Learning Gaussian Mixture Models via Transformer Measure Flows

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

Abstract

We introduce a transformer architecture for approximating Gaussian Mixture Models (GMMs) through a measure-to-measure flow interpretation. Rather than estimating explicit cluster parameters, our model predicts the underlying cluster probability distribution by minimizing Wasserstein distance to the true measure. A key innovation is the flow speed hyperparameter, which adjusts clustering intensity by varying transformer step size and indirectly controlling model depth based on the desired output complexity. Experimental results show performance comparable to or exceeding classical algorithms like K-means, while the synthetic setup provides a lightweight, interpretable sandbox for investigating transformer flow foundations without computational overhead of language-based benchmarks.

1. Introduction

Transformers (Vaswani et al., 2017) have achieved remarkable success across natural language processing, computer vision, and scientific discovery, often setting new benchmarks and enabling large-scale foundation models. However, despite their empirical success, the theoretical understanding of transformers remains limited. A promising theoretical perspective views transformer layers as measure-to-measure flows, interpreting the self-attention mechanism as iterative transformations of probability distributions represented by interacting particles (Sander et al., 2022; Geshkovski et al., 2025). This perspective has recently demonstrated practical relevance in filtering (Bach et al., 2025) and empirical Bayes problems (Teh et al., 2025).

Gaussian Mixture Models (GMMs) present a natural environment to investigate transformers from this distributional viewpoint. This flexible statistical model has found numerous applications across a wide variety of fields (McLachlan & Peel, 2000; Bishop, 2006). Crucially, since both the input (observed data) and output (underlying mixing measure) of a GMM are probability distributions, transformers naturally fit into this framework, enabling a rigorous exploration of their theoretical properties without the high computational or data costs of typical of large-scale language models.

Recent theoretical studies confirm that transformer layers inherently induce a clustering of tokens (Geshkovski et al., 2023; 2025; Chen et al., 2025; Geshkovski et al., 2024) that precisely aligns with the structure of GMMs, where the target mixing measure typically comprises a finite number of atomic supports. Our approach explicitly leverages this clustering property through a novel hyperparameter called *flow speed*, which dynamically adjusts clustering intensity during inference. Unlike standard transformers, whose clustering is constrained by training conditions, our method allows flexible control of clustering behavior, significantly enhancing their effectiveness in distributional approximation tasks such as GMM recovery.

2. Clustering in self-attention dynamics

Following (Geshkovski et al., 2025), we study a simplified transformer setup by setting query, key, and value matrices as identity maps ($Q = K = V = I_d$). Under post-layer normalization, tokens evolve on the unit sphere $\mathcal{S}^{d-1}$. In this idealized setup, a token representation evolves through a continuum of layers indexed by time as

$$\dot{x}(t) = \mathbf{P}_{x(t)} \int e^{\beta\langle x(t),y\rangle} y \, d\mu_t(y), \quad t \in [0,T], \tag{1}$$

[1]MIT Mathematics Department, Cambridge, MA, USA [2]MIT Physics Department, Cambridge, MA, USA [3]MIT Electrical Engineering & Computer Science Department, Cambridge, MA, USA. Correspondence to: Aleksandr Zimin <azimin@mit.edu>.

Accepted at Methods and Opportunities at Small Scale (MOSS), ICML 2025, Vancouver, Canada.

where $\mathbf{P}_x$ projects onto the tangent space at x, and $\mu_t = \frac{1}{N}\sum_{i=1}^{N}\delta_{x_i(t)}$. Geshkovski et al. (2025) show that these so-called *unnormalized self-attentional dynamics* on tokens induce dynamics on μ_t, that correspond to a Wasserstein gradient flow on the interaction energy,

$$E_K[\mu] = \frac{1}{2}\iint K(\langle x, y\rangle)\, d\mu(x)d\mu(y), \tag{2}$$

where both integrals are on S^{d-1} and we use attention kernel $K(z) = e^{\beta z}/\beta$. This perspective implies that the map $t \mapsto E_K[\mu_t]$ is nondecreasing. In fact, under various conditions, μ_t is shown to indeed converge a global maximizer of E_K, which are point masses so that tokens cluster (Geshkovski et al., 2025; Markdahl et al., 2017; Boumal et al., 2024; Chen et al., 2025) though it can get stuck in metastable states with multiple cluster for a long time (Geshkovski et al., 2024). We complement this analysis by analyzing the entropy of μ_t defined as $\mathrm{Ent}(\mu_t) = -\int(\log\mu_t)\mu_t$, where we abuse notation and use μ_t to also denote its density with respect to the surface measure on $\mathcal{S}^{d-1}$. We add a different perspective on this clustering by showing that entropy grows to infinity at least at a linear rate.

The following theorem holds for Wasserstein gradient flows[1] that generalize (1) and are of the form:

$$\dot{x}(t) = \nabla E_K[\mu_t](x(t))\,, \tag{3}$$

where E_K is defined in (2) where K is a spherical positive semi-definite kernel. It shows that entropy grows linearly with t to infinity which is a strong indication that clusters form rapidly.

Theorem 2.1. *Let μ_0 admit a density with respect to the surface measure of $\mathcal{S}^{d-1}$. Let E_K be defined as in (2) where K is a spherical positive semi-definite kernel. Then, the differential entropy increases linearly along the Wasserstein gradient flow on μ_t induced by (3):*

$$\partial_t \mathrm{Ent}(\mu_t) \geq (d-1)\big(E_K[\mu_t] - E_K[\sigma]\big),$$

where σ denotes the uniform probability measure on the sphere, and $E_K[\sigma]$ is the minimal value of $E_K[\mu]$ over all probability measures μ on the sphere. In particular, if $\mu_0 \neq \sigma$, the rate is linear.

The proof is in Appendix A. The linear rate cannot be improved in general. Indeed, Benedetto et al. (2015, Proposition 4.3) show that the above inequality becomes an equality for the Kuramoto model ($\beta = 0$) on the circle ($d = 2$).

In practice, attention heads are parametrized by Key, Query, and Value matrices and are interspersed with fully connected layers. The first message of this paper is that despite its simplicity this model captures the macroscopic behavior of complex, parametrized transformers as illustrated in Figure 1.

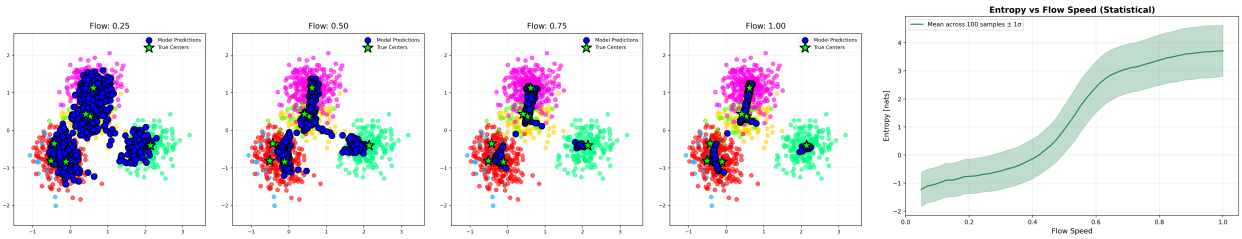

Figure 1. Left: evolution of tokens across layers of our model for one instance. Right: evolution of entropy of tokens across layers averaged over 100 instances.

Given the natural propensity for attention to cluster tokens, we use it to learn GMMs in the next section.

3. Training procedure

Throughout, we consider Gaussian mixtures models where each component has the same spherical covariance matrix[2]. In this model, we observe n independent copies of the random vector $X \sim \mu$ where $\mu = \sum_{i=1}^{C} p_i\mathcal{N}(\mu_i, \sigma^2 I_d)$ and the goal is to learn the *mixing measure* $\nu = \sum_{i=1}^{C} p_i\delta_{\mu_i}$. Note that falls in the notoriously hard Gaussian deconvolution problem since $\mu = \nu \star \mathcal{N}(0, \sigma^2 I_d)$ for which rates of converges are known to be exponentially bad in k (see, e.g., Wu & Yang, 2020).

[1]We refer the reader to Chewi et al. (2025, Chapter 5) for an introduction to Wasserstein gradient flows.

[2]This modeling limitation is not intrinsic to our approach and could easily be lifted in follow-up work.

Unlike traditional EM methods that estimate parameters and suffer from local optima and initialization sensitivity, we directly approximate the cluster distribution by minimizing the average quadratic Wasserstein distance using a method that is closer to NPMLE (Yan et al., 2024; Polyanskiy & Wu, 2020) as it optimizes over the space of distributions, albeit with a different objective function.

Given a sample $X_1, \ldots, X_n \overset{\text{i.i.d.}}{\sim} \mu$, we form their empirical measure $\tilde{\mu} = (1/n)\sum_{i=1}^n \delta_{X_i}$. The transformer acts as a flow map on the space of probability measures to output an estimate $\hat{\nu} = (T_\theta)_\# \tilde{\mu}$ that is hopefully close to ν. In fact, $\hat{\nu}$ is itself an empirical measure over the token representations at the last layer of the transformer and, as such, it depends on the parameters θ of the transformer.

To train the model, we generate N pairs of distributions $(\nu_j, \tilde{\mu}_j)$, $j = 1, \ldots, N$ as follows. First ν_j the mixing measure is drawn at random from an ensemble of discrete measures with a Poisson number of components and Dirichlet weights (see Appendix D.2 for details). Given ν_j, observations $X_1^j, \ldots, X_n^j$ are obtained as independent copies of $X^j \in \mathbb{R}^d$ where $X^j = Z^j + \xi^j$ with $Z^j \sim \nu_j$ independent of $\xi^j \sim \mathcal{N}(0, \sigma^2 I_d)$. Equipped with these pairs, we minimize the loss function $\theta \mapsto \sum_{j=1}^N W_2^2((T_\theta)_\# \tilde{\mu}_j, \nu_j)$ using AdamW; see Appendix for implementation details.

4. Controlling SNR via flow speed regulation

Our experiments demonstrate that the ability of transformers to learn GMMs is directly related to a notion of *signal-to-noise ratio (SNR)* defined as the ratio between the between-class scatter and within-class scatter (see Appendix B).

High SNR corresponds to tight and well-separated clusters while low SNR corresponds to spread out and overlapping clusters. The same set of n points in $\mathbb{R}^d$, could be drawn from few clusters with low SNR or from many clusters with high SNR. Setting an SNR is morally equivalent to setting the number of cluster. Our experiments indicate that transformer models learn to produce mixing measures with an SNR corresponding to the average SNR of their training data; see Figure 2.

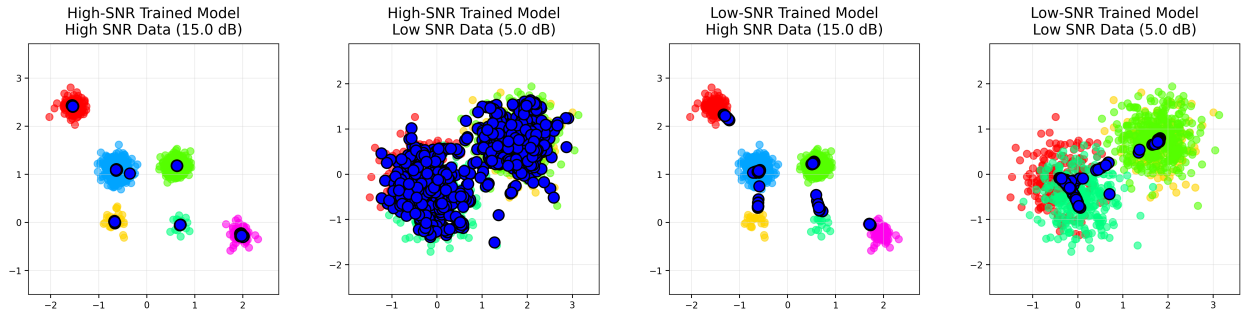

Figure 2. Transformer clustering performance across different training and evaluation scenarios. From left to right: High-SNR trained model on high-SNR data, high-SNR trained model on low-SNR data, low-SNR trained model on high-SNR data, and low-SNR trained model on low-SNR data. Predictions (blue circles) demonstrate varying clustering intensities and effectiveness depending on data complexity. A model trained on high SNR data clusters adequately on similar data but struggles with overlap in complex scenarios. Conversely, the model trained on low SNR data clusters aggressively, possibly condensing clearly separated clusters.

To maintain control on the SNR of the output of the transformer, we leverage the clustering properties of the attention dynamics described in Section 2. Since our theoretical framework indicates long-time clustering, we introduce a new flow speed regulation mechanism that controls the amount of time along which the distribution of tokens evolves along the Wasserstein gradient flow though a specific step-size schedule. This approach is similar in spirit to the recurrent attention blocks employed to scale test-time compute (Bansal et al., 2022; Geiping et al., 2025).

To better match our theoretical framework, where the function form of the vector field is time-invariant, we employ an ALBERT-style (Lan et al., 2020b) architecture where a single attention block (Attention head + fully-connected layer) denoted Attn is shared across layers. Interestingly, this weight-pooling barely impacts the performance of our transformer compared to distinct layers for this task; see Appendix E.4. The flow $\dot{x} = \mathsf{Attn}(x)$ is run until time $T = F(\text{SNR-dB})$ where F is a decreasing function learned from data; see Figure 8 and Appendix C.4 for implementation details).

The continuous-time flow is discretized using a forward Euler scheme with a step size given by the flow-speed parameter $t \in [0, 1]$ that scales attention updates at each Transformer layer. The standard Transformer update $x \to x + \mathsf{Attn}(x)$ is generalized to $x \to x + t \cdot \mathsf{Attn}(x)$. A smaller t results in incremental clustering, suited for high-SNR (simple) datasets since running along the flow would collapse cluster. Larger t encourages more aggressive clustering by traveling longer along the flow and hence tighten clusters. To run the flow until time $T = F(\text{SNR})$, we consider two natural schedules/discretizations summarized in Figure 3. For the *uniform* schedule, the total number of layers L is kept as an exogenous parameter and $t = T/L$, while for the *unit* schedule, L corresponds to the time horizon T and $t = 1$. In the results below, if we don't specify the flow speed, we use uniform. We also compare the performance of flow speed to what we call *No Flow* where we do not adjust to flow speed to account for SNR. Note that when $T = L$ the three methods coincide.

Method	**# Layers**	**Flow Speed**
Uniform	L	$t = T/L$
Unit	$\lfloor T \rfloor$	$t = 1$
No Flow	L	$t = 1$

Figure 3. Step-size schedules

5. Results

First we note that transformers outperform K-means across the board thus expanding their hegemony further into statistical methodology; see Figure 4 and further results in the appendix.

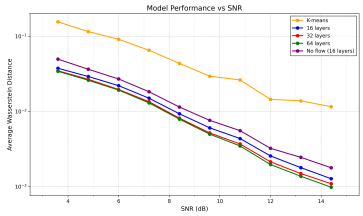

Figure 4. Average model performance for different SNR.

Moreover, our results consistently indicate that both schedules are better than No Flow and that among the two, the Uniform schedule is preferable. This is not surprising as it also give the user the freedom to choose a suitable number of layers. More specifically, we employ both schedules at training and test time with possible mix. Uniform both at training and inference time gives the best performance as indicated in Table 1. Interestingly, the Uniform schedule presents some robustness indicating that it can be employed at inference time by repeating and attention block trained with a different schedule.

Finally, GMM learning is not, in itself, a clustering task. We combine it with a post-processing step with K-means applied to the transformer output. Our results indicate that this step levels the performance of various architectures as K-means rounds the output of various architectures to a estimated mixing measures with similar performance; see Appendix E.5.

	K-means	Uniform → Uniform	Uniform → Unit	Unit → Uniform	Unit → Unit
High SNR	0.0441	0.0016	0.0867	0.0039	0.0029
Moderate SNR	0.0558	0.0080	0.0894	0.0103	0.0120
Low SNR	0.1597	0.0380	0.1095	0.0511	0.0605

Table 1. Comparison of Wasserstein losses for the baseline 16-layer model evaluated under different flow-speed regimes. The notation "Training → Evaluation" indicates the flow-speed parameter used during training (left side) and evaluation (right side).

6. Conclusion

We have established Gaussian Mixture Models (GMMs) as a potent and scalable framework for dissecting the clustering capabilities of transformers. Our findings show that transformers can effectively learn GMMs from synthetic data, surpassing traditional algorithms such as K-means in this domain. More profoundly, this work leverages GMMs to illuminate transformers' behavior as flow maps over probability measures, offering new perspectives on their operational dynamics. The core methodological innovation presented is a flow speed regulation mechanism, implemented through adaptive step sizes, which finely tunes token clustering by managing the evolution duration of token distributions within the self-attention flow.

This research opens up compelling directions for future exploration. A primary focus will be on further advancing transformer efficacy in learning GMMs and tackling a wider array of classical statistical problems. Additionally, thoroughly investigating the impact of our speed regulation mechanism on broader transformer performance is essential. A specific promising avenue involves examining on easy-to-hard instance generalization, following the work of (Geiping et al., 2025).

References

Bach, E., Baptista, R., Calvello, E., Chen, B., and Stuart, A. Learning enhanced ensemble filters, 2025.

Bansal, A., Schwarzschild, A., Borgnia, E., Emam, Z., Huang, F., Goldblum, M., and Goldstein, T. End-to-end algorithm synthesis with recurrent networks: logical extrapolation without overthinking. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.

Benedetto, D., Caglioti, E., and Montemagno, U. On the complete phase synchronization for the Kuramoto model in the mean-field limit. *Communications in Mathematical Sciences*, 13(7):1775–1786, 2015.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006. ISBN 0-387-31073-8.

Boumal, N., Christopher Criscitiello, C., McRae, A., and Rebjock, Q. Synchronization on circles and spheres with nonlinear interactions. Private communication, 2024.

Bourlard, H. and Kamp, Y. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988. doi: 10.1007/BF00332918. URL https://link.springer.com/article/10.1007/BF00332918.

Calinski, T. and Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27, 1974.

Chen, S., Lin, Z., Polyanskiy, Y., and Rigollet, P. Quantitative clustering in mean-field transformer models, 2025.

Chen, Y. and Yang, Y. Cutoff for exact recovery of gaussian mixture models. *IEEE Transactions on Information Theory*, 67 (6):4223–4238, 2021.

Chewi, S., Niles-Weed, J., and Rigollet, P. *Statistical optimal transport*, volume 2364 of *Lecture Notes in Mathematics*. Springer, Cham, 2025.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300, 2013. URL https://papers.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf.

Cuturi, M., Teboul, O., and Vert, J.-P. Differentiable ranking and sorting using optimal transport. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Dasgupta, S. Learning mixtures of gaussians. In *FOCS'99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pp. 634–644. IEEE, 1999.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x.

Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025.

Geshkovski, B., Letrouit, C., Polyanskiy, Y., and Rigollet, P. The emergence of clusters in self-attention dynamics, 2023.

Geshkovski, B., Koubbi, H., Polyanskiy, Y., and Rigollet, P. Dynamic metastability in the self-attention model, 2024.

Geshkovski, B., Letrouit, C., Polyanskiy, Y., and Rigollet, P. A mathematical perspective on transformers, 2025.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2nd edition, 2009. ISBN 9780387848570. doi: 10.1007/978-0-387-84858-7.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2020a.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*, 2020b.

Markdahl, J., Thunberg, J., and Gonçalves, J. Almost global consensus on the n-sphere. *IEEE Transactions on Automatic Control*, 63(6):1664–1675, 2017.

McLachlan, G. J. and Peel, D. *Finite Mixture Models*. Wiley-Interscience, New York, NY, 2000.

Plaut, E. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*, 2018. doi: 10.48550/arXiv.1804.10253. URL https://arxiv.org/abs/1804.10253.

Polyanskiy, Y. and Wu, Y. Self-regularizing property of nonparametric maximum likelihood estimator in mixture models, 2020.

Sander, M. E., Ablin, P., Blondel, M., and Peyré, G. Sinkformers: Transformers with doubly stochastic attention. In *International Conference on Artificial Intelligence and Statistics*, pp. 3515–3530. PMLR, 2022.

Teh, A., Jabbour, M., and Polyanskiy, Y. Solving empirical bayes via transformers, 2025. URL https://arxiv.org/abs/2502.09844.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vempala, S. and Wang, G. A spectral algorithm for learning mixture models. *Journal of Computer and System Sciences*, 68 (4):841–860, 2004.

Wu, Y. and Yang, P. Optimal estimation of Gaussian mixtures via denoised method of moments. *The Annals of Statistics*, 48 (4):1981 – 2007, 2020.

Yan, Y., Wang, K., and Rigollet, P. Learning Gaussian mixtures using the Wasserstein–Fisher–Rao gradient flow. *Ann. Statist.*, 52(4):1774–1795, 2024.

A. Proof of Theorem 2.1

We consider a probability density $\mu_t(x)$ defined on the unit sphere S^{d-1}, normalized so that $\int_{S^{d-1}} \mu_t(x)\, d\sigma(x) = 1$ for all $t \geq 0$. The density evolves according to the Wasserstein gradient flow governed by the kernel $K(\langle x, y\rangle)$, which means it satisfies the continuity equation:

$$\partial_t \mu_t(x) + \mathrm{div}_S(\mu_t(x) v_t(x)) = 0, \quad \text{where} \quad v_t(x) = \int_{S^{d-1}} \nabla_x K(\langle x, y\rangle) \mu_t(y)\, d\sigma(y).$$

We define the differential entropy of the density as:

$$\mathrm{Ent}(\mu_t) = -\int_{S^{d-1}} \mu_t(x) \log \mu_t(x)\, d\sigma(x).$$

First, we compute the derivative of entropy with respect to time. By differentiating under the integral and applying the continuity equation, we have:

$$\partial_t \mathrm{Ent}(\mu_t) = -\int_{S^{d-1}} (1 + \log \mu_t(x)) \partial_t \mu_t(x)\, d\sigma(x).$$

Using the continuity equation, integration by parts, and the closed manifold property of the sphere, this becomes:

$$\partial_t \mathrm{Ent}(\mu_t) = -\int_{S^{d-1}} \langle \nabla_S \mu_t(x), v_t(x)\rangle\, d\sigma(x).$$

Next, we substitute the definition of the velocity field v_t and exchange integrals:

$$\partial_t \mathrm{Ent}(\mu_t) = \iint_{S^{d-1}\times S^{d-1}} \langle \nabla_S \mu_t(x), \nabla_x K(\langle x, y\rangle)\rangle \mu_t(y)\, d\sigma(y) d\sigma(x).$$

Applying a second integration by parts in the variable x, we find:

$$\partial_t \mathrm{Ent}(\mu_t) = -\iint_{S^{d-1}\times S^{d-1}} \mu_t(x)\mu_t(y)\Delta_{S,x}K(\langle x,y\rangle)\, d\sigma(x)d\sigma(y).$$

Because the kernel K is spherical positive semi-definite, it has a spherical harmonic expansion:

$$K(\langle x,y\rangle) = \sum_{\ell=0}^{\infty} c_\ell Z_\ell(\langle x,y\rangle), \quad c_\ell \geq 0.$$

Using the identity for spherical harmonics, $-\Delta_S Z_\ell = \ell(\ell+d-2)Z_\ell$, we rewrite:

$$-\Delta_{S,x}K(\langle x,y\rangle) = (d-1)(K(\langle x,y\rangle) - c_0) + \sum_{\ell=1}^{\infty} c_\ell(\ell-1)(\ell+d-1)Z_\ell(\langle x,y\rangle).$$

The second summation is again positive semi-definite, implying:

$$\partial_t \mathrm{Ent}(\mu_t) \geq (d-1)\iint_{S^{d-1}\times S^{d-1}} \mu_t(x)\mu_t(y)(K(\langle x,y\rangle) - c_0)\, d\sigma(x)d\sigma(y).$$

Recognizing the definition of the interaction energy $E_K[\mu_t]$ and observing that $E_K[\sigma] = c_0$, where σ denotes the uniform probability measure on the sphere and $E_K[\sigma]$ is the minimal value of $E_K[\mu]$ over all probability measures μ on the sphere, we obtain the stated inequality:

$$\partial_t \mathrm{Ent}(\mu_t) \geq (d-1)(E_K[\mu_t] - E_K[\sigma]).$$

B. Signal-to-Noise ratio

We measure the complexity of data from our Gaussian Mixture Model (GMM) using the Signal-to-Noise Ratio (SNR), defined as:

$$\mathrm{SNR} = \frac{\sum_{i=1}^n p_i \|\mu_i - \mu\|^2}{d\sigma^2}, \quad \text{where} \quad \mu = \sum_{i=1}^n p_i\mu_i.$$

Here, the numerator represents the variance between clusters, while the denominator captures variance within clusters. This definition aligns closely with Fisher's criterion used in Linear Discriminant Analysis (LDA) (Fisher, 1936; Hastie et al., 2009) and corresponds to the Calinski–Harabasz index (Calinski & Harabasz, 1974), commonly used to evaluate clustering quality.

Our SNR definition thus provides a direct measure of dataset complexity relevant for both theoretical analyses and practical clustering tasks. Similar metrics are widely used in literature for quantifying cluster separability, informing theoretical bounds on learnability, and guiding practical applications (Dasgupta, 1999; Chen & Yang, 2021; Vempala & Wang, 2004).

C. Transformer Architecture and Training

Given a dataset of noisy observations $X = \{X_j\}_{j=1}^n$ generated by the GMM, we treat these points as an input token sequence for our Transformer. We also compute the dataset's SNR value s, which is fed into a small MLP flow predictor F. The model outputs a sequence of tokens $\widehat{X} = \{\widehat{X}_i\}_{i=1}^n$, where each $\widehat{X}_i$ predicts the cluster center μ_i from which X_i was sampled. We interpret these predictions as an empirical measure $\tilde{\nu}$ and train the model to minimize the squared 2-Wasserstein distance to the ground-truth measure $\tilde{\nu}$:

$$\widehat{\nu} = \frac{1}{n}\sum_{i=1}^n \delta_{\widehat{X}_i}, \quad \tilde{\nu} = \sum_{i=1}^n \tilde{p}_i\, \delta_{\mu_i}, \quad \widehat{p}_i = \frac{\#\{\text{points in cluster } i\}}{n}.$$

The core of our architecture is an Adaptive Transformer, which applies several shared-weight blocks in a cyclic fashion. The total flow strength $T = F(\mathrm{SNR}-\mathrm{dB})$ is distributed among these blocks: each block receives a fraction t of T and updates its tokens via

$$x \;\leftarrow\; x + t\cdot \mathrm{Attn}(x)\,.$$

Here, F is a monotonically decreasing trainable function (we will describe its specific parametrization later in the appendix).

We reuse the same block weights on each iteration, varying only the flow-speed fraction t. Additional optimizations — input/output normalization and orthogonal encoder/decoder layers — are illustrated in Figure 5.

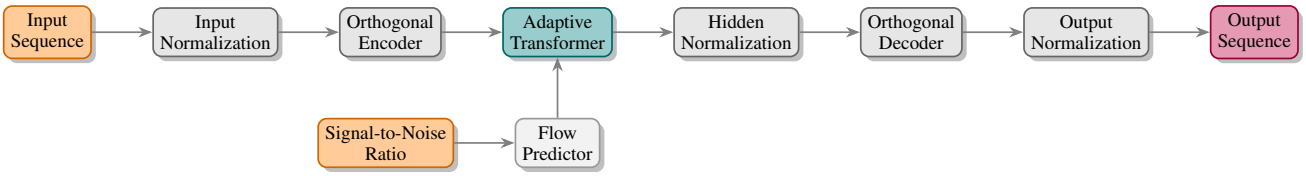

Figure 5. Overview of the Transformer Architecture for GMM Recovery.

C.1. Input Normalization

Each input dataset X is normalized by subtracting the empirical mean and scaling by the mean empirical variance per dimension:

$$X_{\text{norm}} = \frac{X - \mu}{\sigma}, \quad \text{where} \quad \mu = \frac{1}{n}\sum_{j=1}^{n} X_j, \quad \sigma^2 = \frac{1}{nd}\sum_{j=1}^{n} \|X_j - \mu\|^2.$$

This normalization closely relates to data whitening. It ensures translation and scale invariance, meaning that if the input data is shifted by a vector or scaled by a scalar, the model's output correspondingly shifts or scales.

We intentionally avoid full rotation invariance for simplicity and because our data does not exhibit any inherent directional preference.

C.2. Orthogonal Encoding

Following normalization, the data is embedded into a latent space using an orthonormal projection matrix $P \in \mathbb{R}^{d \times d_h}$ with orthonormal rows satisfying $PP^\top = I_d$:

$$X_{\text{latent}} = X_{\text{norm}} P.$$

This step projects the input data from its original dimension d to a higher-dimensional latent space of dimension d_h ($d_h \geq d$), preserving distances, angles, and inner products among data points, retaining intrinsic cluster structures in the latent representation.

C.3. Adaptive Transformer with Flow-Speed Parameter

We adopt an ALBERT-style Transformer (Lan et al., 2020a) with cross-layer parameter sharing, applying several shared-weight attention blocks iteratively. The final flow time T is determined based on the SNR of the data. We reuse the same block weights at each iteration, varying only the flow-speed parameter t.

We distribute the total flow T using two methods:

1. **Uniform Regime**: Fixed number of layers L; each layer receives flow-speed $t = T/L$.

2. **Unit Regime**: Flow-speed $t = 1$ for the first $\lfloor T \rfloor$ layers and $t = 0$ for the remaining layers.

After the Transformer layers, the latent output Z_{raw} is normalized before decoding:

$$Z = \frac{Z_{\text{raw}} - \mu_Z}{\sigma_Z}, \quad \text{where} \quad \mu_Z = \frac{1}{n}\sum_{j=1}^{n} (Z_{\text{raw}})_j, \quad \sigma_Z^2 = \frac{1}{nd}\sum_{j=1}^{n} \|(Z_{\text{raw}})_j - \mu_Z\|^2.$$

C.4. Flow Predictor Parametrization

We parameterize the monotonically increasing function defined on $[0, 1]$ as follows. Given a number of discretization steps N, we define:

$$f_\theta(x) = \sum_{i=1}^{N} w_i \cdot \sigma\left(N \cdot (x - x_i)\right),$$

where $\sigma(\cdot)$ is the sigmoid function. The weights w_i are obtained by applying a softmax function to a learnable vector, ensuring $w_i \geq 0$ and $\sum_i w_i = 1$. The positions x_i are computed as:

$$x_i = \left(1 + \frac{10}{N}\right) \cdot \sigma(\text{learnable vector}) - \frac{5}{N},$$

ensuring that all positions x_i lie within the interval $\left[-\frac{5}{N},\, 1 + \frac{5}{N}\right]$. This choice allows the effective range of the resulting monotonic function $f_\theta(x)$ to potentially be smaller than 1, but still within the interval $[0, 1]$.

The final flow strength function $F(\mathrm{SNR} - \mathrm{dB})$ is computed by appropriately rescaling $f_\theta(x)$ to be monotonically decreasing and to match the range of the SNR-dB values used.

C.5. Orthogonal Decoder

The decoder reconstructs the Transformer's latent representation back to the input space, decomposing it into principal-subspace and residual (null-space) components.

Given the latent code $Z \in \mathbb{R}^{n \times d_h}$ from the encoder–Transformer, the decoder first reconstructs the principal-subspace component using the transpose of the encoder projection matrix P:

$$X_{\text{main}} = ZP^\top. \tag{4}$$

Next, the decoder computes the residual latent component $Z_{\text{res}} = Z - X_{\text{main}}P$ and maps it back to the input space using a learned matrix $Q^\top$:

$$X_{\text{null}} = Z_{\text{res}}Q^\top. \tag{5}$$

The total reconstructed signal is the sum:

$$X_{\text{dec}} = X_{\text{main}} + X_{\text{null}}. \tag{6}$$

This ensures the decoder acts as identity when the Transformer's flow-speed parameter $t = 0$, preserving invertibility. The design combines orthogonal autoencoders and invertible representation learning (Plaut, 2018; Bourlard & Kamp, 1988).

C.6. Output Parameterization

The Transformer directly outputs predictions for each observed point, interpreted as the estimated cluster centers.

These predictions do not require further constraints or normalization, as they are directly matched to the empirical measure of the data.

D. Experiments

D.1. Training Setup

All experiments were conducted on a single NVIDIA A100 80GB SXM4 GPU.

We establish our baseline by training the proposed transformer-based model on synthetic two-dimensional datasets generated from Gaussian Mixture Models (GMMs). Each dataset instance consists of 1000 sample points. To ensure model robustness, we vary the complexity and separability of the data by sampling the SNR-dB uniformly from the interval $[3, 15]$. Additionally, the number of clusters for each synthetic GMM instance is drawn uniformly from $[2, 20]$. This randomized setup covers a wide range of cluster configurations and noise conditions.

Each training epoch consists of 2^{15} synthetic data samples, and training is conducted for 80 epochs. The baseline transformer model comprises 16 identical attention layers, each with 4 attention heads and a latent dimensionality of 32 per head. The model is optimized using the AdamW optimizer with an initial learning rate of 10^{-4}, and a cosine annealing scheduler is applied for learning rate decay.

D.2. Synthetic Data Generation Process

We generate synthetic data from a GMM using the following steps:

1. **Complexity (SNR dB):** Sample a target $\mathrm{SNR}_{\mathrm{dB}}$ from a uniform or truncated log-normal distribution. Convert to linear scale:
$$\mathrm{SNR} = 10^{\mathrm{SNR}_{\mathrm{dB}}/10}.$$
After sampling the cluster means and weights, set noise variance σ^2 to ensure:
$$\sigma^2 = \frac{\sum_{i=1}^{n} p_i \|\mu_i - \mu\|^2/d}{\mathrm{SNR}}, \quad \mu = \sum_{i=1}^{n} p_i \mu_i.$$

2. **Cluster Count:** Sample the number of clusters C from a uniform or truncated Poisson distribution.

3. **Centers:** Draw C independent mean vectors:
$$\mu_i \sim \mathcal{N}(0, I_d), \quad i = 1, \ldots, C.$$

4. **Weights:** Generate mixture proportions:
$$p = (p_1, \ldots, p_n) \sim \mathrm{Dirichlet}(1, \ldots, 1).$$

5. **Assignment:** For a total sample size n, sample cluster assignments:
$$i_j \sim \mathrm{Categorical}(p), \quad j = 1, \ldots, n.$$

6. **Observations:** Generate observations:
$$X_j = \mu_{i_j} + \sigma\, \xi_j, \quad \xi_j \sim \mathcal{N}(0, I_d), \quad j = 1, \ldots, n.$$

D.3. SNR-Scaled Wasserstein Loss

We train our transformer-based model using an SNR-scaled Wasserstein loss to measure the discrepancy between the predicted distribution $\widehat{\nu}$ and the empirical ground-truth distribution $\widetilde{\nu}$.

The squared Wasserstein-2 distance is defined as:

$$W_2^2(\tilde{\nu}, \hat{\nu}) = \inf_{\pi \in \Pi(\tilde{\nu}, \hat{\nu})} \int \|x - y\|^2 d\pi(x, y),$$

where $\Pi(\tilde{\nu}, \hat{\nu})$ is the set of all couplings between $\tilde{\nu}$ and $\hat{\nu}$. The final training objective scales this Wasserstein distance by the data's SNR:

$$\mathcal{L}(\tilde{\nu}, \hat{\nu}) = \mathrm{SNR} \cdot W_2^2(\tilde{\nu}, \hat{\nu}).$$

This SNR-based scaling ensures the loss appropriately reflects the inherent clustering difficulty under varying noise conditions.

While gradients of Wasserstein distances are typically approximated by auto-differentiating through the Sinkhorn algorithm (Cuturi, 2013; Cuturi et al., 2019), we employ a different, computationally simpler, approach tailored to discrete distributions:

Consider two discrete distributions:

$$\mu = \sum_{i=1}^{n} p_i \delta_{x_i}, \quad \nu = \sum_{j=1}^{m} q_j \delta_{y_j}.$$

We perform backpropagation over the positions x_i and y_j, treating the weights p_i, q_j as constants. In this setting, almost surely, the optimal assignment γ_{ij}, represented as an $n \times m$ coupling matrix, remains locally constant. Therefore, the gradient of the Wasserstein distance coincides exactly with the gradient of the quadratic function:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \|x_i - y_j\|^2 \gamma_{ij},$$

where γ_{ij} is the locally constant optimal assignment. Consequently, the gradient computation simplifies to calculating the gradients of this quadratic form directly. In particular, we optimize the Wasserstein distance directly instead of its entropic regularization.

E. Results

E.1. Evaluation Setup and Datasets

We evaluate transformer-based models against classical K-means with the single initialization and with the number of clusters set correctly. Evaluations are conducted on synthetic GMM datasets at three distinct SNR levels, each with moderate cluster complexity (number of clusters sampled from a truncated Poisson distribution with mean=5.0, min=3, max=7):

- **High SNR**: Fixed SNR-dB at 15.
- **Moderate SNR**: Fixed SNR-dB at 10.
- **Low SNR**: Fixed SNR-dB at 5.

Examples of these datasets are illustrated in Figure 6.

For each model and dataset, we compute average Wasserstein and log-Wasserstein losses independently. Using log-Wasserstein loss is advantageous because the expectation of the log-ratio, $\mathbb{E}[\log(\text{model}_1/\text{model}_2)]$, can be directly computed by subtracting the corresponding average log-loss values. The expectation of the log-ratio provides a more robust and numerically stable alternative to the direct ratio, as it mitigates issues arising from division by values close to zero.

E.2. Effect of Number of Layers and Model Performance

We investigate the impact of model depth by comparing transformer models with 16, 32, and 64 layers, as well as a 16-layer transformer without flow prediction, against classical K-means. The goal is to evaluate how increasing the number of layers affects model performance, specifically regarding the accuracy and stability of cluster recovery across varying noise levels and cluster counts.

Tables 2 and 3 summarize the Wasserstein and log-Wasserstein losses, respectively. All transformer-based models significantly outperform classical K-means at low SNR, demonstrating their effectiveness in noisy conditions. At moderate to high SNR levels, deeper transformer models exhibit slightly improved performance, with the no-flow model consistently underperforming compared to other transformer configurations.

Figure 7 visualizes these results, clearly demonstrating the performance gap between transformer models with flow control and the model without flow prediction, particularly at low SNR. Marginal performance improvements are noted when increasing from 16 to 64 layers.

Additionally, Figure 8 shows how flow speed and total flow (defined as flow speed multiplied by the number of layers) vary with SNR-dB for models with different depths. Despite the 64-layer model having four times as many layers as the 16-layer model, the total flow remains almost constant across varying SNR-dB values, highlighting a controlled and stable flow behavior irrespective of layer depth.

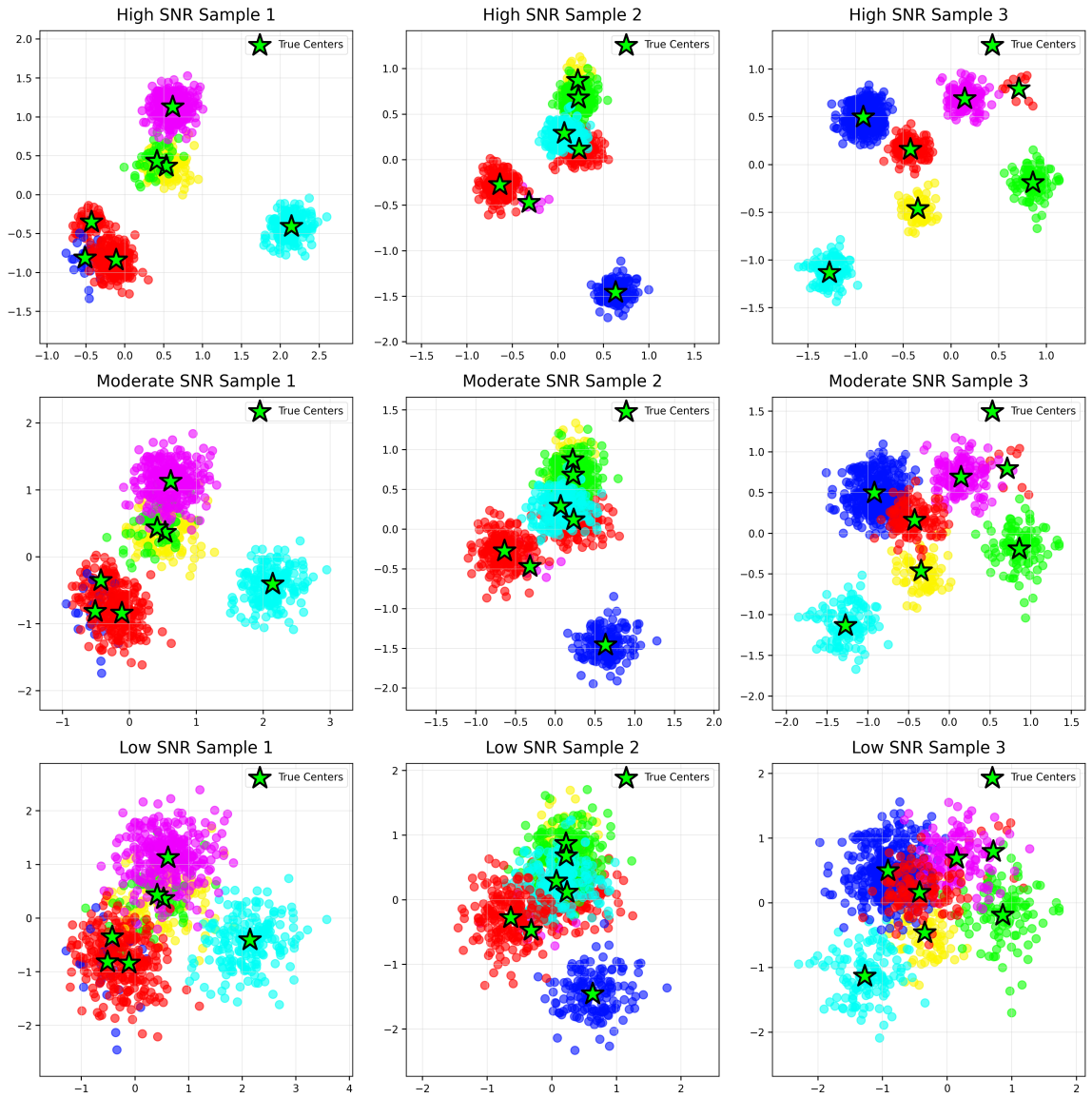

Figure 6. Example synthetic GMM datasets generated at different SNR levels (High, Moderate, and Low). Colors indicate distinct clusters, and stars mark the true cluster centers.

	K-means	16 layers	32 layers	64 layers	No flow (16 layers)
High SNR	0.0441	0.0016	0.0014	0.0013	0.0022
Moderate SNR	0.0558	0.0080	0.0071	0.0069	0.0096
Low SNR	0.1597	0.0380	0.0352	0.0344	0.0462

Table 2. Comparison of Wasserstein losses across transformer models with different numbers of identical attention layers repeated multiple times, classical K-means, and a transformer model without flow prediction. All transformer models use a uniform flow-speed regime.

	K-means	16 layers	32 layers	64 layers	No flow (16 layers)
High SNR	-4.8941	-6.9963	-7.1687	-7.2688	-6.6163
Moderate SNR	-3.7563	-5.3905	-5.5554	-5.6113	-5.1205
Low SNR	-2.2291	-3.7020	-3.7938	-3.8210	-3.4444

Table 3. Comparison of log-Wasserstein losses across transformer models with different numbers of identical attention layers repeated multiple times, classical K-means, and a transformer model without flow prediction. All transformer models use a uniform flow-speed regime.

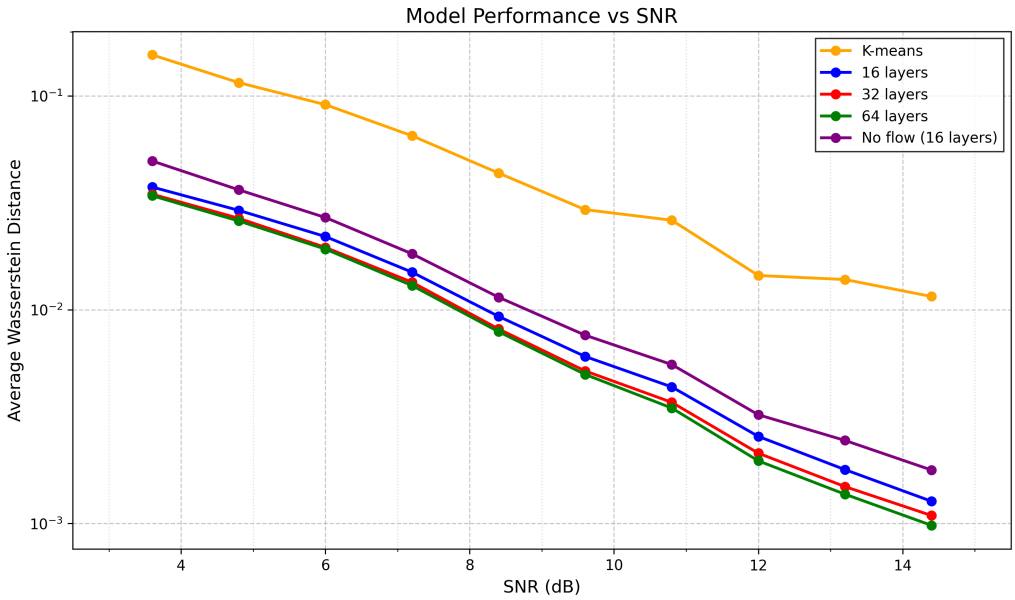

Figure 7. Average model performance measured by Wasserstein distance, comparing transformer models with varying numbers of identical attention layers repeated multiple times under a uniform flow-speed regime and a 16-layer transformer model without flow prediction across SNR levels.

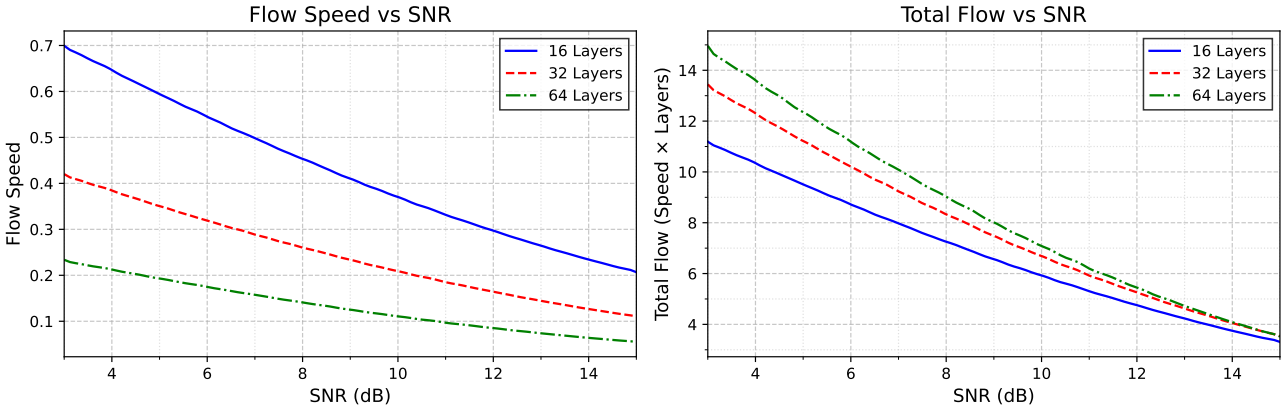

Figure 8. Relationship between learned flow control and SNR-dB, comparing transformer models with 16, 32, and 64 identical attention layers repeated multiple times under a uniform flow-speed regime.

E.3. Comparison of Flow-Speed Regimes

We compared the clustering performance of a 16-layer transformer model across different flow-speed regimes against classical K-means, using previously defined SNR datasets.

Tables 4 and 5 present Wasserstein and log-Wasserstein losses, respectively. Models trained and evaluated on the same flow-speed regime consistently outperform those evaluated on different regimes, highlighting the importance of consistency between training and evaluation. Among these, the Uniform regime notably surpasses the Unit regime, achieving the lowest losses. In contrast, evaluating a model trained with a Uniform flow-speed on an Unit flow-speed yields the poorest performance across all SNR conditions.

Figure 9 further illustrates these results across a continuous range of SNR values. The Uniform → Uniform setting clearly demonstrates superior performance, while Unit flow-speed regimes and cross-regime evaluations lag behind.

	K-means	Uniform → Uniform	Uniform → Unit	Unit → Uniform	Unit → Unit
High SNR	0.0441	0.0016	0.0867	0.0039	0.0029
Moderate SNR	0.0558	0.0080	0.0894	0.0103	0.0120
Low SNR	0.1597	0.0380	0.1095	0.0511	0.0605

Table 4. Comparison of Wasserstein losses for the baseline transformer model with 16 identical attention layers evaluated under different flow-speed regimes. The notation "Training → Evaluation" indicates the flow-speed parameter used during training (left side) and evaluation (right side).

	K-means	Uniform → Uniform	Uniform → Unit	Unit → Uniform	Unit → Unit
High SNR	-4.8941	-6.9963	-2.7754	-5.9572	-6.1945
Moderate SNR	-3.7563	-5.3905	-2.7078	-5.0894	-4.7760
Low SNR	-2.2291	-3.7020	-2.5397	-3.2980	-3.0932

Table 5. Comparison of log-Wasserstein losses for the baseline transformer model with 16 identical attention layers evaluated under different flow-speed regimes. The notation "Training → Evaluation" indicates the flow-speed parameter used during training (left side) and evaluation (right side).

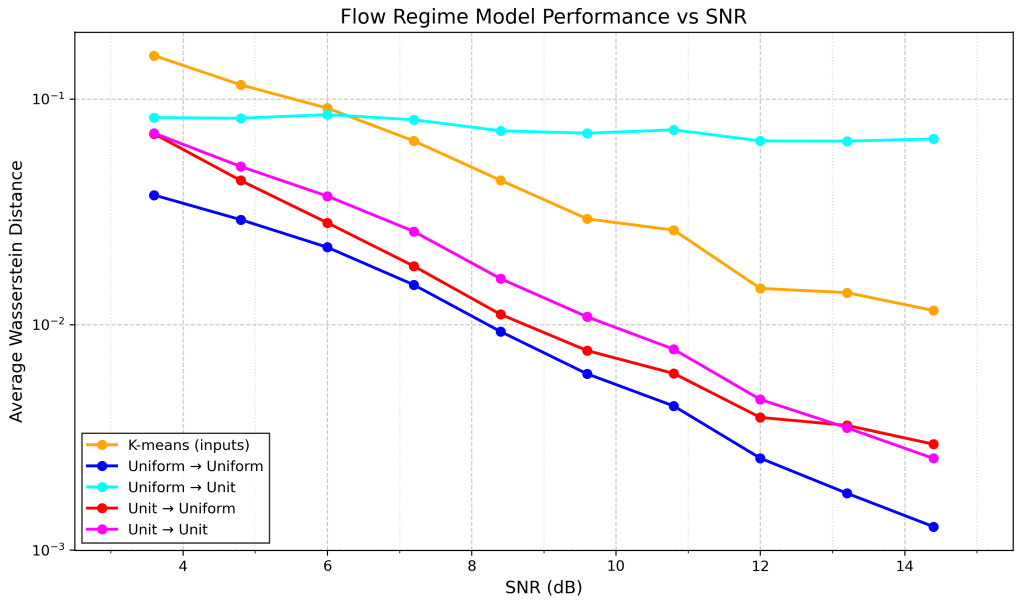

Figure 9. Flow regime model performance vs. SNR. Comparison of average Wasserstein distances across different flow-speed settings.

E.4. Comparison of Identical and Distinct Attention Layers

We compared transformer architectures with identical (ALBERT-style) versus distinct attention layers, using a 16-layer configuration. Tables 6 and 7 summarize the Wasserstein and log-Wasserstein losses, respectively. The improvement gained by using distinct attention layers over identical layers is minor. The adaptive flow and classical K-means are included for reference, showing inferior performance compared to transformer-based models.

Figure 10 visualizes performance across different SNR values. The distinct attention layers slightly outperform identical layers, though the difference remains minimal across the entire range of evaluated SNR values.

	K-means	Identical 16 layers	16 layers (Unit regime)	Distinct 16 layers
High SNR	0.0441	0.0016	0.0029	0.0014
Moderate SNR	0.0558	0.0080	0.0120	0.0071
Low SNR	0.1597	0.0380	0.0605	0.0344

Table 6. Comparison of Wasserstein losses for baseline transformer models with 16 identical attention layers under two different flow-speed regimes (uniform and unit) versus a transformer model with 16 distinct attention layers under a uniform flow-speed regime.

	K-means	Identical 16 layers	16 layers (Unit regime)	Distinct 16 layers
High SNR	-4.8941	-6.9963	-6.1945	-7.0908
Moderate SNR	-3.7563	-5.3905	-4.7760	-5.5600
Low SNR	-2.2291	-3.7020	-3.0932	-3.8187

Table 7. Comparison of log-Wasserstein losses for baseline transformer models with 16 identical attention layers under two different flow-speed regimes (uniform and unit) versus a transformer model with 16 distinct attention layers under a uniform flow-speed regime.

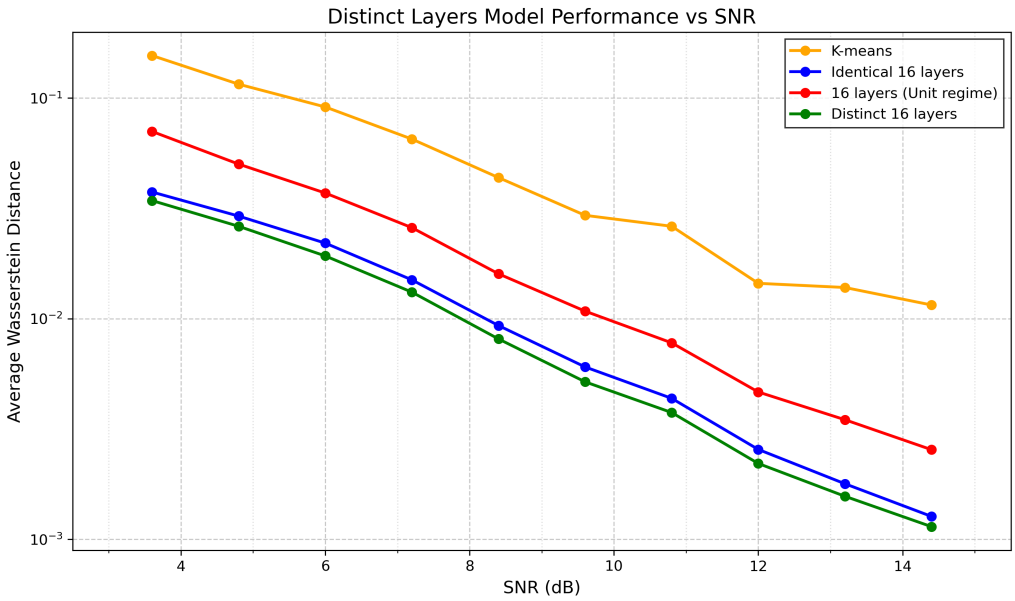

Figure 10. Comparison of transformer models with 16 distinct versus identical attention layers, evaluated under uniform and unit flow-speed regimes. K-means is included as a baseline for reference. Performance is measured by Wasserstein distance across SNR levels.

E.5. Transformer Predictions as Inputs for K-means

We evaluated the effectiveness of directly applying K-means clustering to transformer model predictions, compared with using the input data directly. Performance was assessed across various transformer configurations (16, 32, 64 layers, and a 16-layer model without flow) and compared with classical K-means on raw inputs.

Tables 8 and 9 summarize Wasserstein and log-Wasserstein losses, respectively. Transformer-based methods significantly outperform classical K-means at low SNR, demonstrating their effectiveness in extracting meaningful cluster structures from noisy data. At moderate and high SNR, all transformer-based approaches maintain comparable performance to classical K-means.

Figure 11 illustrates these results, highlighting the marked advantage of transformer predictions over raw inputs in low-SNR scenarios and demonstrating convergence of methods at higher SNR.

	KMeans	KMeans (16 layers)	KMeans (32 layers)	KMeans (64 layers)	KMeans (No Flow)
High SNR	0.0441	0.0634	0.0604	0.0607	0.0631
Moderate SNR	0.0564	0.0584	0.0553	0.0585	0.0605
Low SNR	0.1602	0.0674	0.0652	0.0633	0.0714

Table 8. Comparison of Wasserstein losses obtained by classical K-means applied directly to inputs versus K-means applied to predictions generated by baseline transformer models with 16, 32, or 64 identical attention layers under a uniform flow-speed regime, as well as a baseline transformer model without flow prediction.

E.6. Effect of Varying GMM Sample Size

We evaluated how the performance of the baseline 16-layer transformer model, originally trained on datasets with 1000 GMM samples, varies when the number of GMM samples in the evaluation datasets changes.

Tables 10 and 11 summarize Wasserstein and log-Wasserstein losses for datasets with different SNR conditions and varying sample sizes. As expected, the losses decrease with increasing sample size. Remarkably, the transformer model consistently outperforms classical K-means, even with as few as 100 GMM samples, underscoring its robustness and effectiveness.

	KMeans	KMeans (16 layers)	KMeans (32 layers)	KMeans (64 layers)	KMeans (No Flow)
High SNR	-4.8941	-4.6714	-4.7046	-4.7482	-4.4206
Moderate SNR	-3.7326	-4.1722	-4.2215	-4.1590	-3.9590
Low SNR	-2.2236	-3.2587	-3.3179	-3.3513	-3.1418

Table 9. Comparison of log-Wasserstein losses obtained by classical K-means applied directly to inputs versus K-means applied to predictions generated by baseline transformer models with 16, 32, or 64 identical attention layers under a uniform flow-speed regime, as well as a baseline transformer model without flow prediction.

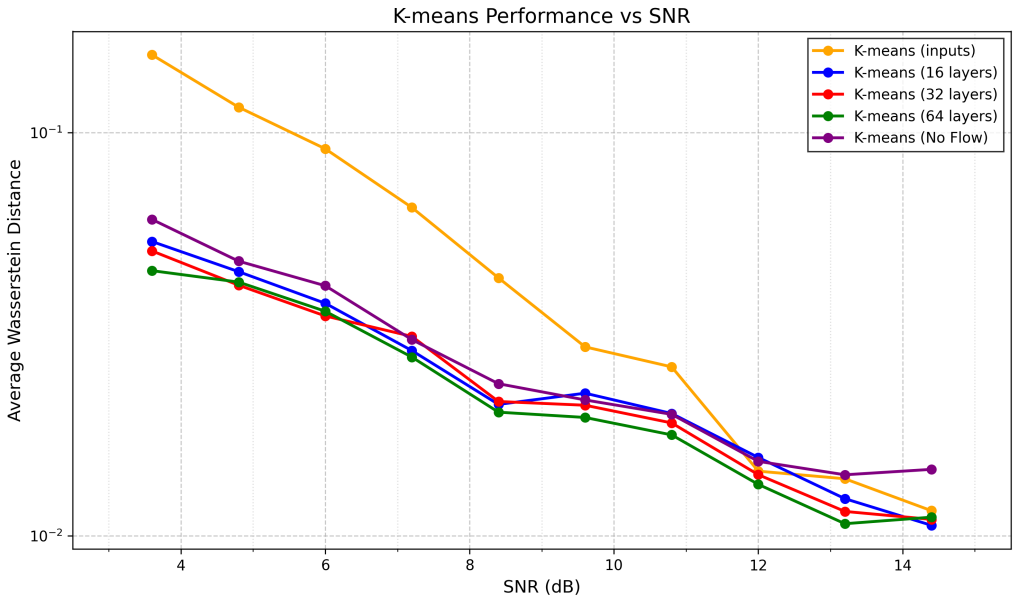

Figure 11. Comparison of K-means clustering performance measured by Wasserstein distance across SNR levels. We apply classical K-means directly to raw input data versus predictions generated by baseline transformer models with 16, 32, or 64 identical attention layers under a uniform flow-speed regime and a baseline transformer model without flow prediction.

Dataset	$n = 100$	$n = 500$	$n = 1000$	$n = 2000$
High SNR	0.0076 / 0.0508	0.0025 / 0.0465	0.0018 / 0.0462	0.0014 / 0.0464
Moderate SNR	0.0351 / 0.0783	0.0140 / 0.0612	0.0094 / 0.0578	0.0069 / 0.0553
Low SNR	0.1248 / 0.2119	0.0553 / 0.1694	0.0394 / 0.1603	0.0302 / 0.1553

Table 10. Comparison of Wasserstein losses for a baseline transformer model with 16 identical attention layers (uniform flow-speed regime) versus classical K-means, evaluated across datasets with varying SNR levels and numbers of GMM samples (n). Each cell lists losses as (Transformer / K-means).

Dataset	$n = 100$	$n = 500$	$n = 1000$	$n = 2000$
High SNR	-5.3273 / -4.1743	-6.4851 / -4.5166	-6.8545 / -4.6274	-7.082 / -4.7578
Moderate SNR	-3.7271 / -3.0604	-4.6974 / -3.3703	-5.1037 / -3.4801	-5.4280 / -3.5404
Low SNR	-2.4590 / -1.9397	-3.3122 / -2.1802	-3.6594 / -2.2411	-3.9352 / -2.2834

Table 11. Comparison of log-Wasserstein losses for a baseline transformer model with 16 identical attention layers (uniform flow-speed regime) versus classical K-means, evaluated across datasets with varying SNR levels and numbers of GMM samples (n). Each cell lists losses as (Transformer / K-means).