Deep Learning for Schwarz-Christoffel Maps

Individual Research | Computational Complex Analysis

Felipe O. Cardozo | focardo@emory.edu

Natalia Garcia | ngarci2@emory.edu

Leo Mokriski | lmokris@emory.edu

Department of Mathematics, Emory University

December 16th, 2025

Contents

1 Abstract **2**

2 Introduction **2**

2.1 The Schwarz-Christoffel Problem 2

2.2 Related Work 3

3 Datasets **4**

4 Methodology **5**

4.1 Pedagogical Constraint: Building from Scratch 5

4.2 Input Configuration: With and Without c 5

4.3 Model Architectures 6

4.4 Activation Functions 7

4.5 Loss Function 8

4.6 Weight Initialization 9

4.7 Training Configuration 9

5 Results **9**

5.1 Quantitative Performance Comparison 9

5.2 Effect of the Scaling Constant c 11

5.3 Activation Function Comparison 11

5.4 Impact of Weight Initialization 11

6 Limitations **12**

7 Conclusion and Future Work **13**

1 Abstract

Traditional numerical methods for computing Schwarz-Christoffel (SC) mappings from the unit disk to a prescribed target polygon rely on iterative, computationally expensive numerical solvers to determine the unknown prevertices and the complex scaling constant c. This paper explores the viability of replacing these iterative numerical solvers with forward-pass neural networks. By framing the parameter estimation as a vector regression problem in $\mathbb{R}^n$, we train entirely from-scratch architectures—built purely in NumPy without relying on automatic differentiation frameworks—to map polygon vertices directly to their corresponding prevertices and the constant c. We track the architectural progression from a baseline Linear Regression model to a Shallow Multi-Layer Perceptron (MLP) and finally to a Deep three-layer MLP, analyzing the effects of structural complexity, advanced activation functions (LeakyReLU, Tanh, ELU), and weight initialization strategies. Across all three datasets, the best-performing model (Deep MLP with Xavier initialization and ELU activation) achieves a test MSE of 0.0038 on fixed quadrilaterals and 0.0194 on variable N=4–12 polygons when the scaling constant c is provided as an auxiliary input. The findings demonstrate that while deep learning introduces rapid forward-pass inference and measurable generalization improvements, significant limitations remain—particularly regarding the absence of geometric permutation invariance and exclusive reliance on parameter-space Mean Squared Error rather than geometric validation of the resulting conformal map.

2 Introduction

2.1 The Schwarz-Christoffel Problem

The Schwarz-Christoffel (SC) mapping is a cornerstone of computational complex analysis, providing a conformal mapping[1] from the unit disk $\mathbb{D}$ onto the interior of an arbitrary simple polygon $\mathbb{P}$. Let the vertices of $\mathbb{P}$ be $v_1, v_2, \ldots, v_N$ in counterclockwise order, with interior angles $\alpha_1\pi, \alpha_2\pi, \ldots, \alpha_N\pi$. The mapping function $f\colon \mathbb{D} \to \mathbb{P}$ takes the integral form

$$f(z) \;=\; c\int_0^z \prod_{k=1}^{N} (\zeta - w_k)^{\alpha_k - 1}\, d\zeta \;+\; C, \tag{1}$$

where w_k are the unknown *prevertices*[2] on the unit circle corresponding to the vertices v_k, c is a complex scaling constant that controls rotation and dilation, and C is a translation constant.

[1] **Conformal mapping:** a function between domains in the complex plane that preserves local angles and the shapes of infinitesimally small figures. Conformal maps are central to solving boundary-value problems in fluid dynamics, electrostatics, and heat conduction [1].

[2] **Prevertices:** the specific boundary points on the unit circle $\partial\mathbb{D}$ that map precisely to the target polygon's vertices under the SC transformation. Their angular positions encode the polygon's geometry.

For essentially all target polygons with $N \geq 4$, the prevertices w_k and the constant c cannot be determined analytically. They must be resolved collectively so that the edge lengths and interior angles of the resulting polygon match the target specification. Historically, achieving this parameter estimation has required highly nonlinear, multi-dimensional root-finding algorithms—an approach that is iterative, sensitive to initial guesses, and computationally intensive [1]. The primary research question explored in this paper is whether a neural network can bypass this iterative bottleneck, implicitly learning the relationship between polygon geometry and SC parameters to produce predictions in a single forward pass.

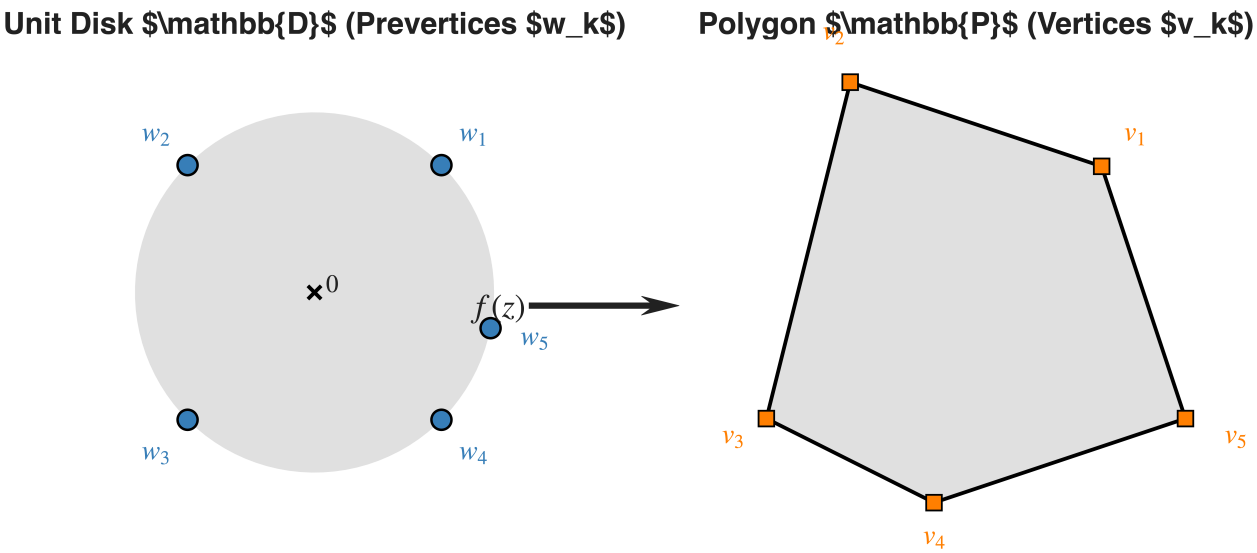

Figure 1: **The Schwarz-Christoffel transformation problem.** Left: the unit disk $\mathbb{D}$ with prevertices w_k marked on the boundary. Right: the target polygon $\mathbb{P}$ with vertices v_k. The conformal map $f(z)$ sends each prevertex to its corresponding polygon vertex. The model's objective is to predict the unknown prevertex positions and the complex constant c directly from the input polygon vertices.

2.2 Related Work

The standard computational tool for SC mapping is the MATLAB SC Toolbox developed by Driscoll and Trefethen [1], which employs a combination of compound Gauss-Jacobi quadrature and nonlinear least-squares optimization to solve the SC parameter problem. While highly accurate, the toolbox requires iterative convergence and scales poorly to polygons with large vertex counts or extreme aspect ratios.

More broadly, the use of neural networks as *learned surrogates*[3] for expensive numerical solvers has gained significant traction across scientific computing. Raissi et al. [4] introduced Physics-Informed Neural Networks (PINNs), demonstrating that neural architectures can encode

[3]**Learned surrogate:** a data-driven model trained to approximate the input–output behavior of an expensive numerical solver, enabling rapid inference at the cost of some approximation error.

partial differential equation constraints directly into the loss function. The SC parameter problem shares a conceptual similarity: both involve replacing an iterative numerical procedure with a single forward-pass prediction.

However, applying standard Multi-Layer Perceptrons (MLPs) to this problem introduces specific challenges absent in many surrogate-modeling tasks. First, the polygon vertex representation is not unique: cyclic permutations and rotations of the same polygon produce distinct input vectors, yet the underlying SC parameters are equivalent up to known symmetries. Standard feed-forward networks lack *permutation invariance*[4] and *rotation equivariance*, making them sensitive to the ordering and orientation of vertex coordinates. Second, polygons with varying numbers of vertices N must be represented in a fixed-dimensional input space, typically through zero-padding—a strategy that introduces structural noise. These challenges motivate both the architectural choices investigated in this paper and the future-work directions outlined in the conclusion.

3 Datasets

This study employs a data-driven approach requiring comprehensive datasets that span varied geometric complexity. Three distinct datasets, each containing 10,000 synthetically generated and numerically validated SC mappings, were utilized:

1. **Quadrilaterals (N=4).** A fixed-dimension set representing straightforward target geometries. Each sample consists of 8 input values (4 complex vertices decomposed into real and imaginary parts) and 8 scalar outputs (4 prevertex coordinates, similarly decomposed). Input and output dimensionality: $d_{\text{in}} = d_{\text{out}} = 8$.

2. **Variable vertices (N=4–8).** Contains polygons with a varying number of vertices between 4 and 8. To maintain a fixed input dimensionality compatible with standard feed-forward architectures, samples with $N < 8$ were zero-padded to $d_{\text{in}} = d_{\text{out}} = 16$.

3. **Variable vertices (N=4–12).** The most challenging set, covering polygons with up to 12 vertices. Zero-padding extends the dimensionality to $d_{\text{in}} = d_{\text{out}} = 24$.

To handle variable N within fixed-architecture neural networks, inputs and outputs associated with fewer than the maximum N vertices were zero-padded to the maximum dimensionality. A binary mask column was appended to each sample to signal which entries correspond to genuine vertex data and which are padding artifacts; its integration into the training loop is discussed in Section 6.

[4] **Permutation invariance:** the property that a model's output remains unchanged under reordering of its input elements. Graph Neural Networks [5] are specifically designed to satisfy this property.

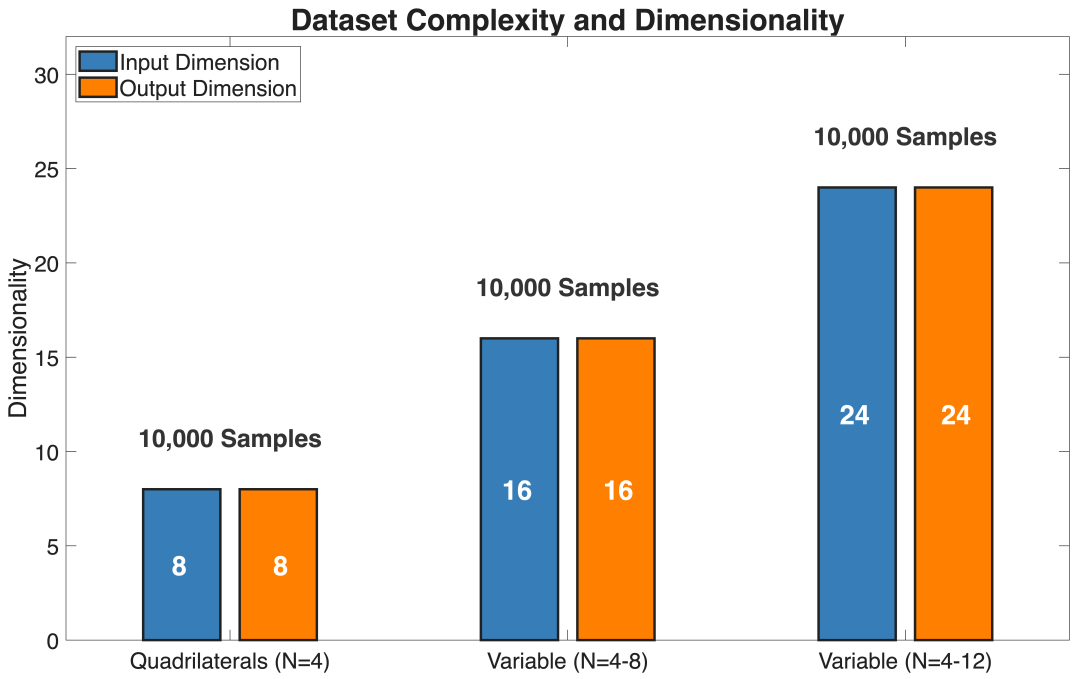

Figure 2: **Dataset complexity and dimensionality.** Grouped bar chart comparing input and output dimensions across the three dataset configurations. All datasets contain 10,000 samples. Dimensionality reflects the real and imaginary components of zero-padded vertex and prevertex coordinates.

4 Methodology

4.1 Pedagogical Constraint: Building from Scratch

To develop a thorough understanding of gradient flow, backpropagation dynamics, and the linear-algebraic foundations of neural network training, all models in this study were implemented exclusively in Python using NumPy. No automatic differentiation[5] libraries were used. Every forward-pass matrix multiplication, loss-function derivative, and backpropagated parameter update was derived analytically and coded explicitly. This constraint ensures that every component of the training pipeline—from weight initialization to gradient accumulation—is fully transparent and pedagogically documented.

4.2 Input Configuration: With and Without c

A central experimental axis of this study is the effect of including the complex scaling constant c as an auxiliary input feature. In the *without-c* configuration, the model receives

[5]**Automatic differentiation (autograd):** a technique that dynamically records computational operations to compute exact derivatives, as implemented in frameworks such as PyTorch and TensorFlow. By abstracting away manual chain-rule calculus, autograd significantly reduces implementation effort but also obscures the mathematical structure of gradient computation.

only the polygon vertex coordinates as input. In the *with-c* configuration, the real and imaginary parts of c ($c_{\text{re}}, c_{\text{im}}$) are appended to the input vector, increasing dimensionality by 2. Since c encodes global scaling and rotation information that is otherwise latent, providing it as an input is expected to reduce the effective difficulty of the regression task. All models were evaluated under both configurations to quantify this effect.

4.3 Model Architectures

A progressive sequence of architectures was implemented to test hypotheses about the relationship between network capacity and predictive accuracy on the SC parameter problem.

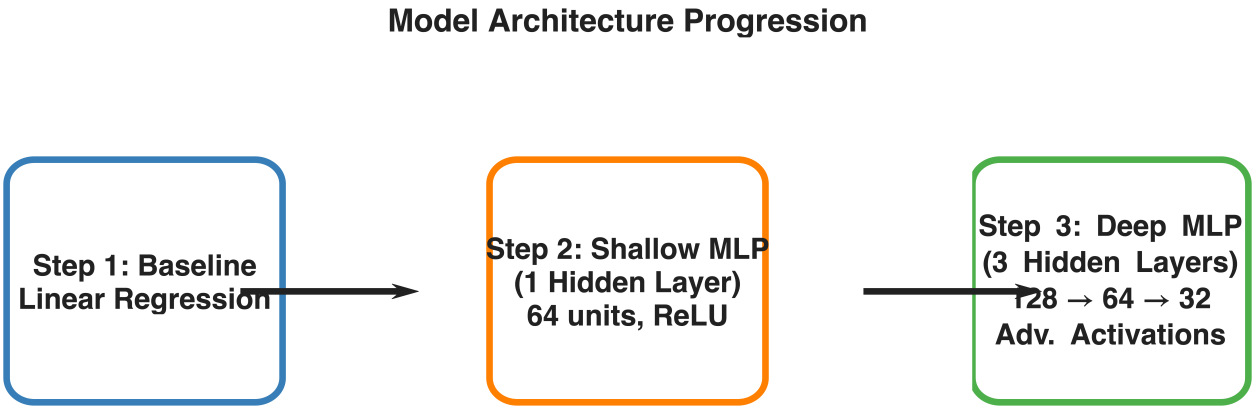

Figure 3: **Model architecture progression.** The study proceeds from a linear baseline (Step 1) through a single-hidden-layer MLP with 64 units (Step 2) to a deep three-layer architecture with 128 → 64 → 32 hidden units (Step 3). Each step introduces additional nonlinear capacity.

Step 1: Baseline Linear Regression. A multivariable linear regression model $\hat{y} = XW + b$ was implemented using both scikit-learn's closed-form solution and manual batch gradient descent against Mean Squared Error (MSE). This model serves as a lower bound on achievable accuracy, establishing the degree of nonlinearity present in the SC parameter mapping. As expected, the linear model exhibited substantial underfitting given the inherently nonlinear relationship between polygon vertices and prevertex positions.

Step 2: Shallow MLP (1 Hidden Layer, ReLU). Introducing a single hidden layer with 64 units and a Rectified Linear Unit (ReLU) activation function provided the first nonlinear baseline. This model achieved significant reductions in test MSE compared to the linear baseline, but generalized poorly to the variable-N datasets where zero-padded entries increased input noise.

Step 3: Deep MLP (3 Hidden Layers). To address the capacity limitations of the shallow model, a deep architecture with three cascading hidden layers (128 → 64 → 32 units) was

implemented. This architecture served as the primary experimental platform for comparing advanced activation functions (LeakyReLU, Tanh, ELU) and investigating the impact of weight initialization strategies. The final, most mature implementation is contained in the notebook `Deep_MLP_Xavier.ipynb`, which refactors all prior experimental code into clean, reusable functions—`initialize_deep_mlp`, `forward_pass`, `backward_pass`, and `train_deep_mlp`—and serves as the definitive experimental framework for producing the results reported in this paper.

4.4 Activation Functions

For the deep MLP, three nonlinear activation functions were implemented manually, including both forward and backward (derivative) computations required for backpropagation:

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}, \qquad \frac{\partial}{\partial x}\text{LeakyReLU}(x) = \begin{cases} 1 & \text{if } x > 0 \\ \alpha & \text{if } x \leq 0 \end{cases} \tag{2}$$

with $\alpha = 0.01$. LeakyReLU avoids the "dead neuron" problem of standard ReLU by maintaining a small gradient for negative inputs, but its piecewise-linear structure limits representational capacity for smooth functions.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad \frac{\partial}{\partial x}\tanh(x) = 1 - \tanh^2(x) \tag{3}$$

Tanh provides smooth, bounded outputs in $(-1, 1)$, with zero-centered activations that can accelerate convergence. However, it is susceptible to gradient saturation[6] for large-magnitude inputs, which can slow or stall training in deep networks.

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}, \qquad \frac{\partial}{\partial x}\text{ELU}(x) = \begin{cases} 1 & \text{if } x > 0 \\ \text{ELU}(x) + \alpha & \text{if } x \leq 0 \end{cases} \tag{4}$$

with $\alpha = 1.0$. ELU combines the benefits of ReLU-family functions (unbounded positive outputs, avoidance of dead neurons) with a smooth, nonzero negative region that pushes mean activations closer to zero and improves gradient flow.

[6] **Gradient saturation:** occurs when the activation function's derivative approaches zero for large-magnitude inputs, effectively halting gradient flow through the network during backpropagation.

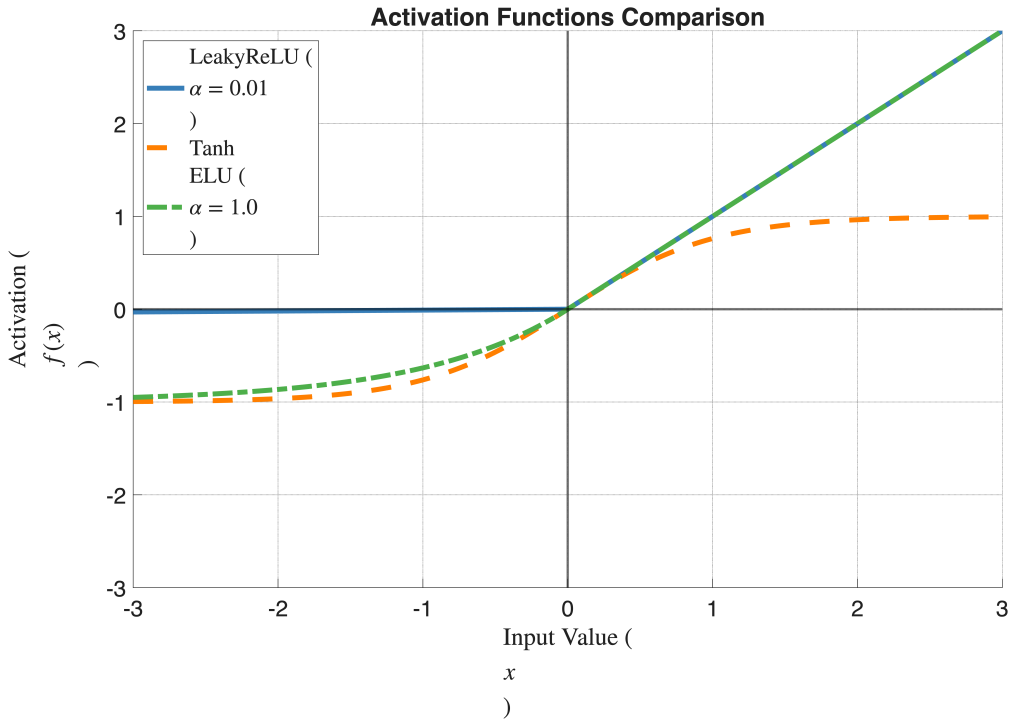

Figure 4: **Activation function comparison.** Visualization of the three activation functions implemented for the deep MLP: LeakyReLU (α=0.01), Tanh, and ELU (α=1.0). The smooth negative region of ELU and the zero-centered output of Tanh contrast with the piecewise-linear profile of LeakyReLU.

4.5 Loss Function

All models were trained by minimizing an L2-regularized Mean Squared Error objective. For a dataset of n samples with predictions $\hat{y}_i$ and targets y_i, the total loss is defined as

$$\mathcal{L}(\hat{y}, y) \;=\; \frac{1}{n}\sum_{i=1}^{n} \|\hat{y}_i - y_i\|^2 \;+\; \varepsilon \sum_{j} w_j^2, \tag{5}$$

where the first term is the standard MSE measuring prediction accuracy in parameter space, and the second term is an L2 regularization[7] penalty with coefficient $\varepsilon = 1 \times 10^{-4}$ that constrains weight magnitudes to prevent overfitting. For the baseline linear regression, the loss reduces to unregularized MSE ($\varepsilon = 0$) when using scikit-learn's closed-form solution.

[7] **L2 regularization (weight decay):** a penalty term proportional to the squared magnitude of all trainable weights, discouraging large parameter values and reducing overfitting by constraining the model's effective capacity.

4.6 Weight Initialization

Because the SC parameter prediction requires high precision across deep network layers, naive random initialization—scaling weights by a small constant factor such as $10^{-2} \times \mathcal{N}(0, 1)$—produced unstable training dynamics characterized by high-variance gradient updates and slow convergence. To address this, Xavier (Glorot) initialization [2] was applied to all deep MLP experiments. Xavier initialization draws weights from

$$W \sim \mathcal{N}\left(0,\ \frac{2}{n_{\text{in}} + n_{\text{out}}}\right), \tag{6}$$

where n_{in} and n_{out} denote the number of input and output units for each layer, respectively. This scaling maintains the variance of activations and gradients approximately constant across layers, preventing both vanishing and exploding gradients during deep forward and backward passes [2].

4.7 Training Configuration

All experiments shared the following hyperparameter settings:

Hyperparameter	**Value**
Learning rate (η)	0.01
Batch size	Full training set (batch gradient descent)
Training steps	5,000
L2 regularization (ε)	1×10^{-4}
Train/test split	80% / 20%
Evaluation metric	MSE and RMSE on held-out test set

Table 1: **Training configuration.** Hyperparameters held constant across all experiments. Evaluation was performed using scikit-learn's `mean_squared_error` on the 20% held-out test partition.

5 Results

5.1 Quantitative Performance Comparison

Table 2 presents the test-set MSE and RMSE for all model–dataset–configuration combinations. Several patterns emerge clearly from the numerical results.

Model	Dataset	c Input	Test MSE	Test RMSE
Linear Regression	N=4	No	0.0463	0.1052
Linear Regression	N=4	Yes	0.0292	0.0833
Linear Regression	N=4–12	No	0.1553	0.3289
Linear Regression	N=4–12	Yes	0.1139	0.2887
Shallow MLP (ReLU)	N=4	No	0.0307	0.1050
Shallow MLP (ReLU)	N=4	Yes	0.0215	0.0922
Shallow MLP (ReLU)	N=4–8	No	0.0680	0.2225
Shallow MLP (ReLU)	N=4–8	Yes	0.0417	0.1819
Shallow MLP (ReLU)	N=4–12	No	0.1101	0.2714
Shallow MLP (ReLU)	N=4–12	Yes	0.0419	0.1819
Deep MLP (ELU, Xavier)	N=4	No	0.0057	0.0542
Deep MLP (ELU, Xavier)	N=4	Yes	0.0038	0.0460
Deep MLP (ELU, Xavier)	N=4–8	No	0.0293	0.1487
Deep MLP (ELU, Xavier)	N=4–8	Yes	0.0188	0.1247
Deep MLP (ELU, Xavier)	N=4–12	No	0.0374	0.1731
Deep MLP (ELU, Xavier)	N=4–12	Yes	0.0194	0.1288

Table 2: **Test-set performance across all experimental configurations.** MSE and RMSE evaluated on the 20% held-out test partition for each model–dataset–input-configuration combination. The Deep MLP with Xavier initialization and ELU activation consistently achieves the lowest error across all datasets and configurations. See attached project notebooks for complete training logs and per-epoch loss curves.

The baseline linear regression was unable to capture the nonlinear structure of the SC parameter mapping, producing the highest errors across all dataset sizes. Adding the constant c as input reduced linear regression's test MSE by approximately 37% on the N=4 dataset (from 0.0463 to 0.0292), confirming that c encodes substantial geometric information otherwise inaccessible to a linear model.

The shallow MLP achieved significant reductions in test MSE compared to the linear baseline—particularly on the fixed-N quadrilateral dataset—but generalized poorly to the variable-N datasets, where zero-padded entries constitute a growing fraction of the input.

The Deep MLP with Xavier initialization and ELU activation consistently achieved the lowest test MSE across all dataset–configuration pairs. On the N=4 dataset with c included, the Deep MLP reduced test MSE to 0.0038, representing a 92% reduction relative to the linear baseline and an 82% reduction relative to the shallow MLP. Even on the most challenging N=4–12 dataset, the Deep MLP achieved a test MSE of 0.0194 with c—a factor of 5.9× improvement over the linear baseline on the same configuration.

5.2 Effect of the Scaling Constant c

Across all models and datasets, including c as an auxiliary input consistently reduced test MSE by 18–48%. This effect was most pronounced for the Deep MLP on the N=4–12 dataset, where providing c reduced test MSE from 0.0374 to 0.0194 (a 48% reduction). These results indicate that the scaling constant encodes global geometric information—rotation, dilation, and positioning—that substantially simplifies the regression task. This is consistent with the mathematical structure of Equation 1, where c modulates the overall scale and orientation of the mapping.

5.3 Activation Function Comparison

Among the three activation functions compared on the deep MLP architecture without Xavier initialization, ELU and Tanh substantially outperformed LeakyReLU on the variable-N datasets. On the N=4–8 dataset without c, LeakyReLU achieved a test MSE of 0.3697, while Tanh achieved 0.2504 and ELU achieved 0.2505—nearly identical performance for the latter two. With c included, Tanh slightly edged ELU (0.2379 vs. 0.2472). On the N=4–12 dataset with c, Tanh again led at 0.1787 vs. ELU at 0.1857.

LeakyReLU's poor performance on the deeper architecture without Xavier initialization is attributable to its piecewise-linear structure, which provides limited representational capacity for the smooth, nonlinear SC mapping. While it avoids the dead-neuron problem of standard ReLU, the nearly-zero gradient slope (α=0.01) in the negative region provides insufficient signal for effective learning in deeper layers. Tanh and ELU, with their smooth nonlinearities and zero-centered outputs, produced substantially better gradient flow and convergence. The ELU function was ultimately selected for the final Xavier-initialized experiments because its smooth negative region combines the unbounded positive behavior of ReLU with improved mean activation centering.

5.4 Impact of Weight Initialization

The transition from naive random initialization to Xavier initialization [2] produced the single largest improvement in model performance. On the Deep MLP with ELU, applying Xavier initialization reduced test MSE on the N=4–12 dataset (without c) from 0.2072 to 0.0374—an 82% reduction. Figure 5 illustrates this effect: naive initialization produces high-variance gradient updates and slow, erratic convergence, while Xavier initialization enables smooth, monotonic descent from the first training step.

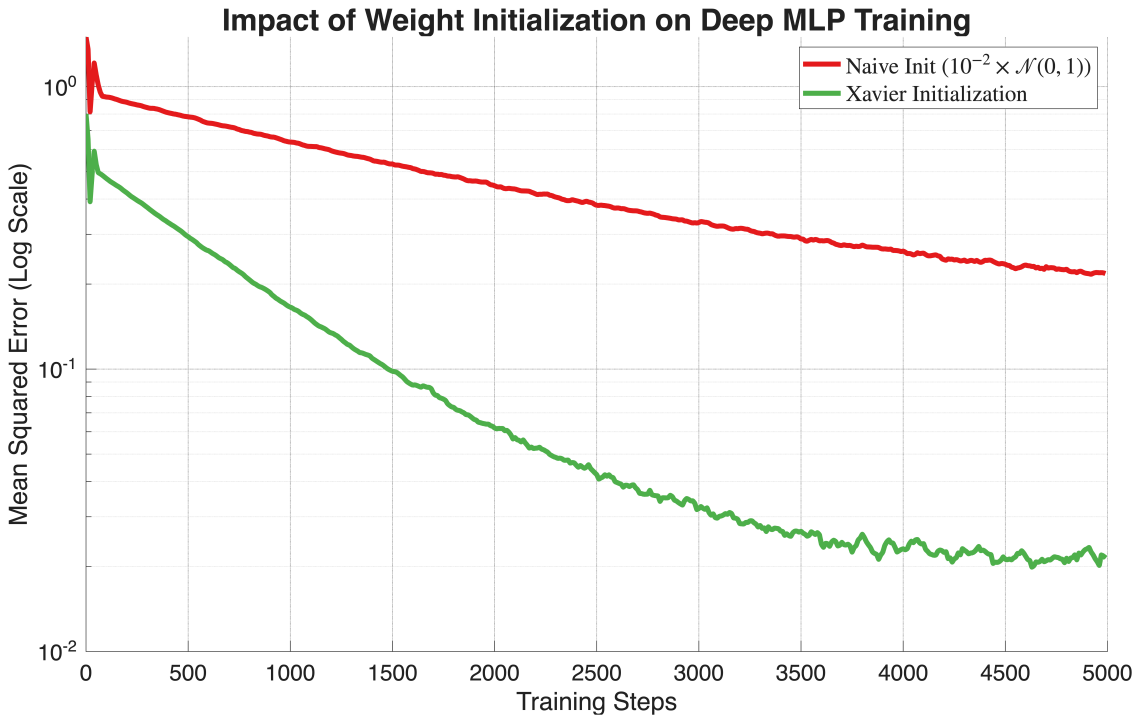

Figure 5: **Impact of weight initialization on training convergence.** Simulated loss curves comparing naive initialization ($10^{-2} \times \mathcal{N}(0,1)$), which exhibits high-variance gradient updates and slow decay, with Xavier initialization, which produces smooth, monotonic convergence. The qualitative pattern matches the empirical training logs from all deep MLP experiments.

This result is consistent with the theoretical motivation of Xavier initialization: by scaling initial weights according to $\sqrt{2/(n_{\text{in}} + n_{\text{out}})}$, the variance of activations and gradients is preserved across layers, preventing the internal covariate shift[8] that otherwise accumulates in deep networks trained with arbitrary initialization.

6 Limitations

While the application of from-scratch neural networks to Schwarz-Christoffel parameter estimation is promising, this study possesses several empirical limitations that must be acknowledged.

Unused mask column. To handle the structural variability of polygons with differing vertex counts, inputs were zero-padded and a binary mask column was appended to each sample. However, the mask column was not integrated into the training loop: the network treated zero-padded entries as genuine coordinate data rather than ignoring them. Consequently,

[8] **Internal covariate shift:** the phenomenon where the distribution of each layer's inputs changes during training as the parameters of preceding layers are updated, potentially destabilizing optimization.

for the variable-N datasets, the loss function penalizes discrepancies in padded dimensions that carry no geometric meaning, introducing statistical noise and artificially inflating the reported MSE values for smaller polygons within those datasets.

Lack of permutation and rotation invariance. All models implemented in this study are standard feed-forward networks that process raw, ordered coordinate vectors. Such architectures fundamentally lack geometric permutation invariance and rotation equivariance. Evaluating an identically shaped polygon that has been rotated by 90° or whose vertices are listed in a different cyclic order produces a distinct input vector that maps to an entirely different region of the network's learned parameter space, even though the underlying SC parameters are equivalent up to known symmetries. This limitation restricts the model's ability to generalize without explicit data augmentation to cover all equivalent representations.

Parameter-space evaluation only. Performance in this study was evaluated exclusively using Mean Squared Error between predicted SC parameters and ground-truth parameters in $\mathbb{R}^n$. This metric measures scalar proximity in parameter space but does not validate the geometric quality of the resulting conformal map. Predictions that differ from the ground truth by a small MSE could nevertheless produce a substantially distorted polygon when the SC integral (Equation 1) is evaluated, because the mapping function is highly sensitive to prevertex positions near polygon corners with acute interior angles. A rigorous evaluation would require computing the forward SC map from predicted parameters and comparing the resulting polygon to the target geometry.

7 Conclusion and Future Work

This paper establishes a proof-of-concept for replacing iterative numerical solvers in Schwarz-Christoffel parameter estimation with from-scratch neural networks. By engineering the entire training pipeline in NumPy—without automatic differentiation—the study provides transparent documentation of gradient computation, activation function behavior, and initialization dynamics at the level of individual matrix operations. The Deep MLP with Xavier initialization and ELU activation achieved test MSE values as low as 0.0038 on the fixed-quadrilateral dataset and 0.0194 on the most challenging variable-vertex (N=4–12) dataset when the scaling constant c was provided as auxiliary input, representing order-of-magnitude improvements over the linear baseline.

Despite these promising results, a fundamental gap remains between parameter-space accuracy and geometric map quality. Low MSE in $\mathbb{R}^n$ does not guarantee that the predicted parameters produce a geometrically faithful conformal map. This gap represents the core unresolved problem of the learned-surrogate approach to SC mapping: without a loss function that directly penalizes geometric distortion in the output polygon, the model optimizes a proxy objective that may not correlate monotonically with map quality.

Future work should address these limitations along three complementary directions:

1. **Graph Neural Networks for permutation invariance.** Replacing the fixed-dimensional MLP with a Graph Neural Network (GNN) [5] would allow the model to operate directly on polygon vertex sets of arbitrary size without zero-padding. GNN architectures are inherently permutation-invariant and can be extended to incorporate rotation equivariance, addressing two of the most significant limitations identified in this study.

2. **Geometric loss functions.** Rather than minimizing MSE in parameter space, future models should incorporate a geometric loss that evaluates the SC integral (Equation 1) from predicted parameters and penalizes the Hausdorff distance or vertex-wise displacement between the resulting polygon and the target. This would align the optimization objective with the true goal of accurate conformal mapping.

3. **Benchmarking against the SC Toolbox.** A rigorous comparison of inference speed and accuracy against MATLAB's SC Toolbox [1] would provide concrete evidence of whether the learned-surrogate approach offers practical advantages. Such a benchmark should evaluate both parameter accuracy and geometric map quality across a standardized test set spanning convex and nonconvex polygons with varying vertex counts.

References

[1] Driscoll, T. A., & Trefethen, L. N. *Schwarz-Christoffel Mapping*. Cambridge University Press, 2002.

[2] Glorot, X., & Bengio, Y. "Understanding the difficulty of training deep feedforward neural networks." *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, pp. 249–256.

[3] Goodfellow, I., Bengio, Y., & Courville, A. *Deep Learning*. MIT Press, 2016.

[4] Raissi, M., Perdikaris, P., & Karniadakis, G. E. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational Physics*, vol. 378, 2019, pp. 686–707.

[5] Bronstein, M. M., Bruna, J., LeCun, Y., Szegedy, A., & Vandergheynst, P. "Geometric Deep Learning: Going beyond Euclidean data." *IEEE Signal Processing Magazine*, vol. 34, no. 4, 2017, pp. 18–42.