AttentionCap: Transformer Based Capacitance Matrix Learning Toward Full-Chip Extraction

Jiechen Huang[1], Hector R. Rodriguez[1], Dingcheng Yang[1], Zuochang Ye[2], Yibo Lin[3], Wenjian Yu[1]*
[1]Dept. Computer Science & Tech., BNRist, Tsinghua Univ., Beijing, China;
[2]School of IC, BNRist, Tsinghua Univ., Beijing, China; [3]School of IC, Peking Univ., Beijing, China.

Abstract

As capacitance extraction accuracy of rule-based pattern matching becomes difficult to sustain at advanced nodes, a growing trend emerges to develop deep-learning-based 2D capacitance models. However, existing MLP- and CNN-based methods constrain their input to fixed metal-layer combinations in a specific process node, limiting their usability in practice. Recognizing the inherent similarity between capacitance matrix and the prevailing attention mechanism, we propose AttentionCap, a customized Transformer for capacitance matrix learning, with a Gram representation framework, a physics-aligned symmetric-attention output layer, and a novel normalized Laplacian loss. We also introduce a process-node embedding to enable multi-node learning. Trained on synthetic data, AttentionCap attains 0.67%/3.99% self/coupling-capacitance error on unseen real designs under a multi-layer and multi-node setting, surpassing the CNN-Cap baseline with 4.6×/5.7× lower self/coupling error and 192× faster inference speed. A pretrained AttentionCap accurately transfers to an unseen node with only 5K samples and 4K finetuning steps. With sufficient accuracy on unseen real designs and strong transferability to new process nodes, AttentionCap offers highly practical value for modern EDA workflows. Code and data are available at https://github.com/THU-numbda/AttentionCap.

Keywords

Capacitance Extraction, Cross-Sectional Pattern Matching, Transformer, Attention Mechanism

ACM Reference Format:
Jiechen Huang[1], Hector R. Rodriguez[1], Dingcheng Yang[1], Zuochang Ye[2], Yibo Lin[3], Wenjian Yu[1]*. 2026. AttentionCap: Transformer Based Capacitance Matrix Learning Toward Full-Chip Extraction. In *63rd ACM/IEEE Design Automation Conference (DAC '26), July 26–29, 2026, Long Beach, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3770743.3804115

1 Introduction

Parasitic capacitance extraction solves the electric coupling among interconnects in integrated circuits (ICs) and delivers a realistic circuit model for accurate timing/signal-integrity/power analysis [5]. It has become a pronounced bottleneck at advanced process nodes, as the accuracy of pattern matching methods becomes difficult to maintain, while accurate field solvers face scalability challenges with the ever-growing design size and complexity [30].

Full-chip capacitance extraction uses pattern matching method with 2D cross-sectional pattern libraries, which match 2D slices of layouts to predefined patterns or interpolation formulas [8]. These libraries require tedious manual design and tuning to maintain accuracy for different process nodes, and are often proprietary, which has motivated the use of deep learning as a compelling alternative [29]. CNN-Cap [27] has established the paradigm of using density grids to represent 2D conductor geometries [2, 23, 26]. It provides distance-aware input for convolutional neural networks (CNNs) to learn the capacitance among conductors. Some studies also adopted coordinate-based input for multilayer perceptrons (MLPs) [3, 18, 31]. However, existing 2D capacitance models either admit only a fixed number of conductors [18, 31] or output only the capacitance of one conductor pair [2, 3, 23, 27], limited by the fixed-dimension nature of MLPs or CNNs. More critically, they exhibit limited generalizability because the model is trained for each metal-layer combination (typically 3 layers like M1-M3-M5) [2, 3, 18, 23, 27, 31], thus requiring numerous datasets and retrained models to cover a process node.

The Transformer architecture [24] has achieved state-of-the-art results with remarkable generalizability across domains such as large language models (LLMs) [10], computer vision [11], and IC design [15, 25]. At its core, the self-attention mechanism learns the pairwise relations among n inputs via an $n \times n$ *attention matrix*, and accordingly refines the representation of each input. By effectively capturing complex relations, attention mechanism endows Transformers with strong representational power, making it the *de facto* foundation of modern AI [21].

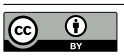
Recognizing the structural similarity that both capacitance matrix and attention mechanism capture an n-to-n interaction, we propose *AttentionCap*, which leverages the prevailing Transformer architecture for capacitance matrix learning toward full-chip extraction. While prior studies mainly focus on individual capacitance

Table 1: AttentionCap vs. CNN-Cap-like methods.

	CNN-Cap-like [2, 23, 27]	AttentionCap
Input represent.	Density grids	Coordinates
Input conductors	n (variable)	n (variable)
Output dimension	Scalar (total or coupling)	n×n **(full matrix)**
Full extraction	n^2 inference passes	**1 inference pass**
Layer combination	Fixed (3 layers)	**Any**
Process coverage*	$\binom{m}{3}$ datasets/models	**1 dataset/model**
Multi-Node	Not explored	**1 model**

* "Process coverage" refers to the necessary datasets and retrained models to cover all 3-layer combinations in an m-layer process node.

This work is supported by National Science and Technology Major Project (2021ZD0114703) and the MIND project (MINDXZ202406). *Corresponding author.



DAC '26, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804115

elements [2, 3, 23, 27], the $n \times n$ *capacitance matrix* fully characterizes an n-conductor layout, quantifying the coupling strength between each conductor pair. This relation is determined not only by their distance but also by other conductors and dielectric environment, which inherently limits distance-based representations such as density grids in CNN-Cap-like methods [27] and 3D-distance-based graphs in GNN-Cap for 3D extraction [17]. In AttentionCap, we input a conductor sequence, train a Transformer to map each conductor to a "capacitance embedding", and dot-product them via symmetric attention to produce the full capacitance matrix. As summarized in Table 1, AttentionCap inherits several key advantages from Transformer: natively support for variable-length inputs, uniform all-to-all interactions without receptive-field limits, and one-pass construction of the full matrix. It is also surprisingly generalizable: one model can accurately cover all metal layers and even multiple process nodes. Our main contributions and results are as follows:

- Based on matrix decomposition and Gram representation framework, we propose AttentionCap, a Transformer specialized toward capacitance physics: order equivariance, a symmetric attention layer enforcing Laplacian structure, and a normalized Laplacian loss for stable training.
- We introduce learnable process-node embeddings for multi-node learning, significantly enhancing AttentionCap's accuracy and new-node transferability, which is highly valuable in practical EDA workflows.
- Through extensive experiments on synthetic training data and unseen real-design test sets, we show that AttentionCap achieves 1.25% and 6.26% errors for self- and coupling-capacitance on unseen designs (averaged on 7nm and 65nm) under the challenging multi-layer setting, largely outperforming CNN-Cap in accuracy and efficiency. Under multi-node training, these errors are further reduced to 0.67% and 3.99%, and the pretrained AttentionCap shows strong transferability to new nodes with only 5K samples and 4K finetuning steps.

2 Preliminaries

2.1 Interconnect Capacitance Extraction

For a system of n conductors, the relation between their electric potentials $\mathbf{u} \in \mathbb{R}^n$ and electric charges $\mathbf{q} \in \mathbb{R}^n$ is described by the *capacitance matrix* $\mathbf{C} \in \mathbb{R}^{n\times n}$ via a linear equation [19]

$$\mathbf{q} = \mathbf{C}\mathbf{u}. \tag{1}$$

The diagonal element C_{ii} is the *total capacitance* of conductor i, and C_{ij} for $i \neq j$ is the *coupling capacitance* between i and j. $\mathbf{C}$ satisfies important physical properties [13]: (1) sign property, $C_{ii} \geq 0$ and $C_{ij} \leq 0$ for $i \neq j$; (2) symmetry, $C_{ij} = C_{ji}$; (3) zero row-sum, $\sum_{j=1}^{n} C_{ij} = 0$. For ease of presentation, we use an equivalent form where couplings are defined as $|C_{ij}|$, so that closer conductors intuitively correspond to larger couplings. Hence, $\mathbf{C}$ is a signless *Laplacian matrix* [20] and is guaranteed to be positive semi-definite.

Field solvers or *pattern matching* is employed in interconnect capacitance extraction to solve $\mathbf{C}$. Field solvers simulate the electrostatic field with finite difference method (FDM), finite element method (FEM), boundary element method (BEM) [28] or floating random walk (FRW) method [14]. While being accurate, they are computationally expensive and cannot scale to full-chip extraction. Pattern matching is an approximate approach for full-chip extraction, which decomposes the 3D layout into 2D cross-sections and estimates capacitance with look-up tables [8].

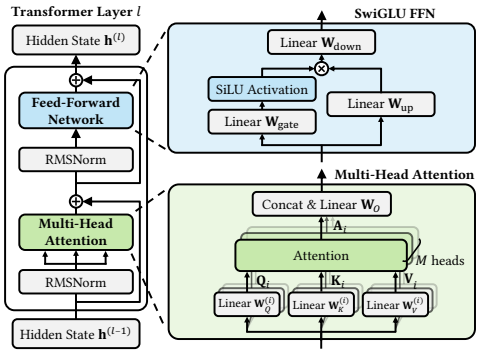


Figure 1: Modern Transformer architecture.

2.2 Deep-Learning-Based Capacitance Extraction for 2D Cross-Sections

Since the design and usage of pattern libraries are highly empirical and not transparent to the research community, there is a growing trend of deep-learning-based 2D cross-sectional capacitance modeling [29]. They can be divided into two categories: (1) **Conductor coordinates as input** were adopted by MLP-based models for process-variation-aware extraction [3], capacitance matrix extraction [18], and adaptive incremental learning [31]. However, they either supported only a fixed number of conductors [16, 18, 31] or predicted only one pairwise coupling [3]. (2) **Density-grid encoding as input** [2, 27] enabled flexibility for variable conductor counts. CNN-Cap [27] leveraged ResNet to process density grids with high capacitance accuracy. ResCap [23] combined conductor-length-based estimation and density-grid MLPs for standard cell designs. However, density-grid models must mark the targets in the grid, and predict only the targeted capacitance. For an n-conductor layout, the grid instantiations and model inference must repeat n^2 times. Moreover, existing models share a critical limitation: they are trained on fixed combinations of (typically) 3 metal layers, necessitating at least $\binom{m}{3}$ separate datasets and retrained models to cover an m-layer process node.

2.3 Attention Mechanism and Modern Transformer Architecture

We briefly review the Transformer architecture [24], along with LLM-era enhancements used in this work, e.g., SwiGLU [22] and RMSNorm [32]. For an input sequence of length n, an L-layer Transformer maps the input embedding $\mathbf{h}^{(0)} \in \mathbb{R}^{n\times d}$ to a final representation $\mathbf{h}^{(L)} \in \mathbb{R}^{n\times d}$, where d is the model dimension. As illustrated in Fig. 1, each layer transforms $\mathbf{h}^{(l-1)} \mapsto \mathbf{h}^{(l)}$ via a multi-head self-attention (MHA) module followed by a position-wise feed-forward network (FFN). Each attention head i uses weight matrices $\mathbf{W}_Q^{(i)}, \mathbf{W}_K^{(i)}, \mathbf{W}_V^{(i)} \in \mathbb{R}^{d\times d_k}$ and MHA concatenates and projects the outputs across M heads with weight $\mathbf{W}_O \in \mathbb{R}^{Md_k\times d}$:

$$\mathbf{O}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^\top}{\sqrt{d_k}}\right)\mathbf{V}_i, \quad \text{MHA}(\mathbf{h}) = [\mathbf{O}_1, \ldots, \mathbf{O}_M]\mathbf{W}_O, \tag{2}$$

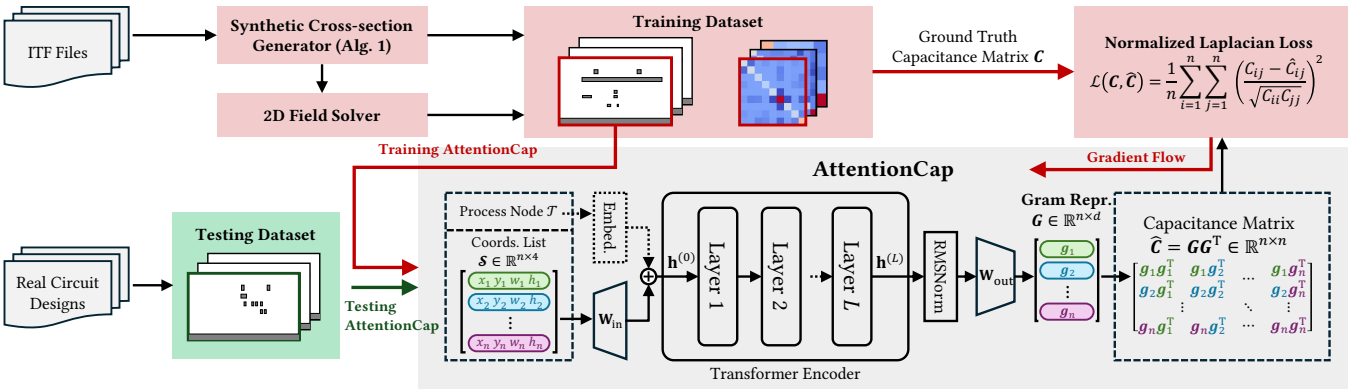


Figure 2: Overview of the AttentionCap framework for capacitance matrix learning.

where $\{\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i\} = \{\mathbf{h}\mathbf{W}_Q^{(i)}, \mathbf{h}\mathbf{W}_K^{(i)}, \mathbf{h}\mathbf{W}_V^{(i)}\}$. In modern architectures, the FFN commonly adopts the SwiGLU form [22]:

$$\mathrm{FFN}(\mathbf{x}) = \left[(\mathbf{x}\mathbf{W}_{\mathrm{up}}) \otimes \sigma(\mathbf{x}\mathbf{W}_{\mathrm{gate}})\right]\mathbf{W}_{\mathrm{down}}^{\top}, \tag{3}$$

where $\mathbf{W}_{\mathrm{up}}, \mathbf{W}_{\mathrm{gate}}, \mathbf{W}_{\mathrm{down}} \in \mathbb{R}^{d\times d_{\mathrm{ff}}}$ are weights, $\otimes$ is element-wise product, and σ is the SiLU activation [22]. Each MHA or FFN module is preceded by RMSNorm [32]:

$$\mathrm{RMSNorm}(\mathbf{x}) = \frac{\mathbf{x}}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2 + \epsilon}} \otimes \boldsymbol{\gamma}, \tag{4}$$

where $\boldsymbol{\gamma} \in \mathbb{R}^d$ is learnable and ϵ is for numerical stability.

3 AttentionCap: Capacitance Transformer

3.1 Problem Formulation

We formulate *capacitance matrix learning* as follows. A layout cross-section with n conductors is parameterized as sequence $\mathcal{S} = [(x_1, y_1, w_1, h_1), \ldots, (x_n, y_n, w_n, h_n)] \in \mathbb{R}^{n\times 4}$, where x_i, y_i, w_i, h_i are the center coordinates, width, and height of the i-th conductor[1]. For a process technology node $\mathcal{T}$, the dielectric stack is fixed (specified in an interconnect technology file (ITF)), so our target is a deterministic but highly non-linear mapping $\mathcal{F}_{\mathcal{T}}$:

$$\mathcal{F}_{\mathcal{T}} : \mathbb{R}^{n\times 4} \to \mathbb{R}^{n\times n}, \quad \mathbf{C} = \mathcal{F}_{\mathcal{T}}(\mathcal{S}). \tag{5}$$

Beyond this, we also explore a more general setting where multi-process-node $\mathcal{F}_{\mathcal{T}_1}, \mathcal{F}_{\mathcal{T}_2}, \ldots$ are learned jointly with a single model.

3.2 Gram Representation Learning

A common strategy for relation modeling is to learn latent embeddings and express the relations via their dot-product [6], which is also the basic idea of attention [24]. We follow this paradigm to decompose learning objective via dot-product, and design AttentionCap under a representation learning framework.

A key observation is that due to the symmetric positive semi-definite nature of capacitance matrix (see Section 2.1), $\mathbf{C}$ can be factorized by dot-product [12], as stated in Lemma 3.1.

[1]Following prior studies, we consider only rectangular Manhattan conductors. AttentionCap could potentially accommodate non-Manhattan geometries with different input embeddings; a systematic study is left for future work.

Lemma 3.1. *(Gram Representation) For any positive semi-definite matrix $\mathbf{C} \in \mathbb{R}^{n\times n}$, there exists $\mathbf{G} \in \mathbb{R}^{n\times n'}$ such that*

$$\mathbf{C} = \mathbf{G}\mathbf{G}^{\top}, \quad \textit{i.e.,}\ C_{ij} = \langle \boldsymbol{g}_i, \boldsymbol{g}_j \rangle, \forall i, j, \tag{6}$$

where $n' \le n$ is the rank of $\mathbf{C}$ [12]. We refer to $\mathbf{G}$ and its row vectors $\{\boldsymbol{g}_i \in \mathbb{R}^{1\times n'}\}$ as the Gram representation of $\mathbf{C}$.

With this decomposition, the target mapping becomes $\mathcal{G}_{\mathcal{T}}$:

$$\mathcal{G}_{\mathcal{T}} : \mathbb{R}^{n\times 4} \to \mathbb{R}^{n\times d}, \quad \mathcal{F}_{\mathcal{T}}(\mathcal{S}) = \mathcal{G}_{\mathcal{T}}(\mathcal{S})\mathcal{G}_{\mathcal{T}}(\mathcal{S})^{\top}. \tag{7}$$

This is a sequence-to-sequence task that transforms each conductor into a Gram vector or "capacitance embedding". We implement $\mathcal{G}_{\mathcal{T}}$ with a Transformer and customized input and output layers:

$$\mathbf{h}^{(0)} = \mathrm{in}(\mathcal{S}),\ \mathbf{h}^{(L)} = \mathrm{Transformer}(\mathbf{h}^{(0)}),\ \hat{\mathbf{G}} = \mathrm{out}(\mathbf{h}^{(L)}). \tag{8}$$

3.3 Model Architecture

Our methodology is outlined in Fig. 2. We propose several physics-aligned specializations based on standard Transformer: no positional encoding, no causal masking, additional input and output layers, and a normalized Laplacian loss.

Input Embedding. The $\mathrm{in}(\cdot)$ in Eq. (8) lifts the conductor coordinates $\mathcal{S} \in \mathbb{R}^{n\times 4}$ to initial embedding $\mathbf{h}^{(0)} \in \mathbb{R}^{n\times d}$. We implement it as a position-wise linear layer with weight matrix $\mathbf{W}_{\mathrm{in}} \in \mathbb{R}^{4\times d}$:

$$\mathbf{h}^{(0)} = \mathcal{S}\mathbf{W}_{\mathrm{in}}. \tag{9}$$

No Positional Encoding. Unlike language tasks, capacitance learning is agnostic to the order of input sequence. More formally, the target mapping $\mathcal{G}_{\mathcal{T}}$ is inherently permutation-equivariant:

$$\mathcal{G}_{\mathcal{T}}(\pi\mathcal{S}) = \pi\mathcal{G}_{\mathcal{T}}(\mathcal{S}), \quad \forall \text{ permutation } \pi. \tag{10}$$

Without positional encoding, the Transformer architecture natively satisfies this equivariance [24]. Therefore, we do not add any positional encoding in AttentionCap.

Transformer Encoder. The $\mathrm{Transformer}(\cdot)$ in Eq. (8) is the L-layer Transformer with modern enhancements, as described in Section 2.3. We adopt the encoder-only architecture, since causal masking is unnecessary in our task.

Symmetric-Attention Output Layer. The $\mathrm{out}(\cdot)$ in Eq. (8) is implemented as an RMSNorm layer followed by a linear layer with

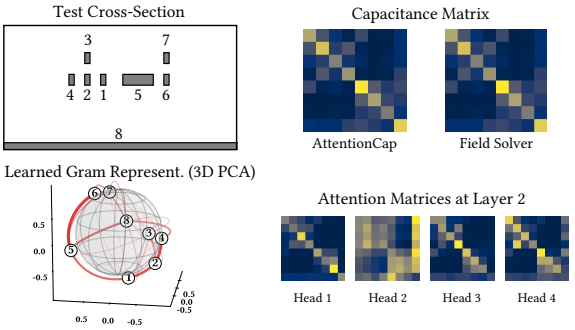


Figure 3: AttentionCap learns meaningful Gram representations for each conductor (visualized via 3D PCA on a unit sphere, line thickness reflecting their dot-products); its internal attention patterns match the target capacitance matrix.

weight $\mathbf{W}_{\text{out}} \in \mathbb{R}^{d\times d}$. The capacitance matrix prediction is then computed by dot-product:

$$\hat{\mathbf{G}} = \text{RMSNorm}(\mathbf{h}^{(L)})\mathbf{W}_{\text{out}}, \quad \hat{\mathbf{C}} = \frac{1}{\sqrt{d}}\hat{\mathbf{G}}\hat{\mathbf{G}}^{\top}. \tag{11}$$

The scaling factor $\frac{1}{\sqrt{d}}$ follows attention mechanism Eq. (2) to stabilize training. Note that Eq. (11) is equivalent to *symmetric attention* [9], so the output layer can be viewed as a symmetric attention module which outputs the attention matrix as the capacitance prediction. Finally, to enforce the Laplacian properties of $\hat{\mathbf{C}}$ (see Section 2.1), we set $\hat{C}_{ii} = \sum_{j\neq i} \hat{C}_{ij}, \forall i$.

Normalized Laplacian Loss. Since capacitance values can span across multiple scales, we propose a novel loss function for capacitance matrix learning. We apply the symmetric Laplacian normalization based on ground truth $\mathbf{C}$:

$$\mathbf{C}' = \mathbf{D}^{-\frac{1}{2}}\mathbf{C}\mathbf{D}^{-\frac{1}{2}}, \quad \hat{\mathbf{C}}' = \mathbf{D}^{-\frac{1}{2}}\hat{\mathbf{C}}\mathbf{D}^{-\frac{1}{2}}, \tag{12}$$

where $\mathbf{D}$ is the diagonal matrix with $D_{ii} = C_{ii}$. The normalized Laplacian loss is defined as:

$$\mathcal{L}(\mathbf{C}, \hat{\mathbf{C}}) = \frac{1}{n}\|\mathbf{C}' - \hat{\mathbf{C}}'\|_F^2 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\frac{C_{ij} - \hat{C}_{ij}}{\sqrt{C_{ii}C_{jj}}}\right)^2, \tag{13}$$

where $\|\cdot\|_F$ is the Frobenius norm. This loss prevents domination by large capacitance values, and is shown to be crucially effective in experiments (see Table 4).

With the Gram representation framework and our architectural enhancements, AttentionCap effectively learns meaningful "capacitance embeddings" with attention patterns that align closely with the target capacitance structure, as shown in Fig. 3.

3.4 Multi-Process-Node Learning

Building capacitance models for new process nodes is often a bottleneck in IC design practice, so a reusable multi-node model could be highly valuable. AttentionCap not only supports multi-node learning, but also benefits from it in two ways: (1) more abundant and diverse samples in the mixed dataset, and (2) strong transferability due to separation of process-node-agnostic features.

Process-Node Embedding. For process nodes $\mathcal{T}_1, \mathcal{T}_2, \ldots$, their capacitance mappings $\mathcal{F}_{\mathcal{T}_1}, \mathcal{F}_{\mathcal{T}_2}, \ldots$ are governed by the different dielectric stacks and wire features. We propose to use a learnable embedding to capture this node-specific information, and add it to the coordinate embedding in Eq. (9):

$$\mathbf{h}^{(0)} = \boldsymbol{\mathcal{S}}\mathbf{W}_{\text{in}} + \text{Embed}(\mathcal{T}_i). \tag{14}$$

This enables AttentionCap to differentiate samples from each node and implicitly inject dielectric-related information into the conductor representations. This extension is lightweight yet effective, as validated in experiments (see Table 3).

Transfer to New Process Nodes. Trained on multi-node data, AttentionCap learns both process-node embeddings and process-node-agnostic features, developing strong transferability to new, unseen nodes. For each new process node $\mathcal{T}_{\text{new}}$, we can add an embedding vector $\text{Embed}(\mathcal{T}_{\text{new}})$ and finetune the pretrained model. Experiments show that a multi-node-pretrained AttentionCap can transfer to $\mathcal{T}_{\text{new}}$ with a quick finetune on few training samples from $\mathcal{T}_{\text{new}}$ (see Fig. 5). This fast adaption is exceptionally valuable in practice, as preparing training data with field solver is the main bottleneck for developing capacitance models for a new process node.

3.5 Synthetic Training Data Generation

Layout data is often scarce and proprietary. More critically, real layouts contain deterministic features such as wire width and layer usage, which may cause overfitting and poor generalizability without careful data curation. Therefore, we propose a synthetic cross-section generator (Alg. 1) that produces diverse training samples, requiring only a process node description (typically an ITF).

Alg. 1 first randomly selects 3, 4, or 5 layers (line 1) and then places a random number of conductors. We use Poisson distribution $P(\cdot)$ to sample the conductor count n (line 2). $\mathcal{N}(\cdot,\cdot)$ and $U\{\cdot\}/(\cdot)$ denote Gaussian and discrete/continuous uniform distributions, respectively. To mimic realistic cross-sections, each conductor is sampled as "continuous" with 10% probability (line 9) or "discrete" with 90% probability (lines 11-12). For discrete width sampling, we apply the exponential-like distribution $E(w_{\min}, w_{\max})$ [27]. Finally, design rules are guaranteed via rejection sampling (lines 13-14).

In our implementation, N is set to 8, resulting in roughly 2-16 conductors per sample. The window width W is technology dependent and determined via field-solver simulation such that the coupling between two M1 conductors with spacing $W/2$ falls below 1% of the total capacitance, following common practice [3, 27]. For instance, W is 10.7μm for a 65nm node and 2.7μm for a 7nm node. We implement $E(w_{\min}, w_{\max})$ as $\mathbb{P}(w = iw_{\min}) \propto (0.75)^i, i = 1, 2, \ldots,$ truncated to $w_{\max}$, which is similar to [27].

4 Experimental Results

4.1 Experiment Setup

All experiments were conducted on Intel Xeon Platinum 8488C CPUs (3.8GHz) and a single NVIDIA RTX 5090 GPU. Synthetic data generation (Alg. 1) and 2D field solver were implemented in C++ with CPU multithreading. Model training and evaluation used PyTorch 2.9 and CUDA 13.0.

Datasets. We use two open-source process nodes (ASAP7 [7] and FreePDK15 [4]) and two industrial nodes that include conformal dielectrics (Real28 and Real65). For each node, we generate 50K synthetic cross-sections using Alg. 1 and solve capacitance

Algorithm 1: Synthetic Training Data Generation.

Input: Process technology $\mathcal{T}$, expected conductor count N, window width W.
Output: A list of randomly placed conductors $\mathcal{S}$.

Randomly select a `LayerSet` of $U\{3,4,5\}$ layers in $\mathcal{T}$;
Sample $n \sim P(N-2)+2$; ▷ ensure $n \geq 2$ and $\mathbb{E}(n) = N$
$\mathcal{S} \leftarrow$ [`substrate`]; ▷ init with an infinite substrate
repeat
  $(y, h, w_{\min}, s_{\min}) \leftarrow$ sample a layer from `LayerSet`;
  Sample $x \sim \mathcal{N}(0, (\frac{W}{6})^2)$; ▷ 3σ range $= W$
  $w_{\max} \leftarrow W - 2|x|$;
  if $U(0,1) < 0.1$ **then**
    Sample $w \sim U(w_{\min}, w_{\max})$;
  else
    Sample $w \sim E(w_{\min}, w_{\max})$;
    $x \leftarrow$ round x to grids $i(w_{\min}+s_{\min}), i=0,\pm1\ldots$;
  if `DesignRuleCheck`$(x, y, w, h, \mathcal{S})$ *passes* **then**
    Add (x, y, w, h) to $\mathcal{S}$;
until $\mathcal{S}$ *has* n *conductors*;

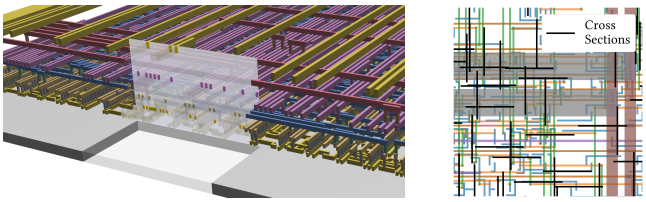


Figure 4: Example cross-sections from ASAP7 real-design test set, in local 3D view (left) and local top view (right).

matrices with 2D field solver, which takes about 16 hours on 24 threads. The synthetic datasets are split 9:1 into training/validation sets. For testing, we prepare three designs for ASAP7 and Real65, respectively, covering multiple circuit types. We uniformly sample 5K cross-sections from each design and retain the 10 center-most conductors in each sample, as shown in Fig. 4. FreePDK15 and Real28 include only synthetic data for training the multi-node model.

Models. We implement **AttentionCap** with $d = 256, d_k = 64, M = 4, d_{\text{ff}} = 512, L = 6$ (4.0M parameters). For large multi-node datasets, we also evaluate **AttentionCap-L** with $d = 384, d_k = 96, M = 4, d_{\text{ff}} = 768, L = 8$ (12M parameters). As a baseline, **AttentionCap w/o attention** (equivalent to a pure **MLP**) removes all attention modules and expands to $d = 512, d_{\text{ff}} = 1024$ (9.7M parameters). We include **CNN-Cap** [27] from its official implementation [1], which is a ResNet34-based model with 7.2M parameters.

Training. All models are trained under a unified setup: AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.999$ and weight decay 10^{-4}; learning rate scheduled from 1.5×10^{-4} to 10^{-5} with 1K warm-up steps and linear decay. We apply horizontal-flip augmentation for training data (simply $x_i \mapsto -x_i$) and the proposed normalized Laplacian loss. We consistently train for 300K steps and select the best validation-loss checkpoint. To maintain roughly 400 effective epochs for different datasets, the training batch size is adjusted according to: batch_size × steps = data_size × epochs. For instance, batch size is 128 for training on the 50K dataset, 512 for 200K. During testing, we apply a batch size of 128 across all experiments.

Evaluation Metrics. Following [27], we report four key metrics in practical capacitance extraction: Err_{tot}, $\text{Ratio}_{\text{tot}}$, Err_{cp}, and Ratio_{cp}. Denote $e_{ij}^{(k)} = \left|\hat{C}_{ij}^{(k)} - C_{ij}^{(k)}\right| / \left|C_{ij}^{(k)}\right|$ for the k-th sample. Err_{tot} and $\text{Ratio}_{\text{tot}}$ are the mean and high-error rate (>5%) of total-capacitance errors $e_{ii}^{(k)}, \forall k, i$; Err_{cp} and Ratio_{cp} are the mean and high-error rate (>10%) of coupling-capacitance errors $e_{ij}^{(k)}, \forall k, i \neq j$. Tiny couplings (less than 1% of its total capacitance) are excluded when calculating these metrics.

4.2 Benchmark on CNN-Cap Public Datasets

We first benchmark AttentionCap on CNN-Cap datasets [27] available at [1], which contain fixed 3-layer combinations at 55nm and 15nm nodes, with 50K cross-sections per setting. Given the density-grid data, we reconstruct each conductor's $\tilde{x}_i, \tilde{w}_i$ based on grid index and use $(\tilde{x}_i, \tilde{w}_i, \text{layer\_id})$ as AttentionCap inputs. The datasets provide single-master capacitances per cross-section, so we train and test AttentionCap on a single row of its output matrix. Validation-set errors are shown in Table 2. AttentionCap consistently outperforms CNN-Cap with remarkable accuracy.

Table 2: Benchmark on CNN-Cap Pattern-C datasets [1, 27].

Model	Node	Layers	Err_{tot}	$\text{Ratio}_{\text{tot}}$	Err_{cp}	Ratio_{cp}
CNN-Cap*	55 nm	(2,3,6)	0.10%	**0**	1.20%	0.30%
AttentionCap			**0.03%**	**0**	**0.35%**	**0.02%**
CNN-Cap*	55 nm	(2,4,6)	0.20%	**0**	1.20%	0.10%
AttentionCap			**0.04%**	**0**	**0.32%**	**0.03%**
CNN-Cap*	15 nm	(1,3,5)	0.20%	**0**	1.80%	0.40%
AttentionCap			**0.03%**	**0**	**0.29%**	**0**
CNN-Cap*	15 nm	(1,3,8)	0.20%	**0**	1.50%	**0**
AttentionCap			**0.02%**	**0**	**0.30%**	**0**

* Results are taken from [27]. CNN-Cap uses separate models for total and coupling capacitance; they are merged into one row here.

4.3 Main Results: Training on Synthetic Data and Testing on Real Designs

Our principle is to train models on synthetic data and test them on unseen real-design cross-sections, under **multi-metal-layer** and **multi-process-node** settings.

Single-Node Learning. For ASAP7 and Real65, we have 50K synthetic samples and 15K test samples each. To train CNN-Cap [27], we discretize each metal layer into a 512-resolution density grid and expand its input channels to 9 (ASAP7) and 8 (Real65) to handle all metal layers. CNN-Cap_{tot} and CNN-Cap_{cp} are the specialized models for total and coupling capacitance [27]. Main results are summarized in Table 3. AttentionCap significantly outperforms CNN-Cap and MLP baselines on both nodes, achieving 0.98%/5.38% total/coupling error on 65nm and 1.52%/7.13% total/coupling error on 7nm, in unseen real-design tests. CNN-Cap is relatively accurate on total capacitance but suffers from large coupling error (>20% average error; ~50% high-error outliers), showing its limited ability to capture complex multi-conductor interactions. MLP (i.e., AttentionCap w/o attention) fails to provide usable predictions,

Table 3: Main Results: training on synthetic data and testing on unseen real designs.

Model	Param.	Node	Valid. Error (synthetic data)				Test Error (real-design)				#Train Samples[a]	Train Time[b]	#Test Samples[a]	Test Time[b]	Test FLOPs
			Err_{tot}	$Ratio_{tot}$	Err_{cp}	$Ratio_{cp}$	Err_{tot}	$Ratio_{tot}$	Err_{cp}	$Ratio_{cp}$					
CNN-Cap_{tot}	7.2M	65 nm	2.83%	15.0%	-	-	4.06%	30.0%	-	-	315 K	2.18 hr	54 K	6.91 s	9.83 B
CNN-Cap_{cp}	7.2M		-	-	9.74%	34.3%	-	-	20.4%	54.4%	1265 K	9.69 hr	184 K	27.6 s	33.1 B
MLP	9.7M		14.5%	76.2%	53.7%	81.9%	23.6%	87.8%	84.0%	87.0%	45 K	0.43 hr	15 K	0.15 s	0.67 B
AttentionCap	4.0M		**0.85%**	**1.65%**	**3.24%**	**6.95%**	**0.98%**	**1.55%**	**5.38%**	**13.4%**	45 K	0.64 hr	15 K	0.28 s	0.28 B
CNN-Cap_{tot}	7.2M	7 nm	0.69%	**0.09%**	-	-	2.11%	9.94%	-	-	315 K	2.18 hr	73 K	11.1 s	13.2 B
CNN-Cap_{cp}	7.2M		-	-	3.89%	6.95%	-	-	25.1%	48.7%	1416 K	9.61 hr	322 K	54.0 s	58.2 B
MLP	9.7M		16.4%	80.9%	62.5%	85.1%	32.4%	90.8%	120%	86.8%	45 K	0.44 hr	15 K	0.15 s	0.86 B
AttentionCap	4.0M		**0.47%**	0.39%	**2.19%**	**3.49%**	**1.52%**	**6.75%**	**7.13%**	**18.9%**	45 K	0.64 hr	15 K	0.24 s	0.35 B
MLP	9.7M	**Mix**	15.2%	78.4%	57.9%	83.7%	30.0%	90.0%	110%	86.7%	180 K	0.52 hr	30 K	0.10 s	1.53 B
AttentionCap	4.0M		0.54%	0.46%	2.55%	4.61%	0.88%	1.07%	5.25%	13.5%	180 K	0.68 hr	30 K	0.47 s	0.63 B
AttentionCap-L	12M		**0.44%**	**0.30%**	**2.08%**	**2.97%**	**0.67%**	**0.28%**	**3.99%**	**9.75%**	180 K	1.02 hr	30 K	0.66 s	1.89 B

[a] All models use the same train/valid/test data. The different sample sizes are due to architectural difference: AttentionCap treats each matrix as a sample, while CNN-Cap treats each capacitance element as an individual sample. We increase CNN-Cap batch size accordingly to align training steps/epochs.
[b] CNN-Cap runtimes also include the cost of constructing density-grid inputs, which is negligible for training but accounts for ~74% of testing time.

highlighting the fundamental role of attention mechanism. Test errors are consistently higher than validation errors, indicating the challenging **out-of-distribution** (OOD) features in real designs. Moreover, models show higher error than on 3-layer datasets (Table 2), which confirms that our multi-layer setting presents much higher learning complexity.

Moreover, AttentionCap is far more efficient than CNN-Cap: 18.5× faster in training and 192× faster (181× fewer FLOPs) in testing, owing to smaller model size and avoidance of density-grid processing. For comparison, field solver takes 6 hours on 24 threads to solve all test samples.

Multi-Node Learning. We mix 7nm, 15nm, 28nm, 65nm synthetic samples to train multi-node models, and test them on 7nm and 65nm unseen real designs. As shown in Table 3, AttentionCap achieves even higher 7nm/65nm test accuracy than models trained only on 7nm/65nm, which implies that AttentionCap can accurately distinguish samples via process-node embeddings, and has been enhanced by samples from other nodes via learning node-agnostic features. The larger AttentionCap-L further improves accuracy given the abundant data, attaining the best **0.67%/3.99% test error** for total/coupling capacitance, which is 4.6×/5.7× lower than CNN-Cap's average error.

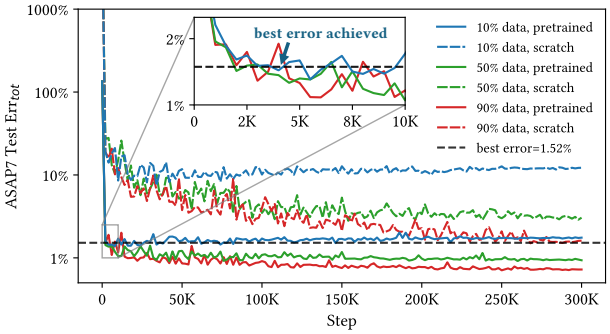


Figure 5: Strong transferability of AttentionCap: with only 5K samples and 4K finetuning steps for new process node (7nm), the pretrained model (on 15nm+28nm+65nm) outperforms the best model trained from scratch.

The learned node-agnostic features make AttentionCap highly transferable to new nodes. We pretrain a multi-node AttentionCap on 15nm+28nm+65nm data, and finetune it with 10%/50%/90% train splits from 7nm data. As shown in Fig. 5, with only a 10% train split (5K samples) and 4K finetuning steps, the pretrained model outperforms the best 7nm model trained from scratch (best error achieved at 240K steps with the 90% train split). This data-efficient adaption is exceptionally valuable in practice, as preparing training data is the main bottleneck. We attribute this multi-node capability to the exceptional representation power of Transformer architecture.

4.4 Ablation Study

We conduct an ablation study with the best model AttentionCap-L on the mixture dataset (7nm+15nm+28nm+65nm). We evaluate training with mean squared error (MSE) and a GELU-based feed-forward network (FFN) variant (using $d_{ff} = 1152$ to match 12M parameters). Test-set errors are reported in Table 4, showing that the proposed normalized Laplacian loss is crucial for training, and the SwiGLU [22] improves capacitance accuracy.

Table 4: Ablation study on AttentionCap-L (12M).

Node	FFN	Loss	Err_{tot}	$Ratio_{tot}$	Err_{cp}	$Ratio_{cp}$
Mix	SwiGLU	MSE	4.91%	34.8%	25.6%	49.7%
	GELU	Laplacian	1.00%	0.89%	5.75%	15.1%
	SwiGLU	Laplacian	**0.67%**	**0.28%**	**3.99%**	**9.75%**

5 Conclusion

We introduce AttentionCap, a Transformer for capacitance matrix learning with Gram representation learning, symmetric-attention output, and normalized Laplacian loss. AttentionCap delivers high accuracy, broad generalization across metal layers and process nodes. Trained exclusively on synthetic data, it achieves 0.67%/3.99% self/coupling error on unseen real designs, with 4.6×/5.7× lower error and 192× higher efficiency than CNN-Cap [27]. It also adapts to a new node with only 5K samples and a quick finetune. These results demonstrate AttentionCap's strong potential as a unified and transferable model for practical capacitance extraction.

References

[1] 2023. CNN-Cap Official Implmentation. https://github.com/ydc123/CNNCap.

[2] Mohamed Saleh Abouelyazid, Sherif Hammouda, and Yehea Ismail. 2022. Accuracy-Based Hybrid Parasitic Capacitance Extraction Using Rule-Based, Neural-Networks, and Field-Solver Methods. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 41, 12 (2022), 5681–5694.

[3] Mohamed Saleh Abouelyazid, Sherif Hammouda, and Yehea Ismail. 2022. Fast and accurate machine learning compact models for interconnect parasitic capacitances considering systematic process variations. *IEEE Access* 10 (2022), 7533–7553.

[4] Kirti Bhanushali and W Rhett Davis. 2015. FreePDK15: An open-source predictive process design kit for 15nm FinFET technology. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*. 165–170.

[5] James Hsueh-Chung Chen, Theodorus E Standaert, Emre Alptekin, Terry A Spooner, and Vamsi Paruchuri. 2014. Interconnect performance and scaling strategy at 7 nm node. In *IEEE International Interconnect Technology Conference*. 93–96.

[6] Giordano Cicchetti, Eleonora Grassucci, Luigi Sigillo, and Danilo Comminiello. 2025. Gramian Multimodal Representation Learning and Alignment. In *International Conference on Learning Representations (ICLR)*.

[7] Lawrence T Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm finFET predictive process design kit. *Microelectronics Journal* 53 (2016), 105–115.

[8] Jason Cong, Lei He, Andrew B. Kahng, David Noice, Nagesh Shirali, and Steve H.-C. Yen. 1997. Analysis and justification of a simple, practical 2 1/2-D capacitance extraction methodology. In *Design Automation Conference (DAC)*. 627–632.

[9] Martin Courtois, Malte Ostendorff, Leonhard Hennig, and Georg Rehm. 2024. Symmetric Dot-Product Attention for Efficient Training of BERT Language Models. In *Findings of the Association for Computational Linguistics (ACL)*. 8002–8011.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 4171–4186.

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations (ICLR)*.

[12] Roger A. Horn and Charles R. Johnson. 2013. Positive Definite and Semidefinite Matrices. In *Matrix Analysis, Second Edition*. Cambridge University Press, Chapter 7.

[13] Jiechen Huang, Shuailong Liu, and Wenjian Yu. 2025. A Parallel Floating Random Walk Solver for Reproducible and Reliable Capacitance Extraction. In *2025 Design, Automation & Test in Europe Conference (DATE)*. IEEE, 1–7.

[14] Jiechen Huang and Wenjian Yu. 2024. Enhancing 3-D Random Walk Capacitance Solver with Analytic Surface Green's Functions of Transition Cubes. In *Proc. DAC*.

[15] Doyun Kim, Jaemin Park, Youngmin Oh, and Bosun Hwang. 2024. TraceFormer: s-parameter prediction framework for PCB traces based on graph transformer. In *ACM/IEEE Design Automation Conference (DAC)*.

[16] Zhixing Li and Weiping Shi. 2020. Layout capacitance extraction using automatic pre-characterization and machine learning. In *International Symposium on Quality Electronic Design (ISQED)*. 457–464.

[17] Lihao Liu, Fan Yang, Li Shang, and Xuan Zeng. 2023. GNN-Cap: Chip-Scale interconnect capacitance extraction using graph neural network. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* 43, 4 (2023), 1206–1217.

[18] Yaoyao Ma, Xiaoyu Xu, Shuai Yan, Yaxing Zhou, Tianyu Zheng, Zhuoxiang Ren, and Lan Chen. 2023. Extraction of interconnect parasitic capacitance matrix based on deep neural network. *Electronics* 12, 6 (2023), 1440.

[19] J.C. Maxwell. 1873. *A Treatise on Electricity and Magnetism*.

[20] Russell Merris. 1994. Laplacian matrices of graphs: a survey. *Linear algebra and its applications* 197 (1994), 143–176.

[21] OpenAI. 2023. GPT-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[22] Noam Shazeer. 2020. GLU variants improve transformer. *arXiv preprint arXiv:2002.05202* (2020).

[23] Jiun-Cheng Tsai, Hsuan-Ming Huang, Wei-Min Hsu, Pei-Ting Lee, Jen-Hang Yang, Heng-Liang Huang, Yen-Ju Su, and Charles H-P Wen. 2025. ResCap: Fast-yet-Accurate Capacitance Extraction for Standard Cell Design by Physics-Guided Machine Learning. In *Asia and South Pacific Design Automation Conference (ASPDAC)*. 1243–1250.

[24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[25] Liangjian Wen, Yi Zhu, Lei Ye, Guojin Chen, Bei Yu, Jianzhuang Liu, and Chunjing Xu. 2022. Layoutransformer: Generating layout patterns with transformer via sequential pattern modeling. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*.

[26] Dingcheng Yang, Haoyuan Li, Wenjian Yu, Yuanbo Guo, and Wenjie Liang. 2023. CNN-Cap: Effective convolutional neural network-based capacitance models for interconnect capacitance extraction. *ACM Transactions on Design Automation of Electronic Systems* 28, 4 (2023), 1–22.

[27] Dingcheng Yang, Wenjian Yu, Yuanbo Guo, and Wenjie Liang. 2021. CNN-Cap: Effective convolutional neural network based capacitance models for full-chip parasitic extraction. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*.

[28] Wenjian Yu, Chao Hu, and Wangyang Zhang. 2009. Variational capacitance extraction of on-chip interconnects based on continuous surface model. In *Proceedings of the 46th Annual Design Automation Conference*. 758–763.

[29] Wenjian Yu, Shan Shen, Dingcheng Yang, Haoyuan Li, Jiechen Huang, and Chunyan Pei. 2025. Deep Learning Inspired Capacitance Extraction Techniques. In *Asia and South Pacific Design Automation Conference (ASPDAC)*. 106–112.

[30] W. Yu, M. Song, and M. Yang. 2021. Advancements and challenges on parasitic extraction for advanced process technologies. In *Asia and South Pacific Design Automation Conference (ASPDAC)*. 841–846.

[31] Ziwei Yu, Shuai Yan, Yaxing Zhou, Xiaoyu Xu, and Zhuoxiang Ren. 2025. AIL DNN: Modeling of IC Interconnect Parasitic Capacitances Based on Adaptive Incremental Learning. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* (2025).

[32] Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in neural information processing systems* 32 (2019).