Machine Learning for Computational Science and Engineering (2026) 2:27
https://doi.org/10.1007/s44379-026-00073-z

RESEARCH

LLMs as designers of physics-constrained neural networks for constitutive modeling: a demonstration on hyperelastic solids

Marius Tacke[1] · Matthias Busch[2] · Kian Abdolazizi[2] · Jonas Eichinger[1] · Kevin Linka[3] · Christian Cyron[1,2] · Roland Aydin[4,5]

Received: 27 March 2026 / Accepted: 27 May 2026

Abstract

Large language model (LLM)-based frameworks extend beyond agents; they also enable the on-demand creation of specialized scientific and engineering tools. We demonstrate this concept in the field of solid mechanics. There, so-called constitutive models describe the relationship between body deformation and mechanical stress. They are essential for both the scientific understanding and industrial application of materials. However, even recent data-driven methods of constitutive modeling, such as constitutive artificial neural networks (CANNs), still require substantial expert knowledge and human labor. We present a framework in which an LLM generates a CANN on demand, tailored to a given material class and dataset provided by the user. The framework covers LLM-based architecture selection, integration of physical constraints, and complete implementation. Evaluation on three benchmark problems demonstrates that LLM-generated CANNs achieve accuracy and generalization comparable to or greater than manually engineered counterparts, while substantially reducing the expertise required for constitutive modeling.

Keywords Large language models (LLMs) · Physics-constrained neural networks · Constitutive artificial neural networks (CANNs) · Automated model generation · Data-driven solid mechanics

1 Introduction

Constitutive models capture our understanding of how materials behave under mechanical load, expressed as mathematical relationships linking stresses to strains. Calibrated with experiments, they predict behavior beyond what was directly tested, including complex cases that are hard or impossible to reproduce in the lab. These predictions allow for realistic mechanical simulations of engineered products and biological tissues, which deepen scientific understanding and reduce the time and cost of component design.

Historically, constitutive behavior was captured by empirically derived symbolic laws such as Mooney–Rivlin, Neo Hookean, and Ogden [1–4]. To reduce the effort of handcrafted laws, data-driven approaches arose: distance minimizing data-driven computing [5, 6], black-box surrogates via neural networks [7, 8], and spline-based interpolants [9–11]. These are typically flexible but data hungry, weak at extrapolation, and difficult to interpret.

Gray box strategies embed physics to improve reliability [12]: PINNs [13, 14]; MIANNs/PANNs that hard enforce mechanics [15–19]; B-spline-based constitutive networks built on the Sussman-Bathe principal-direction formulation [20]; and, central to our benchmarks, CANN families that blend constitutive structure with learning [21–24]. The CANN paradigm has since been extended beyond pure hyperelasticity to viscoelastic CANNs (vCANNs) [25], damage-augmented CANNs capturing Mullins-type stress softening [26], and inelastic CANNs (iCANNs) for finite-strain plasticity, damage and viscoelasticity [27, 28], demonstrating that

✉ Marius Tacke
marius.tacke@hereon.de

✉ Christian Cyron
christian.cyron@hereon.de

✉ Roland Aydin
roland.aydin@uni-saarland.de

1 Helmholtz-Zentrum Hereon, Geesthacht, Germany

2 Hamburg University of Technology, Hamburg, Germany

3 RWTH Aachen University, Aachen, Germany

4 Saarland University, Saarbrücken, Germany

5 German Center for Artificial Intelligence, Saarbrücken, Germany

Published online: 12 June 2026

CANN architectures can represent strongly non-monotonic and history-dependent stress responses while preserving thermodynamic consistency. Related hybrids add neural corrections to mechanistic baselines [29] or learn path dependence via neural ODEs [30]. Beyond constitutive modeling itself, a parallel line of work employs machine-learning surrogates to predict effective material properties and engineering failure risks at a fraction of the cost of direct simulation [31–38]. These approaches cut data needs and aid extrapolation, yet retain black box elements.

In parallel, interpretable methods seek explicit, inspectable laws: symbolic regression [39, 40]; EUCLID style inference from fields and forces [41–44]; and KAN based models that yield closed-form constitutive expressions, including CKANs [45–49]. Despite growing automation, effective use still demands substantial expertise.

LLM-based code generation lowers this barrier by automatically assembling the data processing, model setup, and solver code needed to build simulation or optimization pipelines from plain-language task descriptions. Such frameworks appear across diverse domains, including engineering optimization, PDEs, graph and materials modeling, and chemical engineering [50–56]. A recent thrust centers on bilevel optimization, where an LLM-driven outer loop proposes solution candidates and an inner loop performs numerical calibration and evaluation [57, 58]. Within this pattern, the scientific generative agent (SGA) applies this approach to general scientific hypothesis generation [59], while the constitutive scientific generative agent (CSGA) adapts it for constitutive modeling [60]. Across benchmark stress–strain prediction tasks, CSGA outperforms SGA but remains less accurate than highly specialized methods such as constitutive artificial neural networks (CANNs).

Beyond single-LLM pipelines, agentic systems use multiple LLMs to plan, write, execute, and refine domain-specific code, such as MechAgents for finite-element mechanics [61] and MDAgent for molecular dynamics [62]. These frameworks are evolving from fixed teams to dynamic, task-specific organizations through subtask decomposition and specialized subagents [63, 64], and even toward self-developing 'agent OS' platforms [65]. Related work explores adaptive teaming and coordination [66, 67] or frames agent design as evolutionary search [68].

Motivated by dynamic agent generation, we propose an LLM-driven framework that creates task-specific constitutive artificial neural networks (CANNs) on demand and then immediately uses these self-generated new modules. While the constitutive scientific generative agent (CSGA) improved usability by letting non-experts build constitutive models with LLMs, it lacked the accuracy of specialized CANNs. Our approach combines both strengths: the LLM automatically designs, configures, and calibrates a CANN tailored to each material, offering the simplicity of an LLM interface and the accuracy of CANNs. We call these LLM-generated networks GenCANNs and refer to human-designed ones simply as CANNs. Although the LLM defines each GenCANN in Python code, its role goes far beyond conventional code generation. The actual task is to design a physically meaningful constitutive model that can learn from very small datasets, generalize to unseen loading conditions, and extrap-

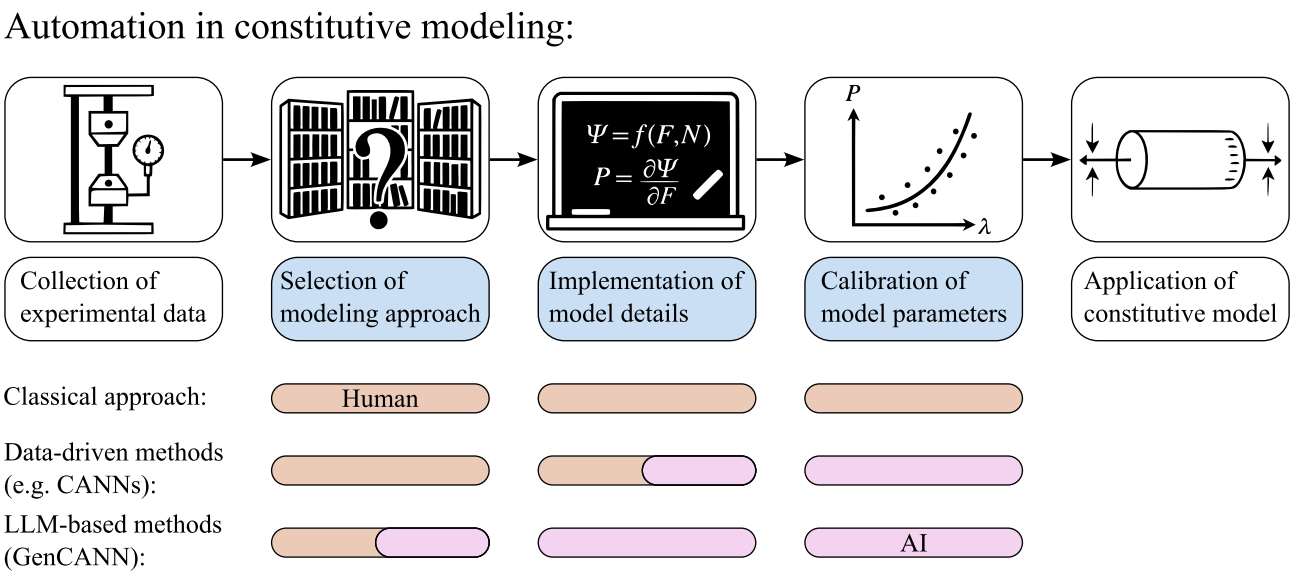

Fig. 1 Evolution of constitutive modeling in solid mechanics over time. Until the 2010s, scientists manually derived constitutive models to describe experimental observations. Recent data-driven methods, such as constitutive artificial neural networks (CANNs), partially automate model implementation and fully automate model calibration. Our framework leverages LLMs to generate CANNs on demand, pushing towards complete end-to-end automation

olate beyond the training range. In this study, we show that LLMs can do exactly that. The progression toward automation enabled by GenCANNs is illustrated in Fig. 1.

The remainder of the paper is organized as follows: Section 2 provides background and introduces benchmark methods. Section 3 describes our approach. Section 4 reports evaluation results, and Section 5 discusses findings and concludes the paper.

2 Background

2.1 Continuum mechanics essentials

Because CANNs embed continuum mechanics directly into their architecture, the method cannot be understood without the underlying theory. In particular, the preprocessing from deformation measures to invariants and the postprocessing from strain energy to stress both follow continuum mechanics. We therefore briefly summarize the foundations relevant to this work here and refer the reader to [69] for a comprehensive treatment.

Material points are labeled by their reference position $\mathbf{X}$ and current position $\mathbf{x}$. The deformation is characterized by the deformation gradient $\mathbf{F}$ and the right Cauchy–Green deformation tensor $\mathbf{C}$:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}, \quad \mathbf{C} = \mathbf{F}^T\mathbf{F}.$$

In simple loading cases, such as uniaxial tension, the deformation can be described by the stretch $\lambda = \frac{l}{l_0}$, which is the ratio of current length l to reference length l_0 and, in this case, corresponds to the relevant diagonal entry of $\mathbf{F}$. For simple shear, where material layers undergo a lateral displacement, the deformation is often characterized by the shear $\gamma = \frac{u}{h}$, the ratio of lateral displacement u to specimen height h, which then coincides with the corresponding off-diagonal entry of $\mathbf{F}$.

The scalar invariants of $\mathbf{C}$ are I_1, I_2, and I_3. Incompressibility means $\det(\mathbf{C}) = 1$, hence $I_3 = 1$:

$$I_1 = \mathrm{tr}(\mathbf{C}), \quad I_2 = \frac{1}{2}(\mathrm{tr}(\mathbf{C})^2 - \mathrm{tr}(\mathbf{C}^2)), \quad I_3 = \det(\mathbf{C}).$$

For isotropic materials, the strain-energy density Ψ depends only on invariants of $\mathbf{C}$, whereas anisotropy requires an additional description of the form of anisotropy. In this work, we only assume transverse isotropy with a single preferred fiber direction $\mathbf{n}$, define the structure tensor $\mathbf{N}$, and form the additional invariants I_4 and I_5:

$$\mathbf{N} = \mathbf{n} \otimes \mathbf{n}, \qquad I_4 = \mathbf{N} : \mathbf{C}, \qquad I_5 = \mathbf{N} : \mathbf{C}^2.$$

A material is considered hyperelastic if its mechanical behavior can be described by a strain energy density function, denoted as Ψ. In this work, we exclusively focus on the concept of hyperelastic materials, which describes rubber and various types of biological tissue in many situations with satisfactory accuracy. The task of constitutive modeling is to define the strain energy Ψ as a function f of the deformation state, that is, $\mathbf{F}$ or $\mathbf{C}$. Once the strain energy function Ψ is known, the isochoric part of the first Piola–Kirchhoff stress, $\mathbf{P}_{iso}$, can be determined. The incompressibility constraint adds a volumetric term, $-p\mathbf{F}^{-T}$, to the total stress $\mathbf{P}$, where p serves as a Lagrange multiplier enforcing incompressibility and corresponds to the hydrostatic pressure:

$$\Psi = f(I_1, I_2, I_4, I_5), \quad \mathbf{P}_{iso} = \frac{\partial \Psi}{\partial \mathbf{F}}, \quad \mathbf{P} = \frac{\partial \Psi}{\partial \mathbf{F}} - p\mathbf{F}^{-T}.$$

The first Piola–Kirchhoff stress $\mathbf{P}$ represents the load per unit area in the undeformed reference configuration, whereas the Cauchy stress $\boldsymbol{\sigma}$ refers to the load per unit area in the deformed spatial configuration. $\boldsymbol{\sigma}$ can be computed from $\mathbf{P}$ using the deformation gradient $\mathbf{F}$ and its determinant J:

$$J = \det(\mathbf{F}), \quad \boldsymbol{\sigma} = J^{-1}\mathbf{P}\mathbf{F}^T.$$

Both $\mathbf{P}$ and $\boldsymbol{\sigma}$ are second-order tensors, typically represented in 3D as 3×3 matrices, where $\mathbf{X}_{ij}$ denotes the entry in row i and column j.

2.2 Constitutive artificial neural networks (CANNs)

As we aim to generate constitutive artificial neural networks (CANNs) on demand using large language models (LLMs), we compare them against CANNs designed by human experts. These models follow the continuum mechanics framework outlined in Section 2.1. They are gray-box models that process the deformation gradient $\mathbf{F}$ in three stages [21]. A deterministic preprocessing block first applies white-box relationships from continuum mechanics to assemble the right Cauchy-Green tensor $\mathbf{C}$ and compute the scalar invariants (such as I_1, I_2, and, for transversely isotropic materials, I_4 and I_5). A black-box feed-forward neural network then maps these invariants to the strain energy density Ψ, where individual invariants are often passed through separate sub-networks whose outputs are combined by a final layer so that the contribution of each deformation mode remains interpretable. Finally, a deterministic postprocessing block obtains the first Piola-Kirchhoff stress $\mathbf{P} = \partial\Psi/\partial\mathbf{F} - p\mathbf{F}^{-T}$ by automatic differentiation of Ψ with respect to $\mathbf{F}$ and adds the incompressibility pressure term. This approach reduces the tensor-to-tensor mapping to a compact scalar regression, enforces thermodynamic consistency, and improves interpretability: because each invariant has a clear geometric

meaning and is processed by a dedicated sub-network whose output enters the strain energy through a final combining layer, the contribution of each deformation mode to Ψ can be inspected individually after training [21]. Training proceeds by backpropagation of a stress-based loss using a stochastic gradient-based optimizer, while activation functions, weight initialization, and the number and width of hidden layers are tuned per material to balance expressivity against overfitting on the typically small stress-strain datasets. For each dataset, we compare against the most accurate publicly established CANN variants [21, 70, 71]. Re-implementing or adapting CANNs remains challenging, as details such as input selection and constraint enforcement must be tailored for each material. Consequently, constitutive modeling in the current paradigm and prior to GenCANN still requires deep expertise.

2.3 Constitutive scientific generative agent (CSGA)

An obvious route to automate constitutive modeling is to use LLMs. Using an LLM as a direct strain-to-stress surrogate is unreliable and misses its strength in code generation. The scientific generative agent (SGA) addresses this by letting an LLM propose, implement, and refine constitutive models [59]. The constitutive scientific generative agent (CSGA) specializes SGA for continuum mechanics by adding assumptions (e.g., isotropy, incompressibility), defining inputs and outputs, suggesting an invariant basis, and enforcing zero stress at the reference state [60]. Concretely, CSGA implements a bilevel optimization: in the outer loop, an LLM receives a prompt that combines the task description, a few stress-strain samples from the experiment, materials-theory hints (e.g., to build the law on invariants of $\mathbf{C}$ or on principal stretches of $\mathbf{F}$, and to ensure $\mathbf{P} = \mathbf{0}$ at $\mathbf{F} = \mathbf{I}$), and a Python code skeleton with an `__init__` method leaving space for the declaration of trainable physical parameters and a `forward` method leaving space for the implementation of a mapping from the predefined input to the predefined output, in our case from the deformation gradient $\mathbf{F}$ to the first Piola-Kirchhoff stress $\mathbf{P}$. The LLM fills this skeleton with a symbolic, parameterized constitutive law. In the inner loop, these continuous parameters are calibrated on the supplied data by gradient-based optimization of a stress-based loss, and the calibrated model together with its loss value is appended to the prompt and returned to the LLM for a fixed number of refinement iterations, so that each new proposal can build on the strengths and weaknesses of previous ones. We use CSGA as our second baseline, complementing CANNs, as it is the most specialized LLM-based framework for constitutive modeling so far. Reported studies [59, 60] show that CSGA outperforms SGA but remains less accurate than CANNs. Its advantage lies in ease of use via a plain-text interface.

3 Method

Current approaches to constitutive modeling follow two paths. Specialized models such as CANNs are highly accurate and inherently satisfy physical constraints such as objectivity and thermodynamic consistency but are difficult to implement. In contrast, LLM-based agents like CSGA are easy to use through a text interface but lack accuracy and do not enforce these constraints. Rather than replacing one with the other, we combine their strengths: the LLM generates, on demand and from scratch, a CANN tailored to the material at hand. We refer to this generated model as GenCANN, short for LLM-generated CANN. This way, the LLM builds on (instead of competes with) decades of research, offering the simplicity of an LLM interface together with the accuracy and consistency of CANNs.

At the core of our pipeline is a large language model, OpenAI's o3, which we use without additional training. We focus on hyperelastic incompressible materials that are either isotropic or transversely isotropic. Figure 2 summarizes the LLM's role in the CANN design process. The LLM receives a two-part prompt. The first part describes the task: to implement a CANN that matches the chosen material class and follows certain coding requirements. It also includes a short summary of the continuum mechanics theory that links stress and strain through strain energy, similar to the Background Section 2.1. Depending on the material class selected by the user, these instructions vary slightly. For transverse isotropy for example, it is also explained how a structure tensor is defined. These specifications in the prompt are required at least by the current generation of LLMs. Our framework translates the user's choice of material class into these precise instructions: while users only need to decide whether a material is isotropic or has a single preferred fiber direction, the framework handles the hard part of turning that decision into a correct implementation. The second part of the prompt includes a rough code skeleton defining class names and public method signatures to support automatic invocation of the generated code. This skeleton deliberately fixes only the external interface required to call the generated module from our framework; it does not prescribe any internal network architecture. As a consequence, the CANN architecture is not predefined but is designed anew by the LLM in each generation run, which is the central methodological difference to all human-designed CANNs in the literature. The generated CANN combines three elements: preprocessing, one or more feedforward neural networks, and postprocessing. All of these are implemented by the LLM. Preprocessing and postprocessing, which include tensor assembly, invariant computation, and stress derivation, are mostly determined by continuum mechanics theory. This structure causes a number of physical constraints to be satisfied by construction rather than by penalty terms in the loss. For example, ther-

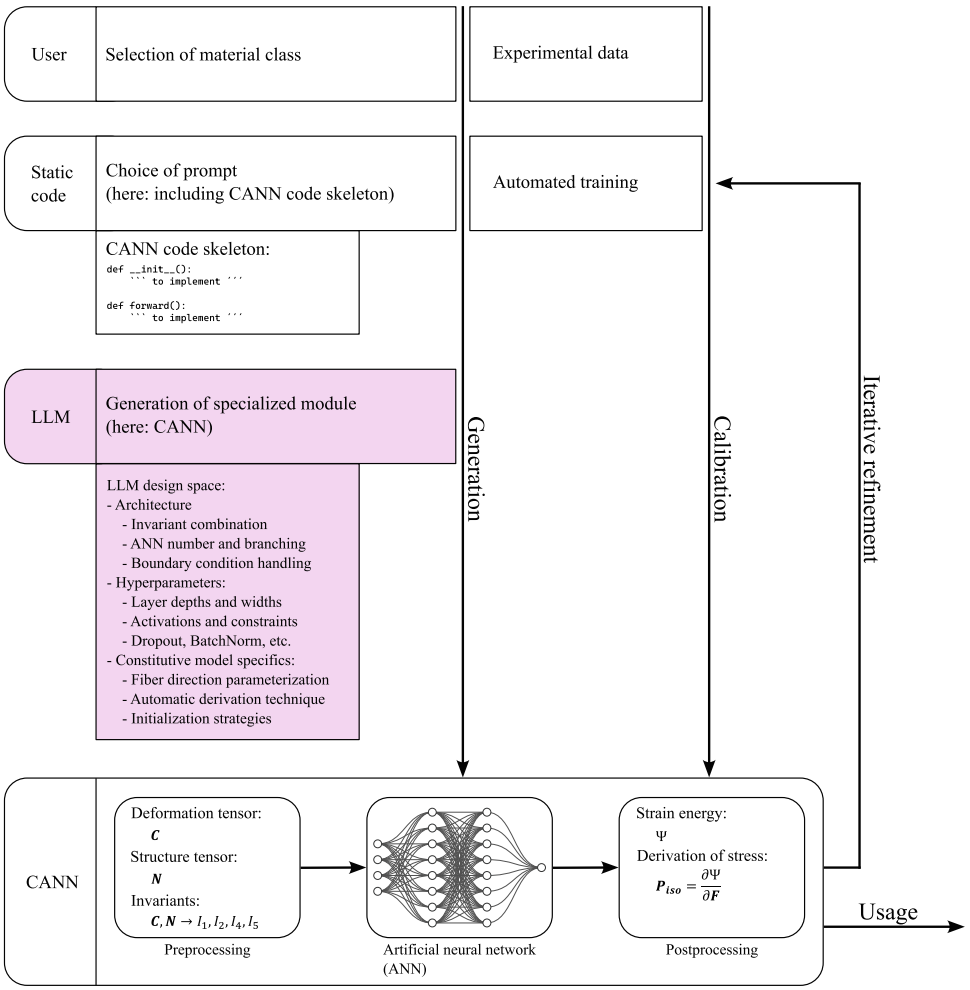

Fig. 2 Process of the LLM-based generation of a constitutive artificial neural network (CANN). Static code translates the material classification into a prompt that includes a task description, continuum mechanics theory, formal requirements, and a rough CANN code skeleton, prompting the LLM to implement a CANN tailored to the specific modeling task. The resulting CANN is then trained and evaluated on experimental data and iteratively refined

modynamic consistency follows from obtaining $\mathbf{P}$ as $\partial\Psi/\partial\mathbf{F}$ via automatic differentiation, while objectivity, stress-tensor symmetry, and material symmetry follow from using only invariants of $\mathbf{C}$ (and, for transverse isotropy, the structure tensor $\mathbf{N}$) as inputs to the feedforward network. The same applies to interpretability: because the LLM-designed feedforward part still operates on the invariants, the per-invariant inspectability of the strain energy carries over to GenCANNs.

The main design freedom of the LLM lies in the feedforward neural networks that map invariants to strain energy. For these networks, the LLM decides on invariant combinations, network architecture and size, activation functions, constraints and regularization, handling and estimation of fiber directions when needed, weight initialization, and treatment of boundary conditions. Concretely, this means that the depth and width of the feedforward network, the choice of activation functions, the initialization strategies, the use of one monolithic network versus several invariant-specific sub-networks, and the way constraints are imposed may all differ substantially between runs and between materials. Concrete, inspectable examples are provided in two places: the full list of LLM-selected architectures per dataset is compiled in Table 1, and a complete end-to-end LLM-generated CANN implementation is reproduced verbatim in Listing 1 of Section D.

Once the CANN is implemented, it is executed and trained on the provided data. The complete script and its R^2 score are sent back to the LLM for three refinement rounds. The best-performing version is kept as the final model. We repeat the complete CANN generation process five times per dataset, present the statistical analysis in Fig. 9, and show the best-performing CANNs in Figs. 3, 4, 5, 6, 7 and 8.

The central technical difference between SGA, CSGA, and our pipeline lies in how specifically the task is posed to the LLM. SGA and CSGA ask for a constitutive model in relatively broad terms, even if CSGA adds domain guidance. Our pipeline instead asks the LLM to implement a constitutive artificial neural network (CANN) for a given material class. This specific request strongly activates the CANN-related knowledge encoded in the model weights, including typical architectural patterns, invariant choices, constraint strategies, and implementation principles learned from the

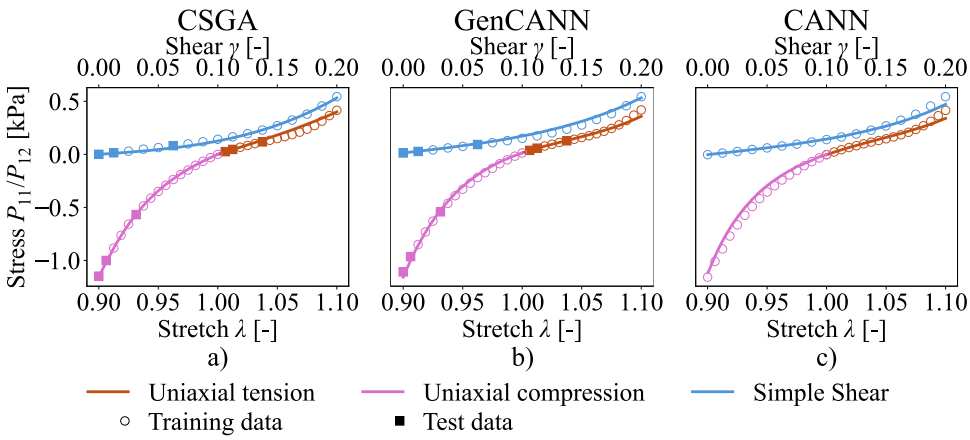

Fig. 3 Brain tissue deformation. Predictions of the LLM-generated constitutive artificial neural network (GenCANN) for mechanical stress induced by brain tissue deformation compared with two baselines: the LLM-based constitutive scientific generative agent (CSGA) and the human-designed CANN

literature. The knowledge is always present in the LLM, but the prompt determines which parts of it are brought to the foreground and used. In this way, the LLM is guided directly into the CANN design space and builds on the established CANN literature.

4 Results

4.1 Brain data

We begin with a dataset on the mechanical behavior of human brain tissue, an established benchmark for hyperelastic constitutive modeling [72–74]. Accurate models support impact simulation, injury prediction, and protective design. Brain tissue is soft, nearly incompressible, strain-stiffening, and asymmetric in tension and compression. The data were collected by [72] through mechanical tests on specimens excised from ten post-mortem human brains (7 male, 3 female, ages 54–81) within 60 hours of death. Multiple regions were sampled, we focus on their cortical gray matter. The tissue was subjected to three loading modes: uniaxial tension, uniaxial compression, and simple shear. For each loading mode, specimens underwent quasi-static loading-unloading cycles up to maximum stretches of approximately $\lambda = 1.1$ in tension,

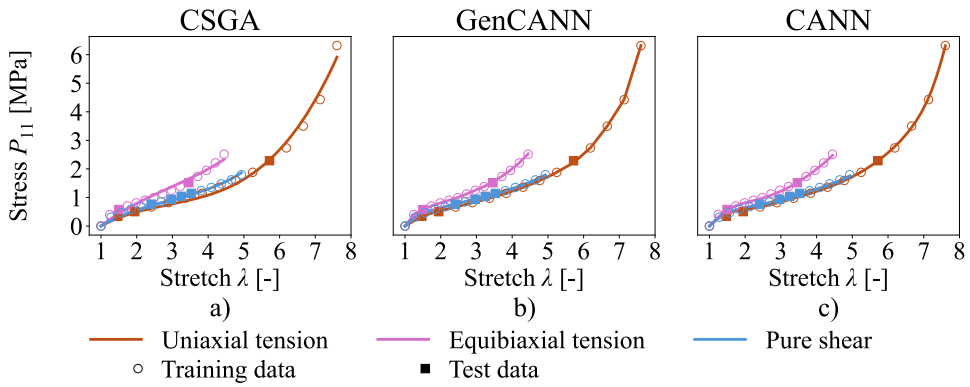

Fig. 4 Rubber deformation. Predictions of the LLM-generated GenCANN for mechanical stress induced by rubber deformation compared with two baselines: the LLM-based CSGA and the human-designed CANN

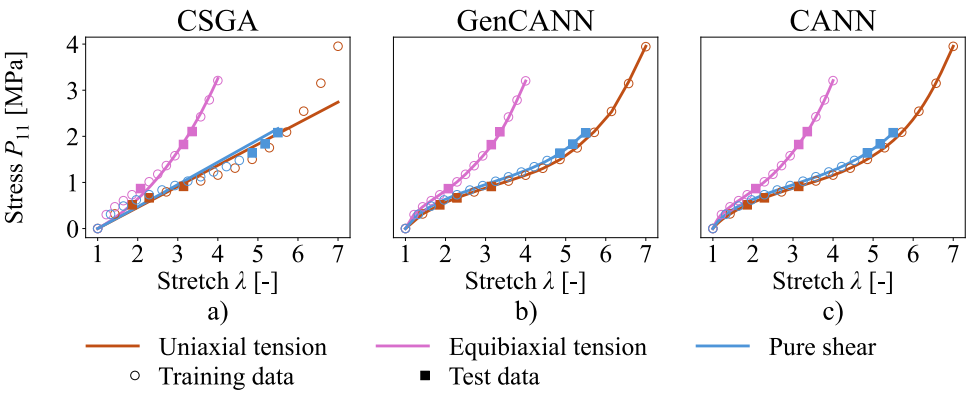

Fig. 5 Synthetic rubber deformation. Predictions of the LLM-generated GenCANN for mechanical stress induced by deformation of a fictitious rubber-like material compared with two baselines: the LLM-based CSGA and the human-designed CANN

$\lambda = 0.9$ in compression, and shear amplitudes of $\gamma = 0.2$, and the mean stress over the hysteresis loop was taken as the effective elastic response. 17 stress–strain points were reported for each loading mode, which makes learning a well-generalizing and extrapolating model challenging. However, such sparse datasets are typical for soft-tissue experiments and representative of standard practice in the field.

We use the best CANN reported in the literature [70], selected from multiple CANN optimization studies on this dataset [22, 23, 70, 74], and the previously introduced CSGA [60] as baselines. Across uniaxial tension, compression, and simple shear, all three methods closely reproduce the measured stress–strain response, as shown in Fig. 3. All models achieve R^2 scores above 0.90 on the brain tissue dataset (see Table 2 for all individual values), with only CSGA showing one score below 0.95 ($R^2 = 0.93$ on uniaxial tension), while the unconstrained GenCANN reaches $R^2 = 0.97$, 1.00, and 0.98 on uniaxial tension, uniaxial compression, and simple shear, respectively, on par with the human-designed CANN ($R^2 = 0.96, 0.99, 1.00$). All three approaches predict training and test points reliably, but were trained on all tested loading conditions. These results confirm that each model captures the complex behavior in the training data, but do not show generalization to unseen loading conditions.

4.2 Rubber data

Rubber is the classic example of a hyperelastic solid, capable of large, reversible strains beyond the scope of linear elasticity. Accurate modeling enables reliable design of components like tires and seals. We study two datasets: Treloar's classic experiments [75] and a separate synthetic dataset that represents a similar fictitious material, provides ground truth for complex loading scenarios, and was introduced in the first publication on CANNs [21]. Both cover the three canonical deformation modes of incompressible rubber elasticity - uniaxial tension, equibiaxial tension, and pure shear - with 15 stretch-stress samples per protocol spanning stretches up to roughly $\lambda = 7$ in uniaxial tension, and we keep the train-test split (alternating samples along each curve) used in the first CANN publication. The experimental data anchor the problem in reality but span only a few loading paths, so a central question is how well models extrapolate to mixed multiaxial states not seen during training. Measuring such states in experiments is often not possible. The synthetic dataset addresses this by including exact stresses for arbitrary deformations from an isotropic, incompressible rubber-like material, enabling a clean assessment of generalization beyond the trained loading paths.

For both rubber datasets, we use the optimal CANN from its initial publication [21] and the CSGA [60] as baselines. Figures 4 and 5 show that GenCANN and CANN match

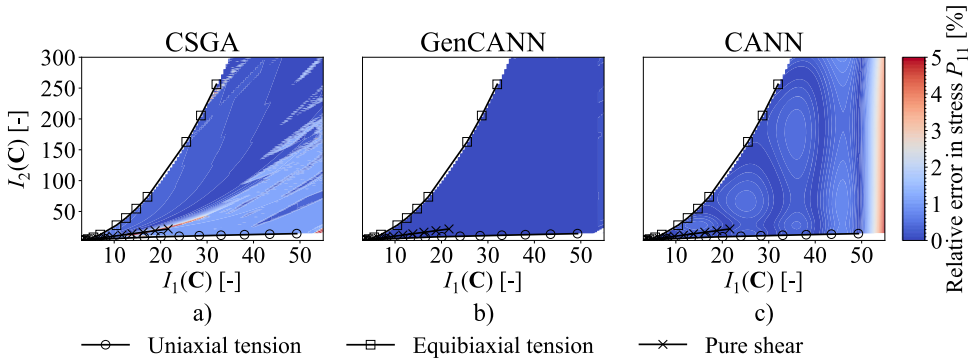

Fig. 6 Invariant plane. Predictions of the LLM-generated GenCANN for mechanical stress induced by deformation of a fictitious rubber-like material evaluated on a plane of biaxial loading states. The three marked paths are included in the training set, while all intermediate states are unseen. The GenCANN is compared with two baselines: the LLM-based CSGA and the human-designed CANN

measured and ground truth stresses with near-perfect accuracy across uniaxial, equibiaxial, and pure shear loading, while CSGA lags behind. These results confirm that CANNs, whether LLM-generated or manually implemented, outperform CSGA even on loading conditions known from training. Quantitatively, on the synthetic rubber dataset CSGA reaches $R^2 = 0.87$ on uniaxial tension and 0.93 on pure shear, whereas both the GenCANN and the human-designed CANN attain $R^2 = 1.00$ on all three loading modes (Table 2). We next evaluate model extrapolation to unseen loading scenarios. For the synthetic material, ground truth stresses can be computed for arbitrary deformations. This enables evaluation on Treloar's invariant plane [76] (Fig. 6), with the first and second invariants spanning the x- and y-axes, respectively. The plane spans from uniaxial to equibiaxial tension, with pure shear at the angular midpoint (note that the axes are scaled differently). Both the baselines CANN and CSGA extrapolate well to the new loading states between the marked paths. The CSGA shows slightly higher errors overall, but extrapolates better to the largest stretches. GenCANN excels in both generalization and extrapolation, performing remarkably well on loading conditions outside its training range. This is a non-trivial test in two senses: the intermediate states between the trained paths involve combinations of I_1 and I_2 that are absent from the training data, while the regions near the boundary of the plane require extrapolation to stretch magnitudes beyond those seen during training. The accuracy of GenCANN across the entire plane indicates that the embedded continuum-mechanical structure effectively constrains the response in data-free regions.

4.3 Skin data

To move beyond isotropy, we next study a transversely isotropic soft tissue: porcine skin. Aligned collagen creates a preferred fiber direction, with higher stiffness along the fibers and greater compliance across them. We use a publicly available biaxial stretch-stress dataset with 402 data points from porcine skin specimens [77, 78]. For the isotropic materials discussed in Sections 4.1 and 4.2, uniaxial stress measurements suffice. However, calibrating models of transversely isotropic materials requires biaxial stress measurements, as they provide directional information essential for capturing fiber-induced anisotropy. The five loading paths are equibiaxial, applying equal stretch in both principal directions; strip-axial, stretching one direction while keeping the other at its initial length; and off-axial, stretching both directions with a stronger bias toward one. The applied stretch was increased monotonically during each test up to a maximum stretch of approximately $\lambda = 1.2$, with stresses recorded in both in-plane directions, yielding two stress components per loading path and a total of ten stress-stretch curves across the five protocols. We assume the tissue is incompressible and that

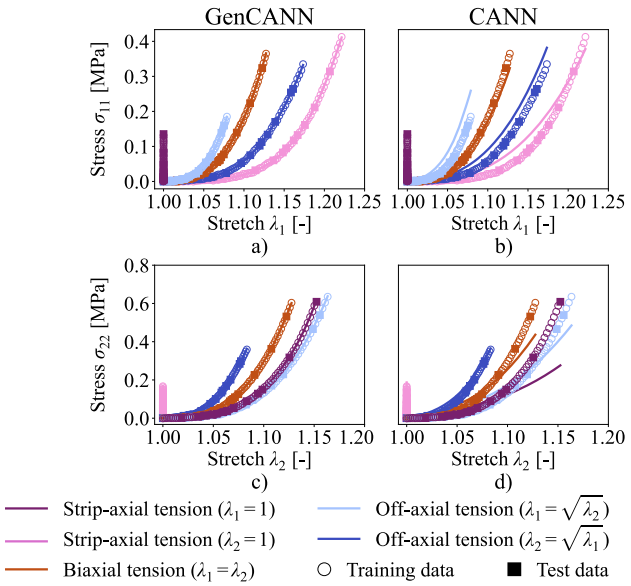

Fig. 7 Porcine skin deformation. Predictions of the LLM-generated GenCANN for stress induced by porcine skin deformation compared with a human-designed CANN as baseline. The LLM-based CSGA previously used as second baseline is not yet capable of modeling transversely isotropic materials

there is no stress acting through the thickness. Our baseline is the CANN variant identified via systematic hyperparameter search for this material [71]. Unlike our approach, this baseline model was trained on all available data points without a dedicated test split. The CSGA has so far only been implemented for isotropic materials, which is why it cannot serve as a baseline for the porcine skin dataset. On this dataset, the GenCANN fits all five loading paths with essentially perfect accuracy, see Fig. 7. It reaches an R^2 score of 1 for every reported stress component. The expert-implemented CANN by [71] shows noticeable errors even on loading paths included in the training, with R^2 scores such as 0.92-0.93 for equibiaxial loading. To check for overfitting, we used leave-one-loading-scenario-out cross-validation, retraining our GenCANN five times and evaluating its performance on the left-out path. As shown in Fig. 8, predictions on unseen paths are less accurate than on paths known from training but remain on par with the manually engineered CANN even though that model was trained on all paths, indicating that the GenCANN does not overfit and extrapolates reasonably well to new biaxial loading states.

4.4 Statistical analysis

In Section 3, we described the CANN generation pipeline, and Fig. 9 summarizes this workflow and the step-wise statistics. We assess our framework along two complementary axes: robustness, that is, its ability to detect and recover from

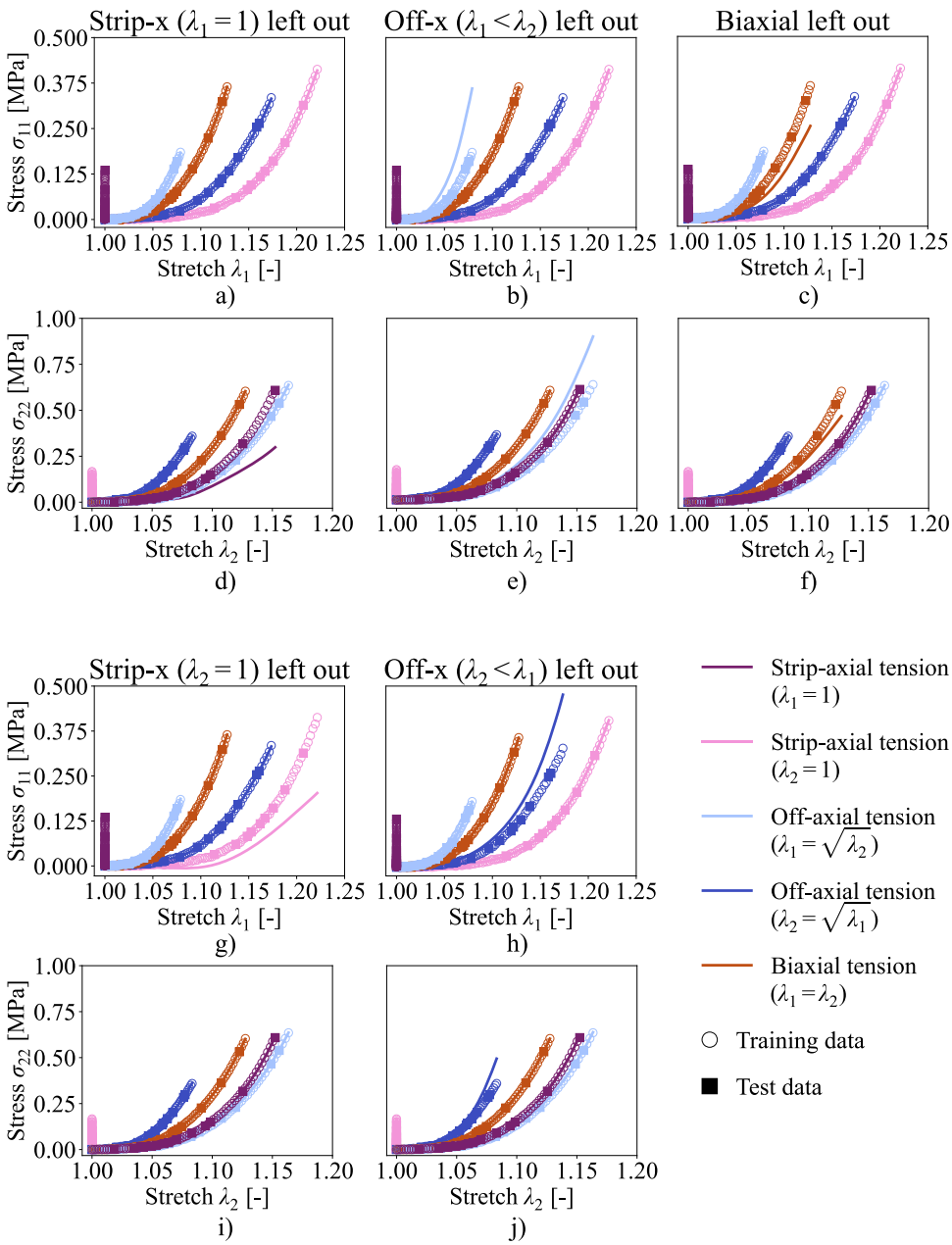

Fig. 8 Leave-one-out cross-validation. Predictions of the LLM-generated GenCANN for stress induced by porcine skin deformation evaluated using a leave-one-loading-scenario-out cross-validation. In five separate training runs, one loading path is excluded each time and used to test how well the GenCANN generalizes to unseen loading paths

faulty model proposals, and reliability, that is, the consistency of its final accuracy across repeated runs.

Each model proposed by the LLM is returned as a Python source string and dynamically executed inside the pipeline. To safeguard against malformed or semantically inconsistent code, the static framework code wraps every point of contact with the dynamically generated module in dedicated try/except guards. These guards cover the critical interactions between static and generated code, namely parsing and importing the LLM-generated script, instantiating the proposed model, performing a forward pass on reference inputs, executing the training loop, and computing the final evaluation metrics. Two classes of failures are distinguished. Syntactic and runtime errors, such as undefined symbols, shape mismatches, or non-differentiable operations, are caught directly by the exception handlers; the captured traceback together with the offending script are returned to the LLM with a request to repair the implementation. Semantic training failures, such as non-convergence within the fixed training budget or numerical breakdowns leading to NaN or Inf stresses, manifest as a negative R^2 score on the training data and are detected by an explicit post-training check that triggers the same correction loop. Across the 124 implementations generated in this study, 31% exhibited syntactic or runtime errors and 5% led to training failures; in every case, the framework detected the issue and obtained a valid, con-

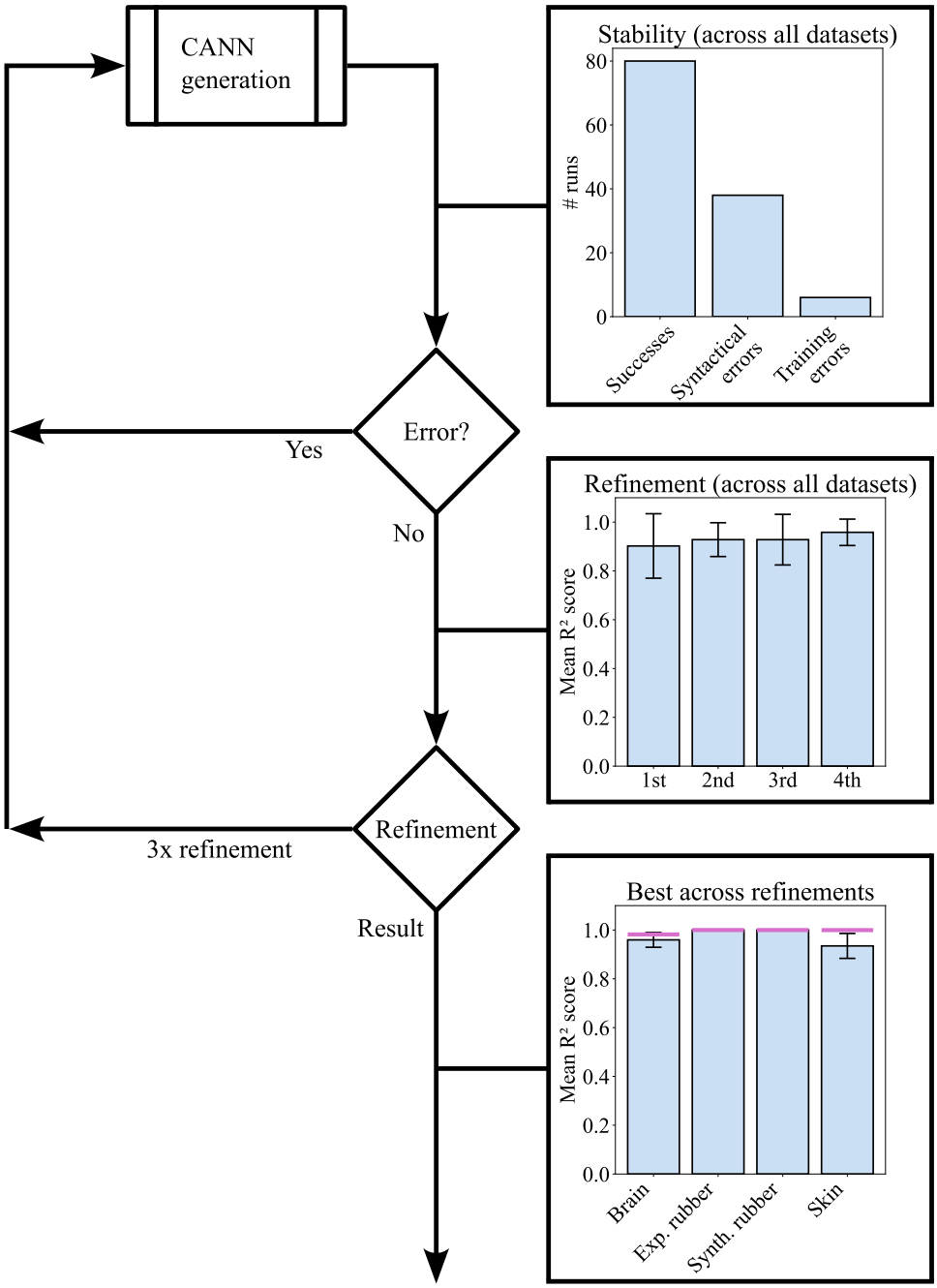

Fig. 9 Stability analysis of our framework. If a generated CANN is invalid Python code (syntactical error) or yields a negative R^2 score (training error), we discard it and repeat the generation. After obtaining a valid CANN, we resend it three times with its R^2 score to the LLM for refinement. We repeat the full process five times per dataset, summarize the results in the lowest bar plot, and mark the best run in pink. The consistently accurate outcomes across runs confirm the stability of our approach

verged model after at most a few retries. Once a valid model is produced, its script and R^2 score are returned to the LLM for three additional refinement rounds, which yield a small but consistent gain in accuracy and a reduction in variance.

Because the LLM is sampled stochastically, a single generation does not fully characterize the behavior of the framework. To assess reliability, we therefore repeated the complete pipeline five times per dataset and report the R^2 score averaged over loading scenarios in the lowest bar plot of Fig. 9. For both rubber datasets, all five runs reached an R^2 score of 1.00 with zero variance across runs. For the brain dataset, the five runs yielded R^2 scores of 0.90, 0.97, 0.98, 0.97, and 0.98 (mean 0.96, standard deviation 0.03), and for the porcine skin dataset 0.91, 0.89, 1.00, 0.99, and 0.89 (mean 0.94, standard deviation 0.05). All 20 runs across the four datasets converged to usable models, with even the worst run still attaining an R^2 score of 0.89 or above, which is comparable to or exceeds the corresponding manually engineered baseline. The best run per dataset is highlighted in pink in Fig. 9, and the predictions of these best runs are shown in Figs. 3-8. The limited variance across runs indi-

cates that even a single generation already produces a usable model with high probability; nevertheless, because the process is partly stochastic, we recommend running multiple generations and selecting the best model, as we do here.

Beyond accuracy, the practical adoption of an LLM-based generation pipeline also depends on its computational cost. Pooled across the four datasets and the five runs per dataset, a complete CANN generation (initial proposal plus three refinement rounds) consumed on average $44{,}624 \pm 29{,}504$ input tokens and $8{,}905 \pm 3{,}428$ output tokens with OpenAI's o3, corresponding to an average API cost of $\$0.16 \pm \$0.09$ per generation at the rates in effect during this study. The resulting expense is therefore on the order of cents per material, negligible compared to the expert-time cost of designing and implementing a CANN by hand.

5 Discussion and conclusions

We were inspired by works such as [63, 64] that create agents for use in LLM-based frameworks on demand. We applied this idea to constitutive modeling by generating specialized constitutive artificial neural networks (CANNs) on demand, each tailored to a specific material. Rather than viewing LLM-based approaches and specialized methods as competing, we integrate them, combining the strengths of both. CANNs provide high accuracy and strict adherence to physical constraints, while LLMs offer an accessible interface and great flexibility while vastly reducing the human expertise required.

In detail, our framework automates constitutive modeling by prompting an LLM to design a CANN that fits the material class and data. The LLM makes all key design choices, including architecture, activation functions, constraints, fiber direction handling, and the full technical implementation. This gives the system high flexibility for modeling new materials. Static code manages prompt selection and model training, which reduces user effort but limits adaptability. User input is minimal, requiring only material classification and data, making the system both powerful and easy to use. In the future, automating the static parts with LLM agents could expand the design space, reduce manual intervention, and improve adaptability to new materials and evolving model requirements even further.

The LLM-generated CANNs (GenCANNs) matched or, in several cases, clearly exceeded the accuracy of human-designed CANNs across the brain, rubber, and skin datasets, supporting the viability of our approach. Among the LLM's design choices, we observed a consistent preference for larger feedforward architectures than the baselines, for example, 256-128-64-3 vs. 100 neurons for brain and 32-32 vs. 16-16 for synthetic rubber (see Table 1), which raises the question of whether the performance gains are due only to increased capacity or whether GenCANNs remain competitive when restricted to the same size as the baselines. To answer this, we repeated the experiments with GenCANNs constrained to exactly match the baseline CANN network sizes. For brain and rubber, the size-constrained and unconstrained GenCANNs performed indistinguishably. Only the skin dataset showed a benefit from the larger, unconstrained network on the training paths, yet even there, the size-constrained GenCANN still clearly surpassed the baseline CANN. While larger models can fit training data more closely, they also increase the risk of overfitting, especially with the small datasets typical of constitutive modeling (e.g., 15 data points per loading path in rubber). Our generalization tests indicate that the LLM-based design remains well balanced: on the invariant plane and in the leave-one-loading-scenario-out cross-validation for skin, both size-constrained and unconstrained GenCANNs generalize to unseen loading states and extrapolate beyond the trained range with remarkable accuracy. The full analysis of the network size, including all plots, is provided in Section A and shows that GenCANNs remain highly competitive even when constrained in size.

Viewed more systematically, the comparison between size-constrained and unconstrained GenCANNs is one instance of a broader question: how strongly does the performance of the framework depend on its individual ingredients - the training data, the imposed physical constraints, and the architectural choices made by the LLM? Several analyses already presented can be read as a first, qualitative sensitivity study along exactly these axes. The size-constrained versus unconstrained comparison isolates the effect of network capacity from all other architectural decisions, the refinement analysis in Appendix C tracks how often the LLM modifies activation functions, weight initialization, and regularization across refinement rounds, and the leave-one-loading-scenario-out cross-validation on the porcine skin dataset probes sensitivity to data coverage. Together, they suggest that network size is the dominant driver of the accuracy gains during refinement, that activation, initialization, and regularization play comparatively minor roles, and that GenCANNs generalize robustly to loading paths excluded from training. A fully variance-based treatment along the same axes - for instance, Sobol indices, SHAP attributions, or controlled noise-injection studies as employed in recent materials and engineering informatics work [79–82] - could push this assessment further and quantify, within a single common framework, the marginal contribution of each physical constraint, hyperparameter, and dataset characteristic. Such an analysis would additionally reveal which prompt elements - for example, the explicit incompressibility instruction or the structure-tensor description for transverse isotropy - are most

critical for the LLM to produce reliable architectures, and is a natural extension of the present work.

A different route to the same goal of reducing expert effort in constitutive modeling is taken by the B-spline-based constitutive neural network of Lee and Bathe [20], which extends the Sussman-Bathe model [9] via regression on uniaxial tension-compression data. Its strength lies in resolving localized features such as peaks and wiggles along a single principal direction, achieved by deliberately forgoing constraints such as polyconvexity. The CANN paradigm on which GenCANN builds makes the opposite trade-off, embedding continuum-mechanical structure to support generalization to unseen multiaxial states (cf. Figs. 6 and 8); the two philosophies are therefore complementary. The monotonic stress-stretch responses in our benchmarks reflect the chosen datasets rather than a limitation of the framework: non-monotonic and history-dependent behaviors such as Mullins-type damage [26], finite-strain plasticity, and viscoelasticity [25, 27, 28] have already been formulated within the CANN paradigm, so extending the GenCANN prompt and code skeleton to let the LLM introduce internal damage or viscous variables is a natural next step.

Future work could extend the paradigm of using LLMs to generate specialized modules on demand to any domain that offers established theoretical scaffolding a neural network should respect, such as fluid mechanics, thermodynamics, electromagnetics, or chemical kinetics. Within this blueprint, LLM-driven synthesis could be benchmarked against classical automated model-design paradigms — neural architecture search, evolutionary algorithms, and reinforcement-learning-based controllers — which search a predefined architecture space rather than drawing on prior knowledge encoded in a pretrained language model. Overall, our results show that LLM-generated, physics-constrained CANNs are ready for real-world applications: embedding GenCANNs into finite-element solvers (e.g., as user material subroutines) and digital-twin environments would enable on-demand material modeling directly within engineering simulation pipelines.

Appendix A: Network architecture evaluation

In Section 3, we described the LLM's design space when generating a CANN. In Section 4, we showed that these GenCANNs achieve very high accuracy. While preprocessing from the deformation gradient to invariants and postprocessing from strain energy to stresses follow established continuum mechanics, the LLM still makes several important choices: which invariant combinations to use, the network architecture and size, activation functions, constraints and regularization, how to handle or estimate fiber directions, and weight initialization. We observed that it often selects larger architectures than the baseline CANNs (Table 1 and Fig. 16). To isolate the effect of capacity from other design choices, we ran a controlled comparison. Each GenCANN was constrained to use exactly the baseline CANN network size and evaluated alongside the unconstrained GenCANNs and the baseline CANNs (Figs. 10–15).

For the brain data (Fig. 10) and the rubber data on trained loading paths (Figs. 11 and 12), the three models are indistinguishable in practice. When we evaluate generalization on the synthetic rubber material using Treloar's invariant plane (Fig. 13), both GenCANNs, size-constrained and unconstrained, reach a remarkably high accuracy and substantially outperform the baseline CANN across unseen mixed biaxial states. For skin (Fig. 14), the unconstrained GenCANN is the only model that perfectly fits all training paths, suggesting that the larger architecture helps for this more complex anisotropic case, yet the size-constrained GenCANN still outperforms the baseline. In the skin leave-one-loading-scenario-out cross-validation (Fig. 15), the size-constrained GenCANN generalizes at least as well as the unconstrained

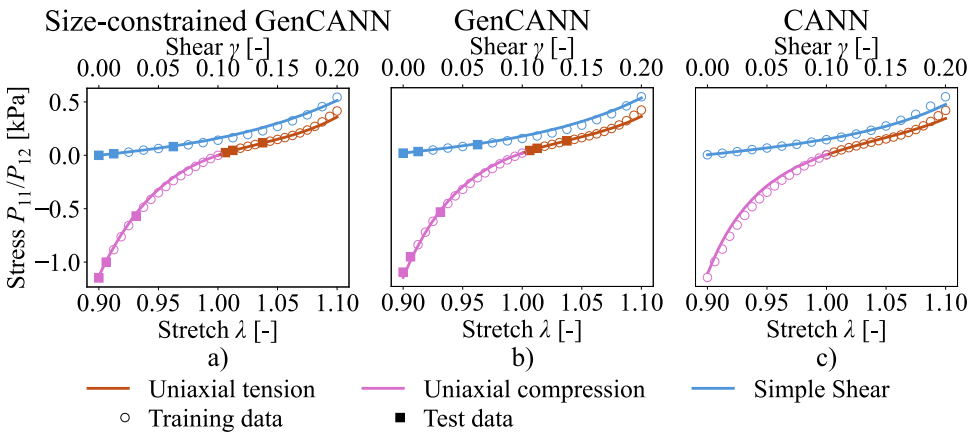

Fig. 10 Brain tissue deformation. Predictions of the LLM-generated constitutive artificial neural network (GenCANN) for mechanical stress induced by brain tissue deformation compared with the human-designed CANN as baseline and an LLM-generated GenCANN constrained to the baseline CANN network size

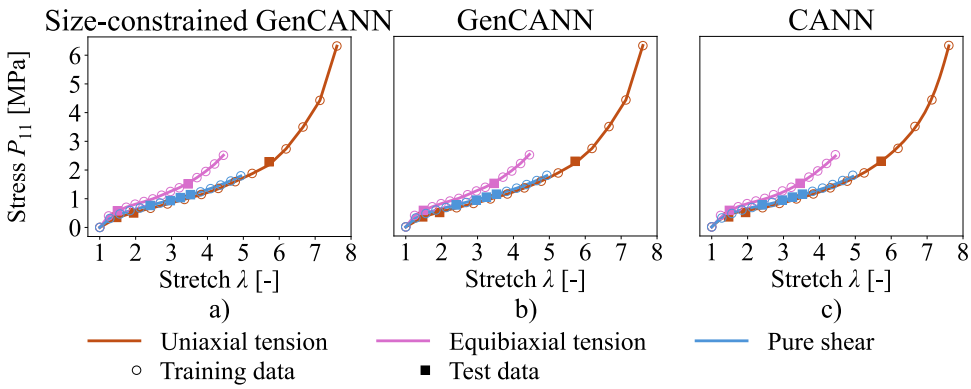

Fig. 11 Rubber deformation. Predictions of the LLM-generated GenCANN for stress induced by rubber deformation compared with the human-designed CANN as baseline and an LLM-generated GenCANN constrained to the baseline CANN network size

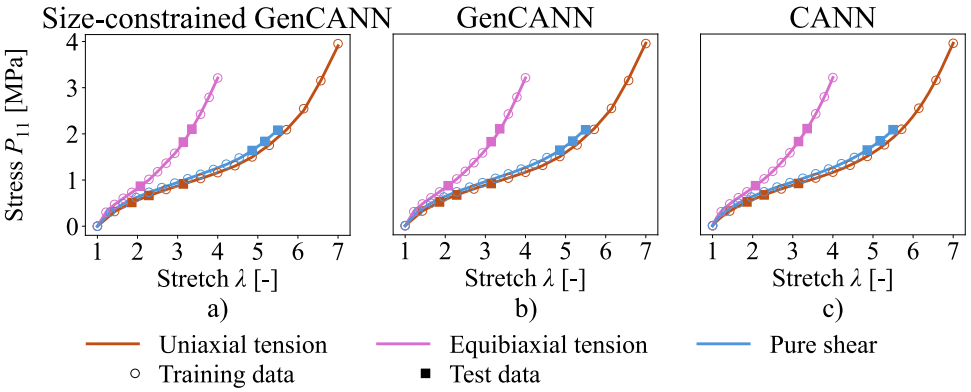

Fig. 12 Synthetic rubber deformation. Predictions of the LLM-generated GenCANN for stress induced by synthetic rubber deformation compared with a human-designed CANN as baseline and an LLM-generated GenCANN constrained to the baseline CANN network size

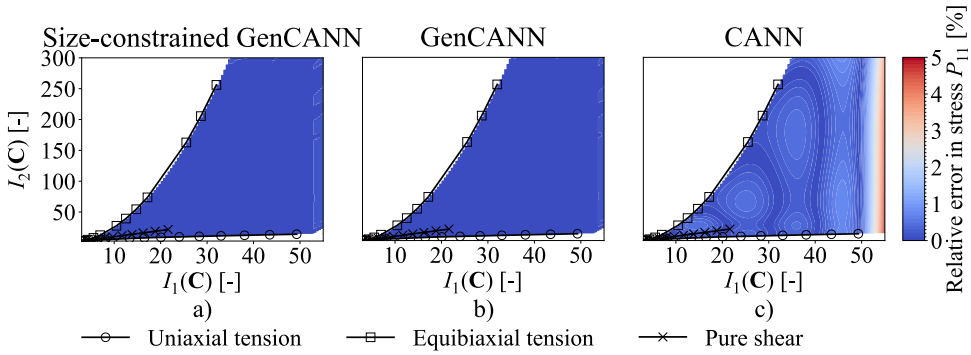

Fig. 13 Invariant plane. Predictions of the LLM-generated GenCANN for stress induced by synthetic rubber deformation evaluated on a plane of biaxial loading states. The three marked paths are in the training set, while all intermediate states are unseen. The GenCANN is compared with a human-designed CANN as baseline and an LLM-generated GenCANN constrained to the baseline CANN network size

model to the left-out path; the unconstrained model fits training paths more tightly but does not generalize better.

We consider the unconstrained GenCANNs the most realistic choice for new materials, where no manually tuned baseline prescribes an architecture. Our aim is to remove manual trial-and-error, not to reproduce legacy sizes. Still, when we do restrict the LLM to baseline sizes, GenCANNs remain highly competitive: only the skin dataset clearly benefits from the larger network, and even there, the size-constrained GenCANN exceeds the baseline. For generalization to unseen states and extrapolation (Figs. 13 and 15), size-constrained GenCANNs are on par with unconstrained ones. Overall, the strong performance of LLM-designed CANNs cannot be attributed to network size alone, and if users prefer smaller models for efficiency, interpretability, or deployment constraints, the GenCANN approach can honor those limits while maintaining high accuracy.

Appendix B: Tables

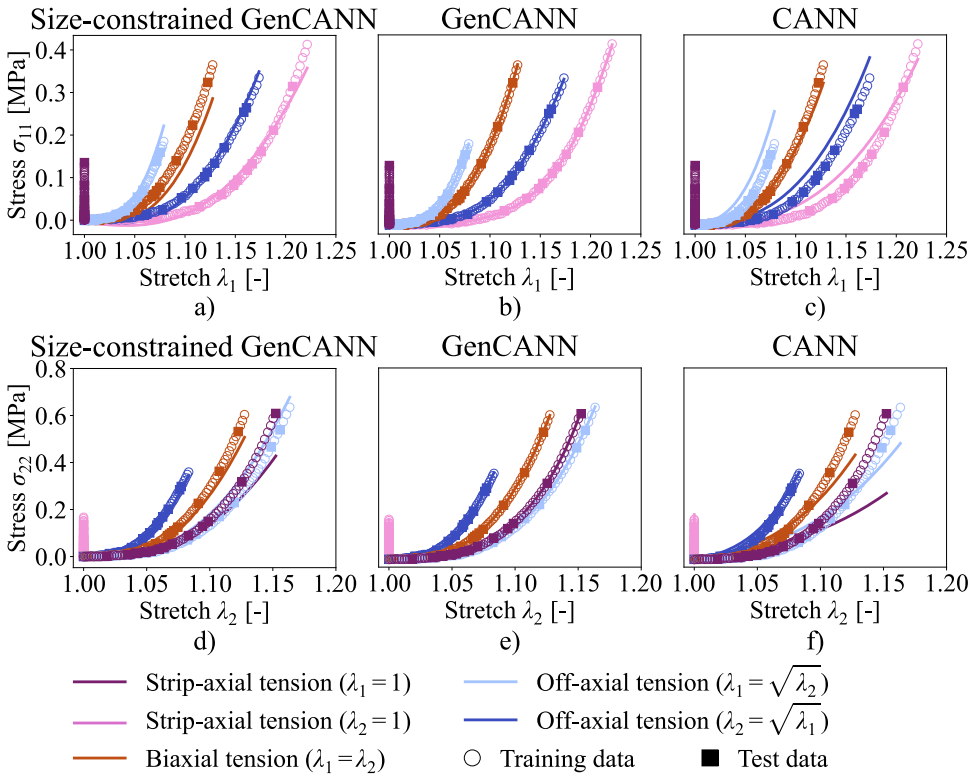

Fig. 14 Porcine skin deformation. Predictions of the LLM-generated GenCANN for stress induced by porcine skin deformation compared with a human-designed CANN as baseline and an LLM-generated GenCANN constrained to the baseline CANN network size

Appendix C: Analysis of the refinement process

As our baselines underwent extensive hyperparameter optimization [21, 70, 71], we allowed the LLMs to refine their initial implementations through three additional rounds. Each complete run, including these refinements, was repeated five times for every dataset. Our goal was to assess the effectiveness of the refinements and understand how the LLMs optimize their models. Because most aspects of the continuum mechanics design space are difficult to categorize or quantify, we focused our analysis on the feedforward neural network at the core of each GenCANN. Figures 16 a) and 17 a) show the best R^2 scores for each run, while Figs. 16 b) and 17 b) present the mean R^2 scores per refinement round, and Figs. 16 c) and 17 c) illustrate the network weights per round. We observed that the LLMs significantly increased the size of the feedforward networks during refinement when not given any constraints. To isolate this effect, we conducted additional experiments where the LLM was constrained to

Table 1 Architectural specifications of the constitutive artificial neural network (CANN), the LLM-generated CANN (GenCANN), and the GenCANN constrained to the baseline CANN network size, as shown in Figs. 3-8 and 10-15

Material	Model	Neurons per hidden layer
Brain	CANN	100
	Size-constrained GenCANN	100
	Unconstrained GenCANN	256, 128, 64, 3
Experimental rubber	CANN	16, 16
	Size-constrained GenCANN	16, 16
	Unconstrained GenCANN	64, 64, 16, 16
Synthetic rubber	CANN	16, 16
	Size-constrained GenCANN	16, 16
	Unconstrained GenCANN	32, 32
Skin	CANN	8, 16
	Size-constrained GenCANN	8, 16
	Unconstrained GenCANN	128, 128, 64, 32

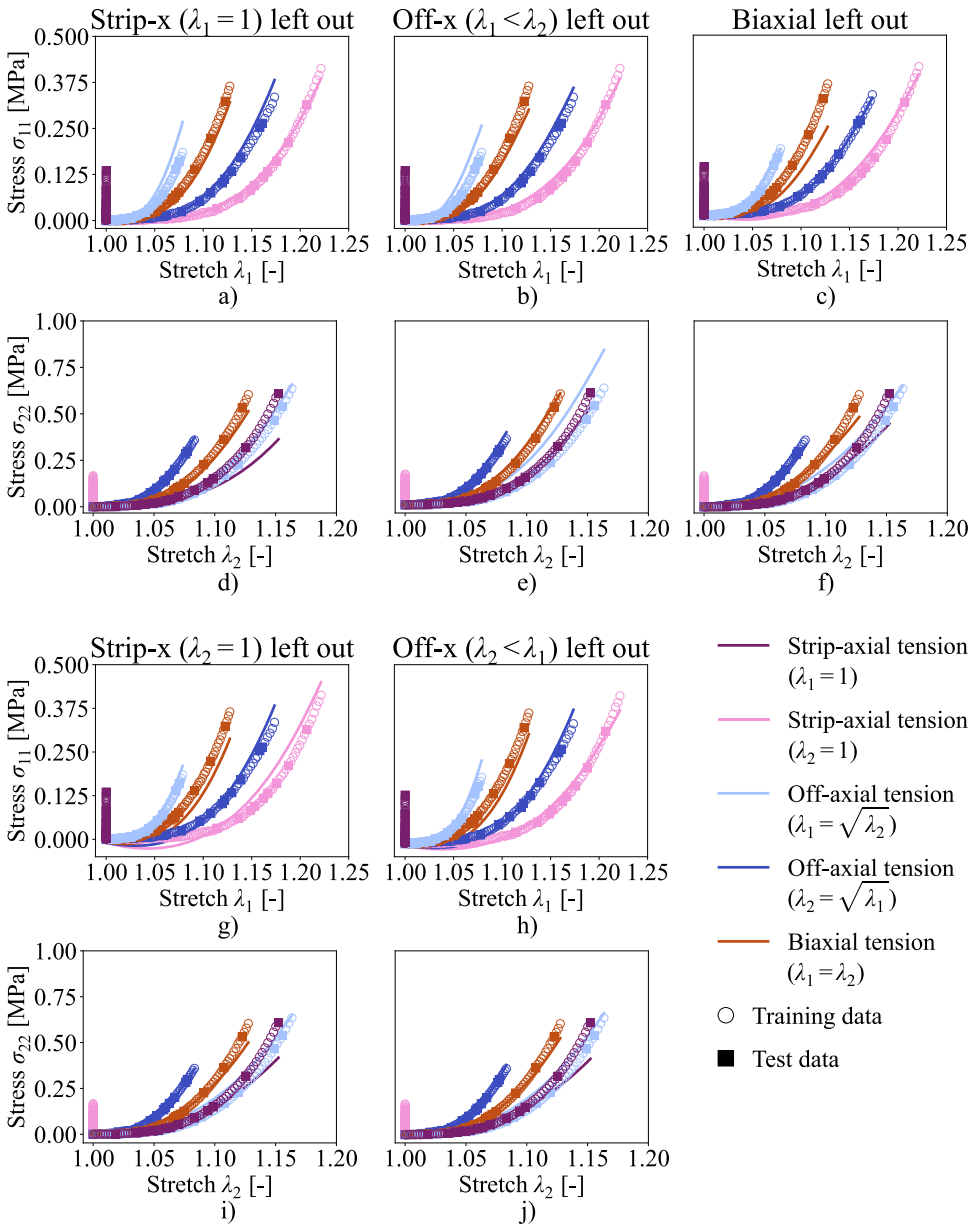

Fig. 15 Leave-one-out cross-validation. Predictions of the LLM-generated GenCANN constrained to the baseline CANN network size for stress induced by porcine skin deformation evaluated using a leave-one-loading-scenario-out cross-validation. In five separate training runs, one loading path is excluded each time and used to test how well the GenCANN generalizes to unseen loading paths

use for its GenCANNs exactly the same network sizes as the baselines. To further explore what the LLM changes during refinement, we plotted the ratio of implemented to possible changes for activation functions, initialization, and regularization settings. As expected, when the LLM was restricted from altering the number of layers and neurons, it experimented more with activation, initialization, and regularization (compare Figs. 16 d)–f) with 17 d)–f)). However, this had little impact on performance: for size-constrained GenCANNs, R^2 scores did not improve significantly across refinement rounds. In contrast, unconstrained GenCANNs showed a significant increase in R^2 scores, particularly for the brain and skin datasets. Accuracy improved and variance decreased, indicating greater reliability. This success appears to stem from enlarging the feedforward networks, which in turn underscores that network size is the most influential hyperparameter in CANNs - a fact the LLM consistently exploits during refinement when allowed. Other hyperparameter refinements contribute marginally and may be secondary in practical settings. Nevertheless, this does not alter the key finding from Section A: even when network size is fixed, size-constrained GenCANNs achieve performance levels close to their unconstrained counterparts.

Table 2 R^2 scores of the constitutive scientific generative agent (CSGA), the constitutive artificial neural network (CANN), the LLM-generated CANN (GenCANN), and the GenCANN constrained to the baseline CANN network size, as shown in Figs. 3-8 and 10-15

Test	CSGA	Size-constrained GenCANN	Unconstrained GenCANN	CANN
Brain				
Uniaxial tension	0.93	0.97	0.97	0.96
Uniaxial compression	1.00	1.00	1.00	0.99
Simple Shear	0.99	0.99	0.98	1.00
Experimental rubber				
Uniaxial tension	0.99	1.00	1.00	1.00
Equibiaxial tension	0.97	1.00	1.00	1.00
Pure shear	0.98	1.00	1.00	1.00
Synthetic rubber				
Uniaxial tension	0.87	1.00	1.00	1.00
Equibiaxial tension	0.98	1.00	1.00	1.00
Pure shear	0.93	1.00	1.00	1.00
Skin				
Strip-X ($\lambda_2 = 1$), stress in 1		0.98	1.00	0.96
Strip-X ($\lambda_2 = 1$), stress in 2		0.98	1.00	0.98
Off-X ($\lambda_2 = \sqrt{\lambda_1}$), stress in 1		0.99	1.00	0.97
Off-X ($\lambda_2 = \sqrt{\lambda_1}$), stress in 2		0.99	1.00	0.95
Equibiaxial ($\lambda_1 = \lambda_2$), stress in 1		0.89	1.00	0.92
Equibiaxial ($\lambda_1 = \lambda_2$), stress in 2		0.97	1.00	0.93
Off-X ($\lambda_1 = \sqrt{\lambda_2}$), stress in 1		0.92	1.00	0.89
Off-X ($\lambda_1 = \sqrt{\lambda_2}$), stress in 2		0.95	1.00	0.90
Strip-X ($\lambda_1 = 1$), stress in 1		0.76	1.00	0.26
Strip-X ($\lambda_1 = 1$), stress in 2		0.89	1.00	0.81

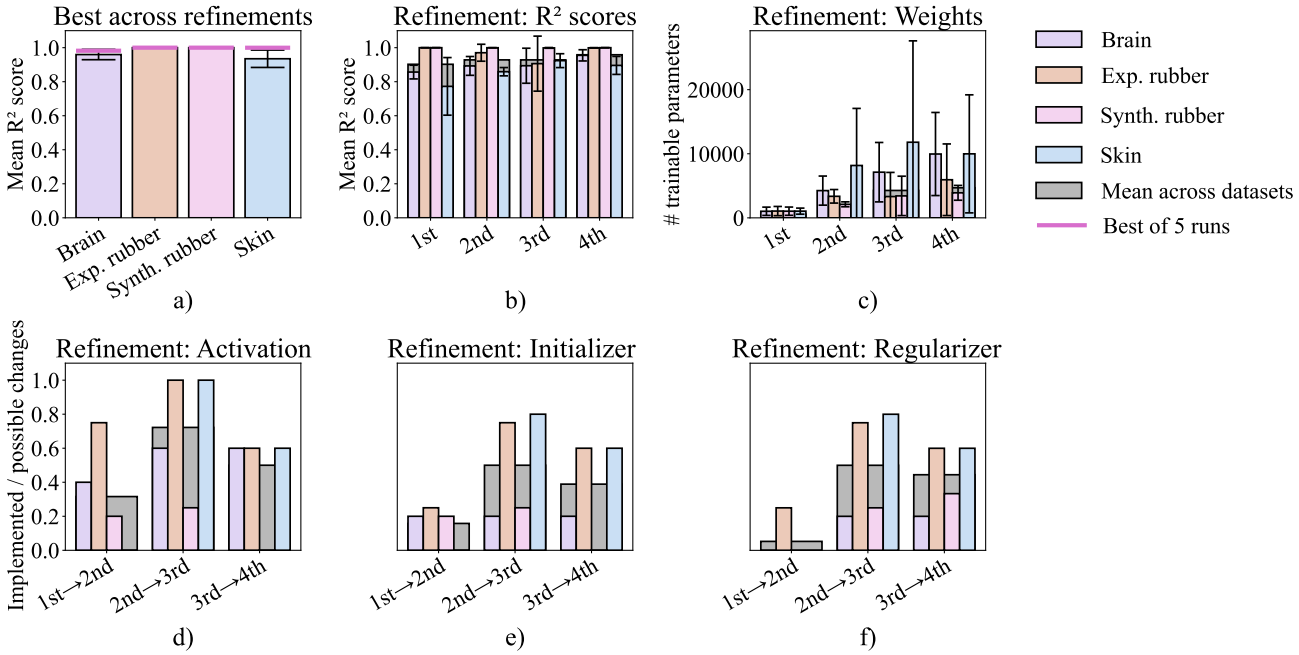

Fig. 16 Analysis of the refinement process for unconstrained generated constitutive artificial neural networks (GenCANNs). Each generated CANN undergoes three successive refinements by the LLM, resulting in four GenCANNs per run, and each run is repeated five times per dataset. Panel (**a**) presents the best-performing GenCANN from each complete run, while (**b**) shows performance per refinement round and (**c**) illustrates the numbers of network weights across refinement rounds. Panels (**d–f**) depict the ratio of implemented to possible changes for activation, initialization, and regularization, where a ratio of 1 indicates that every opportunity to test a new hyperparameter setting was utilized, and a ratio of 0 indicates that the corresponding hyperparameter remained unchanged throughout all refinement rounds

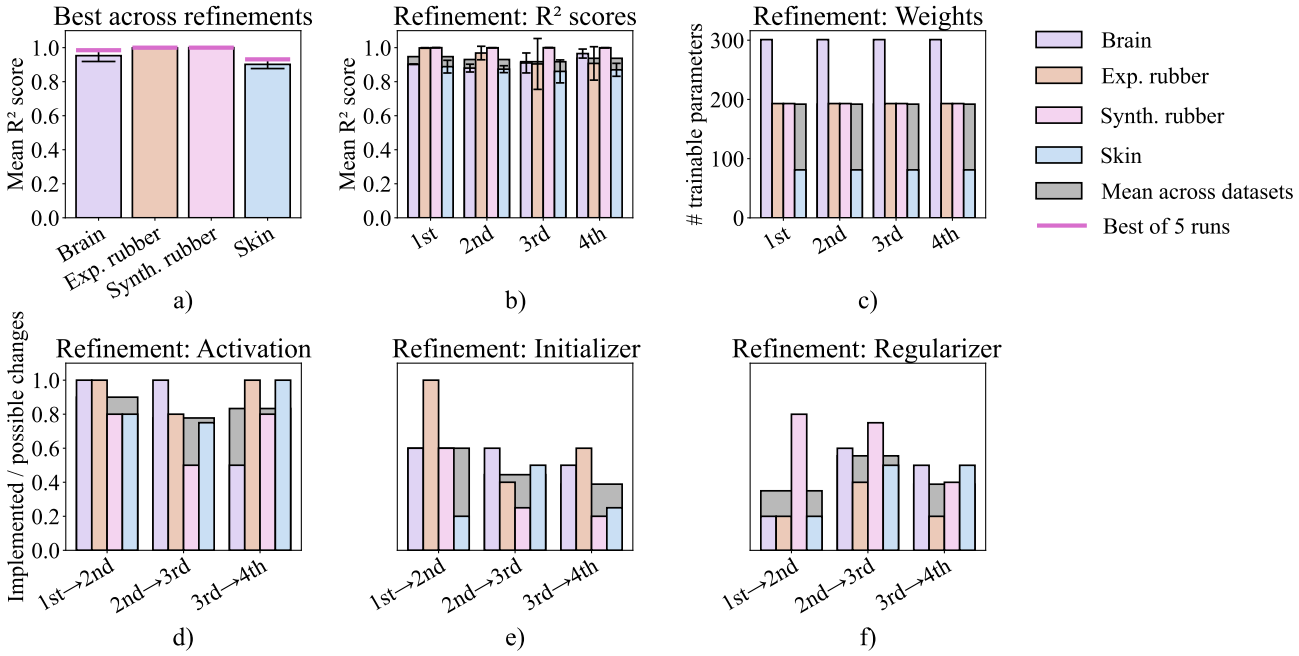

Fig. 17 Analysis of the refinement process for size-constrained generated constitutive artificial neural networks (GenCANNs). Each generated CANN undergoes three successive refinements by the LLM, resulting in four GenCANNs per run, and each run is repeated five times per dataset. Panel (**a**) presents the best-performing GenCANN from each complete run, while (**b**) shows performance per refinement round and (**c**) illustrates the numbers of network weights across refinement rounds. Panels (**d–f**) depict the ratio of implemented to possible changes for activation, initialization, and regularization, where a ratio of 1 indicates that every opportunity to test a new hyperparameter setting was utilized, and a ratio of 0 indicates that the corresponding hyperparameter remained unchanged throughout all refinement rounds

Appendix D: Exemplary GenCANN implementation

Listing 1 Full implementation of LLM-generated constitutive artificial neural network (GenCANN) for porcine skin dataset.

```
import numpy as np
import tensorflow as tf

# ----------------------------------------#
#             Helper layer                #
# ----------------------------------------#
class StructureTensor(tf.keras.layers.Layer):
    """
    Learns the fibre angle alpha  (0  alpha  Pi/2) and returns the structure tensor
    N = n  n with  n = [cos alpha, sin alpha, 0]T.
    """
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.alpha = self.add_weight(
            name        = "alpha",
            shape       = (),
            dtype       = tf.float32,
            initializer = tf.keras.initializers.RandomUniform(0.0, 0.5*np.pi),
            trainable   = True,
        )

    def call(self, F):
        batch = tf.shape(F)[0]
        n = tf.stack(
            [tf.cos(self.alpha), tf.sin(self.alpha), tf.constant(0.0, dtype=F.dtype)],
            axis=0,
        )                                       # (3,)
        n = tf.tile(n[None, :], [batch, 1])    # (B,3)
        return tf.einsum("bi,bj->bij", n, n)  # (B,3,3)

# ----------------------------------------#
#            Psi-network                  #
# ----------------------------------------#
class PsiNetwork(tf.keras.layers.Layer):
    """
    A single multivariate neural network Psi(lambdax, lambday, lambdaz, I, II, IV, V).
    Feeding both the stretches and the invariants makes it easier for the NN
    to discover patterns for extreme stretch states (strip-y, off-y, ).
    """
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.net = tf.keras.Sequential([
            tf.keras.layers.Dense(128, activation="softplus"),
            tf.keras.layers.Dense(128, activation="softplus"),
            tf.keras.layers.Dense(64 , activation="softplus"),
            tf.keras.layers.Dense(32 , activation="softplus"),
            tf.keras.layers.Dense(1  , activation="linear"),
        ])

    def call(self, features):
        return self.net(features)   # (B,1)

# ----------------------------------------#
#                  CANN                   #
# ----------------------------------------#
class CANN(tf.keras.layers.Layer):
    """
    Constitutive Artificial Neural Network for an incompressible, transversely
    isotropic material subjected to planar stretches.
    """
```

```
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.structure_tensor = StructureTensor(name="structure_tensor")
        self.psi_network      = PsiNetwork    (name="psi_network")

    # ----------------------------------------#
    #              Public forward pass        #
    # ----------------------------------------#
    def call(self, inputs):
        """
        Args:
            inputs (list | tuple): (stretch_x, stretch_y)  two tensors of
                                   shape (B,) and dtype float32.

        Returns:
            list(tf.Tensor, tf.Tensor): sigma_xx and sigma_yy (each shape (B,))
        """
        if not isinstance(inputs, (list, tuple)) or len(inputs) != 2:
            raise ValueError("CANN expects [stretch_x, stretch_y] as input.")

        lambdax = tf.reshape(inputs[0], (-1,))
        lambday = tf.reshape(inputs[1], (-1,))
        lambdaz = 1.0 / (lambdax * lambday)       # incompressibility J = 1

        # (B,3,3) deformation gradient  (diagonal for pure stretches)
        F = tf.linalg.diag(tf.stack([lambdax, lambday, lambdaz], axis=1))

        # Isochoric part of the first PK stress
        P_iso = self._compute_P_iso(F, lambdax, lambday, lambdaz)

        # Convert to Cauchy stress and eliminate pressure
        sigma = self._compute_cauchy(F, P_iso)         # (B,3,3)

        return [sigma[:, 0, 0], sigma[:, 1, 1]]

    # ----------------------------------------#
    #             Internal helpers            #
    # ----------------------------------------#
    def _compute_P_iso(self, F, lambdax, lambday, lambdaz):
        """
        Obtain P_iso = Psi/F  via automatic differentiation.
        Feature vector for Psi contains the stretches AND four invariants.
        """
        with tf.GradientTape() as tape:
            tape.watch(F)

            # Right Cauchy-Green tensor
            C  = tf.matmul(tf.transpose(F, [0, 2, 1]), F) # (B,3,3)
            trC  = tf.linalg.trace(C)                     # (B,)
            trC2 = tf.linalg.trace(tf.matmul(C, C))       # (B,)

            I1 = trC[:, None]                             # (B,1)
            I2 = (0.5 * (trC ** 2 - trC2))[:, None]    # (B,1)

            N  = self.structure_tensor(F)
            IV = tf.reduce_sum(C * N,              axis=[-2, -1])[:, None]  # (B,1)
            V  = tf.reduce_sum(tf.matmul(C, C) * N, axis=[-2, -1])[:, None] # (B,1)

            stretches = tf.stack([lambdax - 1.0, lambday - 1.0, lambdaz - 1.0], axis=1)     # (B
,3)
            invariants = tf.concat(
                [I1 - 3.0, I2 - 3.0, IV - 1.0, V - 1.0], axis=1
            ) # (B,4)

            features = tf.concat([stretches, invariants], axis=1) # (B,7)
```

```
            Psi = self.psi_network(features) # (B,1)

        return tape.gradient(Psi, F)    # (B,3,3)

    def _compute_cauchy(self, F, P_iso):
        """
        sigma = (1/J) * P_iso * FT    p I,  with p chosen such that sigma_zz = 0.
        """
        J  = tf.linalg.det(F)  # (B,)
        sigma = tf.matmul(P_iso, tf.transpose(F, [0, 2, 1])) / J[:, None, None] # (B,3,3)

        p  = sigma[:, 2, 2]  # (B,)
        I3 = tf.eye(3, batch_shape=[tf.shape(F)[0]], dtype=F.dtype)
        return sigma - p[:, None, None] * I3

    # ----------------------------------------#
    #           Public getter                 #
    # ----------------------------------------#
    def get_alpha(self):
        """Return the learned fibre direction angle alpha (radians)."""
        return self.structure_tensor.alpha

# ----------------------------------------#
#           Model builder                 #
# ----------------------------------------#
def build_cann_model():
    """
    Creates the Keras model that maps planar stretches to normal Cauchy stresses.

    Returns:
        tf.keras.Model
    """
    stretch_x = tf.keras.Input(shape=(), name="stretch_x")
    stretch_y = tf.keras.Input(shape=(), name="stretch_y")

    sigmax, sigmay = CANN(name="cann")([stretch_x, stretch_y])

    model = tf.keras.Model(
        inputs=[stretch_x, stretch_y],
        outputs=[sigmax, sigmay],
        name="CANN_model",
    )
    return model
```

Author Contributions All authors contributed to the conception and design of the study. M.T. led the conceptualization, methodology, software development, investigation, original draft preparation, and visualization. M.B. contributed to manuscript review and editing as well as visualization. K.A. was responsible for validation, data curation, and manuscript review and editing. J.E. contributed to the investigation, manuscript review and editing, visualization, and supervision. K.L. supported conceptualization, validation, and manuscript review and editing. C.C. contributed to conceptualization, manuscript review and editing, and supervision. R.A. contributed to conceptualization, manuscript review and editing, supervision, and project administration. All authors read and approved the final manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL. This research did not receive funding.

Data Availability The human brain tissue and porcine skin tissue datasets, along with the corresponding CANN code, are available at: https://github.com/LivingMatterLab/CANN. The rubber datasets and the corresponding CANN implementation are available in this repository: https://github.com/ConstitutiveANN/CANN. The implementation of the CSGA can be found here: https://github.com/ConstitutiveSGA/CSGA. Finally, the GenCANN code, including all prompts and along with the complete conversation logs corresponding to the results presented in this manuscript, is available at: https://github.com/gencann26/GenCANN.

Declarations

Competing interests The authors declare no competing interests.

Ethical approval No animals or human subjects were used in this study. All data are taken from the literature.

References

1. Mooney M. A theory of large elastic deformation. J Appl Phys. 1940;11(9):582–92. https://doi.org/10.1063/1.1712836.
2. Rivlin R. Large elastic deformations of isotropic materials. i. fundamental concepts. Philos Trans R Soc London Ser A Math Phys Sci. 1948;240(822):459–90. https://doi.org/10.1098/rsta.1948.0002.
3. Rivlin RS. Large elastic deformations of isotropic materials iv. further developments of the general theory. Philos Trans Royal Soc London Ser A Math Phys Sci. 1948;241(835):379–97. https://doi.org/10.1098/rsta.1948.0024.
4. Ogden RW. Large deformation isotropic elasticity-on the correlation of theory and experiment for incompressible rubberlike solids. Proc R Soc London A Math Phys Sci. 1972;326(1567):565–84. https://doi.org/10.1098/rspa.1972.0026.
5. Kirchdoerfer T, Ortiz M. Data-driven computational mechanics. Comput Methods Appl Mech Eng. 2016;304:81–101. https://doi.org/10.1016/j.cma.2016.02.001.
6. Carrara P, De Lorenzis L, Stainier L, Ortiz M. Data-driven fracture mechanics. Comput Methods Appl Mech Eng. 2020;372:113390. https://doi.org/10.1016/j.cma.2020.113390.
7. Ghaboussi J, Garrett J Jr, Wu X. Knowledge-based modeling of material behavior with neural networks. J Eng Mech. 1991;117(1):132–53. https://doi.org/10.1061/(ASCE)0733-9399(1991)117:1(132).
8. Hashash YM, Jung S, Ghaboussi J. Numerical implementation of a neural network based material model in finite element analysis. Int J Numer Meth Eng. 2004;59(7):989–1005. https://doi.org/10.1002/nme.905.
9. Sussman T, Bathe K-J. A model of incompressible isotropic hyperelastic material behavior using spline interpolations of tension-compression test data. Commun Numer Methods Eng. 2009;25(1):53–63. https://doi.org/10.1002/cnm.1105.
10. Latorre M, Montáns FJ. Extension of the sussman-bathe spline-based hyperelastic model to incompressible transversely isotropic materials. Comput Struct. 2013;122:13–26. https://doi.org/10.1016/j.compstruc.2013.01.018.
11. Dal H, Denli FA, Açan AK, Kaliske M. Data-driven hyperelasticity, part i: A canonical isotropic formulation for rubberlike materials. J Mech Phys Solids. 2023;179:105381. https://doi.org/10.1016/j.jmps.2023.105381.
12. Fuhg JN, Anantha_Padmanabha G, Bouklas N, Bahmani B, Sun W, Vlassis NN, Flaschel M, Carrara P, De_Lorenzis L. A review on data-driven constitutive laws for solids. Arch Comput Methods Eng. (2024). https://doi.org/10.1007/s11831-024-10196-2
13. Hao Z, Liu S, Zhang Y, Ying C, Feng Y, Su H, Zhu J. Physics-informed machine learning: A survey on problems, methods and applications. (2022). arXiv preprint arXiv:2211.08064
14. Liu B, Wang Y, Rabczuk T, Olofsson T, Lu W. Multi-scale modeling in thermal conductivity of polyurethane incorporated with phase change materials using physics-informed neural networks. Renew Energy. 2024;220:119565.
15. As' ad F, Avery P, Farhat C. A mechanics-informed artificial neural network approach in data-driven constitutive modeling. Int J Numer Meth Eng. 2022;123(12):2738–59. https://doi.org/10.1002/nme.6957.
16. Linden L, Klein DK, Kalina KA, Brummund J, Weeger O, Kästner M. Neural networks meet hyperelasticity: A guide to enforcing physics. J Mech Phys Solids. 2023;179:105363. https://doi.org/10.1016/j.jmps.2023.105363.
17. Friedrichs K, Dammaß F, Kalina KA, Kästner M. Precise, efficient and flexible modeling of crystallizing elastomers based on physics-augmented neural networks. Comput Methods Appl Mech Eng. 2026;455:118852.
18. Dammaß F, Kalina KA, Kästner M. Neural networks meet phase-field: A hybrid fracture model. Comput Methods Appl Mech Eng. 2025;440:117937.
19. Klein DK, Fernández M, Martin RJ, Neff P, Weeger O. Polyconvex anisotropic hyperelasticity with neural networks. J Mech Phys Solids. 2022;159:104703.
20. Lee S, Bathe K-J. A constitutive neural network for incompressible hyperelastic materials. Mach Learn Comput Sci Eng. 2025;1(2):31.
21. Linka K, Hillgärtner M, Abdolazizi KP, Aydin RC, Itskov M, Cyron CJ. Constitutive artificial neural networks: A fast and general approach to predictive data-driven constitutive modeling by deep learning. J Comput Phys. 2021;429:110010. https://doi.org/10.1016/j.jcp.2020.110010.
22. Linka K, Pierre SRS, Kuhl E. Automated model discovery for human brain using constitutive artificial neural networks. Acta Bio-

mater. 2023;160:134–51. https://doi.org/10.1016/j.actbio.2023.01.055.
23. McCulloch JA, St. Pierre SR, Linka K, Kuhl E. On sparse regression, lp-regularization, and automated model discovery. Int J Numer Meth Eng. 2024;125(14):7481. https://doi.org/10.1002/nme.7481.
24. Peirlinck M, Linka K, Hurtado JA, Holzapfel GA, Kuhl E. Democratizing biomedical simulation through automated model discovery and a universal material subroutine. Comput Mech. 2025;75(6):1703–23.
25. Abdolazizi KP, Linka K, Cyron CJ. Viscoelastic constitutive artificial neural networks (vcanns)-a framework for data-driven anisotropic nonlinear finite viscoelasticity. J Comput Phys. 2024;499:112704. https://doi.org/10.1016/j.jcp.2023.112704.
26. Zlatić M, Čanadija M. Recovering mullins damage hyperelastic behaviour with physics augmented neural networks. J Mech Phys Solids. 2024;193:105839.
27. Holthusen H, Lamm L, Brepols T, Reese S, Kuhl E. Theory and implementation of inelastic constitutive artificial neural networks. Comput Methods Appl Mech Eng. 2024;428:117063.
28. Holthusen H, Linka K, Kuhl E, Brepols T. A generalized dual potential for inelastic constitutive artificial neural networks: A jax implementation at finite strains. J Mech Phys Solids. (2025):106337
29. Tushar, Kumar S, Chakraborty S. Fusion-based constitutive model (fuce): Toward model-data augmentation in constitutive modeling. Int J Mech Syst Dyn. 2025;5(1):86–100. https://doi.org/10.1002/msd2.70005.
30. Taç V, Rausch MK, Costabal FS, Tepole AB. Data-driven anisotropic finite viscoelasticity using neural ordinary differential equations. Comput Methods Appl Mech Eng. 2023;411:116046. https://doi.org/10.1016/j.cma.2023.116046.
31. Liu B, Vu-Bac N, Zhuang X, Lu W, Fu X, Rabczuk T. Al-demat: A web-based expert system platform for computationally expensive models in materials design. Adv Eng Softw. 2023;176:103398.
32. Liu B, Vu-Bac N, Rabczuk T. A stochastic multiscale method for the prediction of the thermal conductivity of polymer nanocomposites through hybrid machine learning algorithms. Compos Struct. 2021;273:114269.
33. Liu B, Liu P, Wang Y, Li Z, Lv H, Lu W, et al. Explainable machine learning for multiscale thermal conductivity modeling in polymer nanocomposites with uncertainty quantification. Compos Struct. 2025;370:119292.
34. Liu B, Vu-Bac N, Zhuang X, Fu X, Rabczuk T. Stochastic full-range multiscale modeling of thermal conductivity of polymeric carbon nanotubes composites: A machine learning approach. Compos Struct. 2022;289:115393.
35. Liu B, Vu-Bac N, Zhuang X, Fu X, Rabczuk T. Stochastic integrated machine learning based multiscale approach for the prediction of the thermal conductivity in carbon nanotube reinforced polymeric composites. Compos Sci Technol. 2022;224:109425.
36. Liu B, Lu W, Olofsson T, Zhuang X, Rabczuk T. Stochastic interpretable machine learning based multiscale modeling in thermal conductivity of polymeric graphene-enhanced composites. Compos Struct. 2024;327:117601.
37. Xia Y, Zhang C, Wang C, Liu H, Sang X, Liu R, et al. Prediction of bending strength of glass fiber reinforced methacrylate-based pipeline uv-cipp rehabilitation materials based on machine learning. Tunn Undergr Space Technol. 2023;140:105319.
38. Zhang X, Xia Y, Zhang C, Liu B, Wang C, Fang H, Wang J. A prediction model for dyke-dam piping based on data augmentation and interpretable ensemble learning. Eng Failure Anal. (2025):110174
39. Koza JR. Genetic programming: On the programming of computers by means of natural selection (complex adaptive systems). Bradford Book. 1993;1:18.
40. Abdusalamov R, Hillgärtner M, Itskov M. Automatic generation of interpretable hyperelastic material models by symbolic regression. Int J Numer Meth Eng. 2023;124(9):2093–104. https://doi.org/10.1002/nme.7203.
41. Flaschel M, Kumar S, De Lorenzis L. Unsupervised discovery of interpretable hyperelastic constitutive laws. Comput Methods Appl Mech Eng. 2021;381:113852. https://doi.org/10.1016/j.cma.2021.113852.
42. Flaschel M, Kumar S, De Lorenzis L. Discovering plasticity models without stress data. NPJ Comput Mater. 2022;8(1):91. https://doi.org/10.1038/s41524-022-00752-4.
43. Joshi A, Thakolkaran P, Zheng Y, Escande M, Flaschel M, De Lorenzis L, et al. Bayesian-euclid: Discovering hyperelastic material laws with uncertainties. Comput Methods Appl Mech Eng. 2022;398:115225. https://doi.org/10.1016/j.cma.2022.115225.
44. Thakolkaran P, Joshi A, Zheng Y, Flaschel M, De Lorenzis L, Kumar S. Nn-euclid: Deep-learning hyperelasticity without stress data. J Mech Phys Solids. 2022;169:105076. https://doi.org/10.1016/j.jmps.2022.105076.
45. Kolmogorov AN. On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables. Dokl Akad Nauk SSSR. 1956;108:179–82.
46. Liu Z, Ma P, Wang Y, Matusik W, Tegmark M. Kan 2.0: Kolmogorov-arnold networks meet science. (2024). arXiv preprint arXiv:2408.10205.
47. Liu Z, Wang Y, Vaidya S, Ruehle F, Halverson J, Soljačić M, Hou TY, Tegmark M. Kan: Kolmogorov-arnold networks. (2024). arXiv preprint arXiv:2404.19756.
48. Abdolazizi KP, Aydin RC, Cyron CJ, Linka K. Constitutive kolmogorov–arnold networks (ckans): Combining accuracy and interpretability in data-driven material modeling. J Mech Phys Solids. (2025):106212. https://doi.org/10.1016/j.jmps.2025.106212
49. Ji C, Abdolazizi KP, Holthusen H, Cyron CJ, Linka K. Inelastic constitutive kolmogorov-arnold networks: A generalized framework for automated discovery of interpretable inelastic material models. (2026). arXiv preprint arXiv:2602.17750.
50. Rios T, Lanfermann F, Menzel S. Large language model-assisted surrogate modelling for engineering optimization. In: 2024 IEEE Conference on Artificial Intelligence (CAI); (2024). p. 796–803. IEEE. https://doi.org/10.1109/CAI59869.2024.00151
51. Hao H, Zhang X, Zhou A. Large language models as surrogate models in evolutionary algorithms: A preliminary study. Swarm Evol Comput. 2024;91:101741. https://doi.org/10.1016/j.swevo.2024.101741.
52. Wuwu Q, Gao C, Chen T, Huang Y, Zhang Y, Wang J, Li J, Zhou H, Zhang S. PINNsagent: Automated PDE surrogation with large language models. In: Forty-second International Conference on Machine Learning. (2025). https://openreview.net/forum?id=RO5OGOzs6M
53. Li S, Marwah T, Shen J, Sun W, Risteski A, Yang Y, Talwalkar A. Codepde: An inference framework for llm-driven pde solver generation. (2025). arXiv preprint arXiv:2505.08783.
54. Verma S, Goyal A, Mathur A, Anand A, Ranu S. Grail: Graph edit distance and node alignment using llm-generated code. (2025). arXiv preprint arXiv:2505.02124.
55. Huang J, Xing Q, Ji J, Yang B. Code-generated graph representations using multiple LLM agents for material properties prediction. In: Forty-second International Conference on Machine Learning. (2025). https://openreview.net/forum?id=lvvMwGUam6
56. Heyer M, Zhang J, Sugisawa N, Laub J-F, Lapkin A. Automated generation of mechanistic models for chemical process digital twins using reinforcement learning-part i: Conceptual framework and equation generation. ChemRxiv preprint. (2025). https://doi.org/10.26434/chemrxiv-2025-r70bs-v3
57. Chen A, Stanton SD, Alberstein RG, Watkins AM, Bonneau R, Gligorijevic V, Cho K, Frey NC. LLMs are highly-constrained bio-

physical sequence optimizers. In: NeurIPS 2024 Workshop on AI for New Drug Modalities; (2024). https://openreview.net/forum?id=SzfRVq8X07
58. Pandey S, Xu R, Wang W, Chu X. Openfoamgpt: a rag-augmented llm agent for openfoam-based computational fluid dynamics. (2025). arXiv preprint arXiv:2501.06327.
59. Ma P, Wang T-H, Guo M, Sun Z, Tenenbaum JB, Rus D, Gan C, Matusik W.LLM and simulation as bilevel optimizers: A new paradigm to advance physical scientific discovery. In: Forty-first International Conference on Machine Learning; (2024). https://openreview.net/forum?id=hz8cFsdz7P
60. Tacke M, Busch M, Bali K, Abdolazizi K, Linka K, Cyron C, et al. Constitutive scientific generative agent (csga): Leveraging large language models for automated constitutive model discovery. Mach Learn Comput Sci Eng. 2025;1(1):23. https://doi.org/10.1007/s44379-025-00022-2.
61. Ni B, Buehler MJ. Mechagents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and integrate knowledge. Extreme Mech Lett. 2024;67:102131. https://doi.org/10.1016/j.eml.2024.102131.
62. Shi Z, Xin C, Huo T, Jiang Y, Wu B, Chen X, et al. A fine-tuned large language model based molecular dynamics agent for code generation to obtain material thermodynamic parameters. Sci Rep. 2025;15(1):10295. https://doi.org/10.1038/s41598-025-92337-6.
63. Wang Y, Wu Z, Yao J, Su J. Tdag: A multi-agent framework based on dynamic task decomposition and agent generation. Neural Netw. 2025;185:107200.
64. Chen G, Dong S, Shu Y, Zhang G, Sesay J, Karlsson B, Fu J, Shi Y. Autoagents: a framework for automatic agent generation. In: Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence; (2024). p. 22–30. https://doi.org/10.24963/ijcai.2024/3
65. Tang J, Fan T, Huang C. Autoagent: A fully-automated and zero-code framework for llm agents. (2025). arXiv preprint arXiv:2502.05957.
66. Liu Z, Zhang Y, Li P, Liu Y, Yang D. A dynamic LLM-powered agent network for task-oriented agent collaboration. In: First Conference on Language Modeling; (2024). https://openreview.net/forum?id=XII0Wp1XA9
67. Nettem G, Disha M, Aavish GJ, Prasad SS, Natarajan S. Agentflow: A context aware multi-agent framework for dynamic agent collaboration. In: Proceedings of the 17th International Conference on Agents and Artificial Intelligence; (2025). p. 687–93. https://doi.org/10.5220/0013375700003890
68. Yuan S, Song K, Chen J, Tan X, Li D, Yang D. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms. In: Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies; (2025). p. 6192–217. doi: https://doi.org/10.18653/v1/2025.naacl-long.315
69. Holzapfel GA. Nonlinear solid mechanics. A Continuum Approach for Engineering, second print edn. John Wiley & Sons, Chichester (2001).
70. Pierre SRS, Linka K, Kuhl E. Principal-stretch-based constitutive neural networks autonomously discover a subclass of ogden models for human brain tissue. Brain Multiphys. 2023;4:100066. https://doi.org/10.1016/j.brain.2023.100066.
71. Linka K, Tepole AB, Holzapfel GA, Kuhl E. Automated model discovery for skin: Discovering the best model, data, and experiment. Comput Methods Appl Mech Eng. 2023;410:116007. https://doi.org/10.1016/j.cma.2023.116007.
72. Budday S, Sommer G, Birkl C, Langkammer C, Haybaeck J, Kohnert J, et al. Mechanical characterization of human brain tissue. Acta Biomater. 2017;48:319–40. https://doi.org/10.1016/j.actbio.2016.10.036.
73. Budday S, Sommer G, Haybaeck J, Steinmann P, Holzapfel GA, Kuhl E. Rheological characterization of human brain tissue. Acta Biomater. 2017;60:315–29. https://doi.org/10.1016/j.actbio.2017.06.024.
74. Budday S, Ovaert TC, Holzapfel GA, Steinmann P, Kuhl E. Fifty shades of brain: a review on the mechanical testing and modeling of brain tissue. Arch Comput Methods Eng. 2019. https://doi.org/10.1007/s11831-019-09352-w.
75. Treloar L. Stress-strain data for vulcanised rubber under various types of deformation. Trans Faraday Soc. 1944;40:59–70. https://doi.org/10.1039/TF9444000059.
76. G Treloar LR. The physics of rubber elasticity. Oxford University Press, Oxford (2005). https://doi.org/10.1093/oso/9780198570271.001.0001
77. Tac V, Sree VD, Rausch MK, Tepole AB. Data-driven modeling of the mechanical behavior of anisotropic soft biological tissue. Eng Comput. 2022;38(5):4167–82. https://doi.org/10.1007/s00366-022-01733-3.
78. Tac V, Costabal FS, Tepole AB. Data-driven tissue mechanics with polyconvex neural ordinary differential equations. Comput Methods Appl Mech Eng. 2022;398:115248. https://doi.org/10.1016/j.cma.2022.115248.
79. Liu B, Liu P, Lu W, Olofsson T. Explainable artificial intelligence (xai) for material design and engineering applications: a quantitative computational framework. Int J Mech Syst Dyn. 2025;5(2):236–65.
80. Zhang X, Xia Y, Zhang C, Wang C, Liu B, Fang H. An archimedes optimization algorithm based extreme gradient boosting model for predicting the bending strength of uv cured glass fiber reinforced polymer composites. Polym Compos. 2026;47(5):4228–45.
81. Liu B, Vu-Bac N, Zhuang X, Rabczuk T. Stochastic multiscale modeling of heat conductivity of polymeric clay nanocomposites. Mech Mater. 2020;142:103280.
82. Liu B, Penaka SR, Lu W, Feng K, Rebbling A, Olofsson T. Data-driven quantitative analysis of an integrated open digital ecosystems platform for user-centric energy retrofits: A case study in northern sweden. Technol Soc. 2023;75:102347.