GExPC: An Explainable Evolutionary Approach for Crack Segmentation with Probabilistic Circuits

Erik Nielsen
Giovanni Iacca
erik.nielsen@unitn.it
giovanni.iacca@unitn.it
University of Trento
Trento, Italy

Abstract

Crack segmentation is a fundamental task in structural inspection and has traditionally relied on resource-intensive, black-box Deep Learning (DL) models. To address this, we introduce GExPC, an explainable approach combining Probabilistic Neural Circuits (PCNets) with Grammatical Evolution (GE) to automatically discover compact, task-adapted architectures. Evaluated across six benchmarks, GExPC uses fewer than 100 learnable parameters yet achieves competitive performance—particularly in recall and shape preservation—against state-of-the-art DL models requiring millions of parameters. Our results demonstrate that evolutionary search over PC structures provides a highly efficient, interpretable alternative to standard DL for vision-based structural inspection.

CCS Concepts

• **Mathematics of computing** → **Probabilistic inference problems**; • **Computing methodologies** → **Genetic programming**; *Visual inspection.*

Keywords

Grammatical Evolution, Probabilistic Neural Circuits, Crack Segmentation

ACM Reference Format:
Erik Nielsen and Giovanni Iacca. 2026. GExPC: An Explainable Evolutionary Approach for Crack Segmentation with Probabilistic Circuits. In *Genetic and Evolutionary Computation Conference (GECCO Companion '26), July 13–17, 2026, San Jose, Costa Rica.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3795101.3805281

1 Introduction

In civil engineering, crack segmentation is critical for detecting structural anomalies and assessing road reliability. While Deep Learning (DL) methods achieve high-quality segmentations under diverse conditions, state-of-the-art models are highly memory-intensive, opaque due to architectural complexity, and dependent on massive annotated datasets.

To address these limitations, we present GExPC, a novel method for generating explainable, lightweight crack segmentation models. Our approach uses Probabilistic Neural Circuits (PCNets) [11], into which we uniquely integrate gate [7] and residual nodes [8]. This combination enhances structural expressiveness and helps identify highly influential network components. To determine the optimal topology, GExPC employs a two-phase training strategy (Figure 1). First, Grammatical Evolution (GE) explores candidate architectures on a data subset. Second, the selected architecture undergoes full-dataset training to optimize its weights, yielding an efficient, explainable network. Examples of the final network are shown in Figure 2.

We evaluated GExPC across six standard benchmarks: AEL, Crack500, DeepCrack, GAPS384, cracktree200, and CrackSeg9k, described in [1]. Despite using fewer than 100 parameters, orders of magnitude smaller than existing DL approaches, our method consistently achieves competitive performance. This drastic size reduction enables full model interpretability. The code of the proposed pipeline is publicly available on GitHub[1]. To summarize, our main contributions are:

- We propose GExPC, a GE-based method to optimize PCNets.
- We validate it across six crack segmentation datasets, showing competitive performance against state-of-the-art DL models.
- We provide an interpretation of the generated models.

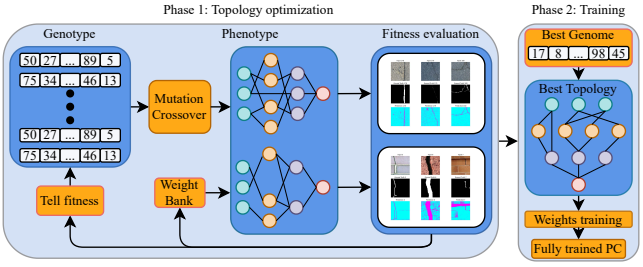


Figure 1: Overall workflow of the proposed GExPC approach.

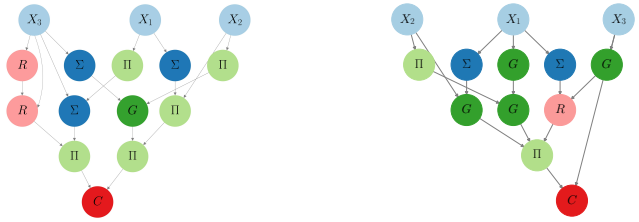
2 Methodology

We now detail the PCNet architecture and its probabilistic semantics, followed by our GE-based, two-phase training procedure designed to automatically discover and optimize efficient network topologies.

[1]https://github.com/NielsenErik/GExPC

GECCO Companion '26, San Jose, Costa Rica

ACM ISBN 979-8-4007-2488-6/2026/07
https://doi.org/10.1145/3795101.3805281



(a) AEL (b) DeepCrack

Figure 2: Examples of PCNet topologies found by GExPC on AEL (a) and DeepCrack (b) datasets. Input nodes are denoted by X_i (with i ranging from 1 to 3, corresponding to the three channels of the images). $\sum$, $\prod$, G, and R indicate, respectively, Sum, Product, Gate, and Residual nodes. C indicates the final Classification node. Note that the architecture supports skip connections, allowing edges to bypass adjacent layers and connect to nodes further down the hierarchy.

2.1 Probabilistic neural circuits

Our approach relies on Einsum networks [4], where input features are transformed into Probability Density Functions (PDFs) and recursively recombined. To process spatial data efficiently, all intermediate nodes operate in log-likelihood space, outputting maps $\ell(x) \in \mathbb{R}^{B\times H\times W}$ for a batch of size B and spatial dimensions $H\times W$. We extend traditional Einsum networks by introducing *Gate* and *Residual* nodes for greater expressiveness.

Input nodes model a batch from a single-feature channel x_i using a mixture of Gaussian ($\mathcal{N}$), Laplace ($\mathcal{L}$), and Student-t (t) distributions. To ensure numerical stability, the gated log-density is computed as $\ell_{\text{in},i}(x_i) = g_i \log \sum_{m\in\{\mathcal{N},\mathcal{L},t\}} \pi_{i,m} p_{i,m}(x_i)$, where g_i is a learnable scaling gate, $\pi_{i,m}$ are softmax-derived mixture weights, and $p_{i,m}$ are the component densities.

Sum and Product nodes dictate the core probabilistic routing. A sum node computes a learnable weighted mixture over its children $\{c_i\}_{i=1}^N$ in log-space: $\ell_\oplus(x) = \log \sum_{i=1}^N \exp(\log w_i + \ell_{c_i}(x))$. A product node assumes independence among its children, simply summing their log-likelihoods: $\ell_\otimes(x) = \sum_{i=1}^N \ell_{c_i}(x)$.

Gate nodes softly select between two children (a and b) using a learnable coefficient $\lambda \in (0,1)$ parameterized by a sigmoid function. This allows the model to suppress less informative branches during the evolutionary search:

$$\ell_{\text{gate}}(x) = \log\big(\exp(\ell_a(x)+\log\lambda) + \exp(\ell_b(x)+\log(1-\lambda))\big).$$

Residual nodes incrementally refine the circuit by mixing a base sub-circuit with an additional corrective component using a learnable sigmoid parameter m:

$$\ell_{\text{res}}(x) = \log\big(\exp(\ell_{\text{base}}(x) + \log(1-m)) + \exp(\ell_{\text{sub}}(x)+\log m)\big).$$

Classifier nodes terminate the network by stacking K class-specific log-likelihoods, obtaining pixel-wise predictions via

$$\arg\max_k \ell_{\text{cls}}(x)_{b,k,h,w}.$$

2.2 Grammatical evolution

To automatically discover optimal PCNet topologies, we employ a Grammatical Evolution (GE) approach that maps integer-valued genomes (codons) into Directed Acyclic Graphs (DAGs). The GE mapper recursively expands a grammar to select production rules for our defined node types (*Input*, *Sum*, *Product*, *Gate*, and *Residual*), strictly enforcing constraints on maximum network depth and branching factor. Codon wrapping and deterministic fallback rules ensure robust decoding when genomes are exhausted [3]. To maximize representational efficiency, a builder module constructs a DAG, rather than a standard tree, by sharing input nodes across the circuit and memorizing identical subtrees for node reuse. The evolutionary loop optimizes these architectures using elitism, pairwise tournament selection, single-point crossover, and random codon mutation. Fitness is evaluated using the validation loss. Selection relies on a lexicographic ordering of fitness followed by model complexity, inherently favoring lighter models. To accelerate convergence, offspring parameters are initialized by averaging compatible parent parameters, enabling effective parameter inheritance across generations (further detailed in Section 2.3).

2.3 Two-phase optimization and training

Evaluating $n_{pop} \times n_{gen}$ architectures on the full dataset is computationally prohibitive. Therefore, GExPC employs a two-phase approach: a fast evolutionary search where candidate topologies are trained on a random data subset, followed by an *a posteriori* full-dataset training of the single best topology. To further accelerate the evolutionary phase, we implement a weight caching mechanism. Genomes are hashed to store and retrieve trained parameters, allowing partial weight reuse for similar topologies. Additionally, we apply Lamarckian inheritance [5] during crossover: offspring networks are initialized by averaging compatible parameters from their parents, speeding up convergence without sacrificing exploratory variance. Throughout both training phases, networks are optimized via Adam using a composite loss function: $\mathcal{L} = \alpha\mathcal{L}_{\text{CE}} + \beta\mathcal{L}_{\text{Dice}} + \gamma(1 - \text{clDice})$. This combines binary cross-entropy ($\mathcal{L}_{\text{CE}}$), soft Dice loss ($\mathcal{L}_{\text{Dice}}$) [2], and a connectivity-aware clDice term based on differentiable soft-skeletonization [1]. We set $\alpha = 0.2$, $\beta = 0.1$, and $\gamma = 0.3$ to prioritize structural skeletonization while maintaining pixel-wise accuracy.

2.4 Evaluation metrics

In crack segmentation, localizing thin, elongated structures is prioritized over exact area estimation. Therefore, our primary evaluation metric is the Centerline Intersection over Union (clIoU) [1]. Using a tolerance of 4 pixels, clIoU emphasizes crack continuity by calculating overlap based on the morphological dilation of ground-truth and predicted skeletons. We also report standard pixel-wise Intersection over Union (IoU), Precision, Recall, F1-score, and number of trainable parameters to provide a comprehensive assessment.

3 Experimental evaluation

The experimental evaluation aims to evaluate GExPC by addressing the following research questions:

RQ1 Can GExPC achieve performance comparable to state-of-the-art DL–based approaches?

RQ2 Does GExPC provide effective results while preserving model transparency?

3.1 Datasets

To ensure a broad evaluation, we tested our approach on six different datasets widely used in the literature: AEL, Crack500, DeepCrack, GAPS384, cracktree200, and CrackSeg9k, described in [1].

3.2 Baselines

We compared GExPC against three widely adopted, representative deep learning architectures of varying complexity: DeepCrack [10], U-Net [6], and U-Net++ [9]. To further contextualize our performance, we include relevant results directly from the recent Omnicrack30k benchmark study [1]. In our analysis (see Figure 3), these external results are denoted with an "(o)" and include CrackFormer, DeepCrackL, HMA, Conglo, Topo, and nnU-Net.

Table 1: GE configuration.

Parameter	Value
Generations	10
Population size	20
Genome length	96
Crossover rate	0.9
Mutation rate	0.05
Elite individuals	2
Max depth	3
Max branching	3
Tournament size	2

Table 2: Training configuration.

Parameter	Value
Epochs (GE phase)	50
Epochs (Full train)	80
Train set (GE phase)	20%
Validation split	20%
Image size	256×256

3.3 Experimental setup

Our evaluation configurations are detailed in Tables 1 and 2. To keep search costs feasible, the GE budget was kept relatively limited; however, larger evolutionary budgets could potentially yield even better models. To accommodate their architectural differences, DL baselines were trained on an NVIDIA RTX 4080 GPU, while GExPC was executed on a 20-core CPU cluster. Both setups enforced a strict 10-hour maximum training time per run. Table 3 reports the performance in terms of clIoU. Due to the stochastic nature of evolutionary algorithms, GExPC was executed five independent times per dataset (reporting mean, standard deviation, and best result), whereas the standard DL baselines were evaluated using a single fixed-seed run.

3.4 RQ1: Performance comparison

We evaluate whether GExPC achieves competitive crack segmentation while drastically reducing parameter count (Table 3 and Figure 3). GExPC performs on par with U-Net and U-Net++ using fewer than 100 parameters, compared to the tens of millions required by CNN baselines. While Holm-Bonferroni ranking based on clIoU shows a statistically significant gap compared to state-of-the-art DeepCrack, it confirms no significant difference between GExPC and the U-Net variants. This proves that effective crack segmentation does not require heavy over-parameterization.

Moreover, while DL models generally achieve higher absolute scores, especially on larger datasets, GExPC yields strong recall and competitive clIoU on datasets like AEL and DeepCrack, effectively capturing thin, elongated structures. It also remains robust

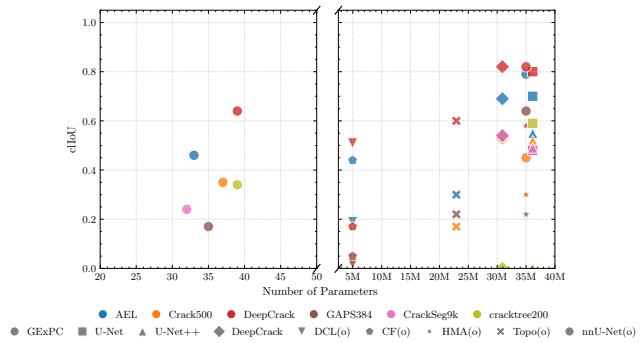


Figure 3: Trade-off between clIoU and the number of learnable parameters for GExPC vs. the DL-based methods. The left part of the plot highlights the GExPC models, whose parameter count lies in the range $[30, 40]$, while the right one displays the DL baselines, with parameter counts ranging from 10^6 to 5×10^7. The color indicates the datasets, while the marker shape indicates the methods. Results denoted by "(o)" are directly taken from the benchmark study [1].

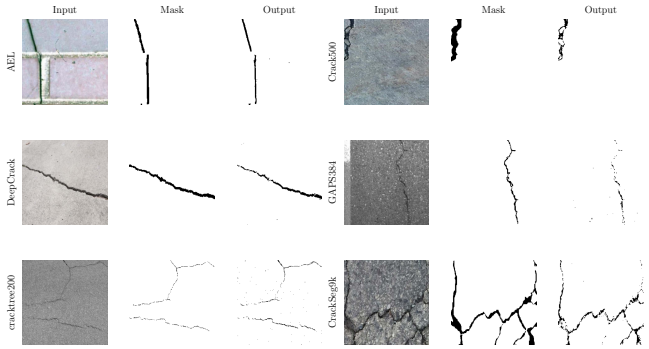


Figure 4: Example of segmentation of GExPC over each dataset. "Mask" represents the ground-truth labels of the datasets, while "Output" shows the resulting segmentation.

in highly imbalanced scenarios (<1% crack pixels). Even on challenging benchmarks (GAPS384, cracktree200) where all methods suffer performance drops, GExPC maintains competitive recall and preserves crack continuity, a crucial aspect for structural inspection.

The trade-off between performance and the number of model parameters is visible in Figure 3, where GExPC occupies a distinct region characterized by very low model complexity and lower (but still practically valid) clIoU scores, in contrast with DL baselines, which instead achieve higher performance but with higher model complexity. These results emphasize that GExPC is not designed to outperform large DL models in absolute terms, but rather to provide a lightweight, explainable, and tractable alternative that remains competitive across diverse datasets. Finally, Figure 4 shows some examples of segmentation over each dataset.

3.5 RQ2: Explainability

The use of PCNets not only enhances the tractability of the solution but also enables the construction of an explainable model. GExPC

Table 3: Quantitative comparison across benchmark datasets. For our method (GExPC), results are reported as mean ± std (best) across five independent runs to indicate the average performance, variance, and the highest achieved performance.

Method	clIoU	IOU	Prec.	Recall	F1	Params
AEL						
GExPC	0.39±.05 (0.46)	0.62±.04 (0.66)	0.36±.07 (0.43)	0.82±.03 (0.80)	0.48±.06 (0.54)	33.0±5.5 (33)
U-Net	0.70	0.79	0.73	0.75	0.73	36.2M
U-Net++	0.55	0.78	0.73	0.77	0.73	36.2M
DeepCrack	0.69	0.83	0.76	0.85	0.80	30.9M
Crack500						
GExPC	0.30±.05 (0.35)	0.57±.01 (0.58)	0.39±.10 (0.51)	0.47±.12 (0.33)	0.38±.01 (0.36)	26.0±11.1 (37)
U-Net	0.49	0.72	0.63	0.69	0.64	36.2M
U-Net++	0.52	0.74	0.59	0.77	0.66	36.2M
DeepCrack	0.53	0.76	0.64	0.79	0.70	30.9M
DeepCrack						
GExPC	0.63±.01 (0.64)	0.65±.01 (0.66)	0.44±.02 (0.47)	0.78±.02 (0.76)	0.51±.01 (0.53)	31.2±5.5 (39)
U-Net	0.81	0.70	0.77	0.82	0.78	36.2M
U-Net++	0.86	0.61	0.81	0.86	0.83	36.2M
DeepCrack	0.82	0.69	0.77	0.85	0.80	30.9M

Method	clIoU	IOU	Prec.	Recall	F1	Params
GAPS384						
GExPC	0.12±.04 (0.17)	0.36±.05 (0.41)	0.02±.00 (0.03)	0.53±.11 (0.42)	0.05±.00 (0.05)	32.4±4.7 (35)
U-Net	0.48	0.66	0.51	0.50	0.47	36.2M
U-Net++	0.49	0.66	0.51	0.51	0.48	36.2M
DeepCrack	0.55	0.70	0.52	0.61	0.54	30.9M
cracktree200						
GExPC	0.27±.07 (0.34)	0.48±.01 (0.49)	0.04±.01 (0.05)	0.41±.11 (0.40)	0.08±.02 (0.09)	35.0±4.9 (39)
U-Net	0.59	0.66	0.48	0.53	0.48	36.2M
U-Net++	0.00	0.49	0.00	0.00	0.00	36.2M
DeepCrack	0.00	0.49	0.00	0.00	0.00	30.9M
CrackSeg9k						
GExPC	0.18±.09 (0.24)	0.41±.20 (0.50)	0.19±.09 (0.21)	0.40±.23 (0.34)	0.18±.06 (0.21)	34.6±5.0 (32)
U-Net	0.48	0.66	0.51	0.50	0.47	36.2M
U-Net++	0.49	0.66	0.51	0.51	0.48	36.2M
DeepCrack	0.55	0.70	0.52	0.61	0.54	30.9M

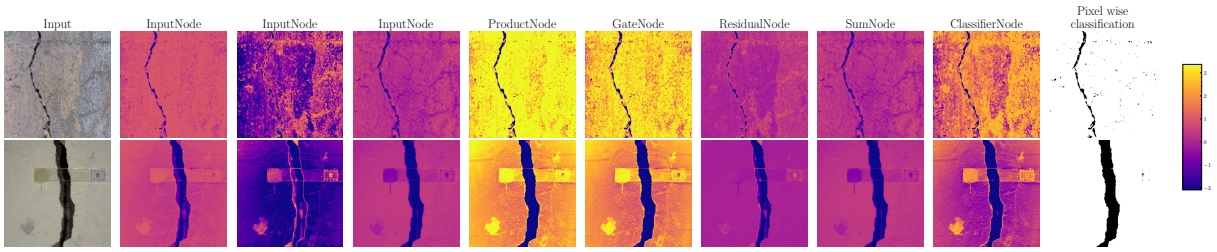


Figure 5: Pixel-wise distributions of the nodes composing the best model found by GExPC trained on the DeepCrack dataset, illustrating the intermediate probabilistic representations that lead to the final pixel-wise binary classification.

allows for a clear global view of the model, since the resulting PCNets contain only a very limited number of learnable parameters, and their weight distributions can be directly inspected and interpreted. Two examples of PCNet topologies found by GExPC are reported in Figure 2. Finally, thanks to the integration of the PCNet, GExPC guarantees transparency in how the input image is progressively mapped into probabilistic representations up to the final output, as shown in the example reported in Figure 5.

4 Conclusions

In this paper, we presented GExPC, an explainable crack segmentation framework based on PCNets. By integrating GE with a structured probabilistic representation, GExPC addresses several key limitations of contemporary DL approaches devised for this problem, namely excessive model complexity and limited explainability. Extensive experiments across six benchmark datasets show that GExPC achieves competitive segmentation performance while relying on orders of magnitude fewer parameters than DL models. Although large DL models tend to attain higher absolute scores, particularly on datasets with abundant training data, GExPC exhibits robust behavior under severe class imbalance and consistently preserves crack continuity. Beyond quantitative performance, a central advantage of the approach is its intrinsic explainability.

Future work will focus on extending the proposed approach toward a mixture-of-experts formulation, leveraging the limited number of parameters to specialize multiple PCs for different crack characteristics. We will also explore whether the proposed approach can achieve strong performance in other application domains and assess its potential in unsupervised or self-supervised settings.

References

[1] Christian Benz and Volker Rodehorst. 2024. Omni-Crack30k: A Benchmark for Crack Segmentation and the Reasonable Effectiveness of Transfer Learning. 3876–3886 pages.
[2] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. 2016. V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. 565-571 pages.
[3] Michael O'Neil and Conor Ryan. 2003. *Grammatical Evolution.* Springer US, Boston, MA, 33–47.
[4] Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van Den Broeck, Kristian Kersting, and Zoubin Ghahramani. 2020. Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits. 7563–7574 pages.
[5] Jonas Prellberg and Oliver Kramer. 2018. Lamarckian Evolution of Convolutional Neural Networks. In *Parallel Problem Solving from Nature (PPSN).* Springer International Publishing, Cham, 424–435.
[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI).* Springer International Publishing, Cham, 234–241.
[7] Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. 2022. Conditional sum-product networks: Modular probabilistic circuits via gate functions. *International Journal of Approximate Reasoning* 140 (2022), 298–313.
[8] Ke Zhang, Miao Sun, Tony X Han, Xingfang Yuan, Liru Guo, and Tao Liu. 2017. Residual networks of residual networks: Multilevel residual networks. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 6 (2017), 1303–1314.
[9] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. 2020. UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *IEEE Transactions on Medical Imaging* 39, 6 (2020), 1856–1867.
[10] Qin Zou, Zheng Zhang, Qingquan Li, Xianbiao Qi, Qian Wang, and Song Wang. 2019. Deepcrack: Learning Hierarchical Convolutional Features for Crack Detection. *IEEE Transactions on Image Processing* 28, 3 (2019), 1498–1512.
[11] Pedro Zuidberg Dos Martires. 2024. Probabilistic Neural Circuits. *AAAI Conference on Artificial Intelligence* 38, 15 (Mar. 2024), 17280–17289.