PolyChristoffel Networks

Obtaining latent manifolds via curvature learning

Mihir Talati

December 19, 2025

Motivation & Intuition

Why another dynamics model?

- Most current NNs are considered Black-Box simulators.
- We know something useful hides in Latent Space (see AlexNet) but we can't extract it due to arbitrary nonlinearities.
- Modern Models solve this by imposing structure on embeddings and evolutions.
- Physics models such as HNNs/LNNs do this via:
    - invariances, conservation laws, interpretable coefficients
    - extrapolation beyond the training distribution
- However we suspect that the geometry of latent spaces encode these properties

Geometric intuition: "forces" as coordinate effects and nonlinearity

- A straight-line path in one coordinate system can look curved in another.
- Example: inertial motion in Cartesian becomes nontrivial in polar coordinates.
- The apparent "forces" come from geometry via **Christoffel symbols**.

$$\ddot{x}^i + \Gamma^i{}_{jk}(x)\,\dot{x}^j \dot{x}^k = 0$$

- Interpret $\Gamma^i{}_{jk}(x)$ as **state-dependent couplings** of velocities into acceleration.
- Goal here: make the **equations of motion** a first-class object:

learn $\Gamma(x)$ so that trajectories follow a geodesic equation.

Graduate DiffGeom in 30 seconds: metric, connection, geodesics

- A (pseudo-)Riemannian metric $g(x)$ defines local inner products:

$$\langle u, v\rangle_x = u^\top g(x)\, v$$

- The Levi–Civita connection (torsion-free, metric-compatible) yields Christoffels:

$$\Gamma^i{}_{jk}(g) = \tfrac{1}{2} g^{i\ell}\Big(\partial_j g_{k\ell} + \partial_k g_{j\ell} - \partial_\ell g_{jk}\Big)$$

- Geodesics are “straightest” paths under that connection:

$$\ddot{x}^i + \Gamma^i{}_{jk}(x)\, \dot{x}^j \dot{x}^k = 0.$$

Model

Model overview

Inputs: initial state/latent x_0, initial velocity v_0 (and optionally observed trajectory).
Learn: a polynomial Christoffel field $\Gamma(x)$ (and optionally an autoencoder).

Observed y_t	Latent/state x_t
$x_t = E(y_t) \Rightarrow$	$\Gamma(x)$ produces geodesic rollout $\hat{x}_{0:T}$
$\hat{y}_t = D(\hat{x}_t)$	Losses enforce dynamics + metric consistency

Design principle: keep the dynamics layer interpretable by making $\Gamma(x)$ a low-degree polynomial.

Parameterization: polynomial Christoffel network

Let $x \in \mathbb{R}^d$. Build monomial features $\phi(x) \in \mathbb{R}^P$ (degree $\leq p$):

$$\phi(x) = \left[1,\ x_1, \dots, x_d,\ x_1^2,\ x_1 x_2, \dots\right].$$

Then define

$$\Gamma^i{}_{jk}(x) = \sum_{m=1}^{P} C^i{}_{jk,m}\, \phi_m(x).$$

- $C^i{}_{jk,m}$ are trainable coefficients (directly interpretable).
- Enforce torsion-free symmetry by construction:

$$\Gamma^i{}_{jk} = \Gamma^i{}_{kj}.$$

Dynamics block: geodesic equation as a neural layer

Use first-order state $z = (x, v)$ with $v = \dot{x}$:

$$\dot{x} = v, \qquad \dot{v}^i = -\Gamma^i{}_{jk}(x)\, v^j v^k.$$

This defines a deterministic vector field $f_\theta(z)$ where parameters are the polynomial coefficients.

Interpretability:

- acceleration is quadratic in velocity and structured by $\Gamma(x)$
- no arbitrary activation needed in the dynamics law

Integrator: Scuffed ResNet rollout (discrete geodesic flow)

A simple differentiable integrator (semi-implicit Euler):

$$v_{t+1} = v_t + \Delta t \, a(x_t, v_t), \quad x_{t+1} = x_t + \Delta t \, v_{t+1},$$

where

$$a^i(x, v) = -\Gamma^i{}_{jk}(x) \, v^j v^k.$$

- Equivalent to stacking residual blocks with shared parameters.
- Stable, CUDA-friendly, easy to backprop through long rollouts.
- Later swap-in: RK4 / differentiable ODE solvers if needed.

Optional: jointly learned autoencoder with manifold constraint

When observations live in y-space (images, fields, etc.), use an encoder/decoder:

$$x_t = E_\psi(y_t), \qquad \hat{y}_t = D_\psi(\hat{x}_t).$$

To enforce latent validity on a chosen manifold $\mathcal{M}$:

$$x \leftarrow \Pi_{\mathcal{M}}(x)$$

(e.g., sphere projection, hyperboloid retraction).

- Joint training: geometry losses backprop through E_ψ.
- Encourages a chart where dynamics are "as geodesic as possible."

Why metric reconstruction / pseudo-Riemannian regularization?

Learning $\Gamma(x)$ freely can fit trajectories but yield a connection that is not Levi–Civita of any metric.

To encourage a **(pseudo-)Riemannian interpretation**, impose constraints consistent with:

- torsion-free: ${\Gamma^i}_{jk} = {\Gamma^i}_{kj}$
- metric compatibility: $\nabla g = 0$
- fixed signature (p, q), non-degeneracy ($\det g \neq 0$)

Metric reconstruction idea: integrate the compatibility equation

Metric compatibility in coordinates:

$$\partial_i g_{jk} = \Gamma^{\ell}_{\ ij}\, g_{\ell k} + \Gamma^{\ell}_{\ ik}\, g_{j\ell}.$$

Treat this as a (path) ODE along a curve $x(s)$ from a basepoint x_0:

$$\frac{\mathrm{d}}{\mathrm{d}s} g_{jk}(s) = \Big(\Gamma^{\ell}_{\ ij}(x(s)) g_{\ell k}(s) + \Gamma^{\ell}_{\ ik}(x(s)) g_{j\ell}(s)\Big) \frac{\mathrm{d}x^i}{\mathrm{d}s}.$$

- Choose $g(x_0) = g_0$ with fixed signature; integrate to get $g(x)$.
- If Γ is not metric-realizable, the reconstructed $g(x)$ becomes **path-dependent**.

Consistency loss: penalize path dependence (loop loss)

Compute $g(x)$ via two different paths:

- Path A: straight line $x_0 \to x$
- Path B: two-segment $x_0 \to x_m \to x$

Then define:

$$\mathcal{L}_{\text{loop}} = \|g^{(A)}(x) - g^{(B)}(x)\|_F^2.$$

Interpretation:

- pushes $\Gamma(x)$ toward connections that preserve some bilinear form
- provides extra signal even if you only supervise trajectories

Pseudo-Riemannian validity losses (practical)

Given reconstructed $g(x)$:

- Symmetry:

$$\mathcal{L}_{\mathrm{sym}} = \|g - g^{\top}\|_F^2$$

- Non-degeneracy barrier (avoid singular metric):

$$\mathcal{L}_{\mathrm{det}} = \mathrm{softplus}\big(\alpha - \log|\det g|\big)$$

- Fixed signature (p, q) via eigenvalue sign penalties:

$$\mathcal{L}_{\mathrm{sig}} = \sum_{i=1}^{d} \mathrm{softplus}\big(-\beta\, s_i \lambda_i(g)\big), \quad s_i \in \{+1, -1\}.$$

Overall objective and training loop

Primary fit: match trajectories (latent or decoded):

$$\mathcal{L}_{\text{traj}} = \frac{1}{T}\sum_t \|\hat{x}_t - x_t\|^2 \quad \text{or} \quad \mathcal{L}_{\text{recon}} = \frac{1}{T}\sum_t \|D(\hat{x}_t) - y_t\|^2.$$

Regularize geometry:

$$\mathcal{L} = \mathcal{L}_{\text{traj/recon}} + \lambda_{\text{loop}}\mathcal{L}_{\text{loop}} + \lambda_{\text{det}}\mathcal{L}_{\text{det}} + \lambda_{\text{sig}}\mathcal{L}_{\text{sig}} + \lambda\|C\|_2^2.$$

- Backprop through (i) rollout integrator and (ii) metric reconstruction.
- Coefficients C remain interpretable (polynomial terms in Γ).

Training: Backprop + Computational Graph + Pytorch

Forward pass (one minibatch)

- Initialize from data (or latent): (x_0, v_0).
- Rollout dynamics for $t = 0, \dots, T-1$ with step Δt:

$$v_{t+1} = v_t + \Delta t \, a(x_t, v_t), \qquad x_{t+1} = x_t + \Delta t \, v_{t+1},$$

  where

$$a^i(x_t, v_t) = -\Gamma^i{}_{jk}(x_t) \, v_t^j v_t^k, \qquad \Gamma^i{}_{jk}(x) = \sum_{m=1}^{P} C^i{}_{jk,m} \, \phi_m(x).$$

- Compute loss on the trajectory (and optional geometry regularizers):

$$\mathcal{L} = \sum_{t=1}^{T} \left\| x_t - x_t^{\text{true}} \right\|^2 + \lambda \, \mathcal{L}_{\text{geom}}.$$

Autodiff view: the rollout is a differentiable computation graph. Gradients flow

$$\mathcal{L} \;\to\; (x_{1:T}, v_{1:T}) \;\to\; a(\cdot) \;\to\; \Gamma(x_t) \;\to\; C.$$

Gradient signal on Christoffel coefficients

Because Γ is **linear** in the coefficients $C^i_{\ jk,m}$, the gradient has a clean form.

At a single step t:

$$\Gamma^i_{\ jk}(x_t) = \sum_{m=1}^{P} C^i_{\ jk,m}\,\phi_m(x_t) \quad \Longrightarrow \quad \frac{\partial \Gamma^i_{\ jk}(x_t)}{\partial C^p_{\ qr,m}} = \delta^i_p\,\delta^q_j\,\delta^r_k\,\phi_m(x_t).$$

Acceleration coupling:

$$a^i_t = -\Gamma^i_{\ jk}(x_t)\,v^j_t v^k_t \quad \Longrightarrow \quad \frac{\partial a^i_t}{\partial C^i_{\ jk,m}} = -\phi_m(x_t)\,v^j_t v^k_t.$$

Chain rule through time (BPTT):

$$\frac{\partial \mathcal{L}}{\partial C^i_{\ jk,m}} = \sum_{t=0}^{T-1} \left\langle \frac{\partial \mathcal{L}}{\partial a^i_t}, -\phi_m(x_t)\,v^j_t v^k_t \right\rangle + \frac{\partial \mathcal{L}_{\text{geom}}}{\partial C^i_{\ jk,m}}.$$

Interpretation: each coefficient update is a weighted sum of *state features* $\phi_m(x_t)$ times *velocity interactions* $v^j_t v^k_t$, so learned geometry can be inspected term-by-term (e.g., which monomials drive which couplings).

Proof of Concept experiment: polar inertial motion

Use a synthetic benchmark with known Christoffels:

$$\Gamma^r_{\theta\theta} = -r, \qquad \Gamma^\theta_{r\theta} = \Gamma^\theta_{\theta r} = \frac{1}{r}$$

(others $= 0$), derived from Euclidean plane in polar coordinates.

- Train $\Gamma(x)$ to reproduce trajectories.
- Use diagnostics to compare learned components to ground truth.
- Verify torsion-free symmetry and reduced path dependence in reconstructed g.

Diagnostics I: Learned Christoffels vs. ground truth

What we plot

Evaluate the learned connection on a grid (r, θ) and extract:

$$\Gamma^r_{\theta\theta}(r,\theta), \quad \Gamma^\theta_{r\theta}(r,\theta)$$

Compare to ground truth:

$$\Gamma^r_{\theta\theta} = -r, \qquad \Gamma^\theta_{r\theta} = \frac{1}{r}.$$

How to interpret

- θ-**invariance:** GT depends only on r, so learned heatmaps should be *nearly constant across* θ.
- **Shape:** $\Gamma^r_{\theta\theta}$ should look like a linear ramp in r; $\Gamma^\theta_{r\theta}$ should decay like $1/r$.
- **Torsion-free check:** $\Gamma^\theta_{r\theta} \approx \Gamma^\theta_{\theta r}$.

Christoffel Diagnostics

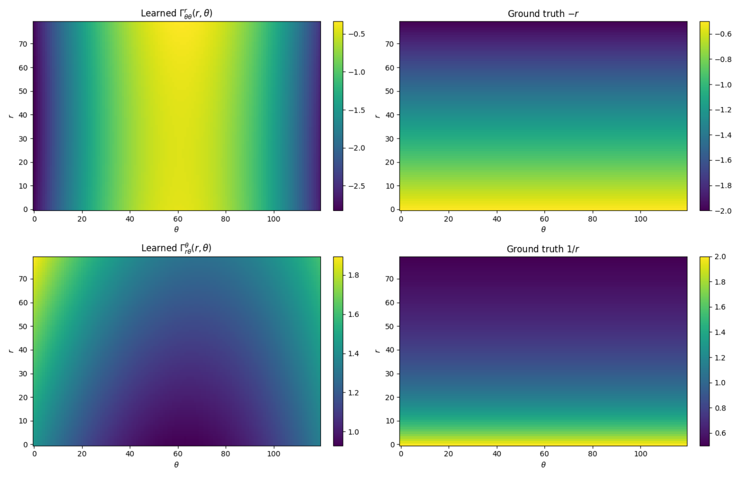

Figure 1: Learned vs. GT Christoffel components on (r, θ) grid.

Diagnostics II: Reconstructed metric and path-dependence (loop error)

Metric target (polar plane)

The Euclidean metric in polar coordinates is:

$$g(r,\theta) = \begin{pmatrix} 1 & 0 \\ 0 & r^2 \end{pmatrix}.$$

We reconstruct g from the learned Γ by integrating metric-compatibility:

$$\nabla g = 0 \quad \Rightarrow \quad \partial_i g_{jk} = \Gamma^{\ell}_{ij} g_{\ell k} + \Gamma^{\ell}_{ik} g_{j\ell}.$$

How to interpret

- **Diagonal structure:** $g_{r\theta} \approx 0$ (off-diagonal heatmap near zero).
- **Component shapes:** $g_{rr} \approx 1$ (flat), $g_{\theta\theta} \propto r^2$ (quadratic in r).
- **Path dependence:** reconstruct g via two paths; the loop error

  $$\|g^{(A)}(x) - g^{(B)}(x)\|_F^2$$

  should be small if Γ is close to Levi–Civita of some metric.

Pretty Graphs Pt 2

Reconstructed g components and loop error heatmap.

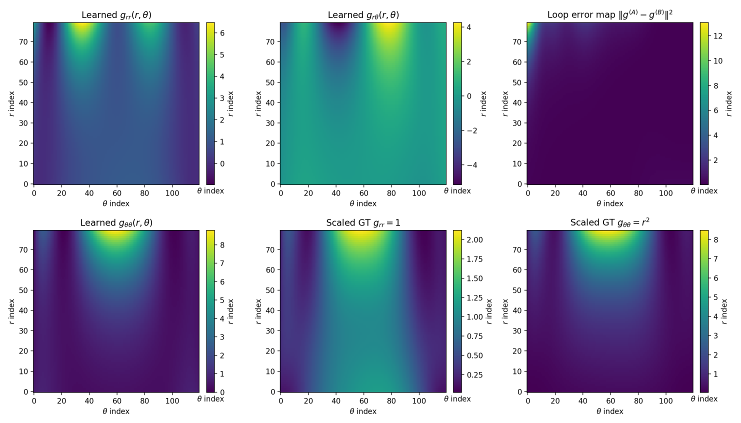

Figure 3: Learned Metric

Pretty Graphs Pt 2 Pt 2

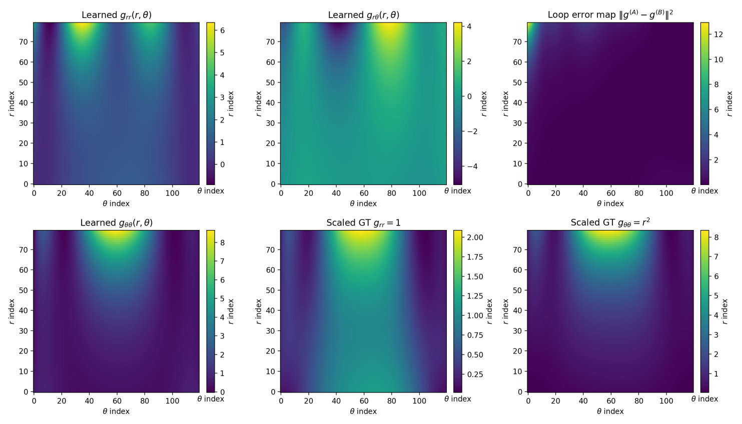

Figure 4: True Metric

While the AutoEncoded metrics are a little harder to interpret due to nonlinear coordinate transforms.

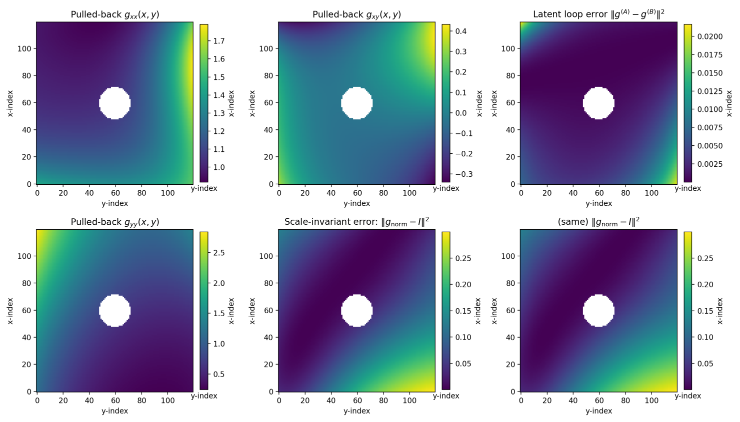

Figure 5: Learned Metric

We can still see what metric values we get by reinterpreting the embeddings under the ground truth metric.

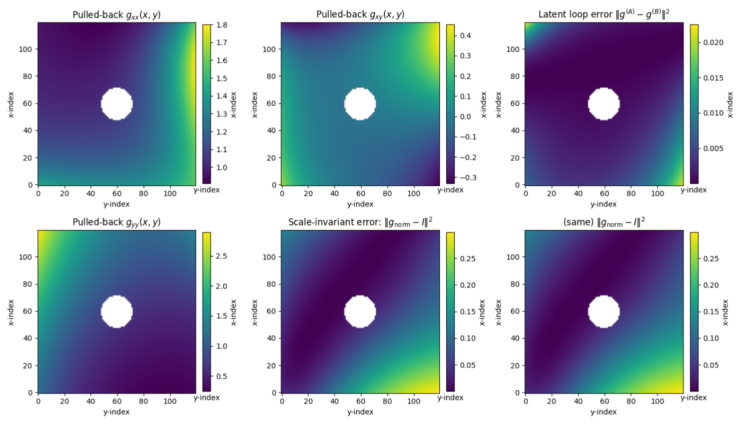

Figure 6: "True"*** Metric

Future Directions

Scaling up: leverage strong autoencoders (CNNs, etc.)

- Many domains already have excellent representation learners:
  - CNN autoencoders / VAEs for images and fields
  - pretrained encoders with semantic latent factors
- Proposal: **freeze or lightly finetune** an existing encoder/decoder, then learn $\Gamma(x)$ (and optionally $g(x)$) on the latent manifold to obtain:
  - interpretable latent dynamics
  - geometry-aware interpolation/extrapolation (geodesics)
  - constraints like signature / nondegeneracy for stability
- The notion of enforcing a Pseudo-Riemannian metric alsso defines a consistent, invariant dot product for vectors at any point on the manifold, which is also precisely how many encoders are trained (e.g in NLP)

Research directions

- **Metric Intervals:** Enforce a notion of "distance" via states through known equivalencies
- **Lorentzian Geometry:** Allow reparametrization of evolution along axes other than time
- **Beyond geodesics:** add forcing/dissipation in a structured way (e.g., geodesic + learned potential or Rayleigh dissipation term).
- **Better metrizability:** directly co-train $g(x)$ and enforce $\Gamma_{\text{poly}} \approx \Gamma(g)$ to guarantee Levi–Civita structure.
- **Topology/coverage:** multiple charts or implicit atlas, while retaining interpretability.
- **Stochastic extensions:** geodesic drift with learned diffusion (SDE in latent).

Takeaways

- Christoffel symbols provide an interpretable parameterization of dynamics:

$$\ddot{x}^i = -\Gamma^i{}_{jk}(x)\,\dot{x}^j\dot{x}^k$$

- Polynomial $\Gamma(x)$ yields a small, inspectable model with calculus-friendly smoothness.
- Metric reconstruction and pseudo-Riemannian regularizers provide extra training signals and geometric validity checks.
- Natural next step: apply on strong latent spaces (e.g., CNN-based autoencoders) to learn **curved latent dynamics**.

References i

- T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural Ordinary Differential Equations," arXiv:1806.07366, 2018.
- S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian Neural Networks," arXiv:1906.01563, 2019.
- E. O. Korman, "Atlas Based Representation and Metric Learning on Manifolds," arXiv:2106.07062, 2021.
- J. Bürge, "Neural geodesic flows," Master's thesis, ETH Zürich, 2025.
- N. He, M. Yang, and R. Ying, "Lorentzian Residual Neural Networks," arXiv:2412.14695, 2025.