and a formal theorem mechanism checks and filters them. Let's consider a practical plan for implementing a project to train a neural network to generate axiom systems.

1) Determine the type of theory and the formalization language and the axiom system:

1) Set of axioms (specific formulas) or axiom schemes (parameterized schemes, for example, Axiom schemas).
2) System requirements: soundness with respect to a given semantics, independence of axioms, minimality (a small number of axioms), completeness/sufficiency for deriving theorems.

2) Data sources and representation of axioms:

1) Training data:
1) Formal theory repositories: Metamath, Mizar, Lean/Coq libraries (for examples of existing axiom systems).
2) Examples of ready-made axiom sets: classical Hilbert axes for propositional and predicate logic, axioms, arithmetic, etc.

2) Representation format:
1) uniform formula syntax and explicit axiom schemes.
2) possibly format for compatibility with existing automatic proof systems.

3) What we return to the neural network:

- specific axiom formulas, or sets of axioms/schemes, possibly with annotation (what type of axiom is this, what language does it belong to).

3) Model architecture:

1) Basic idea: seq2seq/Transformer, which receives a description of goals and produces a set of axioms (or one axiom at a time). You can consider the following options:
1) generating a set of axioms (multiple inference).
2) generating axioms by template: the neural network fills in the gaps within a given axiom scheme (a good way for an initial prototype).
3) graph neural network for representing formulas as trees/graphs and generating via the context of the theory.

2) Integration with the prover:

- after generation, automatically send axioms to an external automatic prover (Prover9, Vampire, E, Z3, Lean/Coq, etc.) to check the derivability of goals, check the consistency and independence of axioms.

3) Training and pretraining:

1) pretraining on well-known examples of axioms and

their applications.
2) additional training in a specific area/language (personalization for the task).

4) How to train a neural network:
1) Approaches: - B-supervised learning: create pairs (context/task, set of axioms). The context can be a description of the theory or a list of goals.
2) Few-shot or prompting: use large language models to propose axioms on the fly, then filter and check using formal methods.
3) Reinforcement learning with feedback from the prover: the agent is rewarded for axioms that allow proving given theorems, without contradictions.
4) Hybrid search + learning: the model proposes candidate axioms, then the proof system finds proofs or detects contradictions; based on this, it adjusts itself [19].
5) What to train:
1) axiom formulas in their original form.
2) axiom schemes (parameterized) with parameters specified.
3) possibly highlighting the role of axioms: tautology, invariant, inductive scheme, etc.

5) Verification and filtering:
1) What must be checked automatically:
1) soundness: all axioms, at least, do not contradict basic theoretical principles; for known formal languages, one can limit oneself to axioms that are instances of known safe schemes.
2) consistency: it is impossible to derive a contradiction from axioms (for example, to prove both P and ¬P from axioms). Use a theorem system prover for subject verification.
3) independence: try to derive each axiom from the others. If it is possible to prove the same without it, it is probably redundant.
4) completeness/need: to what extent do axioms provide the derivation of the required set of theorems.
2) Tools:
1) Automatic provers: Vampire, E, Prover9, Z3, Lean, Coq (for existence checks/direct deductions).
2) Metamath parsers and converters for a unified format.
3) Verification cycle: the neural network proposes axons; the prover verifies; if something is broken, it is filtered and returned for revision.

6) Evaluation metrics:
1) Proof power: which theorems can be proved using the resulting system.
2) Consistency: absence of contradictions within the system.
3) Axiom independence: proportion of independent axioms.
4) Minimal efficiency: how many axioms are needed for the same theory.
5) Generative accuracy: how well the generated axioms correspond to standard formulations (if the goal is to reproduce known sets).

Proof time: the speed of proving theorems.

7) Simple starting experiment (example on propositional logic):
1) Goal: learn to propose a set of 3-4 axioms of a Hilbert-like system.
2) Axioms (example for illustration):
1) A1: $p \to (q \to p)$
2) A2: $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
3) A3: $(\neg p \to \neg q) \to (q \to p)$
4) Inference rule: Modus ponens (p and $p \to q$ imply q).
3) How to test:
1) feed these axioms into a propositional logic prover; use a propositional logic prover (e.g. Prover9) to check that the desired theorems can be proved.
2) test independence: remove one axiom and check that some theorems are no longer derivable.
3) This will give you a working prototype: the neural network learns to formulate axioms that are compatible with the prover.

8) Practical project roadmap:
1) Step 1. Decide on the language and theory. Define a set of theories to start with (e.g. propositional logic and basic predicate logic).
2) Step 2. Collect a dataset of axioms and example theorems (Metamath, etc.). Bring the data to a uniform format.
3) Step 3. Choose a formula representation and set up tokenization.
4) Step 4. Implement a basic model: Transformer-encoder-decoder, which receives a description of the problem/theory language as input and generates one or more formula-axioms.
5) Step 5. Integrate with an automatic prover to check the generated axioms. Gradually improve filtering.
6) Step 6. Evaluation and debugging: measure provable power, independence, consistency. Iterate. - Stage 7. Expansion: add new theories, move to axiom schemes independent of existing basic theories, etc.

9) Important things to remember:
1) A neural network cannot guarantee formal correctness automatically. Be sure to use formal checking tools.
2) Formal systems may not be unique: many different axiom systems lead to one theory; the goal is to find a useful, minimal and independent system.
3) Work in a sawtooth learning cycle: neural network proposal of axioms + strict verifications by proofs + data and goal refinement.

A specific set of tools is selected for the task (in what language to formalize, what proofs to use, what data to collect) [20-21].

3. Developing System of Axioms to Solve a Problem by Its Formulation

Let's consider practical steps to develop a system of axioms to formalize and solve a problem by its formulation.

1) Define the formalization language:
   1) Choose a signature L: a set of non-finding symbols (constants, functions, relations) and their arities.
   2) Determine what will be members of the models: what objects will be the subject of reasoning.
   3) Include equality (usually considered as part of the language).
2) Define target models and semantic interpretation:
   1) What classes of objects should be models (e.g. sets, graphs, groups, numbers, etc.)?
   2) What exactly do you want to prove or disprove within this theory?
3) Introduce definitions (if necessary):
   1) Introducing definitions through axioms helps keep the formalization compact.
   2) Use definitions as abbreviations within the theory to avoid overloading the set of axioms.
4) Provide basic axioms:
   1) Axioms should capture the "essence" of the problem and the properties of objects that are considered true.
   2) Frequently used:
      1) axioms of order (reflexivity, transitivity, antisymmetry);
      2) axioms of operations (associativity, existence of unit/inverse);
      3) principles of existence (existence of objects that have properties, such as zero or unit).
      4) Use axiom schemes where an infinite set of analogous statements is needed (e.g. Peano axioms).
5) Ensure consistency:
   1) Whenever possible, provide a model that satisfies your axioms (model proof). This helps to avoid overly strong or contradictory axioms.
   2) Check the independence of axioms (no axiom should be deducible from the others, if possible).
6) Define the rules of inference:
   1) Choose a system: natural inferences, computational proofs, or Hilbert/nat. inference system, sequential formalism, etc.
   2) Ensure SOUNDNESS: all theorems being deduced are indeed true in the models of the theory. - If possible, take into account completeness with respect to the chosen models, but remember that for some theories it is unattainable (Gödel).
7) Test with examples and basic theorems:
   1) Try to derive several obvious consequences from the axioms.
   2) Provide models if some desired properties are not achieved.
8) Iterative editing:
   1) Add/remove axioms as needed to:
      1) maintain consistency;
      2) reduce redundancy;
      3) protect against inconsistencies;
      4) keep the required inductance/power of the theory.
   2) Strive for minimalism: less is not worse, but it is easier to establish and verify.
   3) Axioms should not be redundant: try to achieve independence of axioms.
   4) Separate purely theoretical axioms and definitions. Definitions can be kept separate to keep the axioms compact.
   5) If you are formulating a problem in a specific area, focus on existing standard theories (group, ring, partially ordered sets, graphs, etc.) this will give ready-made templates of axioms.
   6) Do not forget about limitations: not all problems can be fully formalized or solved by means of axioms. Gödel reminds us of possible incompleteness.

Example 1. Axioms for a group (simplified version)

1) Language L: one binary operation •, constant e.
2) Axioms:
3) Associativity: $\forall a\, \forall b\, \forall c\ (a \bullet (b \bullet c) = (a \bullet b) \bullet c)$
4) Unit: $\forall a\ (e \bullet a = a \ \wedge\ a \bullet e = a)$
5) Invertible element: $\forall a\, \exists b\ (a \bullet b = e \ \wedge\ b \bullet a = e)$
6) Additional properties can be added as needed.

Example 2. Axioms for a partially ordered set (≤)

1) Language L: binary relation ≤.
2) Axioms: - Reflexivity: $\forall x\ (x \le x)$
3) Antisymmetry: $\forall x\, \forall y\ ((x \le y \ \wedge\ y \le x) \to x = y)$
4) Transitivity: $\forall x\, \forall y\, \forall z\ ((x \le y \ \wedge\ y \le z) \to x \le z)$
5) Three axioms establish the structure of a partial order.

Additional properties can be added [22-26].

4. Neural Network Solution to a Problem Based on Its Formulation and the Developed System of Axioms

Let's consider a method of combining formalism and neural network methods: teaching a neural network to understand the formulation of a problem, operate the developed system of axioms and offer a solution or proof, which can then be verified by a formal interpreter [27]. Let's consider the method in the form of practical steps and recommendations.

1) Clarification of the problem and axioms:
   1) Define the goal: prove a theorem, build a solution to the problem, find an optimal plan of action, etc.
   2) List of axioms: which formulas are considered true without proof? What inference rules are allowed?
   3) Input and output format: how are the initial data specified, what steps are allowed, how to record the completeness of the solution.
   4) Correctness boundaries: is strict proof (formal verification) necessary or is correctness in the probabil-

istic sense sufficient?

2) Formalization for the neural network:
   1) What exactly will the neural network predict:
      1) sequence of proof steps (sequence of formulas)
      2) choice of the next inference rule - local transformations of formulas (transformation for formulas)
      3) expansion of the state space (generation of new statements)
   2) How the input will be represented:
      1) tokenized formulas as sequences
      2) graph representations of formulas (trees, expression graphs)
      3) combined migration: formula graph + axioms context
   3) Where the verifier will be placed:
      1) formal verification of each step according to axioms and inference rules
      2) deterministic verification at each step until the end of the proof
3) Data and training:
   1) Dataset:
      1) ready-made sets of proofs in your axiom system: synthetically generated examples: apply axioms to basic types of statements to generate new problems
      2) data with partially filled proofs (hints) for teacher-student learning
   2) Target signals:
      1) probability of each possible inference step
      2) or a specific sequence of correct steps until the proof is complete
      3) reward for a complete correct inference
      4) Model architecture:

- Variants of neural network architectures:
- Transformer: good for sequences of proof steps
- Graph Neural Network (GNN): useful for a structural representation of formulas and the dependencies between them
- Combined approaches: graph encoder + transformer-decoder
- Basically, the model generates the next inference step; the verifier checks for correctness. This allows partly separating the "thinking" of the neural network from the strict logic.
- Search mechanism: the neural network can supervise the search component (e.g., a neural network assistant for Monte Carlo tree search) or work as an auto-generator of a set of candidates, which are then verified.

5) Training and optimization:
   1) Pre-training: train on a large set of correct proof steps (supervised) if there are enough examples.
   2) Relevant learning: the role of the model is to propose the most probable steps, and the reward is the successful completion of the proof.
   3) Verification as part of learning: include penalties in padding learning for steps that do not pass verification.
   4) Curriculum learning: start with simple problems and gradually move to more complex ones.
6) Verification and correctness:
   1) Built-in formal verifier: each proposal/step is checked against the system of axioms and inference rules.
   2) Separation of responsibilities: the neural network produces a candidate solution; verification is done by the summary logic/verifier.
   3) Possibilities of neural network errors: Gödel paradoxes and finite completeness; therefore, zero-guaranteed correctness without verification. Include strict verification at each step.
7) Evaluation and metrics:
   1) Percentage of successfully proven problems.
   2) Average proof length and time to prove.
   3) Robustness to new problems (generalization to more complex examples).
   4) Provability of the solution by the verifier: the proportion of steps that pass the check at each step.
   5) Readability and "understandability" of the proof (for a human reviewer).
8) Practical notes:
   1) Without a formal verifier, the neural network may produce incorrect steps. Always keep the Verifier as an integral part of the solution.
   2) In complex axiomatic systems, completeness and robustness to changes in the axioms require caution: the neural network may "read" the axiom in different ways, so uniform normalization of formulas is important.
   3) Start with simple logic/arithmetic, then move on to more complex systems (e.g., geometry, mathematical logic, programming, etc.).
9) An example of a simplified scenario (for illustration):
   1) Problem: in propositional logic, prove: from A and (A -> B) follows B.
   2) Axioms and rules: modus ponens (MP) as a rule of inference; basic forms of expressions in the form A, A->B, B.
   3) Representation: formulas are tokenized as sequences, the proof step holds references to the statements.
   4) Role of the neural network: generates the next step of the proof, for example:
      1) Fix premise A
      2) Fix the premise A -> B
      3) Apply MP to steps 1 and 2, yielding B
      4) Verifier: checks that step 3 is indeed an application of MP to A and (A>B) yielding B;

steps 1-4 form a correct proof.
Proof complete.

5. Conclusion

Neural networks are becoming very relevant in science because they allow processing huge amounts of data, finding

complex dependencies and speeding up calculations where traditional methods are slow or unsuitable. They complement theory and experiments well, opening up new ways of discovery and optimization in a wide variety of subject areas. Why neural networks are becoming very relevant in science and what tasks they help with:

1) Big data processing: experiments, simulations and sensors generate huge amounts of data that are difficult to analyze using traditional methods.
2) Indirect and complex dependencies: systems are often nonlinear and multidimensional; neural networks are able to identify hidden relationships without an explicitly specified model.
3) Acceleration of calculations: replace expensive simulations with emulators/flows, speeding up the design of materials, chemical formulations and process optimization.
4) Automation and design-oriented science: models help formulate hypotheses, predict results and suggest the most informative experiments.
5) Interdisciplinary integration: from physics and chemistry to biology, climatology and materials science, where the combination of data and models accelerates discovery.

Key areas and example tasks:

1) Emulators and surrogate models: acceleration of expensive computations, climate and biochemical models.
2) Physics-oriented neural networks and physically constrained models: solving and approximating partial differential equations, field reconstruction problems.
3) Neural potentials and molecular design: modeling of material and molecular properties, bond energy prediction, generative approaches to substance design.
4) Generative models and unsupervised learning: design of materials and molecules, autonomous hypothesis search, new structures and formulations.
5) Biological data analytics: analysis of single-cell data, protein structure and dynamics, function prediction.
6) Climate and energy: emulators of complex climate models, optimization of energy systems, forecasting sustainability scenarios.
7) Experimental design and active learning: selecting the most informative experiments and sensors, reducing data costs.
8) Rapid prediction of materials and molecules with target properties, which simplifies the search for experiment candidates and reduces costs.
9) Acceleration of solving complex equations and modeling processes due to trainable emulators. - Improving the structure of explainable results through physics and data: physically motivated architectures and penalties for violating physical laws.
10) Advances in bioinformatics and structural biology (e.g. protein structure prediction, function analysis).
11) Application in climatology and energy for rapid scenario assessment and system optimization.

The relevance of neural networks in science is growing due to the availability of big data, the need to accelerate computations and the need to identify complex dependencies. Success requires a combination of neural networks with subject matter expertise. In the future, it is expected that hybrid neural network AGI methods for solving problems according to their formulations and developed formalized systems will be strengthened, which will contribute to the acceleration of scientific research and scientific and technological progress on the international platform.

Abbreviations

AGI	Artificial General Intelligence
AI	Artificial Intelligence
ML	Machine Learning
NLP	Naturallanguage Processing

Author Contributions

Evgeny Bryndin is the sole author. The author read and approved the final manuscript.

Conflicts of Interest

The author declares no conflicts of interest.

References

[1] Dimitrios Soudris. Reducing Memory Fragmentation with Performance-Optimized Dynamic Memory Allocators in Network Applications. Springer eBooks. 2024.

[2] Clifford Lau. Office of Naval Research contributions to neural networks and signal processing in oceanic engineering. IEEE Journal of Oceanic Engineering. 2024

[3] Mary Lenard. The Application of Neural Networks and a Qualitative Response Model to the Auditor's Going Concern Uncertainty Decision. Decision Sciences. 2024.

[4] Mark Lawley. A Neural Network Integrated Decision Support System for Condition-Based Optimal Predictive Maintenance Policy. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans. 2024.

[5] Feyzullah Temurtas. Harmonic Detection Using Feed Forward Artificial Neural Networks. Sigma. 2024.

[6] Pietro Vecchio. Wind energy prediction using a two-hidden layer neural network. Communications in Nonlinear Science and Numerical Simulation. 2024.

[7] Carlos Uriarte. Solving Partial Differential Equations Using Artificial Neural Networks. 2024. 125 p. URL: https://addi.ehu.es/handle/10810/68335

[8] Yalçın Yılmaz. Multi-purpose neuro-architecture with memristors. 11th IEEE International Conference on Nanotechnology.2025.

[9] Xiaogang Gao. Estimation of physical variables from multichannel remotely sensed imagery using a neural network: Application to rainfall estimation. Water Resources Research. 2025.

[10] Vladimir Krasnopolsky. Some neural network applications in environmental sciences. Part I: forward and inverse problems in geophysical remote measurements. Neural Networks.2025

[11] Edgar Torres, Jonathan Schiefer. Adaptive Physics-informed Neural Networks. Transactions on Machine Learning Research (03/2025).

[12] Evgeny Bryndin. Unambiguous Identification of Objects in Different Environments and Conditions Based on Holographic Machine Learning Algorithms. Britain International of Exact Sciences Journal (BIoEx-Journal). Volume 4. Issue 2. 2022. pp.72-78.

[13] Evgeny Bryndin. Theoretical Foundations of Neural Network Integration of System Software Modules. 2025. In press. *Software Engineering.*

[14] Evgeny Bryndin. Network Training by Generative AI Assistant of Personal Adaptive Ethical Semantic and Active Ontology. International Journal of Intelligent Information Systems Volume.14, Is.2. 2025. pp. 20-25. https://doi.org/10.11648/j.ijiis.20251402.11

[15] Wen Zhang, Juan Li, Xiangnan Chen, Hongtao Yu, Jiaoyan Chen. Neural Axiom Network for Knowledge Graph Reasoning. 2022. URL: https://github.com/JuanLi1621/NeuRAN

[16] Markus Pantsar. Theorem proving in artificial neural networks. European Journal for Philosophy of Science, Volume 14, article 4, (2024).

[17] Levin Hornischer, Zoi Terzopoulou. (2025). Learning How to Vote with Principles: Axiomatic Insights Into the Collective Decisions of Neural Networks. Journal of Artificial Intelligence Research (83), Article 25, 44 pages. https://doi.org/10.1613/jair.1.18890

[18] Fanghua Pei, Fujun Cao, Yongbin Ge. A Novel Neural Network-Based Approach Comparable to High-Precision Finite Difference Methods. Axioms, 14(1), 2025. https://doi.org/10.3390/axioms14010075

[19] Jacek Zurada. Self-Organizing Neural Networks Integrating Domain Knowledge and Reinforcement Learning. 2025, IEEE Transactions on Neural Networks and Learning Systems.

[20] J. Li and et al. Neural Axiom Network for Knowledge Graph Reasoning. Semantic Web, 2022. P. 1-15.

[21] Borko Đ. Bulajic. Application of Machine Learning and Optimization Methods in Engineering Mathematics. Journal Axioms. 198 p.

[22] Kaile Su. Symbolic manipulation based on deep neural networks and its application to axiom discovery. IEEE International Joint Conference on Neural Network. 2017.

[23] Maria Astafurova. Developing Physical Axiomatics: Results and Outcomes. EPJ Web of Conferences 224, 06010 (2019).

[24] Mohamed Y. Syed. An Overview of Axioms. Physical Mathematics, (2022) Volume 13, Is. 3.

[25] Dan Christensen. How an axiomatic system is made? Mathematics, 2024. URL: https://math.stackexchange.com/questions/770328/how-an-axiomatic-system-is-made

[26] Simon Thompson. Adding the axioms to Axiom: Towards a system of automated reasoning in Aldor. Updated: 30 Mar 2025. URL: https://fastercapital.com/content/Axioms--Understanding-the-Role-of-Axioms-in-POA-Proof-Assignments.html

[27] Sebastian Wanker, Jan Pfister, Andrzej Dulny, Gerhard Götz, Andreas Hotho. Identifying Axiomatic Mathematical Transformation Steps using Tree-Structured Pointer Networks. Transactions on Machine Learning Research (01/2025). P. 1-30.