COMPUTER METHODS IN MATERIALS SCIENCE

2026, vol. 26, no. 4, 5–33
https://doi.org/10.7494/cmms.2026.4.1050

Concat-tree genetic programming for graph neural architecture search: Efficient materials property classification toward high-throughput screening

Maciej Krzywda[1*], Mariusz Wermiński[2], Szymon Łukasik[1], Amir H. Gandomi[3]

[1] AGH University of Krakow, Faculty of Physics and Applied Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland.
[2] AGH University of Krakow, Faculty of Metals Engineering and Industrial Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, Poland.
[3] Faculty of Engineering and IT, University of Technology Sydney, 5 Broadway, Ultimo NSW 2007, Australia.

Abstract

We propose Concat-Tree Genetic Programming (CTGP) to automatically discover graph neural network (GNN) architectures for binary classification of DFT-derived materials properties in the JARVIS-DFT database. CTGP is inspired by Cartesian genetic programming (CGP) and classical tree-based genetic programming (GP): genotypes are binary concat trees that decode into explicit, heterogeneous sequential GNN stacks, evolved by mutation-only elitist search. Seven tasks span electronic structure (metal vs. insulator), magnetism, photovoltaic efficiency proxies, topological band character, and thermoelectric transport. Using fixed JARVIS train/validation/test splits and published binarization thresholds, we compare four search configurations – (1 + 1) and (1 + 4) regimes, each with and without a learned surrogate filter – with MLP, RF, DT, and SVM baselines and fair GNN references across four representation levels (L0–L3). L0 uses atomic number only, L1 adds learned elemental descriptors, L2 adds bond-distance summaries, and L3 adds bond-angle statistics. Each configuration uses ten independent random seeds per task. The search yields compact architectures with tens to hundreds of thousands of parameters and millisecond-scale model-forward inference. Accuracy is reported with model complexity, search cost, test AUC, macro-F1, and property-aware analysis of discovered operator sequences. Fixed GNN baselines and random search often match or exceed CTGP on test accuracy, while CTGP remains competitive on selected electronic-structure tasks and provides architecturally transparent stacks for exploratory, property-related interpretation, without claiming causal explainability. On benchmarks with a stable signal above majority-class performance, these surrogates may support rapid screening of large preprocessed graph libraries at low inference cost.

Keywords: graph neural networks, genetic programming, Cartesian genetic programming, concat-tree genetic programming, structure-property relationships surrogate methods, structure-property relationships

* Corresponding author
Authors' e-mails: krzywda@agh.edu.pl, marwer@agh.edu.pl, slukasik@agh.edu.pl, gandomi@uts.edu.au
ORCID IDs: 0000-0002-0758-8498 (M. Krzywda), 0000-0002-0212-1258 (M. Wermiński), 0000-0001-6716-610X (S. Łukasik), 0000-0002-2798-0104 (A. H. Gandomi)
Received: 31.07.2026, accepted: 8.09.2026, published: 24.11.2026
© 2026 Authors. This is an open access publication, which can be used, distributed and reproduced in any medium according to the Creative Commons CC BY 4.0 License requiring that the original work has been properly cited.

1. Introduction

The accelerated discovery of advanced functional materials – such as novel thermoelectrics, topological insulators, and photovoltaic absorbers – relies increasingly on high-throughput screening (HTS) of vast compositional and structural spaces (Choudhary et al., 2020; Jain et al., 2013). While experimental synthesis remains the ground truth, computational filtering using density functional theory (DFT) is widely employed to identify promising candidates (Curtarolo et al., 2013). However, due to the high computational cost of DFT, there is a critical need for ultrafast, accurate surrogate models (Schmidt et al., 2019). Predicting materials properties directly from crystal structure is a central problem in computational materials science (Xie & Grossman, 2018). Graph neural networks (GNNs) encode periodic crystals as atomistic graphs and learn structure-property relations through message passing. Hand-designed models such as CGCNN (Xie & Grossman, 2018) and ALIGNN (Choudhary & DeCost, 2021) show that local bonding geometry, especially bond angles, can strongly influence electronic and transport-property prediction.

Nevertheless, state-of-the-art hand-designed models are often massive and computationally heavy. For instance, architectures incorporating explicit multi-body interactions *via* line graphs significantly increase memory footprint and inference latencies (Choudhary & DeCost, 2021), which creates a computational bottleneck for large-scale high-throughput screening. Furthermore, manual architecture design is labor-intensive and can produce models whose learned decision processes remain difficult to interpret and whose inductive biases are difficult to relate to the underlying materials physics (Roscher et al., 2020). Neural architecture search (NAS) offers an alternative: architectures are discovered automatically rather than specified by hand. Cartesian genetic programming (CGP) (Miller, 2020) represents candidate programs as directed computational graphs encoded by an indexed Cartesian genotype and typically relies strongly on mutation-based variation. Classical tree-based GP, formalized by Koza (1992), represents candidate programs as hierarchical syntax trees. The proposed CTGP method draws inspiration from both paradigms by combining a concat-tree architecture representation with mutation-driven evolutionary search. In this work, we investigate whether the proposed Concat-Tree Genetic Programming (CTGP) approach can discover compact and efficient GNN architectures for binary classification of DFT-derived materials properties in the JARVIS-DFT database.The focus is placed on seven properties spanning electronic structure (metal vs. insulator from band gaps), magnetism, photovoltaic efficiency proxies, topological band character, and thermoelectric transport; tasks for which the relevant local structural descriptors differ markedly. Results are reported as completed search runs under fixed NIST train/validation/test splits and published binarization thresholds, across four node-representation levels (L0–L3). The main contributions of this study are:

- A CTGP NAS protocol for JARVIS-DFT binary property classification, using fixed train/validation/test splits, fixed classification thresholds, and ten completed independent runs per dataset–strategy–representation configuration.
- A systematic comparison of search configurations – including elitist (1 + 1) and small-population (1 + 4) – each with and without surrogate-assisted filtering against classical baselines (MLP, RF, DT, SVM) across four representation levels that progressively incorporate bond-distance and angle information.
- Fair GNN baselines under identical inputs, training, and search budget: fixed homogeneous stacks (GCN, GraphSAGE, GAT, GIN), hyperparameter tuning, and random architecture search matched to the (1 + 1) CTGP evaluation budget.
- Demonstrating the suitability of discovered models for high-throughput materials screening by explicitly reporting model complexity, search wall-clock time, training time, and millisecond-scale inference latency, characterizing the critical accuracy-efficiency trade-off.
- A property-aware analysis of discovered architectures, bridging machine learning and solid-state physics by relating explicit operator choices and representation depth to the physical sensitivity of each target quantity (e.g., hybridization in band-gaps vs. multi-center exchange in magnetism).

2. Related work

2.1. Graph neural networks for materials

Crystal-structure property prediction is commonly formulated as graph learning, where atoms are nodes and bonds or neighbor relations define edges. CGCNN introduced convolutional message passing on periodic graphs with hand-crafted node and edge features. Choudhary and DeCost (2021) showed that many electronic properties depend sensitively on bond angles and local geometric distortions, and proposed ALIGNN, which performs message passing on both the bond graph and its line graph so that triplet (angle) information is propagated explicitly. The representation hierarchy (L0–L3) employed in this study is motivated by the

same observation that atomic number alone is a weak descriptor for bandgap-related quantities, whereas bond distances and angle statistics provide the many-body context that governs hybridization and band character. The same official JARVIS-DFT splits and test-accuracy metrics as in prior work are used, but the target is discovered rather than hand-designed architectures.

2.2. Evolutionary data-driven modeling in materials

Evolutionary computation has a substantial history in materials design, process modeling, and surrogate construction. An early review by Chakraborti (2004) documented applications of genetic algorithms across atomistic materials design, alloy design, and materials processing. Evolutionary neural networks have subsequently been used as data-driven surrogate models; for example, Bhattacharya et al. (2009) combined molecular-dynamics data for the Fe–Zn system with evolutionary neural networks and multi-objective genetic optimization. Tree-based GP has also been investigated as an alternative to neural surrogates. Dimitriu et al. (2009) directly compared neural networks and GP for modeling the strength of ferritic steels, while Giri et al. (2013) introduced bi-objective genetic programming (BioGP), balancing prediction accuracy and symbolic-model complexity, and compared it with an evolutionary neural network (EvoNN) using noisy blast-furnace data. EvoNN and BioGP were subsequently used as surrogate models for cellular-automata simulations of microstructural evolution in dual-phase steel (Halder et al., 2015). These studies establish both neural and tree-based evolutionary surrogate modeling as established approaches in the materials domain. GP has also been applied directly to computational and high-throughput materials data. Yuan and Mueller (2017) used symbolic GP to identify compact predictive models for dielectric breakdown strength from high-throughput materials data. Hernandez et al. (2019) subsequently employed GP-based symbolic regression to construct computationally efficient many-body interatomic potentials from DFT training data. Evolutionary methods have also progressed beyond shallow neural surrogates: Roy et al. (2020) developed the EvoDN/EvoDN2 evolutionary deep-neural-network framework for modeling mechanical properties of micro-alloyed steels, and Mahanta and Chakraborti (2021) applied EvoDN2 to nonlinear and noisy blast-furnace data, comparing it with BioGP and EvoNN. The broader development of evolutionary neural networks, GP, multi-objective optimization, and evolutionary deep learning for materials applications is reviewed comprehensively by Chakraborti (2023).

2.3. Genetic programming and tree-based representations

Genetic programming, formalized by Koza (1992), extends evolutionary computation to the automatic generation of executable computer programs. In classical tree-based GP, candidate solutions are commonly represented as hierarchical syntax trees whose internal nodes correspond to functions or operators and whose terminal nodes represent variables, constants, or other primitive inputs. Evolution proceeds through genetic variation operators such as subtree mutation and crossover, with selection favoring programs that achieve higher fitness. The tree representation is particularly suitable for problems in which solutions are naturally expressed as hierarchical compositions of reusable operations. Beyond symbolic regression and program synthesis, GP has also been employed as a representation for neural architecture design, where the evolved tree specifies the structure of a neural network rather than directly acting as the predictive model (Suganuma et al., 2017). This distinction is directly relevant to the present work: CTGP retains the hierarchical compositional structure of tree-based GP, but its terminal elements encode parameterized GNN operators and the resulting tree is decoded into a trainable neural architecture.

2.4. Cartesian genetic programming and CTGP

Cartesian genetic programming represents candidate programs as directed computational graphs encoded using a Cartesian genotype and commonly relies on mutation as its principal variation mechanism (Miller, 2020). In contrast to the tree-based representation introduced above, CGP typically separates genotype structure from the expressed active computational graph and may contain inactive genetic material. The proposed CTGP method draws on both paradigms: it adopts the hierarchical compositional representation characteristic of tree-based GP while using a mutation-driven evolutionary strategy inspired by CGP. CTGP therefore constitutes a distinct GP representation rather than a variant of canonical CGP.

2.5. Neural architecture search and evolutionary search

Neural architecture search (NAS) automates model design through reinforcement learning, gradient-based methods, and evolutionary algorithms (Elsken et al., 2019). Evolutionary approaches have also been applied

specifically to GNN design. Krzywda (2025) investigated GNN architecture search using a hybrid genetic algorithm with parallel tempering, while Krzywda, Liu, Łukasik and Gandomi (2025) applied linear GP to the automatic design of GNN architectures for node classification. Related work also explored evolutionary optimization of graph convolutional networks for node-classification tasks (Krzywda, Łukasik & Gandomi, 2025a) and CGP for the joint evolution of architecture and training choices in contrastive graph clustering (Krzywda, Łukasik & Gandomi, 2025b). These studies demonstrate that evolutionary representations can be effectively applied to graph-learning architectures, but they focus primarily on generic node classification, graph clustering, or general GNN architecture search rather than periodic crystal graphs and materials-property prediction.

2.6. Surrogate-assisted evolution

Fitness evaluation is particularly expensive in NAS because each candidate architecture must normally be trained before its quality can be assessed (Elsken et al., 2019). Surrogate-assisted evolutionary algorithms reduce this cost by learning an approximate relation between candidate representations and expensive fitness evaluations. This strategy has also been demonstrated directly in computational materials discovery. Jennings et al. (2019), for example, coupled a genetic algorithm with an on-the-fly machine-learning surrogate to reduce the number of expensive energy calculations required during materials search. In the present framework, a pairwise MLP surrogate instead predicts whether a mutated architecture is likely to improve upon its parent. Candidates predicted to be unpromising are rejected before full GNN training, reducing the number of expensive architecture evaluations.

2.7. Deep learning and the complexity of DFT-derived targets

The present results should not be interpreted as demonstrating that the full information content of DFT calculations can be resolved without deep learning. The benchmarks considered here are binary classification problems derived from selected scalar DFT quantities and do not require reconstruction of the complete electronic structure, wavefunctions, charge density, or many-body response. Consequently, informative physically motivated descriptors can be sufficient for high predictive accuracy on some tasks, as also demonstrated by the competitive performance of classical machine-learning baselines. However, good predictive performance on a specific benchmark does not imply that the underlying electronic-structure physics has been fully resolved. Properties governed by hybridization, magnetic exchange, spin-orbit coupling, optical transitions, and transport may depend on nonlinear many-body interactions that are incompletely represented by coarse fixed-length descriptors. GNNs provide a complementary approach by learning relational structural information directly through message passing. The architectures evolved in this work are themselves neural/deep-learning models, although they are intentionally compact and contain only one to six graph-convolution layers. Our results therefore support a task-dependent view: classical methods may be sufficient when informative descriptors capture the dominant predictive signal, whereas graph-based deep learning becomes increasingly useful when the target depends strongly on connectivity, local geometry, and higher-order structural interactions.

2.8. Positioning of the present work

The studies above demonstrate that evolutionary data-driven modeling is already well established in materials science and encompasses evolutionary neural networks, tree-based symbolic GP, evolutionary deep learning, and surrogate-assisted evolutionary optimization. The novelty of the present study therefore does not lie in introducing evolutionary learning to materials modeling. Rather, we investigate the use of CTGP as a NAS mechanism for discovering compact, heterogeneous GNN architectures operating directly on periodic crystal graphs across multiple JARVIS-DFT classification benchmarks. In addition to predictive performance, the study explicitly analyses architecture composition, model complexity, evolutionary search cost, and inference latency, thereby focusing on the accuracy–efficiency trade-off relevant to high-throughput materials screening.

3. Methods

The methodology establishes a CTGP framework for NAS of lightweight GNN architectures for materials property classification. CTGP combines a binary concat-tree genotype, inspired by classical tree-based GP, with a mutation-driven evolutionary search strategy inspired by CGP. Candidate genotypes are decoded into sequential heterogeneous GNN phenotypes composed of GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), GIN (Xu et al., 2019), and simplified graph convolution (SG) layers (Wu et al., 2019).

The experimental design evaluates the resulting architectures on periodic crystal structures from the JARVIS-DFT database across seven binary classification tasks. To investigate the contribution of progressively richer structural information, four hierarchical node-representation levels (L0–L3) are considered. The architecture search is evaluated under multiple evolutionary regimes, including elitist (1 + 1) and small-population (1 + 4) strategies, each studied with and without surrogate-assisted filtering. Predictive performance is analyzed jointly with architecture complexity, search cost, and inference efficiency.

3.1. JARVIS-DFT datasets and classification task

Evaluations are performed on crystal structures from the JARVIS-DFT database (Choudhary & DeCost, 2021; Choudhary et al., 2020), using seven binary classification tasks derived from scalar DFT properties (Tab. 1). Each task compares a computed quantity to a fixed threshold, yielding a positive and negative class with a distinct materials-physics interpretation: bandgap thresholds separate metals from insulators (OptB88vdW and meta-GGA MBJ functionals); the magnetic-moment threshold distinguishes materials with appreciable local spin density; SLME flags compounds with high spectroscopic limited maximum efficiency as photovoltaic candidates; spillage proxies strong spin–orbit-induced band inversion relevant to topological character; Seebeck coefficient and power-factor thresholds identify promising p-type and n-type thermoelectric regimes. Train, validation, and test sets are fixed lists of structure IDs (JIDs) published by NIST; structures are never mixed across splits (Tab. 2).

Tab. 1. JARVIS-DFT binary classification benchmarks

Target	Benchmark ID	Property	Threshold	Class
metal_opt	*optb88vdw_bandgap*	OPT bandgap	≤0.01 ev	metal
metal_mbj	*mbj_bandgap*	MBJ bandgap	≤0.01 ev	metal
magnetic	*magmom_oszicar*	magnetic moment (OSZICAR)	$\|\mu_B\| \geq 0.05$	magnetic
slme	*slme*	spectroscopic limited max efficiency (SLME)	≥10 %	high SLME
spillage	*spillage*	band-structure spillage	≥0.1	high spillage
seebeck_p	*p_seebeck*	p-type Seebeck coefficient	≥100 μV/K	high Seebeck
powerfactor_n	*n_powerfact*	n-type power factor	≥1000 μW/(mK^2)	high power factor

Tab. 2. Fixed train/validation/test split sizes and class balance for each datasets

Target dataset	Split	Total samples	Negative class		Positive class	
			Number of samples	%	Number of samples	%
metal_opt	total	55,713	37,768	–	17,945	–
	train	44,569	30,217	67.8	14,352	32.2
	val.	5,572	3,774	67.7	1,798	32.3
	test	5,572	3,777	67.8	1,795	32.2
metal_mbj	total	18,167	9,808	–	8,359	–
	train	14,535	7,860	54.1	6,675	45.9
	val.	1,817	973	53.5	844	46.5
	test	1,815	975	53.7	840	46.3
magnetic	total	52,210	40,788	–	11,422	–
	train	41,766	32,545	77.9	9,221	22.1
	val.	5,222	4,130	79.1	1,092	20.9
	test	5,222	4,113	78.8	1,109	21.2
slme	total	9,062	5,214	–	3,848	–
	train	7,250	4,188	57.8	3,062	42.2
	val.	906	518	57.2	388	42.8
	test	906	508	56.1	398	43.9

Tab. 2 (cont.)

Target dataset	Split	Total samples	Negative class		Positive class	
			Number of samples	%	Number of samples	%
spillage	total	11,375	3,267	–	8,108	–
	train	9,101	2,628	28.9	6,473	71.1
	val.	1,137	328	28.8	809	71.2
	test	1,137	311	27.4	826	72.6
seebeck_p	total	23,210	13,845	–	9,365	–
	train	18,568	11,017	59.3	7,551	40.7
	val.	2,321	1,407	60.6	914	39.4
	test	2,321	1,421	61.2	900	38.8
powerfactor_n	total	23,210	16,035	–	7,175	–
	train	18,568	12,830	69.1	5,738	30.9
	val.	2,321	1,610	69.4	711	30.6
	test	2,321	1,595	68.7	726	31.3

For every structure, a k-nearest-neighbor graph is constructed with a cutoff of 8 Å and at most 12 neighbors per atom. Node representations range from atomic number only (L0) to learned elemental descriptors (L1), elemental descriptors plus radial-basis summaries of neighbor distances (L2), and the L2 descriptors plus bond-angle statistics (L3), as defined in Tab. 3. From a materials physics perspective, this hierarchy is intentionally designed to isolate the physical drivers of each macroscopic property:

- L0 and L1 capture basic stoichiometry and elemental properties (e.g., electronegativity, valence electrons) (Xie & Grossman, 2018).
- L2 explicitly incorporates interatomic distances *via* RBF encoding. Distance is the primary geometrical parameter governing orbital overlap and hopping integrals in tight-binding models of crystal lattices (Schütt et al., 2018).
- L3 introduces angular statistics, which are vital for determining the local coordination environment, structural distortions, and orbital hybridization (e.g., distinguishing tetrahedral from octahedral coordination). These angular relationships dictate crystal-field splitting, which fundamentally controls electronic band gaps, magnetic exchange interactions, and topological states (Choudhary et al., 2020; Klicpera et al., 2020).

By maintaining a constant graph topology (fixed at an 8 Å cutoff to capture up to the second or third coordination shell) and only varying the node feature dimensionality across levels, this experimental design precisely isolates how much predictive signal each macroscopic property draws from multi-body local geometry versus mere elemental composition.

3.2. Concat-tree genetic programming for GNN search

CTGP is introduced as an evolutionary representation for NAS in which candidate GNN architectures are encoded as variable-depth binary concat trees. The leaf nodes represent parameterized graph-convolution operations, whereas concat nodes define the composition and ordering of these operations. The genotype therefore specifies both the structure of the candidate network and its associated architectural hyperparameters. After decoding, the resulting phenotype is instantiated as a sequential GNN followed by global mean pooling and a multi-layer perceptron classifier. CTGP draws inspiration from both classical tree-based GP (Koza, 1992) and CGP (Miller, 2020), while constituting a distinct representation. In conventional tree-based GP, the evolved tree commonly represents an executable symbolic expression and therefore directly constitutes the predictive model.

In CTGP, by contrast, the tree represents a neural architecture specification rather than the predictor itself: the decoded GNN must subsequently be trained using gradient-based optimization. At the same time, CTGP adopts the mutation-driven search philosophy commonly associated with CGP, but it does *not* employ the canonical Cartesian genotype, fixed computational grid, or inactive-node representation of standard CGP. The term "concat-tree" therefore refers explicitly to the tree-structured genotype used to compose parameterized neural operators. A graph-neural phenotype was selected because periodic crystal structures can naturally be represented as atomistic graphs, enabling message passing to exploit atomic connectivity and local structural environments directly. A conventional symbolic GP predictor would instead require the crystal structure to be

reduced to a fixed-length engineered descriptor vector before prediction. The objective of CTGP is therefore not symbolic regression, but the evolutionary discovery of compact GNN architectures operating directly on crystal graphs. This use of GP as an architecture-generation mechanism is consistent with earlier GP-based NAS approaches, while CTGP specifically targets heterogeneous graph-convolution stacks. Fig. 1 illustrates the genotype-to-phenotype mapping and the mutation operators used in CTGP, whereas Fig. 2 presents representative decoded architectures obtained for the L0–L3 material-representation levels. CTGP uses mutation-only evolutionary variation. This design is inspired by the mutation-dominated search commonly used in CGP and allows local modifications of an existing architecture while retaining much of its current structure.

Tab. 3. Graph node representation levels used in this work

Level	Features	Dimensions	Description
L0	atomic number Z	1	Baseline node representation; matches JARVIS' leaderboard results runs
L1	CGCNN atom embeddings (JARVIS-Tools)	92	ALIGNN-style atomic features
L2	L1 + RBF bond-distance encoding	102	L1 descriptors augmented with 10 Gaussian radial-basis features computed from neighbor distances [Å]. The RBF centers were 0.50, 1.39, 2.28, 3.17, 4.06, 4.94, 5.83, 6.72, 7.61, and 8.50 Å. Neighbor contributions were aggregated using the arithmetic mean of the per-neighbor Gaussian RBF responses, and the resulting features were concatenated to the L1 embedding without additional scaling or normalization
L3	L2 + bond-angle statistics	106	L2 descriptors augmented with the mean, standard deviation, minimum, and maximum cosine of all neighbor-pair angles around each atom, computed from periodic-image-corrected bond vectors. For atoms with fewer than two neighbors, all angular descriptors were set to zero

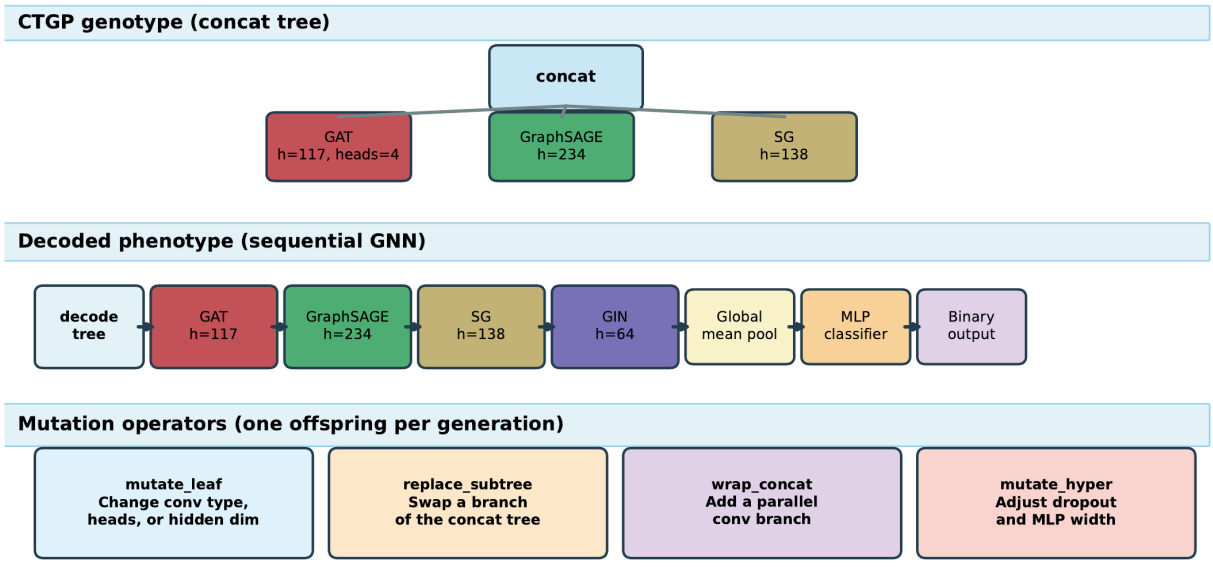


Fig. 1. CTGP genotype as a concat tree (a); decoded sequential GNN phenotype (b), and four mutation operators (c). Crossover is not used

In the present implementation, mutation can modify graph-convolution operators and their hyperparameters, replace or restructure parts of the concat tree, or introduce additional concat structure, as summarized in Fig. 1. Crossover is intentionally excluded from the present study in order to keep the search procedure controlled and to isolate the effect of local mutation-driven architecture variation. Because every substantially modified phenotype must ultimately be trained and evaluated, restricting variation to mutation also avoids introducing an additional experimental factor into an already computationally expensive architecture-search process. The omission of crossover should not be interpreted as implying that recombination is unsuitable for CTGP; subtree-aware or structure-preserving crossover operators constitute a relevant direction for future work.

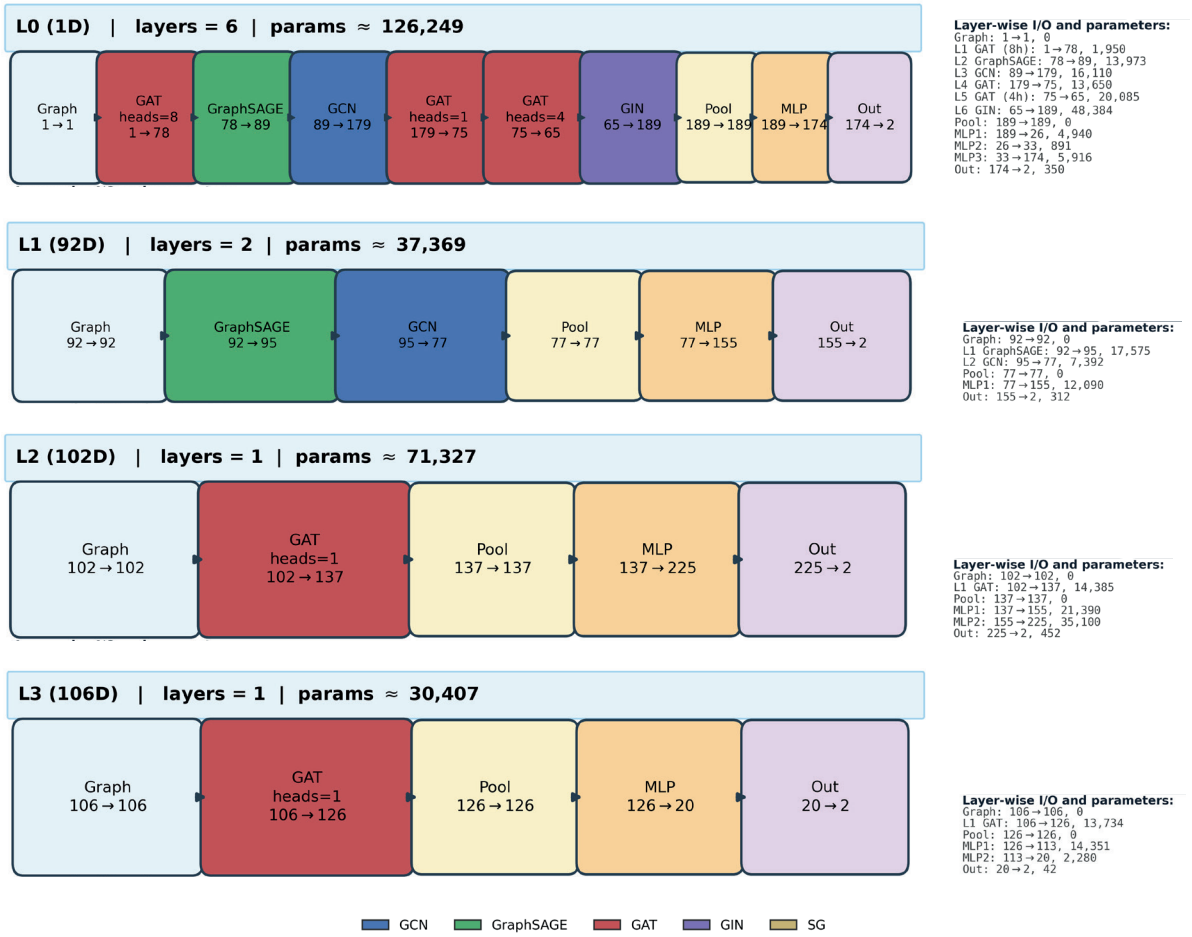


Fig. 2. Representative GNN architectures designed by CTGP at four material-representation levels: L0 (1D), L1 (92D), L2 (102D), and L3 (106D). Each panel shows the decoded message-passing architecture, including convolution types, hidden dimensions, attention heads, where applicable, and total number of trainable parameters. The architectures are presented for structural illustration; quantitative benchmark results are reported separately in Tabs 7–20

Every mutation produces one offspring architecture, which is trained with early stopping on the validation set and scored by validation AUC; the best validation model is reported on the held-out test set using test accuracy as the primary metric.

3.3. Search space

Tab. 4 summarizes the evolvable parameters. Layer types include GCN (Kipf & Welling, 2017), GraphSAGE (Hamilton et al., 2017), GAT (Veličković et al., 2018), and GIN (Xu et al., 2019); the attention-head range applies to GAT layers, and the number of convolutional layers corresponds to the number of graph-convolution operations encoded by the leaves of the concat tree and expressed in the decoded phenotype. Each evaluation trains for at most 50 epochs with a patience of 5; the outer CTGP loop runs for 300 generations. The graph construction (k-NN, 8 Å cutoff, 12 neighbors) is shared across all representation levels, and the learning rate is fixed at 10−3; only the node-feature dimensionality (L0–L3, Tab. 3) varies between levels.

Tab. 4. CTGP search space

Parameter	Range/value
Convolution type	GCN, GraphSAGE, GAT, GIN, SG
Attention heads	1, 2, 4, 8
Hidden dimension	16–256
Number of convolutional layers	1–6
Dropout	0.0–0.8
Classifier MLP layers	1–3 layers
Learning rate	10^{-3} (fixed)
Weight initialization	uniform, Xavier, Kaiming, normal
Graph construction	k-NN, 8 Å cutoff, 12 neighbors
Node features	L0–L3 (see Tab. 3)

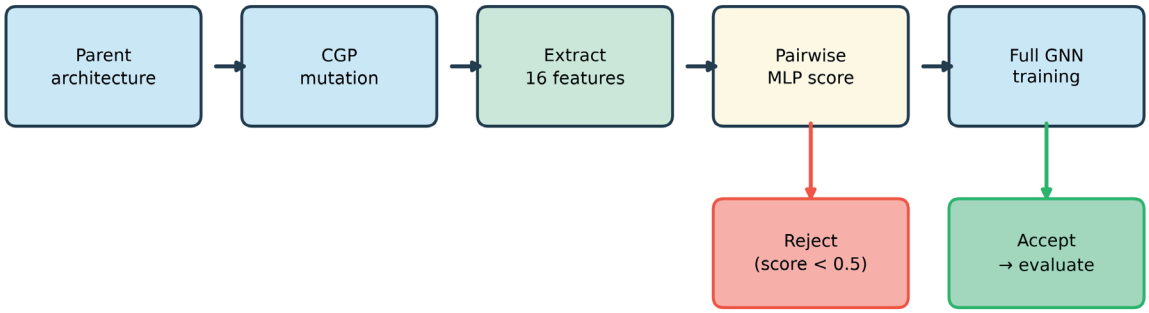


Fig. 3. Surrogate-assisted filtering: a pairwise MLP scores candidates before full GNN training

3.4. Evolutionary strategies

Two elitist mutation-only search regimes are studied, each evaluated with and without a learned surrogate filter, giving four configurations in total: (1 + 1), (1 + 1) with surrogate, (1 + 4), and (1 + 4) with surrogate:

- **(1 + 1) elitist search.** One mutation per generation; the offspring replaces the parent if validation AUC improves.
- **(1 + 4) small population.** Four offspring are generated independently in each generation and are fully trained and evaluated. The offspring with the highest validation AUC competes with the current parent, yielding up to four full candidate evaluations per generation.
- **Surrogate-assisted filtering.** Before full GNN training, a pairwise MLP surrogate scores each parent–offspring pair. Each architecture is encoded by 16 descriptors: (1) number of convolution layers; (2) mean attention-head count; (3–4) mean and standard deviation of convolution hidden widths; (5) dropout; (6) $\log_{10}$ learning rate; (7) normalized weight-initialization index; (8–10) classifier MLP depth, mean hidden width, and standard deviation of hidden widths; (11) trainable parameter count; (12–14) validation AUC, accuracy, and macro-F1; and (15–16) fractions of GCN and GraphSAGE layers in the stack. Descriptors 1–10 and 15–16 are computed from the decoded genotype and training hyperparameters alone. Descriptors 11–14 are metric-derived and are available only for architectures that have already undergone a full training run (the current parent in the elitist chain, and historically evaluated offspring used to retrain the surrogate). At pre-training filter time, no validation metric from full GNN training is available for the unevaluated offspring; we therefore set its parameter count to zero and its validation AUC, accuracy, and macro-F1 to neutral placeholders (0.5), while the parent vector retains the validation metrics from its most recent evaluation. The 32-dimensional concatenation [*x_child*; *x_parent*] is passed to a two-layer MLP that predicts P (offspring improves parent); if this probability falls below 0.5, the candidate is rejected without training. Warm-up lasts 8 generations and the surrogate is retrained every 3 generations with 40 MLP epochs. Fig. 3 illustrates the workflow. The surrogate is applied to both the (1 + 1) and (1 + 4) regimes, yielding the surrogate-assisted variants.
- **Fair GNN baselines.** To ensure a fair comparison, we consider three baseline settings. First, we evaluate fixed-stack GNNs based on GCN, GraphSAGE, GAT, and GIN, using three layers, a hidden dimension of 64, a dropout rate of 0.2, and a learning rate of 10^{-3}; for GAT, we additionally use four attention heads. Second, we consider the same fixed architectures with hyperparameter tuning, performing a grid search over learning rates $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and dropout rates $\{0.0, 0.2, 0.5\}$, resulting in 9 trials per architecture. Finally, we evaluate the random architecture search by uniformly sampling architectures from the same CTGP search space and training each candidate for the same number of epochs. We use a reference budget of 301 full architecture evaluations per run, equal to the evaluation budget of the (1 + 1) CTGP configuration. The random-search baseline therefore provides a budget-matched comparison specifically for CTGP (1 + 1); the (1 + 4) variants use a larger evaluation budget and are reported as additional search configurations.

3.5. Evaluation protocol

The published NIST train/validation/test splits and per-task binarization thresholds are adopted (Tab. 1), with test accuracy as the primary metric and test AUC as a secondary metric. Each search configuration was evaluated using ten completed independent runs for

every task and representation level. Results are reported as the mean ± standard deviation over 10 completed independent runs (n = 10) for every dataset-strategy-representation combination. Independent random seeds were derived separately for each strategy from a fixed base seed, benchmark identifier, method name, and run index; runs were therefore not paired across evolutionary strategies. Classification accuracy was computed by assigning the positive class when the predicted positive-class probability exceeded 0.5 (equivalently, argmax over the two-class softmax output). Classical baselines (MLP, RF, DT, SVM) were trained on fixed-length pooled summaries derived from the same per-atom node descriptors as the GNNs at each representation level, whereas the GNNs additionally used the k-NN graph connectivity.

For each crystal graph, baseline inputs were formed by concatenating the element-wise mean, maximum, and standard deviation of node features across atoms. Features were standardized with a StandardScaler fit on the training split only. The MLP baseline used a 128-64-2 fully connected network with ReLU activations, dropout rates of 0.3 and 0.2, Adam optimization ($lr = 10^{-3}$), and early stopping with a patience of 15 on validation AUC (up to 100 epochs). The random forest used 100 trees with a maximum depth of 10; the decision tree also used a maximum depth of 10; the support-vector machine used an RBF kernel. All sklearn baselines used fixed default hyperparameters without tuning. Each baseline run used the same deterministic seeding function as CTGP, but the seeds differed across methods because the method identifier enters the seed hash. Thus, the comparison controls descriptor content but is not a strictly identical-input comparison: only the GNN models consume explicit message passing over the crystal graph. Within each search run, model selection uses validation AUC only; test metrics are never used for selection. CTGP and fair GNN models were trained with Adam ($lr = 10^{-3}$), binary cross-entropy loss, and mini-batches of 32 graphs. Beyond predictive accuracy, model complexity (parameter count), search cost (wall-clock time and number of full evaluations), per-architecture training time, and inference latency per sample are logged.

3.6. Reproducibility and code availability

Each architecture evaluation trains one GNN with early stopping (at most 50 epochs, patience 5). Let $T \leq 50$ denote the number of epochs actually run, $|B|$ the number of training mini-batches per epoch, $|E|$ the number of edges in the k-NN crystal graph, d the hidden width, and $L \in [1, 6]$ the number of graph-convolution layers. The wall-clock cost of one full training run is:

$$O(T \cdot |B| \cdot |E| \cdot d \cdot L) \tag{1}$$

dominated by message passing over edges in each epoch. A single forward pass at inference scales as $O(|E| \cdot d \cdot L)$ per graph and does not depend on the outer search loop.

The outer CTGP evolutionary loop runs for 300 generations. In the (1 + 1) regime, one offspring is fully trained per generation (Neval ≈ 301 trainings, including the initial parent). In the (1 + 4) regime, four offspring are independently generated, fully trained, and evaluated per generation (Neval ≈ 1,201, including the initial parent). The offspring with the highest validation AUC competes with the current parent. The total search cost is therefore:

$$O(\text{Neval} \cdot T \cdot |B| \cdot |E| \cdot d \cdot L), \quad \text{with Neval} \in \{301, 1{,}201\} \tag{2}$$

before surrogate filtering. Surrogate-assisted variants score each parent–offspring pair with a lightweight pairwise multilayer perceptron (negligible cost relative to full GNN training) and reject unpromising candidates before training, reducing the realized Neval (Tab. 13). Empirically, on L0 benchmarks, the mean wall-clock search time per run is 16.9 h for (1 + 1) and 47.6 h for (1 + 4), falling to 4.9 h and 13.0 h with the surrogate (3.5× and 3.7× speed-ups, respectively; Tab. 14, Fig. 8). The final discovered models require 2–7 min to retrain; inference latency is 0.28–0.40 ms per test graph (Tab. 14).

The source code, experimental configurations, and scripts required to reproduce the results presented in this study are publicly available online (Krzywda, 2026).

4. Results

4.1. Predictive performance across JARVIS-DFT classification tasks

Tab. 5 summarizes the four CTGP strategies on L0 (atomic-number node features); the unified comparison with classical baselines appears in Tab. 7. Tab. 6 ranks the strongest methods per benchmark by mean test accuracy. Fig. 4 compares, for each property, the strongest classical baseline, the four CTGP strategies, and the ALIGNN reference on L0 inputs. RF and DT baselines lead on several band-structure-related tasks where pooled graph descriptors capture much of the signal, even without convolutional message passing. At L0, the distance from the published ALIGNN value is task-dependent. The largest deficits oc-

cur for SLME, p-type Seebeck, and the two bandgap tasks; the magnetic deficit is smaller. Because ALIGNN uses a different feature and architecture pipeline, these values are contextual rather than like-for-like. With only elemental identity as input, CTGP is competitive with the strongest classical baselines on several benchmarks while producing explicit, compact architectures. The bandgap and magnetic-moment tasks remain the most challenging at L0: metal/insulator discrimination and local-moment classification both depend on hybridization and crystal-field splitting that are poorly encoded by atomic number alone. On spillage, our best single run reached test accuracy is 0.7564, a property for which spin–orbit coupling and band inversion are sensitive to local geometry. These patterns motivate the representation-level analysis in the next subsection.

Tab. 5. CTGP evolutionary strategy results on L0 (atomic-number node features)

Method	Metric	*m_opt*	*m_mbj*	*mag.*	*slme*	*spill.*	*seeb.*	*pf_n*
CTGP 1 + 1	accuracy	0.764 ±0.044	0.762 ±0.028	0.841 ±0.018	0.611 ±0.075	0.709 ±0.088	0.750 ±0.046	0.628 ±0.138
CTGP 1 + 1	AUC	0.845 ±0.028	0.846 ±0.028	0.803 ±0.038	0.675 ±0.037	0.763 ±0.079	0.839 ±0.032	0.577 ±0.033
CTGP 1 + 1 + S	accuracy	0.666 ±0.167	0.648 ±0.106	0.699 ±0.219	0.594 ±0.074	0.711 ±0.065	0.629 ±0.125	0.655 ±0.085
CTGP 1 + 1 + S	AUC	0.733 ±0.134	0.743 ±0.082	0.652 ±0.085	0.653 ±0.048	0.688 ±0.118	0.710 ±0.109	0.526 ±0.030
CTGP 1 + 4	accuracy	0.755 ±0.061	0.719 ±0.079	0.849 ±0.036	0.617 ±0.065	0.672 ±0.172	0.713 ±0.092	0.672 ±0.045
CTGP 1 + 4	AUC	0.828 ±0.071	0.809 ±0.073	0.815 ±0.070	0.687 ±0.025	0.761 ±0.009	0.806 ±0.080	0.598 ±0.035
CTGP 1 + 4 + S	accuracy	0.642 ±0.220	0.627 ±0.128	0.803 ±0.021	0.609 ±0.070	0.656 ±0.176	0.600 ±0.155	0.640 ±0.126
CTGP 1 + 4 + S	AUC	0.802 ±0.062	0.763 ±0.062	0.699 ±0.066	0.674 ±0.012	0.762 ±0.009	0.747 ±0.057	0.572 ±0.041

Tab. 6. Top methods per benchmark ranked by mean test accuracy (L0)

Dataset	Method	Test ACC	Rank	Difference from best [pp]
metal_opt	RF	0.771	1	best
	DT	0.767	2	−0.5
	CTGP 1 + 1	0.764	3	−0.7
	CTGP 1 + 4	0.755	4	−1.7
metal_mbj	CTGP 1 + 1	0.762	1	best
	RF	0.750	2	−1.2
	DT	0.729	3	−3.2
	CTGP 1 + 4	0.719	4	−4.3
magnetic	DT	0.853	1	best
	RF	0.851	2	−0.2
	CTGP 1 + 4	0.849	3	−0.4
	CTGP 1 + 1	0.841	4	−1.2
slme	DT	0.701	1	best
	RF	0.699	2	−0.2
	SVM	0.663	3	−3.8
	MLP	0.628	4	−7.3
spillage	RF	0.764	1	best
	DT	0.748	2	−1.7
	SVM	0.740	3	−2.5
	MLP	0.739	4	−2.6

Tab. 6 (cont.)

Dataset	Method	Test ACC	Rank	Difference from best [pp]
seebeck_p	RF	0.776	1	best
	DT	0.768	2	−0.9
	CTGP 1 + 1	0.750	3	−2.6
	SVM	0.715	4	−6.1
powerfactor_n	RF	0.702	1	best
	MLP	0.687	2	−1.5
	SVM	0.687	3	−1.5
	DT	0.682	4	−2.1

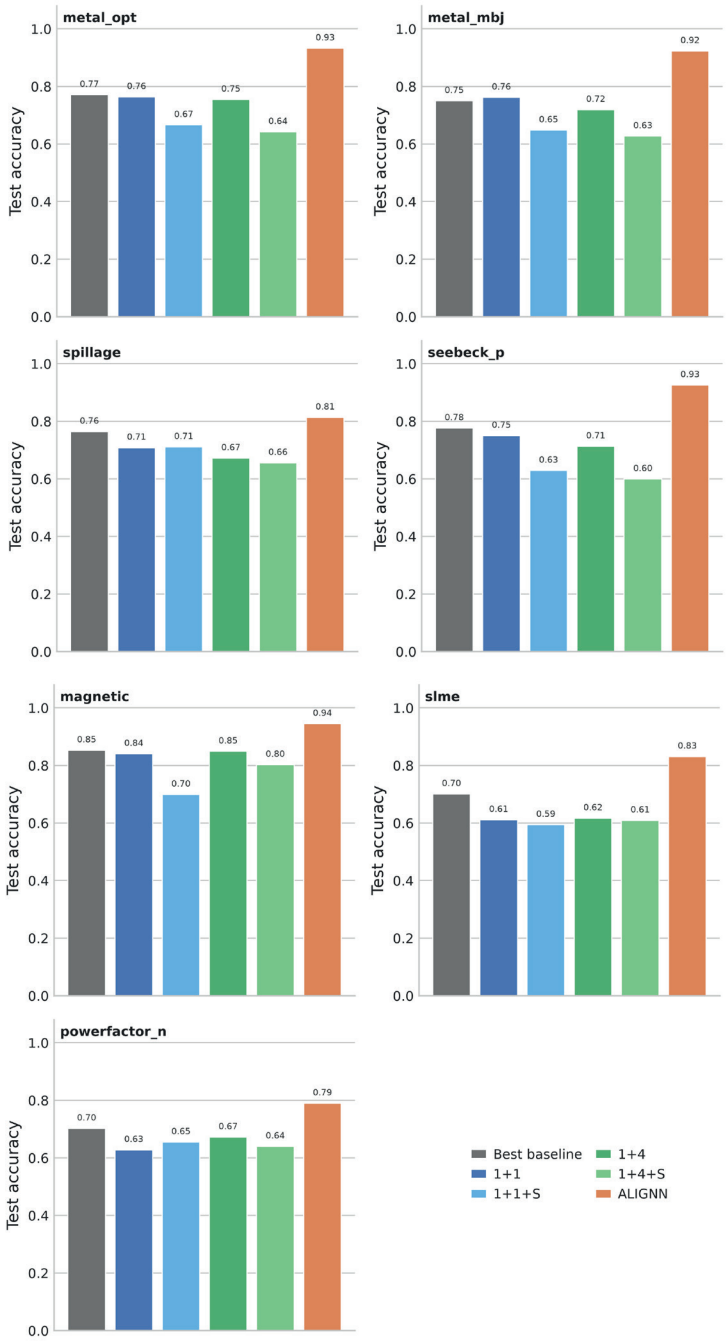


Fig. 4. Contextual comparison of L0 models with published ALIGNN leaderboard accuracy. CTGP bars show mean test accuracy ±1 SD over ten runs; ALIGNN is a single published reference value obtained with a different architecture and feature pipeline

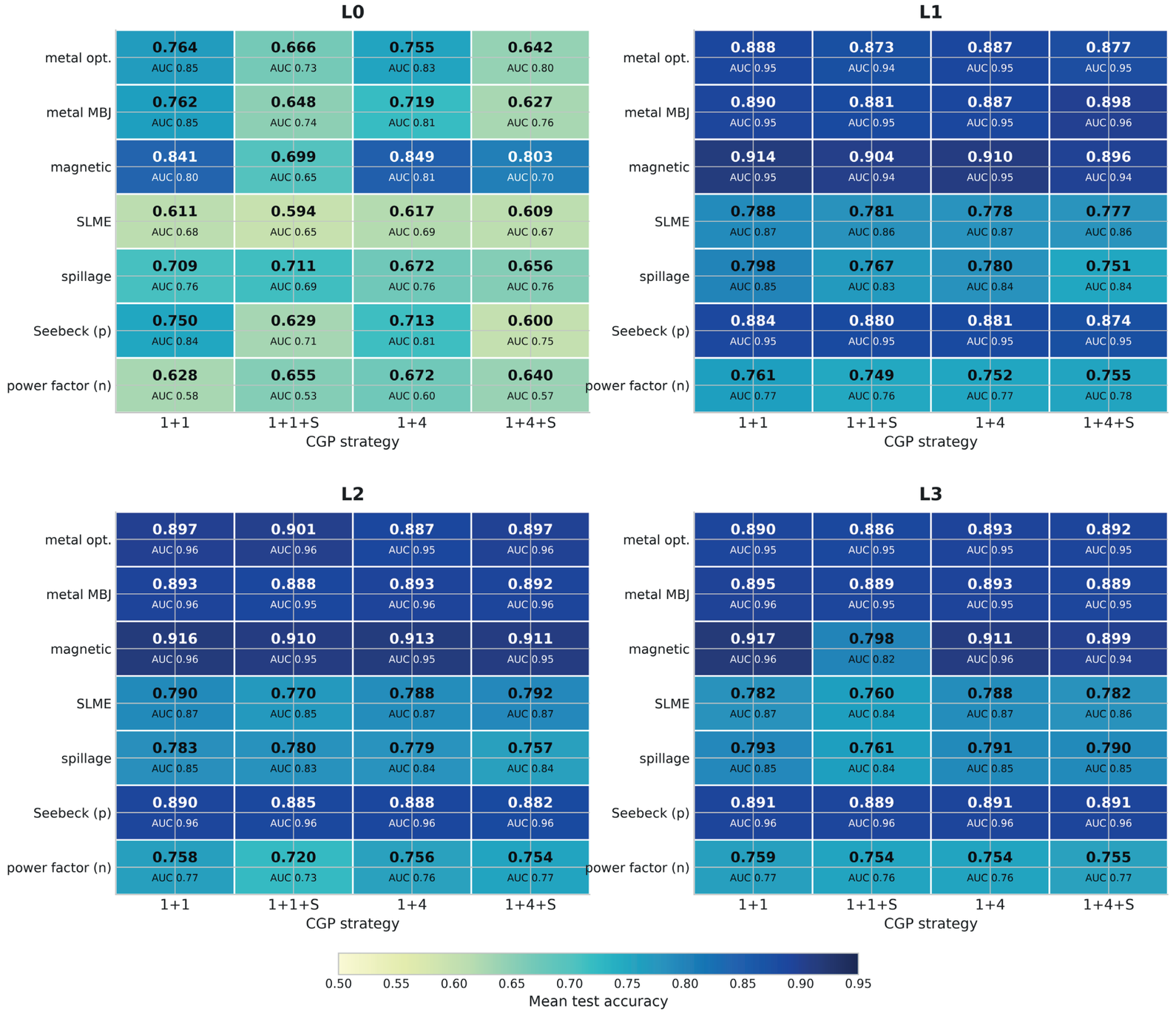


Fig. A. Mean test accuracy by dataset and CTGP strategy for each representation level (L0–L3)

4.2. Representation levels L0–L3

Tab. 6–9 present, for each representation level, a single unified comparison of the four classical baselines (MLP, RF, DT, SVM) and the four CTGP strategies (1 + 1, 1 + 1 + S, 1 + 4, 1 + 4 + S) on identical inputs. All methods at a given representation level are derived from the same underlying node descriptors. Classical models consume fixed-length pooled summaries, whereas GNN-based methods also exploit explicit graph connectivity through message passing. The comparison therefore controls descriptor content across representation levels, while not constituting a strictly identical model-input comparison. Tab. 3 defines the node-feature hierarchy, and Fig. A visualizes the mean accuracy per dataset and CTGP strategy across all four levels. The heatmaps make two patterns immediately visible: first, the L0-to-L1 gain is largest for SLME, *metal_mbj*, *seebeck_p*, and *metal_opt* in the displayed CTGP results; second, surrogate-assisted strategies are sometimes close to their plain counterparts at L1–L3, but they show clear accuracy losses in several L0 cells and a marked instability for magnetic classification at L3. These comparisons are descriptive because no formal significance test is reported.

Tab. 7. Unified comparison at representation level L0: classical baselines and CTGP strategies on identical inputs. Each cell reports mean ±SD over 10 completed runs

Dataset	Baselines				CTGP strategies			
	MLP	**RF**	**DT**	**SVM**	**1 + 1**	**1 + 1 + S**	**1 + 4**	**1 + 4 + S**
Test accuracy								
metal_opt	0.736 ±0.002	0.771 ±0.000	0.767 ±0.000	0.734 ±0.009	0.764 ±0.044	0.666 ±0.167	0.755 ±0.061	0.642 ±0.220
metal_mbj	0.674 ±0.002	0.750 ±0.000	0.729 ±0.000	0.674 ±0.009	0.762 ±0.028	0.648 ±0.106	0.719 ±0.079	0.627 ±0.128
magnetic	0.704 ±0.217	0.851 ±0.009	0.853 ±0.008	0.818 ±0.009	0.841 ±0.018	0.699 ±0.219	0.849 ±0.036	0.803 ±0.021
slme	0.628 ±0.052	0.699 ±0.000	0.701 ±0.009	0.663 ±0.000	0.611 ±0.075	0.594 ±0.074	0.617 ±0.065	0.609 ±0.070
spillage	0.739 ±0.003	0.764 ±0.000	0.748 ±0.008	0.740 ±0.009	0.709 ±0.088	0.711 ±0.065	0.672 ±0.172	0.656 ±0.176
seebeck_p	0.704 ±0.002	0.776 ±0.009	0.768 ±0.007	0.715 ±0.000	0.750 ±0.046	0.629 ±0.125	0.713 ±0.092	0.600 ±0.155
powerfactor_n	0.687 ±0.007	0.702 ±0.007	0.682 ±0.000	0.687 ±0.007	0.628 ±0.138	0.655 ±0.085	0.672 ±0.045	0.640 ±0.126
Test AUC								
metal_opt	0.744 ±0.001	0.840 ±0.000	0.822 ±0.000	0.719 ±0.000	0.845 ±0.028	0.733 ±0.134	0.828 ±0.071	0.802 ±0.062
metal_mbj	0.742 ±0.002	0.839 ±0.000	0.805 ±0.008	0.738 ±0.000	0.846 ±0.028	0.743 ±0.082	0.809 ±0.073	0.763 ±0.062
magnetic	0.659 ±0.060	0.841 ±0.000	0.817 ±0.000	0.721 ±0.000	0.803 ±0.038	0.652 ±0.085	0.815 ±0.070	0.699 ±0.066
slme	0.672 ±0.006	0.779 ±0.000	0.725 ±0.009	0.696 ±0.007	0.675 ±0.037	0.653 ±0.048	0.687 ±0.025	0.674 ±0.012
spillage	0.768 ±0.001	0.809 ±0.009	0.766 ±0.000	0.719 ±0.009	0.763 ±0.004	0.688 ±0.118	0.761 ±0.009	0.762 ±0.009
seebeck_p	0.755 ±0.001	0.856 ±0.000	0.830 ±0.000	0.755 ±0.000	0.839 ±0.032	0.710 ±0.109	0.806 ±0.080	0.747 ±0.057
powerfactor_n	0.592 ±0.013	0.680 ±0.000	0.617 ±0.009	0.507 ±0.000	0.577 ±0.033	0.526 ±0.030	0.598 ±0.035	0.572 ±0.041

Tab. 8. Unified comparison at representation level L1: classical baselines and CTGP strategies on identical inputs. Each cell reports mean ±SD over 10 completed runs

Dataset	Baselines				CTGP Strategies			
	MLP	RF	DT	SVM	1 + 1	1 + 1 + S	1 + 4	1 + 4 + S
Test accuracy								
metal_opt	0.855 ±0.001	0.873 ±0.000	0.862 ±0.000	0.864 ±0.000	0.888 ±0.006	0.873 ±0.017	0.887 ±0.009	0.877 ±0.030
metal_mbj	0.885 ±0.002	0.894 ±0.000	0.890 ±0.000	0.888 ±0.000	0.890 ±0.007	0.881 ±0.027	0.887 ±0.005	0.898 ±0.007
magnetic	0.895 ±0.002	0.899 ±0.000	0.897 ±0.000	0.901 ±0.000	0.914 ±0.004	0.904 ±0.010	0.910 ±0.002	0.896 ±0.018
slme	0.793 ±0.008	0.796 ±0.000	0.786 ±0.000	0.780 ±0.000	0.788 ±0.005	0.781 ±0.024	0.778 ±0.018	0.777 ±0.022
spillage	0.792 ±0.003	0.807 ±0.000	0.775 ±0.000	0.789 ±0.000	0.798 ±0.010	0.767 ±0.040	0.780 ±0.011	0.751 ±0.069
seebeck_p	0.870 ±0.002	0.882 ±0.000	0.854 ±0.000	0.873 ±0.000	0.884 ±0.006	0.880 ±0.011	0.881 ±0.005	0.874 ±0.020
powerfactor_n	0.755 ±0.005	0.753 ±0.007	0.739 ±0.000	0.759 ±0.000	0.761 ±0.010	0.749 ±0.018	0.752 ±0.008	0.755 ±0.008
Test AUC								
metal_opt	0.930 ±0.001	0.944 ±0.000	0.919 ±0.000	0.933 ±0.000	0.952 ±0.003	0.943 ±0.009	0.952 ±0.004	0.947 ±0.014
metal_mbj	0.948 ±0.000	0.954 ±0.000	0.911 ±0.000	0.946 ±0.000	0.953 ±0.003	0.946 ±0.010	0.951 ±0.004	0.958 ±0.000
magneti	0.934 ±0.002	0.938 ±0.000	0.887 ±0.000	0.932 ±0.000	0.952 ±0.002	0.942 ±0.010	0.949 ±0.002	0.937 ±0.014
slme	0.864 ±0.001	0.876 ±0.007	0.820 ±0.008	0.853 ±0.000	0.869 ±0.007	0.865 ±0.018	0.869 ±0.014	0.863 ±0.016
spillage	0.850 ±0.002	0.861 ±0.000	0.786 ±0.007	0.839 ±0.009	0.850 ±0.008	0.827 ±0.029	0.837 ±0.007	0.838 ±0.011
seebeck_p	0.942 ±0.000	0.952 ±0.000	0.899 ±0.009	0.939 ±0.007	0.955 ±0.003	0.952 ±0.007	0.954 ±0.002	0.949 ±0.012
powerfactor_n	0.756 ±0.003	0.784 ±0.008	0.708 ±0.007	0.772 ±0.008	0.772 ±0.013	0.762 ±0.024	0.766 ±0.017	0.778 ±0.012

Tab. 9. Unified comparison at representation level L2: classical baselines and CTGP strategies on identical inputs. Each cell reports mean ± SD over 10 completed runs

Dataset	Baselines				CTGP Strategies			
	MLP	RF	DT	SVM	1 + 1	1 + 1 + S	1 + 4	1 + 4 + S
Test accuracy								
metal_opt	0.862 ±0.002	0.883 ±0.000	0.864 ±0.000	0.875 ±0.009	0.897 ±0.005	0.901 ±0.004	0.887 ±0.010	0.897 ±0.007
metal_mbj	0.888 ±0.001	0.890 ±0.009	0.884 ±0.009	0.891 ±0.007	0.893 ±0.004	0.888 ±0.018	0.893 ±0.006	0.892 ±0.016
magnetic	0.898 ±0.001	0.899 ±0.009	0.898 ±0.008	0.906 ±0.007	0.916 ±0.004	0.910 ±0.012	0.913 ±0.010	0.911 ±0.020
slme	0.796 ±0.001	0.791 ±0.009	0.777 ±0.007	0.784 ±0.008	0.790 ±0.010	0.770 ±0.029	0.788 ±0.018	0.792 ±0.009
spillage	0.803 ±0.001	0.811 ±0.008	0.764 ±0.009	0.799 ±0.008	0.783 ±0.018	0.780 ±0.015	0.779 ±0.017	0.757 ±0.063
seebeck_p	0.871 ±0.002	0.881 ±0.009	0.865 ±0.008	0.877 ±0.008	0.890 ±0.001	0.885 ±0.013	0.888 ±0.008	0.882 ±0.018
powerfactor_n	0.754 ±0.005	0.751 ±0.008	0.726 ±0.007	0.764 ±0.009	0.758 ±0.004	0.720 ±0.061	0.756 ±0.004	0.754 ±0.020

Dataset	Baselines				CTGP Strategies			
	MLP	RF	DT	SVM	1 + 1	1 + 1 + S	1 + 4	1 + 4 + S
Test AUC								
metal_opt	0.936 ±0.001	0.946 ±0.000	0.913 ±0.000	0.943 ±0.008	0.959 ±0.003	0.957 ±0.003	0.952 ±0.006	0.958 ±0.008
metal_mbj	0.951 ±0.000	0.956 ±0.009	0.904 ±0.008	0.952 ±0.008	0.956 ±0.002	0.951 ±0.007	0.957 ±0.003	0.955 ±0.010
magnetic	0.938 ±0.002	0.941 ±0.009	0.878 ±0.009	0.938 ±0.009	0.956 ±0.003	0.948 ±0.011	0.955 ±0.006	0.952 ±0.014
slme	0.869 ±0.000	0.878 ±0.009	0.805 ±0.008	0.861 ±0.007	0.870 ±0.008	0.846 ±0.040	0.868 ±0.016	0.868 ±0.009
spillage	0.862 ±0.003	0.865 ±0.008	0.774 ±0.007	0.851 ±0.009	0.846 ±0.008	0.831 ±0.021	0.843 ±0.007	0.843 ±0.008
seebeck_p	0.946 ±0.000	0.953 ±0.009	0.905 ±0.009	0.945 ±0.007	0.958 ±0.003	0.957 ±0.005	0.957 ±0.004	0.955 ±0.004
powerfactor_n	0.758 ±0.006	0.785 ±0.009	0.705 ±0.007	0.781 ±0.008	0.772 ±0.007	0.732 ±0.058	0.761 ±0.009	0.770 ±0.022

Tab. 10. Unified comparison at representation level L3: classical baselines and CTGP strategies on identical inputs. Each cell reports mean ± SD over 10 completed runs

Dataset	Baselines				CTGP Strategies			
	MLP	RF	DT	SVM	1 + 1	1 + 1 + S	1 + 4	1 + 4 + S
Test accuracy								
metal_opt	0.871 ±0.001	0.887 ±0.009	0.867 ±0.007	0.882 ±0.007	0.890 ±0.010	0.886 ±0.014	0.893 ±0.012	0.892 ±0.017
metal_mbj	0.887 ±0.009	0.894 ±0.007	0.877 ±0.008	0.893 ±0.009	0.895 ±0.002	0.889 ±0.013	0.893 ±0.007	0.889 ±0.019
magnetic	0.897 ±0.001	0.899 ±0.000	0.896 ±0.000	0.907 ±0.008	0.917 ±0.004	0.798 ±0.174	0.911 ±0.009	0.899 ±0.027
slme	0.806 ±0.005	0.797 ±0.009	0.774 ±0.007	0.790 ±0.008	0.782 ±0.009	0.760 ±0.042	0.788 ±0.013	0.782 ±0.019
spillage	0.813 ±0.004	0.810 ±0.007	0.779 ±0.009	0.801 ±0.007	0.793 ±0.008	0.761 ±0.061	0.791 ±0.008	0.790 ±0.015
seebeck_p	0.875 ±0.002	0.885 ±0.007	0.867 ±0.007	0.876 ±0.007	0.891 ±0.006	0.889 ±0.005	0.891 ±0.006	0.891 ±0.006
powerfactor_n	0.756 ±0.006	0.752 ±0.009	0.724 ±0.008	0.763 ±0.008	0.759 ±0.008	0.754 ±0.018	0.754 ±0.007	0.755 ±0.008
Test AUC								
metal_opt	0.942 ±0.002	0.951 ±0.007	0.921 ±0.007	0.948 ±0.007	0.955 ±0.005	0.950 ±0.008	0.955 ±0.005	0.954 ±0.009
metal_mbj	0.954 ±0.000	0.959 ±0.009	0.896 ±0.009	0.955 ±0.008	0.956 ±0.002	0.952 ±0.007	0.955 ±0.005	0.953 ±0.009
magnetic	0.940 ±0.001	0.939 ±0.007	0.875 ±0.000	0.938 ±0.009	0.958 ±0.003	0.815 ±0.203	0.957 ±0.007	0.942 ±0.026
slme	0.870 ±0.001	0.884 ±0.009	0.800 ±0.009	0.862 ±0.008	0.871 ±0.009	0.839 ±0.054	0.870 ±0.010	0.864 ±0.013
spillage	0.864 ±0.002	0.860 ±0.009	0.783 ±0.007	0.852 ±0.007	0.849 ±0.004	0.840 ±0.016	0.847 ±0.011	0.847 ±0.010
seebeck_p	0.950 ±0.001	0.956 ±0.009	0.907 ±0.008	0.948 ±0.007	0.960 ±0.003	0.957 ±0.004	0.957 ±0.004	0.960 ±0.003
powerfactor_n	0.765 ±0.009	0.789 ±0.007	0.704 ±0.009	0.786 ±0.007	0.768 ±0.012	0.763 ±0.026	0.761 ±0.014	0.766 ±0.010

Moving from L0 to the richer CGCNN-style embeddings (L1–L3) (Xie & Grossman, 2018) improves the mean accuracy on several benchmarks – most consistently on spillage and SLME – where bond distances and angle statistics carry direct physical meaning. For spillage and SLME, the additional geometric descriptors approximate the information that ALIGNN encodes through its line-graph updates (Choudhary & DeCost, 2021), as triplet geometry governs band hybridization relevant to topological character and optical absorption edges. Thermoelectric tasks (Seebeck, power factor) show smaller but non-uniform gains, consistent with transport coefficients being controlled by the band structure near the Fermi level rather than by coarse local geometry alone.

4.3. Comparison with published reference models

Tab. 11 compares the best CTGP architecture discovered in this study at each representation level against ALIGNN (Choudhary et al., 2020) and, where available, feature-engineered reference models from the literature. ALIGNN alternates bond-graph and line-graph convolutions so that angle information is incorporated explicitly; the L3 representation supplies mean bond-angle cosines as node features but does not construct a line graph. Using the dated leaderboard values (Tab. 11), the performance deficits of the best discovered CTGP models relative to the state-of-the-art reference models, in descending order of magnitude, are as follows: n-type power factor (5.8 pp), SLME (5.0 pp), MBJ bandgap and p-type Seebeck coefficient (4.2 pp each), spillage (3.8 pp), OPT bandgap (3.5 pp), and magnetic moment (3.2 pp). On spillage and powerfactor, the best discovered models approach published reference accuracies more closely, especially as the representation level increases. The "Gap" column quantifies, in percentage points, how far the best CTGP model across L0–L3 stands from the reference leader for each property.

4.4. Comparison of evolutionary strategies

Tab. 12 aggregates performance by strategy over L0 CTGP runs. Fig. 5 shows the distribution of per-run test accuracy; Fig. 6 reports validation-AUC convergence. The boxplots confirm that (1 + 4) attains the highest median accuracy on the harder thermoelectric and SLME tasks, while (1 + 1) shows wider dispersion but lower search cost. Convergence curves rise quickly in the first 50–80 generations and then plateau, indicating that most architectural improvement is found early in the search rather than through late exploitation. At L0, (1 + 1) has the highest aggregate mean accuracy and the lowest search time, whereas (1 + 4) improves selected tasks at a substantially larger evaluation budget. Strategy preference is therefore task- and budget-dependent. Surrogate filtering reduces the wall-clock search cost substantially (Fig. 7, Tab. 13). For (1 + 1), the mean search time falls from 17 h (plain) to 5 h (surrogate), a 3.5× speed-up; for (1 + 4), from 48 h to 13 h (3.7×). Fig. 8 shows that rejection rates are highest for the (1 + 4) + S magnetic run (96%) and for both surrogate thermoelectric runs (82%); the bandgap rejection rates are lower (50–69%). In these tasks, validation landscapes are flatter and the surrogate must prune more aggressively to recover the speed-up seen in Fig. 7; on spillage and SLME, the filter is more permissive, and accuracy tracks the plain strategies more closely. Per-dataset search hours, rejection rates, and test accuracy are listed in Tab. 13.

Tab. 11. Comparison with the published JARVIS Leaderboard state of the art. For each benchmark, the current leaderboard leader (model and team, access: 30/07/2026), the ALIGNN reference, and the best CTGP architecture at each representation level are listed; the final column is the smallest CTGP gap to the leader

Dataset	Published leaderboard				Best CTGP (this work)			
	Leader model (team)	ACC	ALIGNN ACC	L0	L1	L2	L3	Gap – leader [pp]
metal_opt	*matminer_xgboost* (UofT)	0.9361	0.933	0.764	0.888	0.901	0.893	−3.5
metal_mbj	*matminer_xgboost* (UofT)	0.9399	0.923	0.762	0.898	0.893	0.895	−4.2
magnetic	*matminer_xgboost* (UofT)	0.9489	0.945	0.849	0.914	0.916	0.917	−3.2
slme	*matminer_rf* (UofT)	0.8422	0.831	0.617	0.788	0.792	0.788	−5.0
spillage	*matminer_xgboost* (UofT)	0.8364	0.814	0.711	0.798	0.783	0.793	−3.8
seebeck_p	*matminer_xgboost* (UofT)	0.9332	0.926	0.750	0.884	0.890	0.891	−4.2
powerfactor_n	*matminer_xgboost* (UofT)	0.8186	0.790	0.672	0.761	0.758	0.759	−5.8

Tab. 12. Aggregated CTGP performance by evolutionary strategy (L0)

Strategy	Surrogate	Test ACC	Parameters	Search [h]
1 + 1	no	0.723 ±0.102	168,176	16.9
1 + 1	yes	0.658 ±0.128	145,442	4.9
1 + 4	no	0.690 ±0.106	133,878	47.6
1 + 4	yes	0.637 ±0.139	207,986	13.0

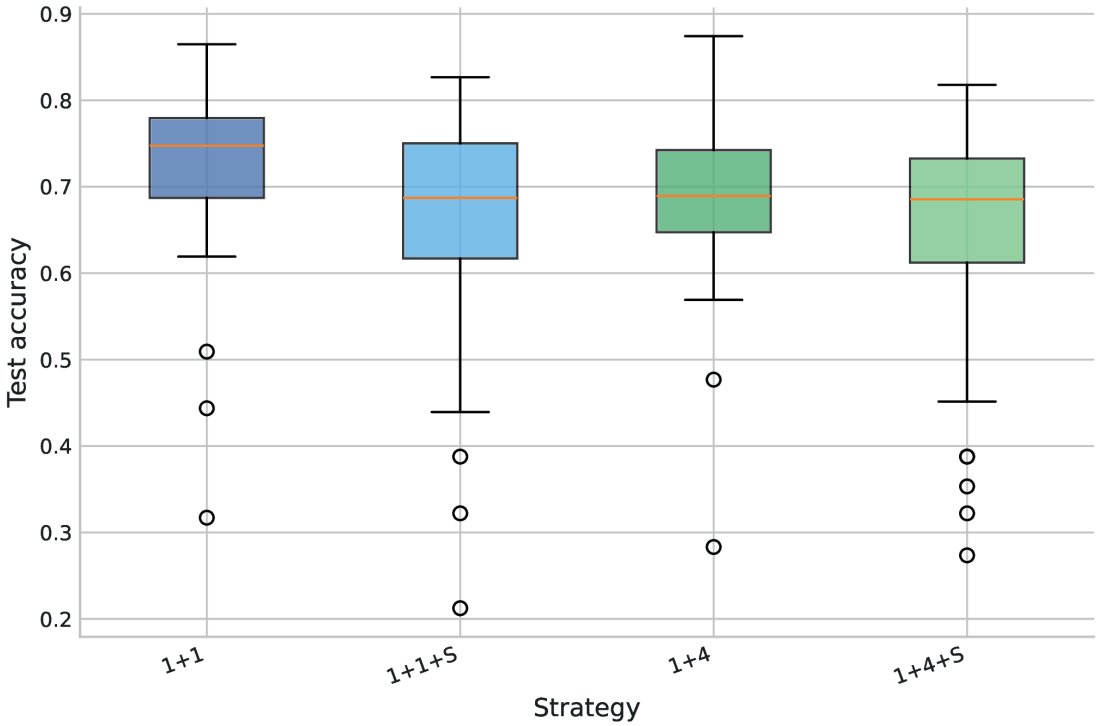


Fig. 5. Distribution of per-run test accuracy across CTGP strategies on L0 benchmarks

Tab. 13. Surrogate filtering efficiency on L0 benchmarks

Dataset	Strategy	Reject. [%]	Evals	Search [h]	Test ACC
metal_opt	1 + 1 + S	68	95	14.1	0.666 ±0.167
metal_opt	1 + 4 + S	50	601	40.4	0.642 ±0.220
metal_mbj	1 + 1 + S	68	95	4.5	0.648 ±0.106
metal_mbj	1 + 4 + S	69	378	17.8	0.627 ±0.128
magnetic	1 + 1 + S	68	95	8.5	0.699 ±0.219
magnetic	1 + 4 + S	96	49	1.3	0.803 ±0.021
slme	1 + 1 + S	68	95	1.4	0.594 ±0.074
slme	1 + 4 + S	69	378	6.4	0.609 ±0.070
spillage	1 + 1 + S	68	95	1.6	0.711 ±0.065
spillage	1 + 4 + S	55	543	8.9	0.656 ±0.176
seebeck_p	1 + 1 + S	82	54	2.3	0.629 ±0.125
seebeck_p	1 + 4 + S	82	214	11.2	0.600 ±0.155
powerfactor_n	1 + 1 + S	82	54	1.7	0.655 ±0.085
powerfactor_n	1 + 4 + S	82	214	8.4	0.640 ±0.126

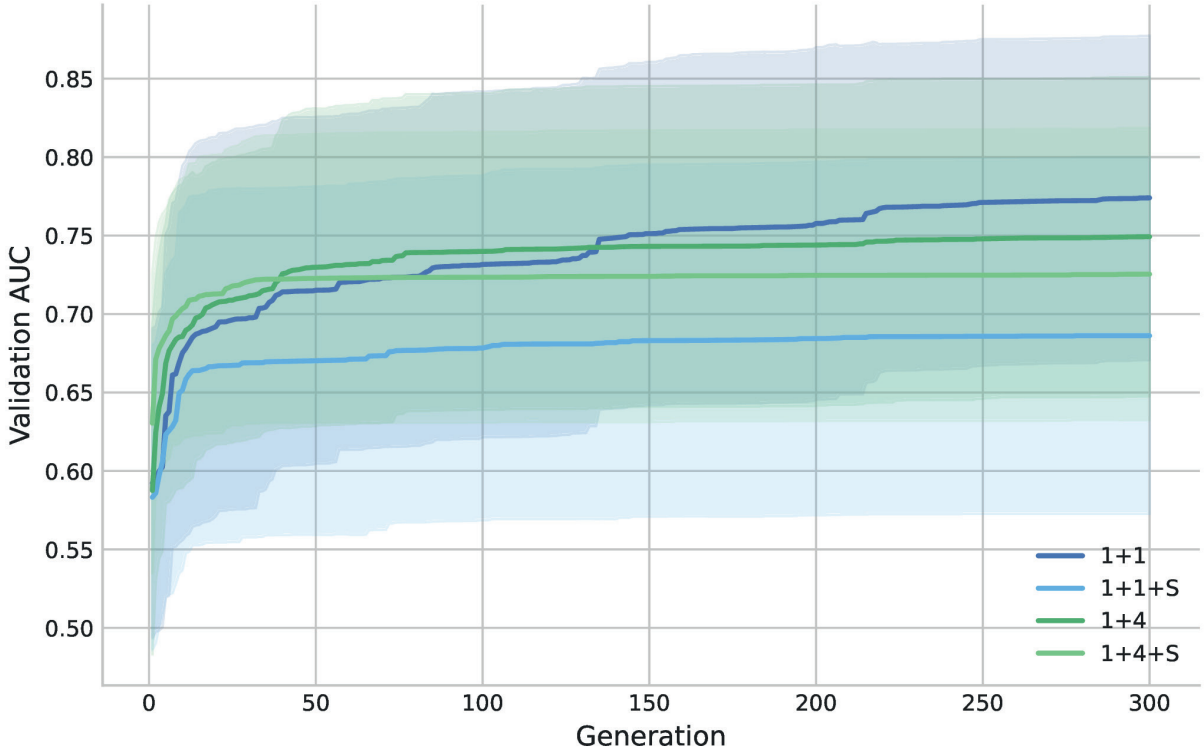


Fig. 6. Validation AUC convergence over CTGP generations, averaged per strategy

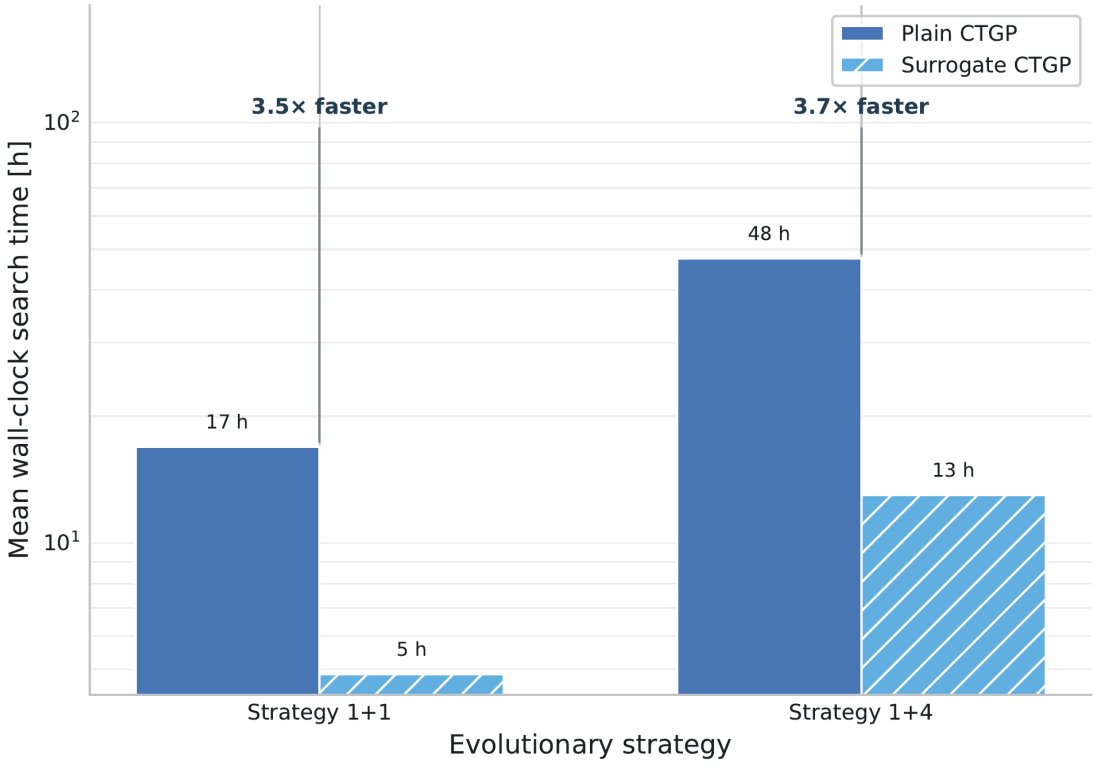


Fig. 7. Mean wall-clock search time for plain versus surrogate-assisted CTGP on L0 runs; mark the speed-up factor

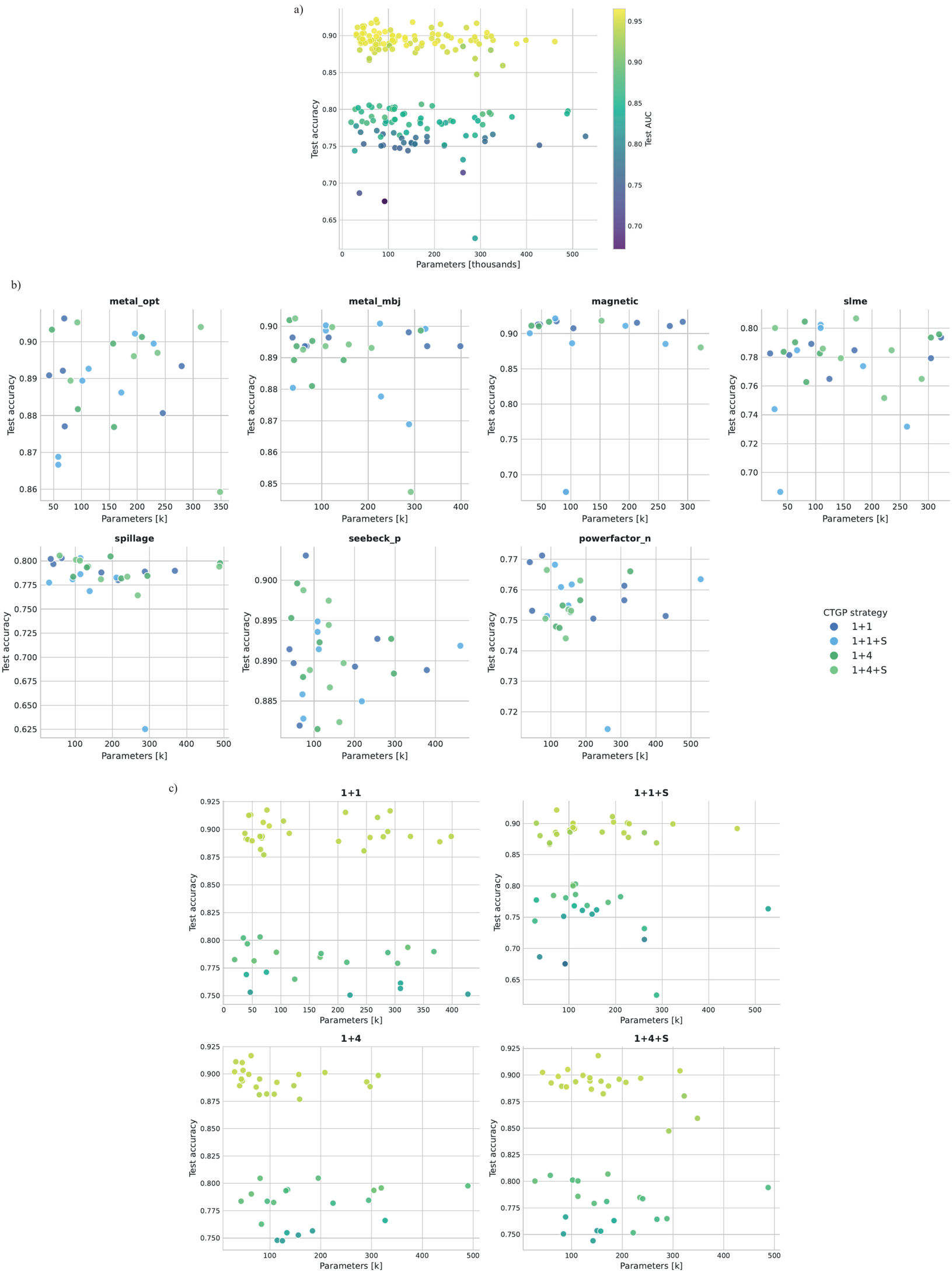


Fig. B. Accuracy versus model size for CTGP runs at representation level L3:
a) pooled view; b) per-strategy facets; c) per-benchmark facets

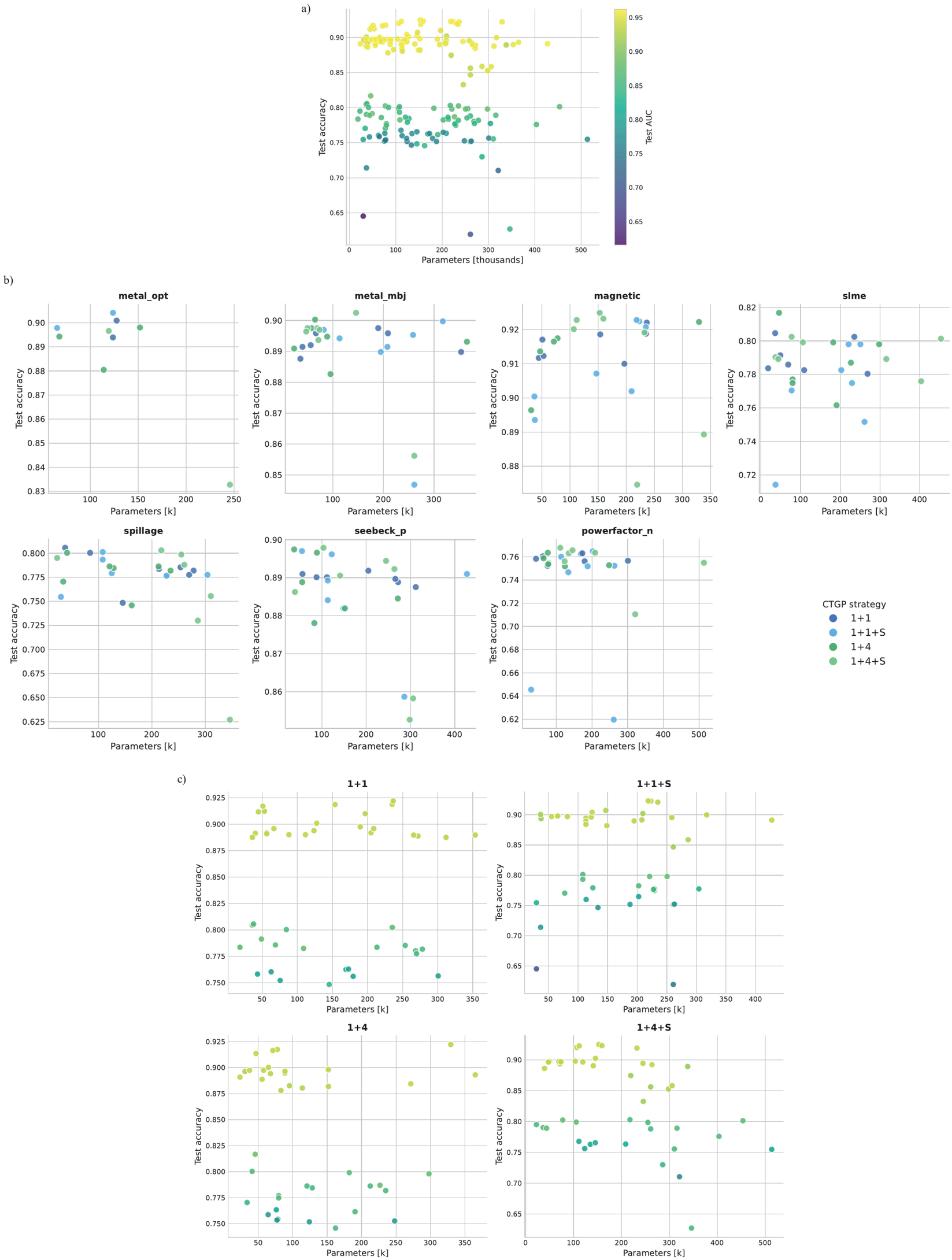


Fig. C. Accuracy versus model size for CTGP runs at representation level L2:
a) pooled view; b) per-strategy facets; c) per-benchmark facets

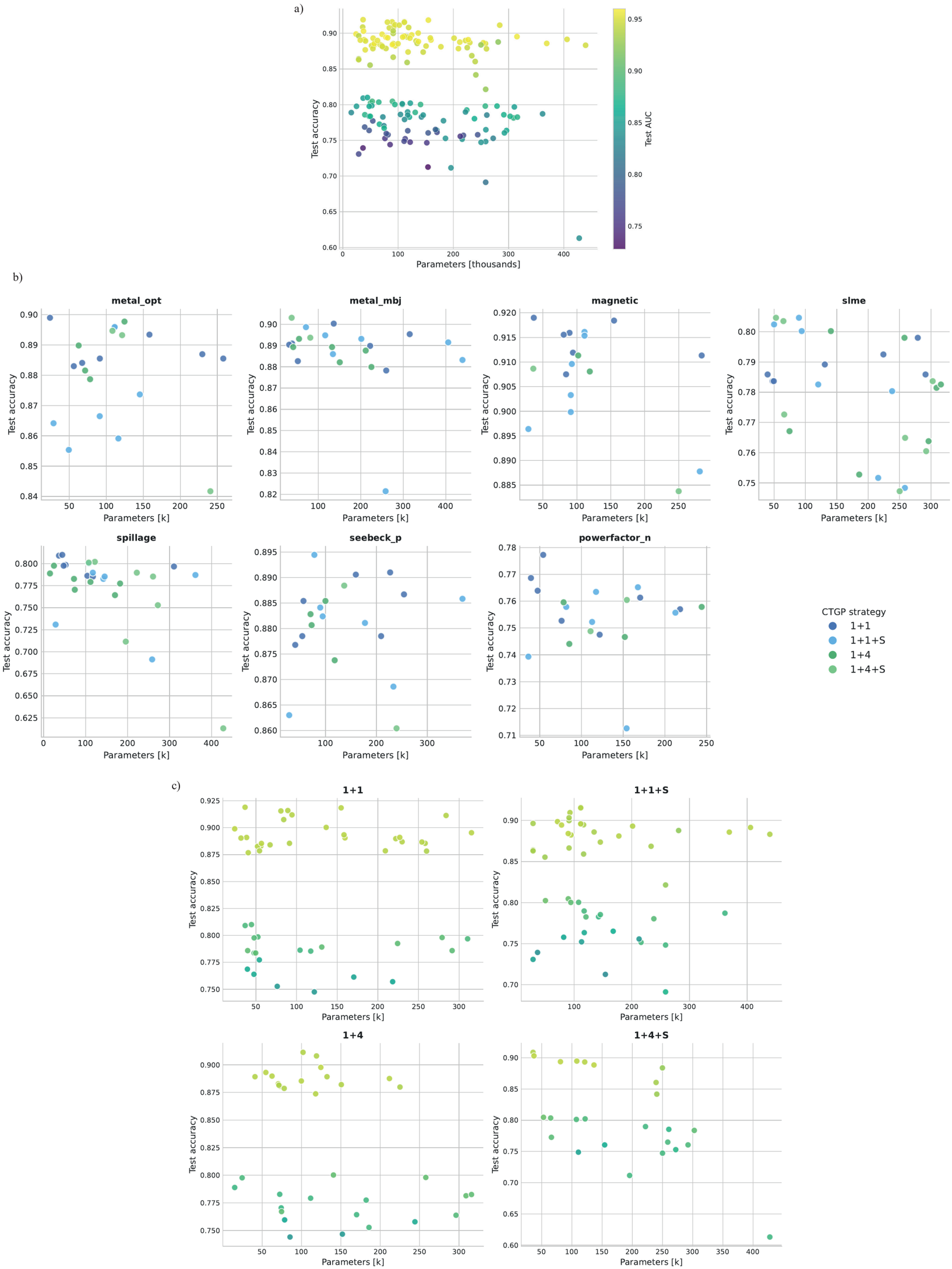


Fig. D. Accuracy versus model size for CTGP runs at representation level L1:
a) pooled view; b) per-strategy facets; c) per-benchmark facets

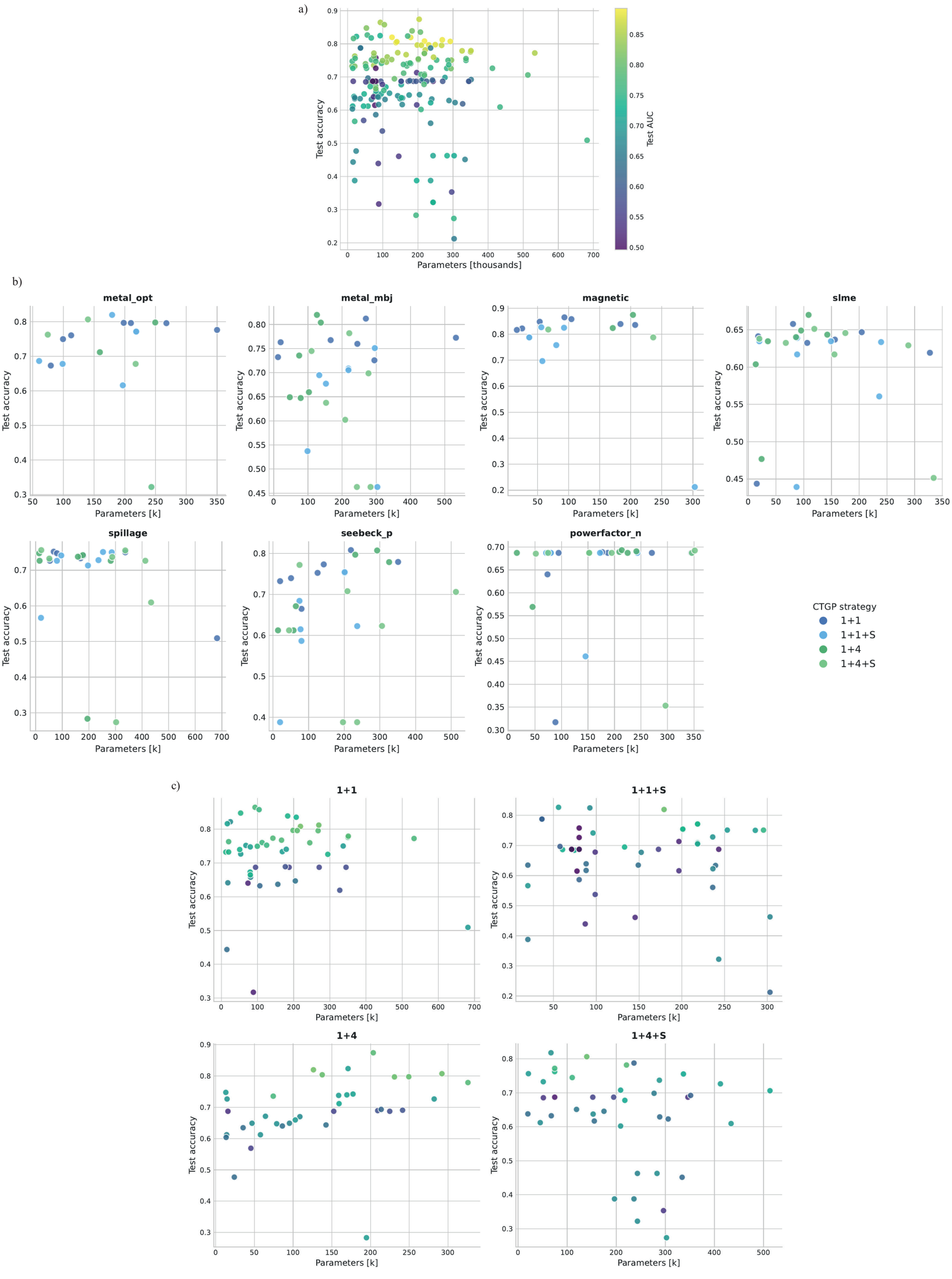


Fig. E. Accuracy versus model size for CTGP runs at representation level L0:
a) pooled view; b) per-strategy facets; c) per-benchmark facets

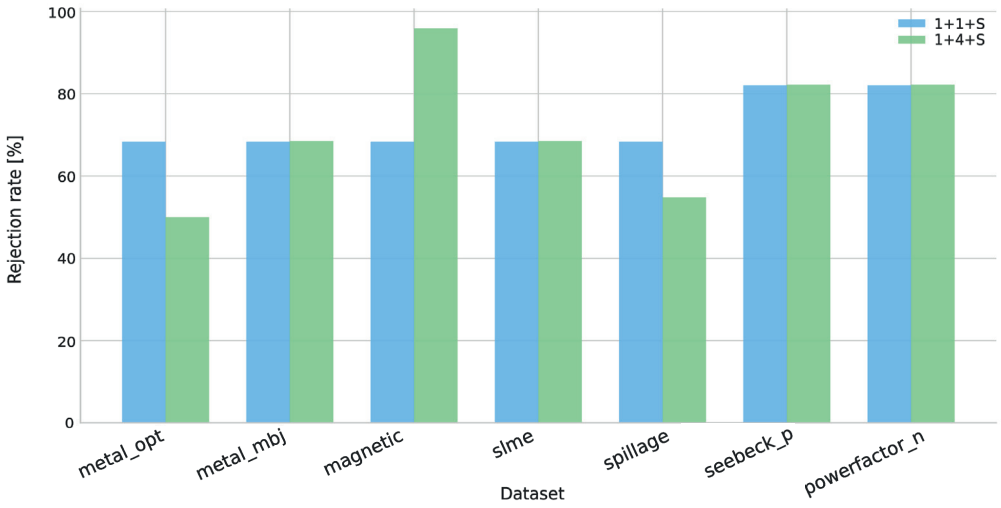


Fig. 8. Mean surrogate rejection rate per benchmark and surrogate strategy

4.5. Model complexity, search cost, and inference

A central claim of this work is that CTGP discovers compact and efficient architectures; therefore, their cost is quantified explicitly. Tab. 14 summarizes, per strategy, the model size (parameter count), search wall-clock time, per-model training time, and inference latency per sample. All experiments were conducted on a workstation equipped with an AMD Ryzen 9 7950X processor (16 cores, 32 threads), 128 GB of RAM, and two NVIDIA RTX 4090 GPUs. We report results from ten completed independent runs for every dataset-method combination included in the analysis. The software environment consisted of Python 3.11, PyTorch 2.8, PyTorch Geometric 2.5.1, CUDA 12.2, and cuDNN 9.10.2. Inference latency was measured using a single complete pass over the test split, without warm-up iterations. To ensure reliable timing measurements, CUDA synchronization was performed immediately before and after the evaluation. All experiments were executed in a single-process, single-GPU setting. The reported inference time includes the complete model evaluation pipeline, while preprocessing, graph construction, and disk or network I/O were excluded from the timing measurements.

Figs B–E relate test accuracy to model size at every representation level; Fig. 9 adds the search-time dimension. Across all levels, the discovered models occupy a favorable region of the accuracy–size plane: most contain on the order of 10^4 to 3×10^5 parameters, yet reach accuracies close to their level specific best. Within each level, the pooled panels show that test AUC correlates with accuracy but that the most accurate points are not always the largest models. The per-strategy facets reveal that the (1 + 4) and (1 + 4) + surrogate tend to occupy the upper-left region (higher accuracy at moderate size), whereas the (1 + 1) variants spread more widely. The per-benchmark facets show a task-specific structure: spillage and SLME exhibit a clearer accuracy–size trade-off at L2/L3, while thermoelectric tasks show more scatter, reflecting noisier labels. At L0, the smallest models already capture most of the achievable accuracy; at L1–L3, the richer features shift the frontier upward without a commensurate growth in parameters, indicating that accuracy gains come from better structural information rather than from larger networks.

Tab. 14. Efficiency of discovered CTGP architectures on L0: model size, search cost, training and inference time (mean over completed runs). Parameter counts in thousands; search time is wall-clock per run; training time is per final-model fit; inference latency is per test sample

Strategy	Parameters	Search time [h/run]	Final-fit training time [min]	Model-forward latency [ms/sample]
1 + 1	168×10^3	16.9	7.0	0.40
1 + 1 + S	145×10^3	4.9	3.4	0.34
1 + 4	134×10^3	47.6	4.2	0.31
1 + 4 + S	208×10^3	13.0	2.2	0.28

Fig. 9 places mean search wall-clock time on the horizontal axis: (1 + 1) clusters at low cost with moderate accuracy, (1 + 4) extends toward higher accuracy at roughly two to four times the search budget, and surrogate variants overlap their plain counterparts in accuracy while occupying lower search-time regions. Fig. 10 breaks latency down by benchmark and strategy; all configurations remain below a few milliseconds per sample, with only weak dependence on model depth, confirming that the compact discovered architectures are suitable for high-throughput screening.

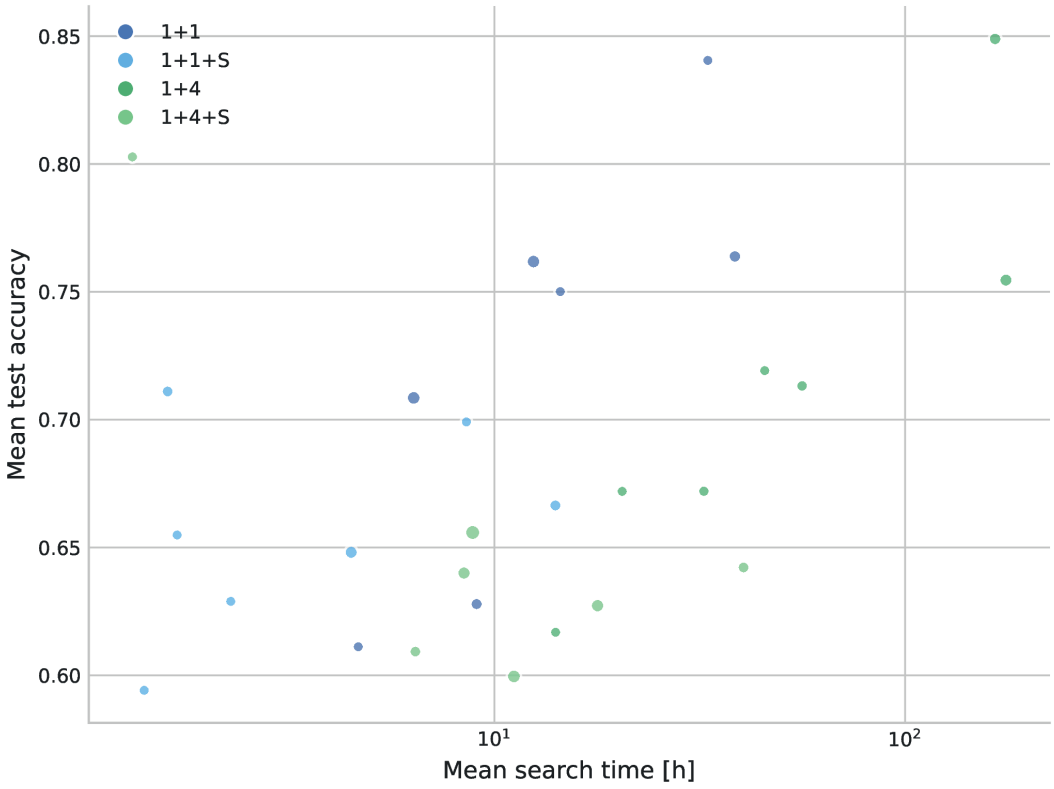


Fig. 9. Search cost versus mean test accuracy; marker size encodes parameter count

4.6. Implications for high-throughput screening

Tab. 15 lists the best CTGP run per benchmark with its decoded layer stack, and Fig. 11 shows operator frequencies in the best solutions. GraphSAGE and GAT together account for the majority of convolution slots in the best runs, with GIN and SG appearing mainly on thermoelectric and SLME tasks where local aggregation without attention suffices. The per-strategy bars show that (1 + 4) runs explore a slightly broader operator mix than (1 + 1), consistent with the larger evaluation budget. Tab. 16 reports run-to-run stability of the operator vocabulary, where operator stability is the mean pairwise Jaccard similarity of the convolution-operator sets selected by runs of the same benchmark (1.0 = identical operator sets across runs). Several trends align with the physics of the target properties.

The selected operator patterns permit exploratory comparison with known materials-physics considerations, but they do not establish a direct causal link. The analysis is post hoc and no operator-level ablation or statistical test is reported. For electronic structure tasks (metal_opt, metal_mbj), the evolved architectures predominantly select attention-based (GAT) and sampling-based (GraphSAGE) operators (as seen in Tab. 15 and Fig. 11). From a crystallographic perspective, in complex or distorted coordination environments, not all nearest neighbors contribute equally to the electronic density of states near the Fermi level. GAT layers can represent unequal neighbor contributions, which is qualitatively compatible with direction-dependent local environments. However, no bond-level attention analysis is reported here, so the learned coefficients should not be interpreted as evidence of dominant hybridization pathways or simulated covalent bonding.

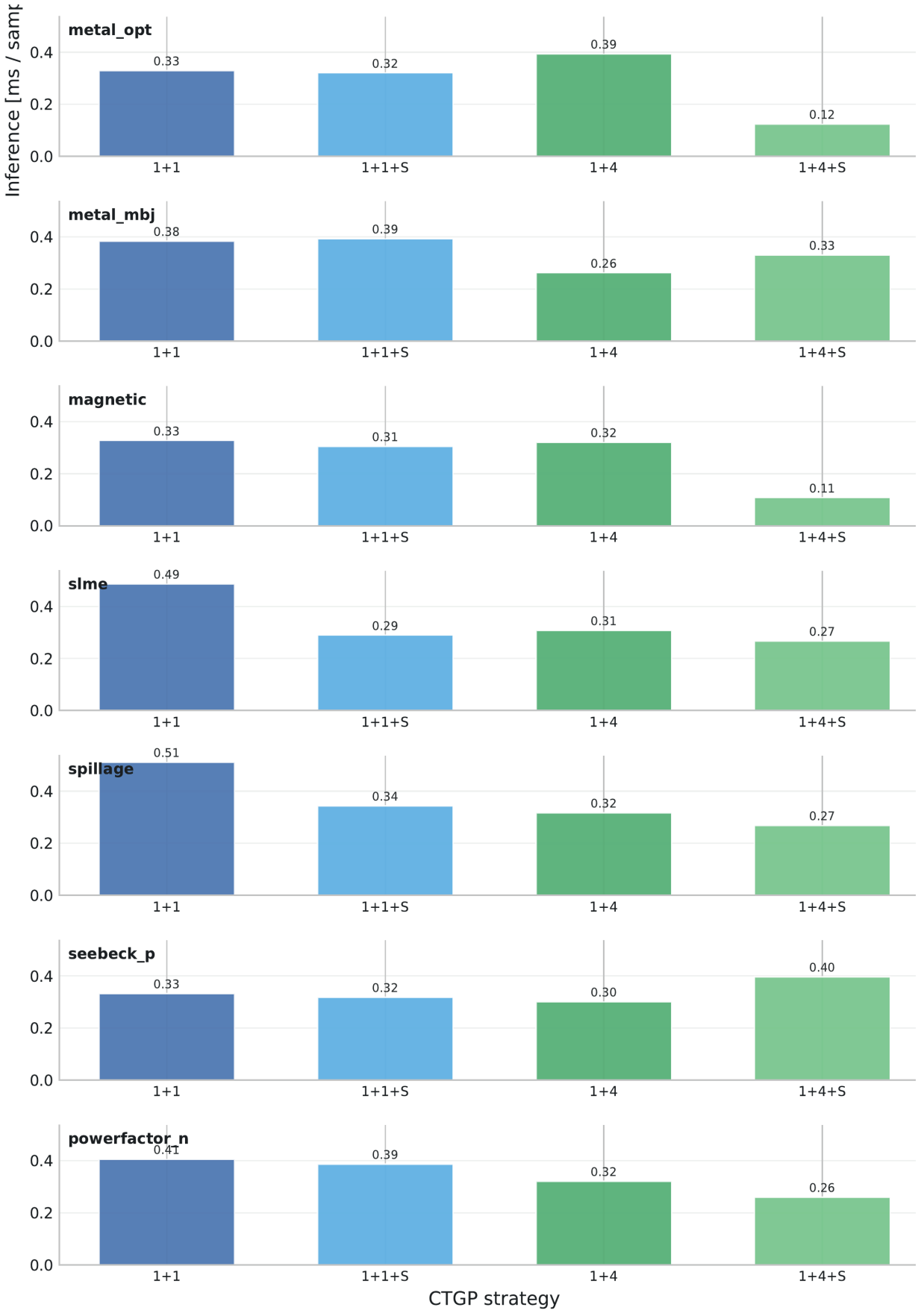


Fig. 10. Mean inference latency per benchmark (one panel per dataset) and CTGP strategy

The best L0 magnetic architecture contains four convolution layers, whereas several other tasks in Tab. 15 contain six. The present evidence therefore does not show a unique or consistent preference for deeper magnetic networks. A task-wise depth analysis across all runs would be required to test this hypothesis.

Regarding SLME and spillage, performance improves for several tasks at L2/L3, but the operator and depth evidence presented in Tabs 14–15 and Fig. 11 is restricted to L0. We therefore do not infer an L2/L3 depth preference from the current architecture analysis.

In contrast, for thermoelectric transport (Seebeck coefficient, power factor) operator diversity is highest and run-to-run stability is lowest for these datasets. This structural variance mirrors the complex physics of macroscopic transport. Unlike electronic band gaps – which are largely determined by local static geometry – thermoelectric coefficients are dynamic transport properties governed by the precise shape of the Fermi surface, carrier effective masses, and relaxation times. Because static geometric graph features (even at L3) are only indirect proxies for these transport dynamics, the search algorithm explores a wider variety of local aggregation operators (GIN, SG) without converging on a single dominant message-passing paradigm.

Tab. 15. Best CTGP architecture per benchmark (single run, L0)

Dataset	Method	Layer stack	Params	ACC
magnetic	CTGP (1 + 4)	GraphSAGE → GraphSAGE → GraphSAGE → GCN	203,490	0.874
metal_mbj	CTGP (1 + 4)	GAT → GraphSAGE → GCN → GAT → GAT → GIN	126,249	0.820
metal_opt	CTGP surrogate (1 + 1)	GraphSAGE → GraphSAGE → GraphSAGE → GCN → SG → SG	179,245	0.819
powerfactor_n	CTGP (1 + 4)	GraphSAGE → GraphSAGE → GraphSAGE → GAT → GraphSAGE → GAT	213,753	0.693
seebeck_p	CTGP (1 + 1)	GraphSAGE → GraphSAGE → GAT → GraphSAGE → SG → GIN	218,700	0.808
slme	CTGP (1 + 4)	GAT → GraphSAGE → SG → GIN → GAT → SG	108,735	0.670
spillage	CTGP surrogate (1 + 4)	GCN → GCN	21,090	0.756

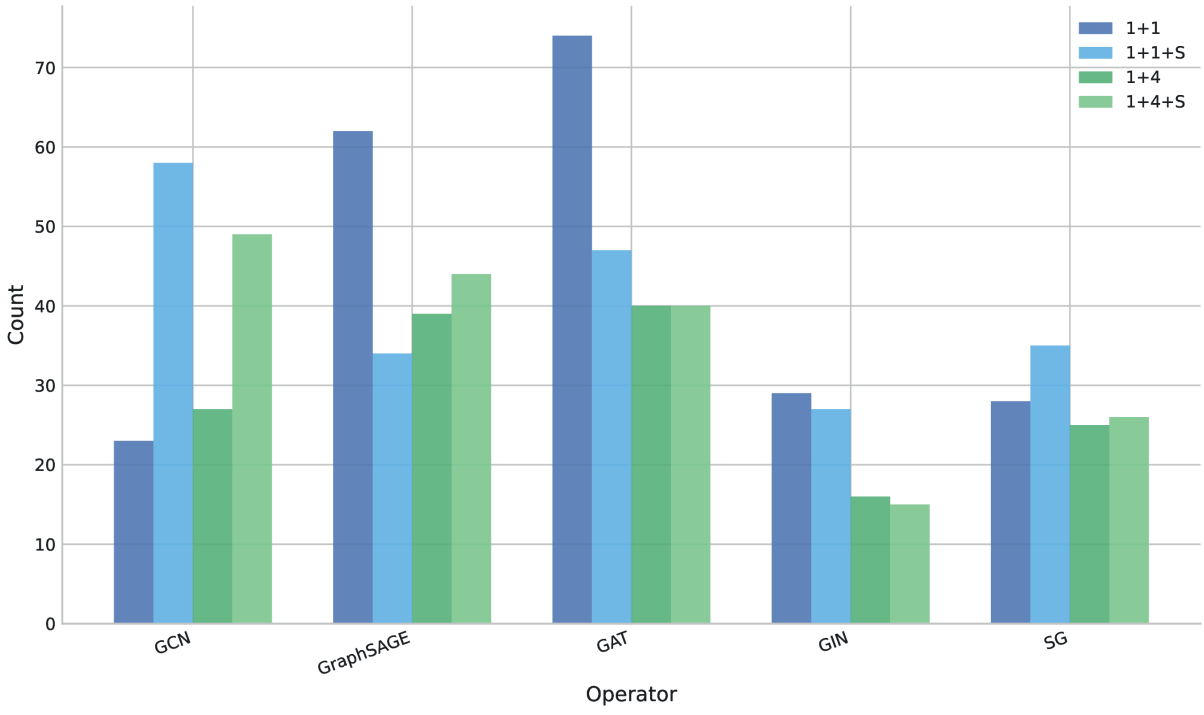


Fig. 11. Frequency of GCN, GraphSAGE, GAT, GIN, and SG operators in the analyzed final L0 architectures, grouped by CTGP strategy

Tab. 16. Stability of discovered architectures across independent benchmark runs (L0)

Strategy	Runs per benchmark	Mean layers	Top operator (share)	Operator stability
1 + 1	10	4.3	GAT (34%)	0.56
1 + 1 + S	10	4.0	GCN (29%)	0.44
1 + 4	10	4.2	GAT (27%)	0.51
1 + 4 + S	10	4.5	GCN (28%)	0.42

Beyond predictive accuracy, the primary requirement for integrating computational models into materials engineering workflows is inference speed. As detailed in Tab. 14, the CTGP-discovered architectures exhibit inference latencies ranging from 0.28 to 0.40 milliseconds per sample on average. In the context of High-Throughput Screening (HTS), this efficiency is transformative. At the measured model-forward latency of 0.28–0.40 ms per sample, inference over one million already preprocessed crystal graphs would require approximately 5–7 min under the reported hardware conditions. This estimate excludes graph construction, feature computation, and I/O and should therefore not be interpreted as end-to-end screening time. Nevertheless, it demonstrates that model inference itself is unlikely to constitute the dominant bottleneck in a high-throughput screening pipeline.

4.7. CTGP versus fair GNN baselines

A central question in NAS is whether evolutionary discovery of heterogeneous convolution stacks offers benefits beyond random sampling of the same architecture space. We therefore use the 301-evaluation CTGP (1 + 1) setting as the primary budget-matched reference for random search, while the higher-budget (1 + 4) variants are reported separately as additional evolutionary search configurations. To address this, we compare the four CTGP strategies against three controlled GNN baselines trained on identical crystal graphs, NIST splits, node features, and optimization settings (Sections 3.4 and 3.5):

- fixed (per seed, the best homogeneous GCN, GraphSAGE, GAT, or GIN stack with default hyperparameters);
- fixed + hparam (the same four architectures with a 3 × 3 grid over learning rate and dropout);
- random search (uniform sampling from the CTGP architecture space with 301 full training evaluations per run, matching the (1 + 1) CTGP budget).

Tabs 17–20 report the results separately for representation levels L0–L3. Each table entry gives test accuracy, test AUC, and test macro-F1 as mean ± standard deviation over completed independent runs. Within each benchmark column, the highest accuracy is shown in bold, the highest AUC in italics, and the highest macro-F1 underlined. This presentation allows a direct assessment of whether CTGP improves ranking quality (AUC), hard classification (accuracy), or class-balanced performance (macro-F1) relative to fixed and randomly sampled architectures.

At L0 (Tab. 17), where node inputs reduce to atomic number, Fixed + hparam attains the strongest test accuracy on six of seven benchmarks, and Random search is competitive on the remaining tasks. CTGP strategies trail both fair baselines on most properties, indicating that with minimal featurization, predictive performance is driven primarily by graph convolution and hyperparameter tuning rather than by heterogeneous stack discovery. Notable exceptions include spillage, where CTGP (1 + 1) achieves the best test AUC, suggesting that evolved operator mixtures can improve class ranking even when accuracy remains below the tuned fixed stacks.

Tab. 17. Three-way comparison at L0 (atomic-number node features): classical ML baselines (MLP, RF, DT, SVM), CTGP evolutionary strategies, and fair GNN baselines

Method	*metal_opt*	*metal_mbj*	*magnetic*	*slme*	*spillage*	*seebeck_p*	*powerfactor_n*
MLP	0.736 ±0.002 0.744 ±0.001 0.658 ±0.002	0.674 ±0.002 0.742 ±0.002 0.666 ±0.003	0.704 ±0.217 0.659 ±0.060 0.409 ±0.103	0.628 ±0.052 0.672 ±0.006 0.593 ±0.069	0.739 ±0.003 0.768 ±0.001 0.651 ±0.004	0.704 ±0.002 0.755 ±0.001 0.684 ±0.004	0.687 ±0.000 0.592 ±0.013 0.407 ±0.000
RF	0.771 ±0.000 0.840 ±0.000 0.707 ±0.000	0.750 ±0.000 0.839 ±0.000 0.748 ±0.000	0.851 ±0.000 0.841 ±0.000 0.721 ±0.000	0.699 ±0.000 0.779 ±0.000 0.688 ±0.000	0.764 ±0.000 0.809 ±0.000 0.683 ±0.000	0.776 ±0.000 0.856 ±0.000 0.767 ±0.000	0.702 ±0.000 0.680 ±0.000 0.475 ±0.000
DT	0.767 ±0.000 0.822 ±0.000 0.715 ±0.000	0.729 ±0.000 0.805 ±0.000 0.728 ±0.000	0.853 ±0.000 0.817 ±0.000 0.731 ±0.000	0.701 ±0.000 0.725 ±0.000 0.693 ±0.000	0.748 ±0.000 0.766 ±0.000 0.665 ±0.000	0.768 ±0.000 0.830 ±0.000 0.759 ±0.000	0.682 ±0.000 0.617 ±0.000 0.535 ±0.000

Tab. 17 (cont.)

Method	*metal_opt*	*metal_mbj*	*magnetic*	*slme*	*spillage*	*seebeck_p*	*powerfactor_n*
SVM	0.734 ±0.000 0.719 ±0.000 0.635 ±0.000	0.674 ±0.000 0.738 ±0.000 0.665 ±0.000	0.818 ±0.000 0.721 ±0.000 0.649 ±0.000	0.663 ±0.000 0.696 ±0.000 0.645 ±0.000	0.740 ±0.000 0.719 ±0.000 0.631 ±0.000	0.715 ±0.000 0.755 ±0.000 0.690 ±0.000	0.687 ±0.000 0.507 ±0.000 0.407 ±0.000
CTGP 1 + 1	0.764 ±0.044 0.845 ±0.028 0.736 ±0.038	0.762 ±0.028 0.846 ±0.028 0.761 ±0.028	0.841 ±0.018 0.803 ±0.038 0.697 ±0.044	0.611 ±0.075 0.675 ±0.037 0.583 ±0.119	0.709 ±0.088 0.763 ±0.004 0.583 ±0.086	0.750 ±0.046 0.839 ±0.032 0.742 ±0.044	0.628 ±0.138 0.577 ±0.033 0.399 ±0.075
CTGP 1 + 1 + S	0.666 ±0.167 0.733 ±0.134 0.586 ±0.198	0.648 ±0.106 0.743 ±0.082 0.600 ±0.184	0.699 ±0.219 0.652 ±0.085 0.482 ±0.160	0.594 ±0.074 0.653 ±0.048 0.528 ±0.137	0.711 ±0.065 0.688 ±0.118 0.547 ±0.096	0.629 ±0.125 0.710 ±0.109 0.573 ±0.176	0.655 ±0.085 0.526 ±0.030 0.415 ±0.020
CTGP 1 + 4	0.755 ±0.061 0.828 ±0.071 0.715 ±0.033	0.719 ±0.079 0.809 ±0.073 0.709 ±0.089	0.849 ±0.036 0.815 ±0.070 0.720 ±0.100	0.617 ±0.065 0.687 ±0.025 0.587 ±0.094	0.672 ±0.172 0.761 ±0.009 0.519 ±0.165	0.713 ±0.092 0.806 ±0.080 0.624 ±0.199	0.672 ±0.045 0.598 ±0.035 0.457 ±0.067
CTGP 1 + 4 + S	0.642 ±0.220 0.802 ±0.062 0.531 ±0.250	0.627 ±0.128 0.763 ±0.062 0.575 ±0.192	0.803 ±0.021 0.699 ±0.066 0.541 ±0.142	0.609 ±0.070 0.674 ±0.012 0.573 ±0.109	0.656 ±0.176 0.762 ±0.009 0.527 ±0.165	0.600 ±0.155 0.747 ±0.057 0.531 ±0.210	0.640 ±0.126 0.572 ±0.041 0.399 ±0.034
Fixed	0.789 ±0.010 0.844 ±0.012 0.752 ±0.013	0.729 ±0.019 0.801 ±0.023 0.727 ±0.019	0.847 ±0.005 0.813 ±0.016 0.715 ±0.015	0.651 ±0.005 0.693 ±0.005 0.639 ±0.004	0.752 ±0.003 0.757 ±0.000 0.620 ±0.024	0.741 ±0.027 0.809 ±0.020 0.718 ±0.051	0.690 ±0.002 0.601 ±0.027 0.424 ±0.009
Fixed + hparam	0.816 ±0.006 0.883 ±0.007 0.782 ±0.013	0.804 ±0.008 0.871 ±0.006 0.803 ±0.009	0.860 ±0.001 0.850 ±0.006 0.759 ±0.010	0.655 ±0.006 0.695 ±0.007 0.647 ±0.003	0.754 ±0.003 0.760 ±0.003 0.622 ±0.021	0.778 ±0.006 0.857 ±0.008 0.764 ±0.011	0.693 ±0.003 0.619 ±0.010 0.450 ±0.021
Random search	0.805 ±0.007 0.866 ±0.008 0.771 ±0.009	0.781 ±0.012 0.857 ±0.014 0.779 ±0.013	0.853 ±0.017 0.835 ±0.033 0.745 ±0.055	0.642 ±0.006 0.687 ±0.010 0.628 ±0.008	0.715 ±0.036 0.762 ±0.005 0.643 ±0.007	0.778 ±0.007 0.855 ±0.007 0.766 ±0.007	0.690 ±0.002 0.609 ±0.004 0.425 ±0.013

At L1 (Tab. 18), CGCNN-style embeddings narrow the gap. Random search and fixed + hparam remain the strongest overall methods, but CTGP (1 + 4) + surrogate reaches the best test accuracy, AUC, and macro-F1 on *metal_mbj*, and CTGP (1 + 1) again leads spillage AUC. These results suggest that architecture search becomes more informative once the node representation encodes chemically meaningful descriptors, particularly for electronic-structure classification.

At L2 (Tab. 19), with additional bond-distance features, Random search dominates both accuracy and AUC on most benchmarks. CTGP remains close on several electronic-structure tasks but does not consistently outperform the fair baselines. The pattern indicates diminishing headroom for heterogeneous stack discovery as geometric information is explicitly supplied in the input features.

At L3 (Tab. 20), fixed + hparam and random search again lead on most accuracy and F1 metrics, while CTGP is competitive on selected columns (e.g. magnetic AUC for CTGP 1 + 4).

Tab. 18. Three-way comparison at L1 (CGCNN-style coordinates): classical ML, CTGP, and fair GNN baselines

Method	*metal_opt*	*metal_mbj*	*magnetic*	*slme*	*spillage*	*seebeck_p*	*powerfactor_n*
MLP	0.855 ±0.001 0.930 ±0.001 0.836 ±0.001	0.885 ±0.002 0.948 ±0.000 0.885 ±0.002	0.895 ±0.002 0.934 ±0.002 0.835 ±0.004	0.793 ±0.008 0.864 ±0.001 0.790 ±0.008	0.792 ±0.003 0.850 ±0.002 0.728 ±0.004	0.870 ±0.002 0.942 ±0.000 0.864 ±0.002	0.755 ±0.005 0.756 ±0.003 0.664 ±0.010
RF	0.873 ±0.000 0.944 ±0.000 0.857 ±0.000	0.894 ±0.000 0.954 ±0.000 0.893 ±0.000	0.899 ±0.000 0.938 ±0.000 0.830 ±0.000	0.796 ±0.000 0.876 ±0.000 0.789 ±0.000	0.807 ±0.000 0.861 ±0.000 0.722 ±0.000	0.882 ±0.000 0.952 ±0.000 0.877 ±0.000	0.753 ±0.000 0.784 ±0.000 0.628 ±0.000
DT	0.862 ±0.000 0.919 ±0.000 0.844 ±0.000	0.890 ±0.000 0.911 ±0.000 0.890 ±0.000	0.897 ±0.000 0.887 ±0.000 0.838 ±0.000	0.786 ±0.000 0.820 ±0.000 0.778 ±0.000	0.775 ±0.000 0.786 ±0.000 0.716 ±0.000	0.854 ±0.000 0.899 ±0.000 0.850 ±0.000	0.739 ±0.000 0.708 ±0.000 0.634 ±0.000
SVM	0.864 ±0.000 0.933 ±0.000 0.847 ±0.000	0.888 ±0.000 0.946 ±0.000 0.888 ±0.000	0.901 ±0.000 0.932 ±0.000 0.842 ±0.000	0.780 ±0.000 0.853 ±0.000 0.774 ±0.000	0.789 ±0.000 0.839 ±0.000 0.707 ±0.000	0.873 ±0.000 0.939 ±0.000 0.868 ±0.000	0.759 ±0.000 0.772 ±0.000 0.658 ±0.000
CTGP 1 + 1	0.888 ±0.006 0.952 ±0.003 0.873 ±0.006	0.890 ±0.007 0.953 ±0.003 0.889 ±0.007	0.914 ±0.004 0.952 ±0.002 0.867 ±0.008	0.788 ±0.005 0.869 ±0.007 0.783 ±0.005	0.798 ±0.010 0.850 ±0.008 0.729 ±0.018	0.884 ±0.006 0.955 ±0.003 0.878 ±0.006	0.761 ±0.010 0.772 ±0.013 0.692 ±0.017

Method	*metal_opt*	*metal_mbj*	*magnetic*	*slme*	*spillage*	*seebeck_p*	*powerfactor_n*
CTGP 1 + 1 + S	0.873 ±0.017 0.943 ±0.009 0.858 ±0.017	0.881 ±0.027 0.946 ±0.010 0.881 ±0.027	0.904 ±0.010 0.942 ±0.010 0.852 ±0.017	0.781 ±0.024 0.865 ±0.018 0.777 ±0.025	0.767 ±0.040 0.827 ±0.029 0.707 ±0.031	0.880 ±0.011 0.952 ±0.007 0.873 ±0.012	0.749 ±0.018 0.762 ±0.024 0.676 ±0.035
CTGP 1 + 4	0.887 ±0.009 0.952 ±0.004 0.870 ±0.011	0.887 ±0.005 0.951 ±0.004 0.886 ±0.005	0.910 ±0.002 0.949 ±0.002 0.855 ±0.007	0.778 ±0.018 0.869 ±0.014 0.771 ±0.020	0.780 ±0.011 0.837 ±0.007 0.712 ±0.010	0.881 ±0.005 0.954 ±0.002 0.876 ±0.005	0.752 ±0.008 0.766 ±0.017 0.667 ±0.034
CTGP 1 + 4 + S	0.877 ±0.030 0.947 ±0.014 0.862 ±0.029	0.898 ±0.007 0.958 ±0.000 0.898 ±0.007	0.896 ±0.018 0.937 ±0.014 0.846 ±0.013	0.777 ±0.022 0.863 ±0.016 0.770 ±0.023	0.751 ±0.069 0.838 ±0.011 0.709 ±0.048	0.874 ±0.020 0.949 ±0.012 0.869 ±0.017	0.755 ±0.008 0.778 ±0.012 0.671 ±0.047
Fixed	0.886 ±0.001 0.951 ±0.001 0.871 ±0.001	0.891 ±0.001 0.952 ±0.002 0.890 ±0.001	0.911 ±0.003 0.948 ±0.003 0.862 ±0.006	0.794 ±0.008 0.873 ±0.008 0.790 ±0.010	0.794 ±0.003 0.842 ±0.002 0.727 ±0.009	0.883 ±0.004 0.955 ±0.001 0.879 ±0.003	0.756 ±0.007 0.759 ±0.013 0.679 ±0.013
Fixed + hparam	0.893 ±0.003 0.954 ±0.001 0.877 ±0.004	0.894 ±0.002 0.954 ±0.002 0.894 ±0.002	0.911 ±0.001 0.949 ±0.003 0.863 ±0.003	0.795 ±0.006 0.873 ±0.008 0.792 ±0.006	0.800 ±0.004 0.849 ±0.004 0.743 ±0.009	0.888 ±0.006 0.956 ±0.002 0.883 ±0.005	0.758 ±0.003 0.762 ±0.018 0.692 ±0.012
Random search	0.895 ±0.003 0.957 ±0.001 0.881 ±0.004	0.897 ±0.003 0.957 ±0.001 0.896 ±0.004	0.914 ±0.003 0.953 ±0.002 0.868 ±0.007	0.802 ±0.003 0.879 ±0.008 0.798 ±0.003	0.793 ±0.005 0.844 ±0.012 0.730 ±0.002	0.885 ±0.005 0.955 ±0.002 0.879 ±0.006	0.764 ±0.005 0.786 ±0.007 0.702 ±0.009

Tab. 19. Three-way comparison at L2 (+ structural descriptors): classical ML, CTGP, and fair GNN baselines

Method	*metal_opt*	*metal_mbj*	*magnetic*	*slme*	*spillage*	*seebeck_p*	*powerfactor_n*
MLP	0.862 ±0.002 0.936 ±0.001 0.843 ±0.002	0.888 ±0.001 0.951 ±0.000 0.887 ±0.001	0.898 ±0.001 0.938 ±0.002 0.838 ±0.003	0.796 ±0.001 0.869 ±0.000 0.792 ±0.002	0.803 ±0.001 0.862 ±0.003 0.747 ±0.003	0.871 ±0.002 0.946 ±0.000 0.865 ±0.001	0.754 ±0.005 0.758 ±0.006 0.660 ±0.012
RF	0.883 ±0.000 0.946 ±0.000 0.868 ±0.000	0.890 ±0.000 0.956 ±0.000 0.890 ±0.000	0.899 ±0.000 0.941 ±0.000 0.831 ±0.000	0.791 ±0.000 0.878 ±0.000 0.785 ±0.000	0.811 ±0.000 0.865 ±0.000 0.727 ±0.000	0.881 ±0.000 0.953 ±0.000 0.876 ±0.000	0.751 ±0.000 0.785 ±0.000 0.623 ±0.000
DT	0.864 ±0.000 0.913 ±0.000 0.846 ±0.000	0.884 ±0.000 0.904 ±0.000 0.884 ±0.000	0.898 ±0.000 0.878 ±0.000 0.837 ±0.000	0.777 ±0.000 0.805 ±0.000 0.773 ±0.000	0.764 ±0.000 0.774 ±0.000 0.704 ±0.000	0.865 ±0.000 0.905 ±0.000 0.860 ±0.000	0.726 ±0.000 0.705 ±0.000 0.623 ±0.000
SVM	0.875 ±0.000 0.943 ±0.000 0.858 ±0.000	0.891 ±0.000 0.952 ±0.000 0.891 ±0.000	0.906 ±0.000 0.938 ±0.000 0.851 ±0.000	0.784 ±0.000 0.861 ±0.000 0.778 ±0.000	0.799 ±0.000 0.851 ±0.000 0.722 ±0.000	0.877 ±0.000 0.945 ±0.000 0.871 ±0.000	0.764 ±0.000 0.781 ±0.000 0.665 ±0.000
CTGP 1 + 1	0.897 ±0.005 0.959 ±0.003 0.883 ±0.002	0.893 ±0.004 0.956 ±0.002 0.893 ±0.004	0.916 ±0.004 0.956 ±0.003 0.870 ±0.007	0.790 ±0.010 0.870 ±0.008 0.786 ±0.010	0.783 ±0.018 0.846 ±0.008 0.723 ±0.008	0.890 ±0.001 0.958 ±0.003 0.885 ±0.001	0.758 ±0.004 0.772 ±0.007 0.676 ±0.015
CTGP 1 + 1 + S	0.901 ±0.004 0.957 ±0.003 0.886 ±0.005	0.888 ±0.018 0.951 ±0.007 0.887 ±0.018	0.910 ±0.012 0.948 ±0.011 0.863 ±0.018	0.770 ±0.029 0.846 ±0.040 0.765 ±0.031	0.780 ±0.015 0.831 ±0.021 0.704 ±0.029	0.885 ±0.013 0.957 ±0.005 0.880 ±0.013	0.720 ±0.061 0.732 ±0.058 0.651 ±0.048
CTGP 1 + 4	0.891 ±0.009 0.954 ±0.006 0.875 ±0.010	0.893 ±0.006 0.957 ±0.003 0.893 ±0.006	0.913 ±0.010 0.955 ±0.006 0.865 ±0.020	0.788 ±0.018 0.868 ±0.016 0.784 ±0.020	0.779 ±0.017 0.843 ±0.007 0.723 ±0.014	0.888 ±0.008 0.957 ±0.004 0.882 ±0.008	0.756 ±0.004 0.761 ±0.009 0.676 ±0.022
CTGP 1 + 4 + S	0.865 ±0.045 0.946 ±0.017 0.852 ±0.043	0.892 ±0.016 0.955 ±0.010 0.891 ±0.016	0.911 ±0.020 0.952 ±0.014 0.868 ±0.023	0.792 ±0.009 0.868 ±0.009 0.788 ±0.009	0.757 ±0.063 0.843 ±0.008 0.712 ±0.042	0.882 ±0.018 0.955 ±0.004 0.878 ±0.017	0.754 ±0.020 0.770 ±0.022 0.691 ±0.016
Fixed	0.892 ±0.003 0.954 ±0.002 0.876 ±0.003	0.897 ±0.003 0.955 ±0.003 0.897 ±0.003	0.914 ±0.001 0.954 ±0.000 0.868 ±0.002	0.796 ±0.007 0.873 ±0.007 0.791 ±0.008	0.798 ±0.002 0.846 ±0.006 0.732 ±0.009	0.889 ±0.004 0.957 ±0.004 0.884 ±0.004	0.758 ±0.006 0.764 ±0.011 0.676 ±0.015
Fixed + hparam	0.898 ±0.003 0.957 ±0.001 0.883 ±0.005	0.899 ±0.005 0.956 ±0.003 0.899 ±0.005	0.917 ±0.002 0.955 ±0.002 0.873 ±0.003	0.794 ±0.009 0.867 ±0.012 0.790 ±0.010	0.795 ±0.005 0.847 ±0.005 0.735 ±0.016	0.892 ±0.005 0.959 ±0.003 0.887 ±0.005	0.760 ±0.003 0.767 ±0.007 0.687 ±0.014
Random search	0.902 ±0.003 0.962 ±0.002 0.888 ±0.002	0.895 ±0.005 0.958 ±0.004 0.894 ±0.005	0.917 ±0.000 0.958 ±0.000 0.874 ±0.000	0.804 ±0.008 0.881 ±0.008 0.800 ±0.009	0.804 ±0.010 0.859 ±0.006 0.745 ±0.010	0.893 ±0.003 0.959 ±0.002 0.888 ±0.003	0.764 ±0.006 0.790 ±0.011 0.700 ±0.016

Tab. 20. Three-way comparison at L3 (+ bond descriptors): classical ML, CTGP, and fair GNN baselines

Method	*metal_opt*	*metal_mbj*	*magnetic*	*slme*	*spillage*	*seebeck_p*	*powerfactor_n*
MLP	0.871 ±0.001 0.942 ±0.002 0.853 ±0.001	0.887 ±0.000 0.954 ±0.000 0.887 ±0.000	0.897 ±0.001 0.940 ±0.002 0.837 ±0.004	0.806 ±0.005 0.870 ±0.001 0.803 ±0.005	0.813 ±0.004 0.864 ±0.002 0.757 ±0.009	0.875 ±0.002 0.950 ±0.001 0.869 ±0.002	0.756 ±0.006 0.765 ±0.009 0.665 ±0.016
RF	0.887 ±0.000 0.951 ±0.000 0.871 ±0.000	0.894 ±0.000 0.959 ±0.000 0.893 ±0.000	0.899 ±0.000 0.939 ±0.000 0.831 ±0.000	0.797 ±0.000 0.884 ±0.000 0.790 ±0.000	0.810 ±0.000 0.860 ±0.000 0.725 ±0.000	0.885 ±0.000 0.956 ±0.000 0.880 ±0.000	0.752 ±0.000 0.789 ±0.000 0.622 ±0.000
DT	0.867 ±0.000 0.921 ±0.000 0.848 ±0.000	0.877 ±0.000 0.896 ±0.000 0.877 ±0.000	0.896 ±0.000 0.875 ±0.000 0.835 ±0.000	0.774 ±0.000 0.800 ±0.000 0.767 ±0.000	0.779 ±0.000 0.783 ±0.000 0.725 ±0.000	0.867 ±0.000 0.907 ±0.000 0.861 ±0.000	0.724 ±0.000 0.704 ±0.000 0.626 ±0.000
SVM	0.882 ±0.000 0.948 ±0.000 0.866 ±0.000	0.893 ±0.000 0.955 ±0.000 0.893 ±0.000	0.907 ±0.000 0.938 ±0.000 0.853 ±0.000	0.790 ±0.000 0.862 ±0.000 0.786 ±0.000	0.801 ±0.000 0.852 ±0.000 0.726 ±0.000	0.876 ±0.000 0.948 ±0.000 0.871 ±0.000	0.763 ±0.000 0.786 ±0.000 0.663 ±0.000
CTGP 1 + 1	0.890 ±0.010 0.955 ±0.005 0.874 ±0.013	0.895 ±0.002 0.956 ±0.002 0.895 ±0.002	0.913 ±0.004 0.954 ±0.003 0.865 ±0.005	0.782 ±0.009 0.871 ±0.009 0.776 ±0.010	0.793 ±0.008 0.849 ±0.004 0.726 ±0.010	0.891 ±0.006 0.960 ±0.003 0.885 ±0.007	0.759 ±0.008 0.768 ±0.012 0.680 ±0.017
CTGP 1 + 1 + S	0.886 ±0.014 0.950 ±0.008 0.871 ±0.015	0.889 ±0.013 0.952 ±0.007 0.889 ±0.013	0.836 ±0.111 0.864 ±0.132 0.781 ±0.114	0.760 ±0.042 0.839 ±0.054 0.755 ±0.046	0.761 ±0.061 0.840 ±0.016 0.712 ±0.043	0.889 ±0.005 0.957 ±0.004 0.884 ±0.005	0.754 ±0.018 0.763 ±0.026 0.684 ±0.025
CTGP 1 + 4	0.893 ±0.012 0.955 ±0.005 0.877 ±0.014	0.893 ±0.007 0.955 ±0.005 0.892 ±0.007	0.913 ±0.003 0.958 ±0.002 0.863 ±0.004	0.788 ±0.013 0.870 ±0.010 0.783 ±0.014	0.791 ±0.008 0.847 ±0.011 0.714 ±0.026	0.891 ±0.006 0.957 ±0.004 0.886 ±0.006	0.754 ±0.007 0.761 ±0.014 0.677 ±0.027
CTGP 1 + 4 + S	0.892 ±0.017 0.954 ±0.009 0.877 ±0.018	0.889 ±0.019 0.953 ±0.009 0.889 ±0.019	0.899 ±0.027 0.942 ±0.026 0.841 ±0.051	0.782 ±0.019 0.864 ±0.013 0.777 ±0.020	0.790 ±0.015 0.847 ±0.010 0.728 ±0.013	0.891 ±0.006 0.960 ±0.003 0.886 ±0.006	0.755 ±0.008 0.766 ±0.010 0.674 ±0.025
Fixed	0.888 ±0.004 0.952 ±0.002 0.872 ±0.005	0.896 ±0.003 0.954 ±0.002 0.896 ±0.003	0.916 ±0.003 0.956 ±0.002 0.872 ±0.004	0.785 ±0.007 0.867 ±0.004 0.782 ±0.007	0.800 ±0.005 0.846 ±0.004 0.730 ±0.013	0.887 ±0.003 0.955 ±0.003 0.882 ±0.003	0.759 ±0.003 0.762 ±0.004 0.679 ±0.012
Fixed + hparam	0.897 ±0.002 0.956 ±0.002 0.881 ±0.003	0.895 ±0.003 0.955 ±0.003 0.895 ±0.003	0.918 ±0.002 0.957 ±0.001 0.874 ±0.002	0.796 ±0.007 0.873 ±0.007 0.792 ±0.006	0.804 ±0.003 0.855 ±0.007 0.743 ±0.012	0.890 ±0.003 0.955 ±0.002 0.884 ±0.003	0.761 ±0.003 0.771 ±0.009 0.695 ±0.013
Random search	0.900 ±0.004 0.960 ±0.002 0.886 ±0.005	0.895 ±0.004 0.958 ±0.002 0.894 ±0.004	0.920 ±0.005 0.959 ±0.002 0.877 ±0.006	0.804 ±0.012 0.882 ±0.008 0.799 ±0.013	0.801 ±0.004 0.857 ±0.005 0.749 ±0.006	0.897 ±0.007 0.961 ±0.004 0.892 ±0.007	0.760 ±0.006 0.775 ±0.003 0.692 ±0.002

Taken together, the fair-baseline comparison shows that CTGP does not uniformly outperform fixed GNNs or random architecture search. Under the 301-evaluation budget shared by CTGP (1 + 1) and random search, evolutionary selection provides task-dependent rather than universal gains. The higher-budget (1 + 4) configurations are reported separately and should not be interpreted as strictly budget-matched comparisons with random search. Instead, the main benefits of CTGP lie in:

- interpretable, explicitly decoded convolution sequences;
- competitive performance on selected electronic-structure and topological-proxy tasks;
- favorable accuracy–size–search-cost trade-offs, as documented in Section 4.5.

5. Discussion

The results demonstrate that CTGP-based NAS can produce lightweight, architecturally transparent GNN architectures for JARVIS-DFT classification tasks without hand-designing layer stacks. The property-aware trends in Section 4.6 suggest that the search does not merely fit labels: operator choices and depth correlate with the structural length scales and geometric sensitivities known from the materials-physics literature (Choudhary et al., 2020; Xie & Grossman, 2018).

5.1. Representation and physical descriptors

The stepwise gain from L0 to L3 is largest for spillage and SLME, properties tied to band hybridization and optical absorption edges; Fig. A shows this as a systematic brightening of the corresponding rows from L0 through L3. This is consistent with ALIGNN's finding that explicit angle information improves electronic-structure predictions: the L2/L3 node features supply bond-distance and angle statistics without a dedicated line-graph module, and CTGP compensates partly by selecting operators that aggregate

the multi-hop neighborhood context. Bandgap and magnetic tasks benefit less from featurization alone, indicating that closing the gap to ALIGNN on these targets likely requires architectural mechanisms – not just richer inputs – for triplet-level message passing.

5.2. Comparison to reference models and baselines

With L0 inputs, CTGP is competitive with classical ML baselines on several benchmarks while offering explicit convolution sequences. The discovered models are compact in absolute parameter count and have low latency under the reported conditions. No like-for-like parameter or latency comparison with ALIGNN was conducted. Tab. 15 reports operator sequences, Fig. 2 reports selected widths, and dropout values are not shown; the results support architectural transparency rather than prediction-level explainability.

5.3. Evolutionary strategies

The four strategies analyzed (1 + 1, 1 + 1 surrogate, 1 + 4, 1 + 4 surrogate) serve distinct roles (Tab. 12, Figs 6 and 7). (1 + 1) is the cheapest per run; (1 + 4) typically offers the best cost–quality trade-off among completed runs. Surrogate-assisted search (Fig. 3) can reduce wall-clock time substantially (Fig. 7, Tab. 13), but Fig. 8 shows that the benefit depends on how aggressively the filter prunes candidates on each property.

5.4. Efficiency and deployability

Figs B–E describe accuracy-size-cost trade-offs among the evaluated CTGP runs. They do not establish a Pareto frontier relative to hand-designed GNNs because no hand-designed model is plotted with complexity and timing measured under the same conditions. Fig. 10 shows latency variation by benchmark and strategy. Graph size and convolution type were not independently varied or analyzed, so their separate contributions to latency cannot be inferred from this figure.

5.5. Interpretability

Tab. 15 reports operator sequences, Fig. 2 reports selected widths, and dropout values are not shown; the results support architectural transparency rather than prediction-level explainability. Relating these sequences to the property-specific patterns in Section 4.6 provides a bridge between computational search and materials interpretation that is difficult to obtain from monolithic hand-designed models.

5.6. Limitations

The (1 + 1) and (1 + 4) searches use different evaluation budgets, so their differences cannot be attributed solely to offspring-population size. Surrogate filtering changes both computational cost and the number of evaluated candidates and can reduce accuracy. The property-aware interpretation is post hoc and is not supported by operator ablations or prediction-level explanations. A single graph construction (8 Å cutoff, at most 12 neighbors) is used and line-graph message passing is outside the search space. Test-set handling is reported transparently, as described in Section 3.5.

6. Conclusion

In this work, concat-tree genetic programming (CTGP), a tree-structured and mutation-driven evolutionary NAS framework inspired by both tree-based GP and CGP, was introduced for the automatic discovery of lightweight GNN architectures for materials property classification. Evaluated across seven diverse JARVIS-DFT benchmarks and four node-representation levels, the discovered models achieve competitive benchmark accuracy with modest parameter counts, although they remain below the dated leaderboard leaders in Tab. 11.

Crucially for computational materials engineering, the CTGP-evolved architectures provide millisecond-scale inference latencies. This extreme computational efficiency overcomes the bottlenecks associated with heavy graph models, making them promising surrogate classifiers, subject to validation of end-to-end screening time and generalization beyond the fixed JARVIS splits. By enabling the rapid evaluation of millions of candidate structures, this approach provides experimental researchers with a practical, ultrafast tool for the early-stage filtering of novel functional materials.

Furthermore, our property-aware analysis demonstrates that the evolutionary search bridges machine learning and solid-state physics. The observed operator and depth patterns are qualitatively consistent with selected materials-physics considerations, but they do not establish causal physical mechanisms. Geometric descriptors are introduced at L2 and L3, not L1. These features improve several tasks, but the study does not provide a like-for-like computational-overhead comparison with ALIGNN.

Future work will focus on integrating these ultra-fast surrogate models into end-to-end materials discovery pipelines and digital twin frameworks. Ultimately, coupling these highly efficient computational screening methods with experimental validation will further enhance the communication between data-driven materials design and laboratory synthesis.

Acknowledgments

The work was undertaken within the framework of fundamental statutory research financed by the Ministry of Science and Higher Education (Poland), Grant No. 16.16.110.663. We would like to thank Alexandra Elbakyan for her work on the accessibility of the science.

CRediT author statement

Maciej Krzywda: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, visualization, writing – original draft, writing – review & editing; Mariusz Wermiński: formal analysis, writing – original draft, writing – review & editing; Szymon Łukasik: conceptualization, supervision, writing – review & editing; Amir H. Gandomi: conceptualization, supervision, writing – review & editing.

Data availability

The datasets used in this study are publicly available through the JARVIS-DFT database maintained by the National Institute of Standards and Technology (NIST) (https://jarvis.nist.gov/). The experiments were conducted using the official train, validation, and test splits provided by the JARVIS-Leaderboard (https://pages.nist.gov/jarvis_leaderboard/). The source code, experimental configurations, and scripts required to reproduce the results are publicly available at https://github.com/mkrzywda/ctgt.

References

Bhattacharya, B., Dinesh Kumar, G. R., Agarwal, A., Erkoç, Ş., Singh, A., & Chakraborti, N. (2009). Analyzing Fe–Zn system using molecular dynamics, evolutionary neural nets and multi-objective genetic algorithms. *Computational Materials Science*, *46*(4), 821–827. https://doi.org/10.1016/j.commatsci.2009.04.023

Chakraborti, N. (2004). Genetic algorithms in materials design and processing. *International Materials Reviews*, *49*(3–4), 246–260. https://doi.org/10.1179/095066004225021909

Chakraborti, N. (2023). *Data-Driven Evolutionary Modeling in Materials Technology*. CRC Press. https://doi.org/10.1201/9781003201045

Choudhary, K., & DeCost, B. (2021). Atomistic line graph neural network for improved materials property predictions. *npj Computational Materials*, *7*, Article 185. https://doi.org/10.1038/s41524-021-00650-1

Choudhary, K., Garrity, K. F., Reid, A. C. E., DeCost, B., Biacchi, A. J., Hight Walker, A. R., Trautt, Z., Hattrick-Simpers, J., Gilad Kusne, A., Centrone, A., Davydov, A., Jiang, J., Pachter, R., Cheon, G., Reed, E., Agrawal, A., Qian, X., Sharma, V., Zhuang, H., … Tavazza, F. (2020). The joint automated repository for various integrated simulations (JARVIS) for data-driven materials design. *npj Computational Materials*, *6*, Article 173. https://doi.org/10.1038/s41524-020-00440-1

Curtarolo, S., Hart, G. L. W., Buongiorno Nardelli, M., Mingo, N., Sanvito, S., & Levy, O. (2013). The high-throughput highway to computational materials design. *Nature Materials*, *12*(3), 191–201. https://doi.org/10.1038/nmat3568

Dimitriu, R. C., Bhadeshia, H. K. D. H., Fillon, C., & Poloni, C. (2009). Strength of ferritic steels: Neural networks and genetic programming. *Materials and Manufacturing Processes*, *24*(1), 10–15. https://doi.org/10.1080/10426910802539796

Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, *20*(55), 1–21. https://doi.org/10.48550/arXiv.1808.05377

Giri, B. K., Pettersson, F., Saxén, H., & Chakraborti, N. (2013). Genetic programming evolved through bi-objective genetic algorithms applied to a blast furnace. *Materials and Manufacturing Processes*, *28*(7), 776–782. https://doi.org/10.1080/10426914.2013.763953

Halder, C., Madej, L., Pietrzyk, M., & Chakraborti, N. (2015). Optimization of cellular automata model for the heating of dual-phase steel by genetic algorithm and genetic programming. *Materials and Manufacturing Processes*, *30*(4), 552–562. https://doi.org/10.1080/10426914.2014.994765

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In I. Guyon, U. von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 1024–1034). Curran Associates. https://papers.nips.cc/paper_files/paper/2017/hash/5dd9db5e033da9c6fb5ba83c7a7ebea9-Abstract.html

Hernandez, A., Balasubramanian, A., Yuan, F., Mason, S. A. M., & Mueller, T. (2019). Fast, accurate, and transferable many-body interatomic potentials by symbolic regression. *npj Computational Materials*, *5*, Article 112. https://doi.org/10.1038/s41524-019-0249-1

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., & Persson, K. A. (2013). Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, *1*(1), Article 011002. https://doi.org/10.1063/1.4812323

Jennings, P. C., Lysgaard, S., Strabo Hummelshøj, J., Vegge, T., & Bligaard, T. (2019). Genetic algorithms for computational materials discovery accelerated by machine learning. *npj Computational Materials*, *5*, Article 46. https://doi.org/10.1038/s41524-019-0181-4

Kipf, T. N., & Welling, M. (2017). *Semi-supervised classification with graph convolutional networks*. Openreview.net. https://openreview.net/forum?id=SJU4ayYgl

Klicpera, J., Groß, J., & Günnemann, S. (2020). *Directional message passing for molecular graphs*. Openreview.net. https://openreview.net/forum?id=WBCf7qTwcR

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

Krzywda, M. (2025). Graph neural network architecture search via hybrid genetic algorithm with parallel tempering. In M. Cha, C. Park, N. Park (Eds.), *CIKM '25: Proceedings of the 34th ACM International Conference on Information and Knowledge Management* (pp. 6793–6796). Association for Computing Machinery. https://doi.org/10.1145/3746252.3761661

Krzywda, M. (2026). *CTGP – Concat-Tree Genetic Programming for GNN Architecture Search* [Source code]. Github. https://github.com/mkrzywda/ctgt

Krzywda, M., Liu, Y., Łukasik, S., & Gandomi, A. H. (2025). Unveiling the search space of simple contrastive graph clustering with Cartesian genetic programming. In G. Ochoa (Ed.), *GECCO '25 Companion: Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 2380–2383). Association for Computing Machinery. https://doi.org/10.1145/3712255.3734538

Krzywda, M., Lukasik, S., & Gandomi, A. H. (2025a). Applying evolutionary techniques to enhance graph convolutional networks for node classification: case studies. In M. Bolanowski, M. Ganzha, L. Maciaszek, M. Paprzycki, D. Ślęzak (Eds.), *Proceedings of the 20th Conference on Computer Science and Intelligence Systems (FedCSIS 2025)* (pp. 321–326). PTI, IEEE. https://doi.org/10.15439/2025F0041

Krzywda, M., Łukasik, S., & Gandomi, A. H. (2025b). Linear genetic programming for design graph neural networks for node classification. In G. Ochoa, B. Filipič (Eds.), *GECCO '25 Companion: Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 2167–2171). Association for Computing Machinery. https://doi.org/10.1145/3712255.3734278

Mahanta, B. K., & Chakraborti, N. (2021). Evolutionary data driven modelling and many objective optimization of non linear noisy data in the blast furnace iron making process. *Computer Methods in Materials Science*, *21*(3), 163–175. https://doi.org/10.7494/cmms.2021.3.0733

Miller, J. F. (2020). Cartesian genetic programming: its status and future. *Genetic Programming and Evolvable Machines*, *21*(1–2), 129–168. https://doi.org/10.1007/s10710-019-09360-6

Roscher, R., Bohn, B., Duarte, M. F., & Garcke, J. (2020). Explainable machine learning for scientific insights and discoveries. *IEEE Access*, *8*, 42200–42216. https://doi.org/10.1109/ACCESS.2020.2976199

Roy, S., Saini, B. S., Chakrabarti, D., & Chakraborti, N. (2020). Mechanical properties of micro-alloyed steels studied using a evolutionary deep neural network. *Materials and Manufacturing Processes*, *35*(6), 611–624. https://doi.org/10.1080/10426914.2019.1660786

Schmidt, J., Marques, M. R. G., Botti, S., & Marques, M. A. L. (2019). Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, *5*, 83. https://doi.org/10.1038/s41524-019-0221-0

Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A., & Müller, K.-R. (2018). SchNet – A deep learning architecture for molecules and materials. *The Journal of Chemical Physics, 148*(24), 241722. https://doi.org/10.1063/1.5019779

Suganuma, M., Shirakawa, S., & Nagao, T. (2017). A genetic programming approach to designing convolutional neural network architectures. In P. A. N. Bosman (Ed.), *GECCO '17: Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 497–504). Association for Computing Machinery. https://doi.org/10.1145/3071178.3071229

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). *Graph attention networks*. ArXiv. https://doi.org/10.48550/arXiv.1710.10903

Wu, F., Zhang, T., de Souza Jr., A. H., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). *Simplifying Graph Convolutional Networks*. ArXiv. https://doi.org/10.48550/arXiv.1902.07153

Xie, T., & Grossman, J. C. (2018). Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical Review Letters*, *120*(14), Article 145301. https://doi.org/10.1103/PhysRevLett.120.145301

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). *How powerful are graph neural networks?* ArXiv. https://doi.org/10.48550/arXiv.1810.00826

Yuan, F., & Mueller, T. (2017). Identifying models of dielectric breakdown strength from high-throughput data via genetic programming. *Scientific Reports*, *7*, Article 17594. https://doi.org/10.1038/s41598-017-17535-3