Learning Gaussian Mixture Models via Transformer Measure Flows

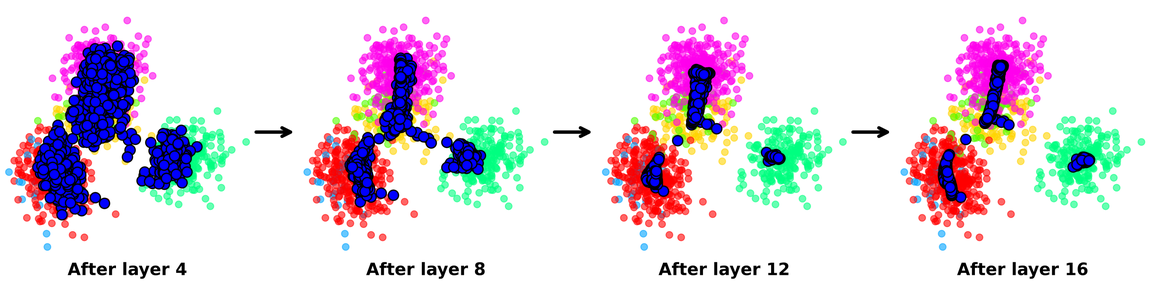
Aleksandr Zimin [1] Anastasiia Kutakh [2] Yury Polyanskiy [3] Philippe Rigollet [1]

MIT [1]Mathematics [2]Physics [3]EECS

Transformers as Clustering Machines

Transformers implements *measure-to-measure* Wasserstein flows: tokens progressively cluster. We introduce **adaptive control** of clustering strength.

Layers progressively tighten clusters as the flow evolves.



Why GMMs?

- Task: Recover clusters from noise
- Mechanism: Transformers implement reverse heat flow (deconvolution)
- Data: Cheap training data generated on the fly

Attention as a Reverse Heat Flow

From tokens to measures. Tokens at layer t: $x_1(t), \ldots, x_n(t) \in \mathcal{S}^{d-1}$

$$\mu_t = \frac{1}{n} \sum_{j=1}^{n} \delta_{x_j(t)}$$

Self-attention is an (ascending) gradient flow

$$x_i(t+1) = x_i(t) + \tau \, \nabla_x \log \Phi_{\mu_t}(x_i)$$

with

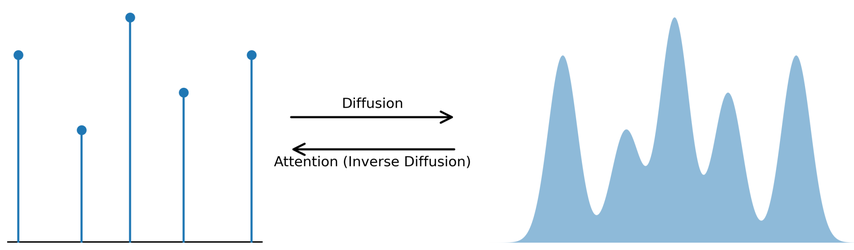
$$\Phi_\mu(x) = \int K(x,y)\, d\mu(y)\,, \quad K(x,y) = \exp(\beta \langle x, y \rangle)$$

PDE As $\tau \to 0$, layers ℓ become time $t = \ell\tau$:

$$\partial_t \mu_t = -\nabla \cdot (\mu_t \, \nabla \log \Phi_{\mu_t}) \quad \text{(Wasserstein gradient flow)}$$

Backward heat flow For β large, [3, 2].

$$\Phi_\mu \approx \beta \, \mu \;\Rightarrow\; \partial_t \mu_t = -\beta \, \Delta \, \mu_t$$



Heat flow = Diffusion / Reverse heat flow = Deconvolution

Adaptive Flow Control

From fixed to adaptive. Control clustering via flow time T:

$$\partial_t x(t) = \mathrm{Attn}(x(t)), \quad t \in [0, T]$$

Discrete implementation. Variable step per layer:

$$x^{(\ell+1)} = x^{(\ell)} + t_\ell \cdot \mathrm{Attn}(x^{(\ell)}), \quad \sum_{\ell=1}^{L} t_\ell = T$$

Uniform: $t_\ell = T/L$ (continuous control)

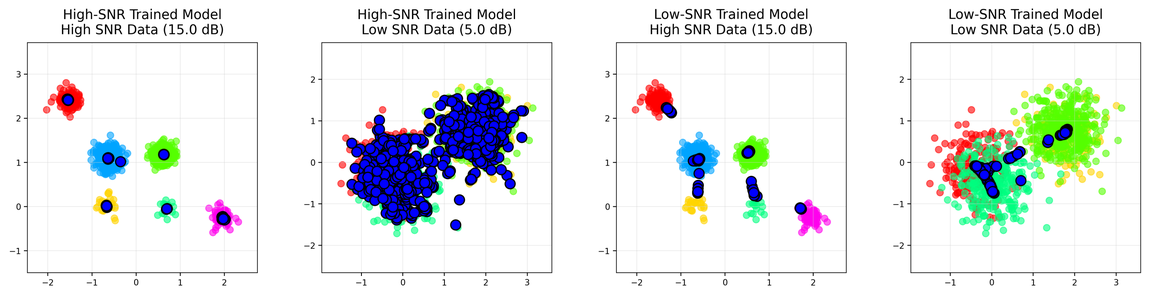
Unit: $t_\ell \in \{0, 1\}$ (integer steps)

Set $T = F(\mathrm{SNR})$ to adapt clustering to data complexity without retraining the model

How Flow Speed Adapts to Data Complexity

Flow time $T = F(\mathrm{SNR})$ is learned to match data difficulty.

- **High SNR (well-separated clusters):** Short flow time—clusters are already visible, only mild refinement needed
- **Low SNR (overlapping clusters):** Long flow time—aggressive clustering to extract structure from noise



Models trained on different SNR data learn different clustering strategies.

Training via Wasserstein Loss Optimization

Data. Input/output pairs $(\hat{\mu}_j, \nu_j)$ generated on the fly

Output: $\nu = \sum_{i=1}^{C} p_i \delta_{\mu_i}$ (mixing measure)

Input: $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} \delta_{X_i}\,, X_i \overset{\text{iid}}{\sim} \nu \star \mathcal{N}(0, I_d)$ (empirical measure)

Learning objective. Squared Wasserstein distance:

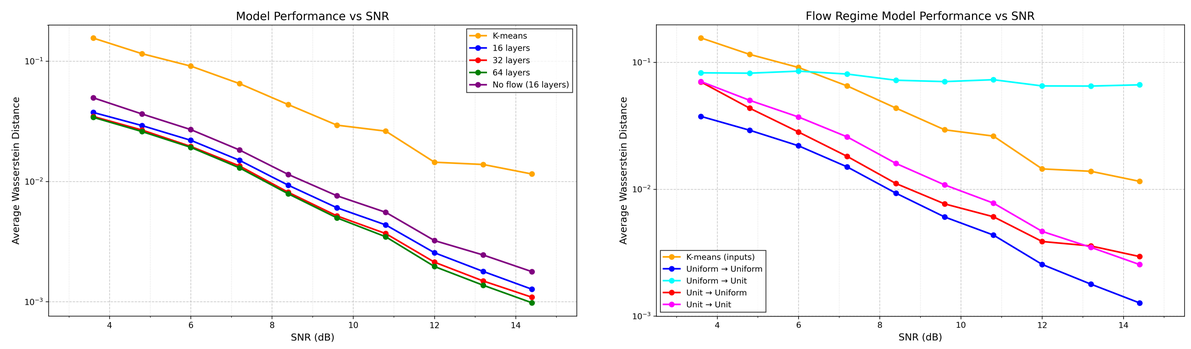
$$\mathcal{L}(\theta) = \sum_j W_2^2(T_\theta)_\# \hat{\mu}_j, \nu_j)$$

Where T_θ is the Transformer flow map.

Direct gradient computation. Wasserstein loss not differentiable (Sinkhorn [1]) so we use locally constant OT plan:

- Gradient simplifies to: $\nabla W_2^2 = \sum_{i,j} \gamma_{ij}^* \nabla \|x_i - y_j\|^2$
- Backprop through positions only, treating weights as constants

Results: 10-100× Performance Gain Over K-means



(a) Depth comparison: K-means vs transformers with 16, 32, 64 layers (all adaptive) + 16-layer no-flow variant ($t = 1$ constant).

(b) Flow schedule ablation: How to distribute flow time T across layers—*Uniform* ($t = T/L$) vs *Unit* ($t = 1$ for $\lfloor T \rfloor$ layers).

Method	High (15 dB)	Mid (10 dB)	Low (5 dB)
K-means baseline	0.0441	0.0558	0.1597
Transformer 16-layer (adaptive)	**0.0016**	**0.0080**	**0.0380**
No-flow 16-layer (t=1)	0.0022	0.0096	0.0462

[1] Cuturi, Teboul, and Vert. *NeurIPS*, 2019.

[2] Geshkovski, Letrouit, Polyanskiy, and Rigollet. *Bull. AMS*, 2025.

[3] Sander, Ablin, Blondel, and Peyré. *AISTATS*, 2022.