American Journal of Mathematical and Computer Modelling
2025, Vol. 10, No. 4, pp. 110-120
https://doi.org/10.11648/j.ajmcm.20251004.11

Research Article

Neural Network Axiomatic Solver Coaching AGI Method for Solving Scientific and Practical 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 solver coaching AGI method for solving scientific and practical problems according to their formulations and developed systems of axioms.

Keywords

Axiomatic Solver Coaching Method, AGI, Neural Network Solution, Scientific Problems, Practical Results

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 [5, 6]. Neural network models themselves extract useful representations from the data. One architecture can be applied to different tasks (transfer learning, fine-tuning) [7]. Rapid prototyping and process acceleration. Rapid iterations, automation of individual processes (generative, classification, forecasting tasks) [8]. 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 [9]. Cloud, GPU/TPU, 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, recom-

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

Received: 2 September 2025; **Accepted:** 13 September 2025; **Published:** 9 October 2025

mendations. Personalization and adaptation to the user based on large volumes 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, GDPR, 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 [10-14]. 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.

The neural network method begins with a clear description of the task and goals of the project. 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, CNN/ViT 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 consider 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):
   a. Physics-informed neural networks (PINNs): neural networks are trained so that their output satisfies the equations and laws of physics (Putin's PDE/ODE, 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.
   b. 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.
   c. 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):
   a. 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.
   b. 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.
   c. 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):
   a. 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.
   b. NSM/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):
   a. 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.
   b. 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:
   a. SINDy and its 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.

b. 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:

a. Physics and engineering: solving and identifying equations for plasma, fluid, materials, heat transfer, acoustics, aerodynamics.

b. Climatology and geoscience: climate models, wave processes, substance transfer, conservation of energy and mass in models.

c. Chemistry and materials science: reaction modeling, quantum-mormon approximation, finding effective descriptions of chemical processes.

d. Biology and medicine: biochemical networks, reaction kinetics, pathogen propagation, energy conservation in biomechanical systems.

e. 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: PINN/hydride with PDE, Hamiltonian/Lagrangian NN for mechanics, neural logic for reasoning, or SINDy 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/JAX for neural networks, libraries for differentiable modeling, tools for automatic differentiation of equations (for PINNs), 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 a Neural Network with New Axioms

Training a neural network with new axioms involves introducing formal axioms into the training process or into the model's reasoning so that it follows them when making decisions, predictions, and generalizations. Below are the main methods and practical steps.

Initial parameters:

1) Domain (language, images, texts, KG, etc.)
2) Axioms (formal notation)
3) Data and their quality during tests.

Main methods:

1) Soft constraints at the training stage: a warning is added to the loss function when the axioms are violated.
2) Neuro-symbolic approaches: a hybrid architecture is used, where part of the knowledge is laid down in the form of rules, and part is processed by the neural network.
3) Inductive or differentiable reasoning: the model is trained not only to predict, but also to prove conclusions based on the axioms.
4) Working with knowledge graphs (KG) and ontologies: indices and embeddings on OWL/RDFS axioms are respected during training.

The choice depends on the task and the available data.

Practical steps. Popular approaches and frameworks:

1) Soft constraints via differentiable logic:

a. Logic Tensor Networks (LTN) — introduce vector representation of predicates and the discipline of logical inference as a differentiable problem.

b. Neural Theorem Prover (NTP) — trainable logical 3D sphere for differentiable proof.

2) Deep Probabilistic/Neural integration:

DeepProbLog — combines neural modules with ProbLog logic; useful if you need probabilistic inferences with axioms.

3) Inductive-proof networks:

Differentiable logic and ground rules — you can train a model so that it satisfies the rules through warnings in the loss.

4) Knowledge graphs and ontologies:

TensorLog, Nachos-like approaches — generalization of KG embeddings under logical axioms.

Step-by-step implementation in practice.

1) Formalize axioms:

Translate domain rules into a form that can be made differentiable.

Examples:

a. All people are mortal: $\forall x\ \text{Human}(x) \rightarrow \text{Mortal}(x)$

b. If y is a child of x and x is a human, then y is also a human:

$\forall x \forall y\ (\text{Parent}(x,y) \wedge \text{Human}(x)) \rightarrow \text{Human}(y)$

c. Define a set of entities and predicates (Human, Mortal,

Parent, etc.).

2) Choose an approach and tool:
   a. If you need a simple addition of constraints to the training of a neural model, start with a soft-constraint method via LTN or a standard loss extension.
   b. If you want a hybrid of neuron and symbolics and probabilistic inference, use DeepProbLog.
   c. If the problem is highly reasoning-intensive and you want to train proofs, use Neural Theorem Prover or other differentiable instances.
   d. For KG-oriented problems, consider TensorLog or similar methods.
3) Define representations:
   a. Predicates: neural modules return the probability of truth of pred(x1, x2,...).
   b. Objects: define a set of constants (individuals) or use temporal primitive generation.
   c. Variable binding: if partial domain coverage is required, use subset sampling for quantifiers ∀. 4) Introduce axiom failure measure:
   d. For each axiomatic rule, compute the failure measure and add it to the overall loss function.
   e. Example of simple form of failure for axiom ∀x (Human(x) → Mortal(x)):
   f. Let H(x) and M(x) be the values of neural networks in [0,1] for each x.
   g. Axiom failure: L_axiom = mean over sample x from the derivative of function A→B, e.g. L = mean(H(x) * (1 - M(x))).
   h. Total failure: L_total = L_data + λ * L_axiom, where λ controls the strength of the constraint. 5) Training process:
   i. Train in the usual way with gradient descent, periodically test how well the axioms are met on validation/test.
   j. If the axioms constrain learning too much, reduce λ or use annealing (gradually increasing the weight of the axioms).
4) Evaluation and debugging:
   a. Check if axioms contradict each other.
   b. Test on problems that require inferences using axioms.
   c. Try different normalization options and different t-norms (for fuzzy logic) - this affects the behavior of the axioms.
   d. Monitor the performance and quality of predictions: the goal is not to destroy predictions, but to complement them with axioms.

Simple illustrative example:
   a. Problem: people and mortality. Axiom: ∀x Human(x) → Mortal(x).
   b. Representations: H(x) = probability that x is human; M(x) = probability that x is mortal.
   c. Losses:
   d. L_data: regular loss on data (e.g. cross-entropy on Human/ Mortal predicates, if applicable).
   e. L_axiom = mean_x [H(x) * (1 - M(x))].
   f. L_total = L_data + λ * L_axiom.
   g. Training: optimize L_total. If H(x) is high and M(x) is low, the axiom will warn the model and tighten M(x).

Recommendations for choosing:
   a. Start simple: add one or two soft constraints to a standard problem and see the effect.
   b. If the problem requires robust rule-based inference, try DeepProbLog or NTP.
   c. For scalability and working with KG, TensorLog approaches and generalizations towards neural embeddings are useful.

3. 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:
   a. Set of axioms (specific formulas) or axiom schemes (parameterized schemes, for example, Axiom schemas).
   b. 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:
   a. Training data:
      i. Formal theory repositories: TPTP (a set of theorem and axiom statements for logics of varying complexity), Metamath, Mizar, Lean/Coq libraries (for examples of existing axiom systems).
      ii. Examples of ready-made axiom sets: classical Hilbert axes for propositional and predicate logic, ZFC axioms, arithmetic, etc.
   b. Representation format:
      i. uniform formula syntax (e.g. prefix notation or LISP-like syntax) and explicit axiom schemes.
      ii. possibly a strict TPTP/FOF format for compatibility with existing automatic proof systems.
      iii. What we return to the neural network:
      iv. 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:
   a. 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:

i. generating a set of axioms (multiple inference).
ii. generating axioms by template: the neural network fills in the gaps within a given axiom scheme (a good way for an initial prototype).
iii. graph neural network for representing formulas as trees/graphs and generating via the context of the theory.

b. 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.

c. Training and pretraining:

i. pretraining on well-known examples of axioms and their applications.
ii. additional training in a specific area/language (personalization for the task).

4) How to train a neural network:

a. Approaches:

i. B-supervised learning: create pairs (context/task, set of axioms). The context can be a description of the theory or a list of goals.
ii. Few-shot or prompting: use large language models to propose axioms on the fly, then filter and check using formal methods.
iii. Reinforcement learning with feedback from the prover: the agent is rewarded for axioms that allow proving given theorems, without contradictions.
iv. Hybrid search + learning: the model proposes candidate axioms, then the proof system finds proofs or detects contradictions; based on this, it adjusts itself [19].

b. What to train:

i. axiom formulas in their original form.
ii. axiom schemes (parameterized) with parameters specified.
iii. possibly highlighting the role of axioms: tautology, invariant, inductive scheme, etc.

5) Verification and filtering:

a. What must be checked automatically:

i. 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.
ii. 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.
iii. independence: try to derive each axiom from the others. If it is possible to prove the same without it, it is probably redundant.
iv. completeness/need: to what extent do axioms provide the derivation of the required set of theorems.

b. Tools:

i. Automatic provers: Vampire, E, Prover9, Z3, Lean, Coq (for existence checks/direct deductions).
ii. TPTP/Mizar/Metamath parsers and converters for a unified format.
iii. Verification cycle: the neural network proposes axons; the prover verifies; if something is broken, it is filtered and returned for revision.

6) Evaluation metrics:

i. Proof power: which theorems can be proved using the resulting system.
ii. Consistency: absence of contradictions within the system.
iii. Axiom independence: proportion of independent axioms.
iv. Minimal efficiency: how many axioms are needed for the same theory.
v. Generative accuracy: how well the generated axioms correspond to standard formulations (if the goal is to reproduce known sets).
vi. Proof time: the speed of proving theorems.

7) Simple starting experiment (example on propositional logic):

a. Goal: learn to propose a set of 3-4 axioms of a Hilbert-like system.

b. Axioms (example for illustration):

i. A1: $p \to (q \to p)$
ii. A2: $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$
iii. A3: $(\neg p \to \neg q) \to (q \to p)$

c. Inference rule: Modus ponens (p and $p \to q$ imply q).

d. How to test:

i. 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.
ii. test independence: remove one axiom and check that some theorems are no longer derivable.
iii. This will give you a working prototype: the neural network learns to formulate axioms that are compatible with the prover.

8) Practical project roadmap:

i. Step 1. Decide on the language and theory. Define a set of theories to start with (e.g. propositional logic and basic predicate logic).
ii. Step 2. Collect a dataset of axioms and example theorems (from TPTP, Metamath, etc.). Bring the data to a uniform format.
iii. Step 3. Choose a formula representation and set up tokenization.
iv. 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.
v. Step 5. Integrate with an automatic prover to check the generated axioms. Gradually improve filtering.

vi. 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:
   a. A neural network cannot guarantee formal correctness automatically. Be sure to use formal checking tools.
   b. 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.
   c. 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].

4. 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:
   a. Choose a signature L: a set of non-finding symbols (constants, functions, relations) and their arities.
   b. Determine what will be members of the models: what objects will be the subject of reasoning.
   c. Include equality (usually considered as part of the language).
2) Define target models and semantic interpretation:
   a. What classes of objects should be models (e.g. sets, graphs, groups, numbers, etc.)?
   b. What exactly do you want to prove or disprove within this theory?
3) Introduce definitions (if necessary):
   a. Introducing definitions through axioms helps keep the formalization compact.
   b. Use definitions as abbreviations within the theory to avoid overloading the set of axioms.
4) Provide basic axioms:
   a. Axioms should capture the "essence" of the problem and the properties of objects that are considered true.
   b. Frequently used:
   c. axioms of order (reflexivity, transitivity, antisymmetry);
   d. axioms of operations (associativity, existence of unit/inverse);
   e. principles of existence (existence of objects that have properties, such as zero or unit).
   f. Use axiom schemes where an infinite set of analogous statements is needed (e.g. Peano axioms).
5) Ensure consistency:
   a. Whenever possible, provide a model that satisfies your axioms (model proof). This helps to avoid overly strong or contradictory axioms.
   b. Check the independence of axioms (no axiom should be deducible from the others, if possible). 6) Define the rules of inference:
   c. Choose a system: natural inferences, computational proofs, or Hilbert/nat. inference system, sequential formalism, etc.
   d. 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).
6) Test with examples and basic theorems:
   a. Try to derive several obvious consequences from the axioms.
   b. Provide models if some desired properties are not achieved.
7) Iterative editing:
   a. Add/remove axioms as needed to:
   b. maintain consistency;
   c. reduce redundancy;
   d. protect against inconsistencies;
   e. keep the required inductance/power of the theory.
   f. Strive for minimalism: less is not worse, but it is easier to establish and verify.
   g. Axioms should not be redundant: try to achieve independence of axioms.
   h. Separate purely theoretical axioms and definitions. Definitions can be kept separate to keep the axioms compact.
   i. 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.
   j. 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)

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

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

a. Language L: binary relation ≤.
b. Axioms: - Reflexivity: $\forall x\ (x \le x)$
c. Antisymmetry: $\forall x\, \forall y\ ((x \le y \ \wedge\ y \le x) \to x = y)$
d. Transitivity: $\forall x\, \forall y\, \forall z\ ((x \le y \ \wedge\ y \le z) \to x \le z)$
e. Three axioms establish the structure of a partial order.

Additional properties can be added [22-24].

5. 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 [25]. Let's consider the method in the form of practical steps and recommendations.

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

What exactly will the neural network predict:
   a. sequence of proof steps (sequence of formulas).
   b. choice of the next inference rule.
   c. local transformations of formulas (conversion to formulas).
   d. expansion of the state space (generation of new statements).

How the input will be represented:
   a. tokenized formulas as sequences.
   b. graph representations of formulas (trees, expression graphs).
   c. combined migration: formula graph + axioms context.

Where the verifier will be placed:
   a. formal verification of each step according to axioms and inference rules.
   b. deterministic verification at each step until the end of the proof.

3) Data and training:

Dataset:
   a. ready-made proof sets in your axiom system (if any: TPTP and other sets for theorems in logic) - synthetically generated examples: apply axioms to basic types of statements to generate new problems.
   b. data with partially completed proofs (hints) for training.

Target signals:
   a. probability of each possible inference step.
   b. or a specific sequence of steps until the proof is complete.
   c. reward for a fully correct inference (for RL learning).

4) Model architecture:

Variants of neural network architectures:
   a. Transformer: good for sequences of proof steps.
   b. Graph Neural Network (GNN): useful for a structural representation of formulas and the dependencies between them.
   c. Combined approaches: graph encoder + transformer-decoder.
   d. Essentially, the model generates the next inference step; the verifier checks for correctness. This allows us to partially define the thinking of the neural network.
   e. Search mechanism: the neural network can guide the search component (e.g. a neural network assistant for Monte Carlo tree search) or work as an auto-generation of a set of candidates, which are then verified.

5) Training and optimization:
   a. Pre-training: train on a large set of correct proof steps (supervised) if there are enough examples.
   b. Relevant learning: the role of the model is to propose the most probable steps, and the reward is the successful completion of the proof.
   c. Verification as part of training: include penalties in padding training for steps that do not pass verification.
   d. Curriculum learning: start with simple problems and gradually move to more complex ones.
6) Verification and correctness:
   a. Built-in formal verifier: each proposal/step is checked against a system of axioms and inference rules.
   b. Separation of concerns: the neural network produces a candidate solution; verification is done by the summary logic/verifier.
   c. 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:
   a. Percentage of successfully proven problems.
   b. Average proof length and proof time.
   c. Robustness to new problems (generalization to more complex examples).
   d. Provability of the solution by the verifier: the proportion of steps that pass the check at each step.
   e. Readability and understandability of the proof for a human.
8) Practical notes:
   a. Without a formal verifier, the neural network may produce incorrect steps. Always keep the Verifier as an integral part of the solution.
   b. In complex axiomatic systems, completeness and robustness to changes in axioms require clarity.
   c. Start with simple logic/arithmetic, then move on to more complex systems (e.g. geometry, mathematical logic, programming, etc.).
9) Example of a simplified scenario (for illustration):

a. Problem: in propositional logic, prove: from A and (A -> B) follows B.
b. Axioms and rules: modus ponens (MP) as a rule of inference; basic forms of expressions in the form A, A->B, B.
c. Representation: formulas are tokenized as sequences, the proof step holds references to the assertions.
d. Role of the neural network: generates the next step of the proof, for example:
   i. Fix the premise A.
   ii. Fix the premise A -> B.
   iii. Apply MP to steps 1 and 2, which yields B.
   iv. Verifier: checks that step 3 is indeed an application of MP to A and (A>B) and yields B; steps 1-4 form a correct proof.
   v. Proof complete.

Formalized systems for finding solutions using solvers

Formalized systems for finding solutions using solvers are formal languages and mechanisms that allow one to describe a problem as a search in a state space and automatically find sequences of actions that lead to a goal.

What does this include:

1) Formalization of the problem, what states exist, what actions can be performed, how transitions between states occur, how the goal and evaluation of actions are set.
2) State space, a graph or pseudo graph whose nodes are states and edges are actions.
3) Search, an algorithm or method that explores the state space and plots a path from the initial state to the goal state.
4) Additionally, heuristics, time/memory constraints, requirements for the optimality of the solution.

Main stages of formalization:

State-space search:

- Definition of the problem: S is a set of states, S0 is the initial, A(s) are admissible actions from s, T(s, a) is a transition to a new state, G(s) is a test of the goal, and an assessment of c(s, a) is possible on demand.

Objective: find a path s0 → s1 → … → sg, where G(sg) is true.

Planning:

1) Examples of formalizations: STRIPS, PDDL. Modeling the world through predicates and actions with preconditions and effects.
2) Implemented by intelligent digital twins for automated execution of sequences of actions. - Constraint problems (CSP):
3) Variables, value domains, a set of constraints between variables.
4) Solution by search with support for constraint propagation.

SAT/SMT:

- Transformation of the problem into a Boolean formula (SAT) or a theory with numerical elements (SMT) and its solution by special solvers.

Proof/logical proof:

- Find a proof of a theorem or deductive inference via resolution, natural conclusions, etc.

Game-based search for solutions:

- Strategy generation, decision tree, evaluation functions.

Structure of a typical formal problem (universal scheme):

1) S: set of states
2) S0 ⊆ S: set of initial states
3) A(s): set of admissible actions in state s
4) T(s, a) -> s': transition (next state)
5) G(s): goal test - (on demand) c(s, a): action evaluation

Goal: find a sequence of actions a1, a2,..., ak and a connected path through states leading to state s, where G(s) is true and with a minimum sum.

Popular search algorithms:

1) Uninformed: BFS, DFS, IDS (iterative depth constraint)
2) Informed: A* (f(n)=g(n)+h(n)), Greedy, IDA*
3) Properties: completeness (if finite space), optimality (for A* with admissible/consistent heuristics)

Heuristics:

1) admissible: do not overestimate the real value
2) consistent: $h(n) \leq c(n, a) + h(n')$ for transition $n \rightarrow n'$

Examples of formalizations with short examples:

1) 8-puzzle as a state-space problem
2) State: arrangement of 3x3 tiles (one empty cell)
3) Actions: move empty cell up/down/left/right
4) Goal: configuration according to a given pattern (e.g. 1 2 3 / 4 5 6 / 7 8 _)
5) Cost: 1 per move
6) Heuristic: number of tiles out of place, or Manhattan distance summed over all tiles
7) Ship route or route planning
8) State: current position/location and task status
9) Action: move along route, perform tasks
10) Goal: reach target location or perform set of tasks
11) CSP — judging by variables
12) Example: sports betting schedule where each slot is assigned a timestamp and a conflict constraint.

Practical issues and choice of formalism:

1) Which problem is more convenient to formalize? Do you need optimality of the solution or any solution is enough?
2) How big is the state space? Are there natural heuristics?
3) Do you need integration with logic/proofs or is a constraint enough?
4) What tools are available: SAT/SMT solvers, planners (Fast Downward), CSP solvers, libraries for A*, etc.

Useful resources:

1) Formalisms and algorithms for searching intelligent digital twins.
2) Tools: SAT/SMT solvers (Z3, CVC4), planners (Fast Downward), libraries for CSP and search.

6. Goal Coaching

Goal coaching is a systematic approach to setting goals, planning actions, supporting and tracking progress so that you move towards your results more effectively and with high motivation. Reasons:

1) The main idea: combine a clear vision of the goal, a reasonable action plan and regular feedback.
2) Often use models and tools that help turn abstract desires into concrete steps and deadlines (e.g. GROW, SMART goals, OKR, action plan).

Practical framework:

GROW (Goal, Reality, Options, Will):

1) Goal: what exactly do you want to achieve? Clarify the outcome and success criteria.
2) Reality: where are you now? What resources and limitations do you have?
3) Options: what steps can you take? What alternatives are available?
4) Will/Wrap-up (Action plan and responsibility binding): what specific step will you take, when, and how will you monitor progress?

You can also use SMART goals and OKRs to formalize goals.

Step by step: how to launch coaching to achieve goals:

1) Defining goals:
   a. Formulate the goal in a specific form: what exactly do you want to achieve, by what deadline, what indicators will be the result.
   b. Example of SMART: "Increase sales by 20% in 3 months by attracting 15 new clients and increasing conversion on the site to 2.5%."
2) Audit of current reality:
   a. Assess the current situation: what is already there, what works, what is in the way, what resources are available.
   b. Identify key barriers and weaknesses.
3) Generating options (Options):
   a. Brainstorming without restrictions: what actions can be taken? What are the alternatives and paths?
   b. Choose 3-5 most realistic options.
4) Action plan and responsibility:
   a. Choose a specific plan with small steps (next steps). - Assign deadlines and responsibilities, if you have a team.
   b. Break the goal into Weekly/weekly tasks and Daily-small steps.
5) Control and adjustment:
   a. Regularly monitor progress: what has been done, what is not working, what has changed.
   b. Make adjustments to the plan, taking into account new experience and data.
6) Support and motivation:
   a. Set regular sessions/checkpoints.
   b. Use supports: diary, habit tracker, progress reports, accountability partner.
7) Tools and formats that can be used:
   a. Coaching questionnaires (to develop thinking and motivation):
   b. What exactly do you want to achieve by the end of this month/quarter?
   c. What is stopping you right now?
   d. What steps can you take today/this week?
   e. What resources do you have, and where can you get more?
   f. Templates and documents
   g. SMART goal template
   h. Action plan (when, what, how)
   i. Progress and obstacles log
   j. Habit and KPI tracker
   k. Methods for increasing efficiency
   l. OKR: Objectives and Key Results for large goals
   m. WOOP: Wish, Outcome, Obstacle, Plan
   n. Dividing goals into short sprints (2-4 weeks)
   o. Session format - 60-minute coaching session twice a month plus weekly mini-meetings or check-ins
   p. At the beginning of the program
   q. diagnostics and goal setting, then regular checks and adjustments
8) Example of the structure of one coaching session (60 minutes):
   a. 5 minutes: check-in and mood, quick review of progress since the last meeting
   b. 10 minutes: progress according to plan, what has been accomplished, what are the results
   c. 15 minutes: working on the current goal (diving into questions on GROW)
   d. 15 minutes: searching for alternatives and choosing the next step (Options)
   e. 10 minutes: agreement on a specific step for the next week, setting expectations and KPIs
   f. Homework: one specific step to take by the next session

Examples of coaching questions (for different phases):

a. Goal phase:
   i. What exactly will be achieved in a specific measurable form?
   ii. Why is this important to you now?
b. Reality phase:
   i. What resources will help you? What obstacles do you see?
   ii. What of what you are doing now works best?
c. Options phase:
   i. What other options cannot be missed?
   ii. What steps will be the simplest and most realistic?
d. Will/plan phase:
   i. What step will you take in the next 24 hours?
   ii. What can get in the way, and how will you deal with it?
   iii. How will you track progress and when do you plan

to adjust the course?

9) Recommendations:
   a. Avoid overly general goals categorically, make goals specific and measurable.
   b. Lack of planning, it is difficult to move without a detailed step plan, break it down into small tasks.
   c. Set up regular checks.
   d. Think through obstacles and plans to get around them in advance.
   e. Don't try to do everything at once - choose 1-2 main areas for the period.
   f. Not a clear enough description of what goals you want to achieve and in what context (career, health, finances, education, personal life, etc.).
   g. You need to clearly formulate SMART goals, offer an action plan, select a suitable coaching format and provide ready-made templates (goals, action plan, progress log, OKR).

7. 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 in CFD, climate and biochemical models.
2) Physics-oriented neural networks (PINNs) 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

CFD	Contract For Difference
CSP	Constraint Problems Content Security Policy
CNN	Cable News Network
CVC	Card Verification Code
GPU	Graphics Processing Unit
GDPR	General Data Protection Regulation
GNN	Graph Neural Network
FOF	Fund of Funds
IDA	Interactive Disassembler Assembly
KPI	Key Performance Indicators
LISP	List Processing
NN	Neural Network
NSM	Naval Strike Missile
ODE	Ordinary Differential Equation
OKR	Objectives Key Results
PDE	Processing Development Environment

PDDL	Planning Domain Definition Language
PINN	Physically Informed Neural Networks
SAT	Scholastic Aptitude Test
SMART	Specific Measurable Achievable Relevant Time
SMT	Simultaneous Multithreading is a Technique
STRIPS	Separate Trading of Registered Interest and Principal of Securities
TPTP	Test & Performance Tools Platform
ViT	Vision Transformer
WOOP	Wish-Outcome-Obstacle-Plan

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

[2] *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.*

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

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

[5] *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.*

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

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

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

[9] *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

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

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

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

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

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

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