GAUSSIAN JOINT EMBEDDINGS FOR SELF-SUPERVISED REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Self-supervised representation learning traditionally relies on deterministic predictions or contrastive objectives, which struggle with multi-modal ambiguities and often require heuristic asymmetries to prevent dimensional collapse. We propose a shift to generative joint modeling via Gaussian Mixture Joint Embeddings (GMJE). By explicitly modeling the joint density of context and target representations, GMJE replaces black-box prediction with closed-form conditional inference, natively providing uncertainty estimation and covariance-aware geometric regularization. We identify a fundamental optimization vulnerability in empirical batch modeling, the *Mahalanobis Trace Trap*, and resolve it through a suite of robust parametric, topological, and adaptive architectures. Further, we show that standard contrastive learning is a degenerate, non-parametric special case of GMJE, motivating our introduction of a dynamic Sequential Monte Carlo (SMC) memory bank to replace standard uniform FIFO queues. Empirical evaluations confirm that GMJE recovers complex multi-modal structures, achieves competitive discriminative performance, and constructs a continuous latent manifold capable of high-fidelity unconditional generative sampling.

1 INTRODUCTION

Joint-embedding architectures have become a leading alternative to reconstruction-based self-supervised learning (SSL): they encode views of an input into a latent space in which semantic dependencies can be captured and predicted (Assran et al., 2023; LeCun, 2022; Assel et al., 2025). *Contrastive* methods such as InfoNCE, SimCLR, and MoCo attract positive pairs and repel negatives (van den Oord et al., 2019; Chen et al., 2020; He et al., 2020), balancing alignment and uniformity on the hypersphere (Wang & Isola, 2022), but typically need many negatives, via large batches or memory queues (Chen et al., 2020; He et al., 2020). *Non-contrastive predictive* methods such as BYOL and JEPA instead predict one representation from another (Grill et al., 2020; Assran et al., 2023). Two issues arise. First, when a context is compatible with several targets (as in masking or partial observation), the Bayes-optimal squared-loss predictor is the conditional mean, which can fall between modes in regions unsupported by data (Bishop, 1994; 2006b). Second, regression losses impose no global dispersion constraint, so collapse is avoided with predictor asymmetry, stop-gradients, EMA targets, or variance–covariance penalties (Grill et al., 2020; Chen & He, 2021; Tian et al., 2021; Bardes et al., 2022; Halvagal et al., 2023).

We propose to replace deterministic latent prediction with *generative joint modeling*. **Gaussian Joint Embeddings (GJE)** model the joint density $p(z_c, z_t)$ of context and target embeddings, so that $-\log p(z_c, z_t) = -\log p(z_t \mid z_c) - \log p(z_c)$ splits into a conditional matching term and a marginal term whose log-determinant provides covariance-dependent volume regularization. Because a single Gaussian cannot represent multi-modal structure, we introduce **Gaussian Mixture Joint Embeddings (GMJE)**. We then show that naive optimization with empirical batch covariances falls into a *Mahalanobis Trace Trap*, in which the data-fit term becomes a constant and the cross-view attractive signal vanishes, and we develop architectures that avoid it. Our contributions are:

- **A probabilistic joint-embedding paradigm.** GJE yields closed-form conditional predictors with uncertainty estimates and built-in variance–covariance control, in both a dual

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

(sample-space, Gaussian-process) and a primal (feature-space) form, which we show coincide for a linear kernel (Section 3).

- **GMJE and remedies for the Trace Trap.** We extend GJE to joint Gaussian mixtures, whose conditional is a closed-form, context-routed mixture, and propose three trap-free instantiations: learnable global prototypes (GMJE-Proto), a context-only Mixture Density Network (GMJE-MDN), and topology-adaptive Growing Neural Gas (GMJE-GNG) (Section 4).
- **The InfoNCE bridge and SMC memory banks.** We show that InfoNCE is exactly the conditional likelihood of a degenerate, non-parametric, isotropic GMJE, which reveals the FIFO queue as a uniform prior over mixture components; we replace it with a Sequential Monte Carlo (SMC) particle bank that reweights and resamples negatives (Sections 4.1 and 4.2).
- **Empirical evidence.** On synthetic multi-valued alignment, GMJE recovers the conditional branches that deterministic and unimodal models average over; on CIFAR-10, the SMC bank improves over FIFO under a tight memory budget; on MNIST, GMJE defines a latent density that supports unconditional sampling better than post-hoc or unimodal alternatives (Section 5).

2 Related Work

Contrastive and non-contrastive SSL. Contrastive predictive coding introduced InfoNCE (van den Oord et al., 2019); MoCo decoupled dictionary size from batch size with a momentum encoder and a FIFO queue (He et al., 2020); SimCLR relies on strong augmentation and large in-batch negative sets (Chen et al., 2020). Hard-negative mining (Robinson et al., 2021; Kalantidis et al., 2020) and debiasing (Chuang et al., 2020) address class collision heuristically. Clustering methods such as DeepCluster and SwAV use prototypes (Caron et al., 2019; 2021a). BYOL, SimSiam, and DINO avoid negatives through asymmetry (Grill et al., 2020; Chen & He, 2021; Caron et al., 2021b), whose role has been analyzed theoretically (Tian et al., 2021; Halvagal et al., 2023; Zhuo et al., 2023); VICReg regularizes feature variance and covariance symmetrically (Bardes et al., 2022). GMJE instead derives both alignment and dispersion from a single joint likelihood.

Deterministic and probabilistic JEPA. JEPA predicts target representations in latent space (LeCun, 2022; Assran et al., 2023) and has been extended to video (Bardes et al., 2024), other modalities, and world models (Terver et al., 2026). Its predictor is deterministic and trained with a regression loss. Probabilistic (Huang, 2026b), bidirectional (Huang, 2026a), and rule-informed (Huang & Raza) variants have been proposed; a full joint-density view of JEPA-style learning is what GJE/GMJE provide.

Probabilistic building blocks. We build on Gaussian processes (Rasmussen & Williams, 2005) and random Fourier features (Rahimi & Recht, 2007), mixture density networks for multi-valued inverse problems (Bishop, 1994), universal approximation by Gaussian mixtures (Li & Barron, 1999; Norets, 2010), Growing Neural Gas (Fritzke, 1994), and Sequential Monte Carlo (Doucet et al., 2001; Chopin & Papaspiliopoulos, 2020). An extended review is given in Section A.

3 Gaussian Joint Embeddings

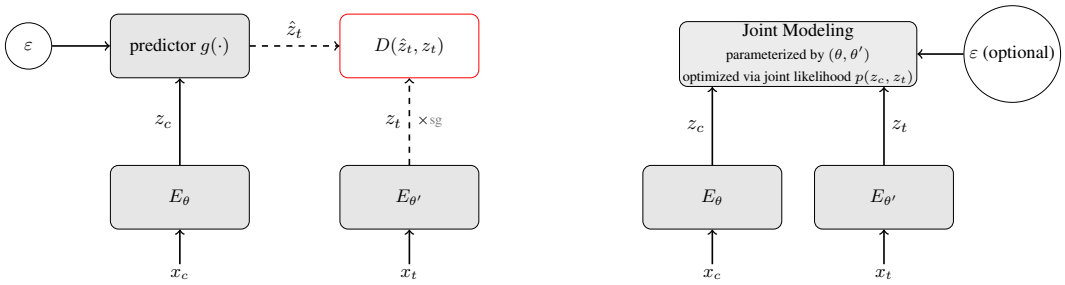
Context and target views are encoded as $z_c = E_\theta(x_c) \in \mathbb{R}^{d_c}$ and $z_t = E_{\theta'}(x_t) \in \mathbb{R}^{d_t}$ (Fig. 1). GJE maximizes the joint log-likelihood of matched pairs,

$$\max_{\theta,\theta'} \mathbb{E}_{(x_c,x_t)\sim\mathcal{D}}\big[\log p\big(E_\theta(x_c), E_{\theta'}(x_t)\big)\big], \qquad \log p(z_c, z_t) = \log p(z_t \mid z_c) + \log p(z_c), \quad (1)$$

under a Gaussian assumption, which can be instantiated over the N samples of a batch (dual) or over the $d = d_c + d_t$ latent features (primal). We write N for batch size, M for memory-bank size, and K for the number of mixture components.

Dual (sample-space) GJE. Replacing the JEPA predictor with a Gaussian process $z_t = f(z_c) + \epsilon$, $f \sim \mathcal{GP}(0, k)$, and treating the d_t target channels as independent GPs sharing the context Gram matrix K_{cc} gives the GP marginal likelihood (Rasmussen & Williams, 2005)

$$\mathcal{L}_{\text{Dual-GJE}} = \tfrac{1}{2}\operatorname{Tr}\big(Z_t^\top K_{cc}^{-1} Z_t\big) + \tfrac{d_t}{2}\log|K_{cc}|, \quad (2)$$



(a) Classic JEPA (b) GJE / GMJE

Figure 1: (a) Classic JEPA encodes two views and aligns them by deterministic latent prediction with a distance loss (red) and a stop-gradient (sg). (b) GJE/GMJE model the joint density of context and target embeddings and derive predictions by probabilistic conditioning. ε denotes optional side information (e.g., actions or noise).

with the non-parametric predictor $\mu_{t|c} = k_{*c}K_{cc}^{-1}Z_t$ and a distance-aware variance $\Sigma_{t|c} = k_{**} - k_{*c}K_{cc}^{-1}k_{c*}$. Equation (2) is a *conditional* likelihood: it regularizes only the context volume, so symmetric optimization collapses ($Z_t \to \mathbf{0}$), and it must be trained with an EMA target and a stop-gradient (Section H); exact inference also costs $\mathcal{O}(N^3)$, which random Fourier features reduce (Section I).

Primal (feature-space) GJE. Modelling the concatenated embedding $z = [z_c^\top, z_t^\top]^\top$ as $\mathcal{N}(\mathbf{0}, C_{\text{joint}})$, with blocks $C_{cc}, C_{ct}, C_{tc}, C_{tt}$, gives the joint NLL

$$\mathcal{L}_{\text{Primal-GJE}} = \tfrac{1}{2N}\sum_{i=1}^{N} z_i^\top C_{\text{joint}}^{-1} z_i + \tfrac{1}{2}\log|C_{\text{joint}}|, \tag{3}$$

and the closed-form linear predictor

$$\mu_{t|c} = C_{tc}C_{cc}^{-1}z_c^*, \qquad \Sigma_{t|c} = C_{tt} - C_{tc}C_{cc}^{-1}C_{ct}. \tag{4}$$

Because it models the *joint* likelihood, primal GJE has symmetric regularizers for both views and costs only $\mathcal{O}(d^3)$. For a linear kernel, dual and primal GJE coincide exactly (Section K).

The Mahalanobis Trace Trap. If C_{joint} is the empirical batch covariance $\frac{1}{N}Z^\top Z$, the data-fit term in Eq. (3) cancels identically:

$$\tfrac{1}{2N}\textstyle\sum_i z_i^\top C_{\text{joint}}^{-1} z_i = \tfrac{1}{2}\operatorname{Tr}\Big(C_{\text{joint}}^{-1}\,\tfrac{1}{N}\textstyle\sum_i z_i z_i^\top\Big) = \tfrac{1}{2}\operatorname{Tr}(I_d) = \tfrac{d}{2}. \tag{5}$$

The expansive, cross-view Mahalanobis gradient is therefore exactly zero, and the optimizer is left with $+\frac{1}{2}\log|C_{\text{joint}}|$, which it minimizes by shrinking all embeddings towards the origin (dimensional collapse). The same cancellation occurs for mixtures with batch-estimated component covariances (Section N). Remedies must therefore decouple the covariance used in the data-fit term from the current batch, e.g., via learnable or EMA-tracked parameters, conditional factorization, or a memory of particles, or must invert the regularizer to *maximize* entropy (Bardes et al., 2022).

Unimodality and homoscedasticity. Both GJE forms induce a single Gaussian conditional, so when a context admits several targets the predictive mean averages the modes (Bishop, 1994); primal GJE additionally has a constant predictive covariance independent of z_c^*. These limitations motivate GMJE.

4 Gaussian Mixture Joint Embeddings

Gaussian mixtures approximate any sufficiently smooth density arbitrarily well in L^1 (Li & Barron, 1999; Norets, 2010), even with isotropic components (McLachlan & Basford, 1988; Bishop, 1994). GMJE models the joint embedding as a K-component mixture,

$$p(z_c, z_t) = \textstyle\sum_{k=1}^{K} \pi_k \mathcal{N}\left(\begin{bmatrix} z_c \\ z_t \end{bmatrix} \middle| \begin{bmatrix} \mu_{c,k} \\ \mu_{t,k} \end{bmatrix}, \begin{bmatrix} \Sigma_{cc,k} & \Sigma_{ct,k} \\ \Sigma_{tc,k} & \Sigma_{tt,k} \end{bmatrix}\right). \tag{6}$$

Both marginals remain mixtures, e.g., $p(z_t) = \sum_k \pi_k \mathcal{N}(z_t \mid \mu_{t,k}, \Sigma_{tt,k})$, and the conditional is a closed-form mixture (Bishop, 2006b),

$$p(z_t \mid z_c) = \sum_k \gamma_k(z_c)\, \mathcal{N}(z_t \mid \mu_{t|c,k}, \Sigma_{t|c,k}), \qquad \gamma_k(z_c) = \frac{\pi_k \mathcal{N}(z_c \mid \mu_{c,k}, \Sigma_{cc,k})}{\sum_j \pi_j \mathcal{N}(z_c \mid \mu_{c,j}, \Sigma_{cc,j})}, \tag{7}$$

where each component predicts as in Eq. (4). The context acts as a *router*: it selects which component predicts the target, so GMJE can place conditional mass on several distinct branches. Unlike a conditional mixture model of $p(z_t \mid z_c)$ alone, the joint formulation keeps the marginal $p(z_c)$ and its geometric regularization, so both encoders can be trained symmetrically (architecture in Fig. 6, Section C). We now give trap-free ways to learn the components.

GMJE-Proto: learnable global prototypes. We learn K joint prototypes $\mu_k = [\mu_{c,k}^\top, \mu_{t,k}^\top]^\top$ (akin to SwAV/DeepCluster prototypes (Caron et al., 2021a; 2019)) and a shared covariance Σ as parameters, and minimize the NLL by a differentiable, relaxed EM computed with the log-sum-exp trick:

$$\mathcal{L}_{\text{GMJE-Proto}} = -\log \sum_{k=1}^{K} \exp\Big(\log \pi_k - \tfrac{1}{2}(Z-\mu_k)^\top \Sigma^{-1}(Z-\mu_k) - \tfrac{1}{2}\log|\Sigma|\Big). \tag{8}$$

Because Σ is a parameter rather than the batch covariance, the trap does not arise. Gradients of the LSE are the responsibilities $\gamma_k(Z)$, which pull each embedding and its winning prototype together (soft clustering), while the $\frac{1}{2}\log|\Sigma|$ penalty compresses component volume; to fit a diverse dataset without inflating Σ, the prototypes must spread across the latent space. Local compression thus transfers the dataset's variance onto globally spread prototypes, without a separate uniformity loss. Component-specific Σ_k are a direct extension (Section C).

GMJE-MDN: conditional mixtures with an information bottleneck. Fixed prototypes do not adapt to per-instance ambiguity. Mixture density networks (Bishop, 1994) predict input-dependent mixture parameters, but a parameter network that sees the full joint embedding Z learns the identity $\mu_k(Z) = Z$ with vanishing variance (*identity collapse*, Section L). We therefore let the parameter network observe *only* z_c and factorize the joint NLL:

$$\mathcal{L}_{\text{GMJE-MDN}} = \underbrace{\tfrac{1}{2} z_c^\top \Sigma_c^{-1} z_c + \tfrac{1}{2}\log|\Sigma_c|}_{-\log p(z_c)\text{: marginal regularization}} \underbrace{-\log \textstyle\sum_k \pi_k(z_c)\, \mathcal{N}\big(z_t \mid \mu_k(z_c), \Sigma_k(z_c)\big)}_{-\log p(z_t|z_c)\text{: conditional MDN}}, \tag{9}$$

where Σ_c is a global (e.g., EMA-tracked) context covariance, which avoids the trap because it lags the current batch. The MDN term provides heteroscedastic, multi-modal routing, while the marginal term penalizes both volume inflation and degenerate shrinkage, keeping the context space full-rank. Encoders and parameter network are trained jointly without stop-gradients (Fig. 7).

GMJE-GNG: discovering K and topology. To avoid fixing K and to reduce sensitivity to initialization, GMJE-GNG maintains the prototypes as nodes of a Growing Neural Gas graph (Fritzke, 1994): for each joint embedding, the nearest prototype and its graph neighbours move towards it; accumulated quantization error triggers insertion of a new prototype between the highest-error node and its worst neighbour; and aged edges are pruned so that disjoint clusters separate. The cost is $\mathcal{O}(K_{\max} \cdot 2d)$ per step, negligible relative to an $\mathcal{O}(Md)$ memory bank (Algorithm 2, Section C).

4.1 Contrastive learning as non-parametric GMJE

Instead of K learned components, treat every pair in a memory bank of size M as an equiprobable mode with isotropic, block-diagonal covariance $\Sigma = \mathrm{diag}(\tau I, \tau I)$, i.e., a kernel density estimate $p(z_c, z_t) = \frac{1}{M}\sum_m \mathcal{N}(z_c \mid z_c^{(m)}, \tau I)\, \mathcal{N}(z_t \mid z_t^{(m)}, \tau I)$.

Proposition 1 (InfoNCE as degenerate GMJE)**.** *Assume L^2-normalized embeddings, that each stored context prototype is represented by its paired target key ($z_c^{(m)} \approx z_t^{(m)}$), and the low-temperature limit $\tau \to 0$ in which only the matching mode m^* contributes to the numerator. Then the conditional NLL of a positive pair under this non-parametric GMJE is*

$$-\log p(z_t \mid z_c) = -\log \frac{\exp(z_c^\top z_t/\tau)}{\sum_{m=1}^{M} \exp(z_c^\top z_t^{(m)}/\tau)} + C, \tag{10}$$

which is the InfoNCE loss (van den Oord et al., 2019) up to an additive constant C.

Algorithm 1 SMC-GMJE: Dynamic Memory Bank Optimization via SMC

Require: Stream of query batches $Z_{query} = \{(z_c, z_t)\}_{b=1}^{B}$ generated by encoders, memory bank size M, temperature τ.

Initialize memory bank $\mathcal{M}$ with M random target embeddings and uniform weights $W \leftarrow 1/M$.

for each training step t with incoming batch Z_{query} **do**

  // Loop A: Encoders Optimization (Requires Gradients)

  Forward Pass: Compute SMC-weighted InfoNCE loss $\mathcal{L}$ (Eq.47) using the current memory bank $\mathcal{M}$ and its weights W.

  Backward Pass: Compute gradients $\nabla\mathcal{L}$ and update both encoder networks' parameters via gradient descent.

  // Loop B: SMC Memory Bank Optimization (Gradient-Free)

  Combine: Pool $\mathcal{M}_{pool} \leftarrow \mathcal{M} \cup \{z_t \in Z_{query}\}$, size $M + B$.

  Prior Weights: $W_{prior}^{(m)} \leftarrow \frac{1}{M+B}$ for new batch, scale old weights by $\frac{M}{M+B}$.

  Importance Update: For each particle $m \in \{1 \dots M + B\}$:

    $\tilde{W}_t^{(m)} \leftarrow W_{prior}^{(m)} \times \frac{1}{B} \sum_{z_c \in Z_{query}} \exp(z_c^T z_t^{(m)}/\tau)$

  Normalize: $W_{pool}^{(m)} \leftarrow \tilde{W}_t^{(m)} / \sum_{j=1}^{M+B} \tilde{W}_t^{(j)}$

  // Execute Sequential Importance Resampling

  Sample M particles from $\mathcal{M}_{pool}$ with replacement, proportional to $W_{pool}^{(m)}$.

  Replace $\mathcal{M}$ with these sampled particles.

  Reset all memory bank weights to uniform: $W \leftarrow 1/M$.

end for

Algorithm Note: as we execute Sequential Importance Resampling (SIR) at every step, the probability mass is perfectly transferred into the physical multiplicity (cloning) of the particles, allowing the memory bank weights to be reset to $1/M$. Consequently, during the forward pass, standard unweighted InfoNCE (Cross-Entropy) over the resampled bank is mathematically exact and strictly equivalent to evaluating the explicit SMC-weighted loss in Eq.47.

The derivation (Section D) uses Bayes' rule on the KDE, with $\|z_c - z_c^{(m)}\|^2 = 2 - 2z_c^\top z_c^{(m)}$ under normalization. The bridge makes explicit two restrictions of contrastive learning: (i) the cross-covariance block is fixed to zero, so views are coupled only by forcing $z_c \approx z_t$ (invariance learning), whereas GMJE's Σ_{ct} can model cross-view transformations; and (ii) every instance is its own mode, which causes class collision (Chuang et al., 2020) and requires many modes M to estimate the denominator, whereas parametric GMJE uses $K \ll M$ semantic prototypes.

4.2 SMC MEMORY BANKS

Under Proposition 1, the memory bank *is* a non-parametric mixture approximating $p(z_c)$, and a FIFO queue (He et al., 2020) imposes a uniform prior $\pi_m = 1/M$ and discards keys by age rather than informativeness. We instead treat the bank as a weighted particle system $\{(Z^{(m)}, W^{(m)})\}$ updated by Sequential Monte Carlo (Doucet et al., 2001; Chopin & Papaspiliopoulos, 2020), which turns Eq. (10) into an SMC-weighted InfoNCE loss with denominator $\sum_m W^{(m)} \exp(z_c^\top z_t^{(m)}/\tau)$. At each step, (1) the bank and the incoming batch are pooled; (2) each particle's weight is updated by its average likelihood under the current queries,

$$\widetilde{W}_t^{(m)} = W_{t-1}^{(m)} \, \mathbb{E}_{z_c \in Z_{\text{query}}} \big[\exp(z_c^\top z_t^{(m)}/\tau)\big], \tag{11}$$

so that hard negatives near the current manifold gain weight; and (3) M particles are resampled in proportion to the normalized weights, after which the weights are reset to $1/M$ (making unweighted InfoNCE on the resampled bank exact). The update is gradient-free, reuses dot products from the forward pass, and keeps the per-step cost at $\mathcal{O}(BMd)$, as in MoCo; the effective sample size $\text{ESS} = 1/\sum_m (W^{(m)})^2$ monitors concentration. SMC-GMJE is thus a drop-in replacement for the FIFO queue that performs principled hard-negative mining (Algorithm 1); it generalizes to full-covariance particles (Section M).

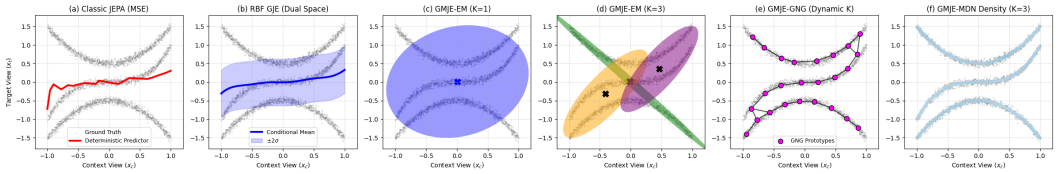


Figure 2: Predictive distributions on Dataset A (ground truth in grey). (a) Classic JEPA collapses to the conditional mean between branches. (b) The dual RBF baseline has a flexible mean but a unimodal predictive band. (c) GMJE-EM ($K = 1$) fits one global ellipse. (d) GMJE-EM ($K = 3$) separates the mass into rigid ellipses. (e) GMJE-GNG places prototypes along the non-linear ridges. (f) GMJE-MDN most closely matches the ground-truth conditional density.

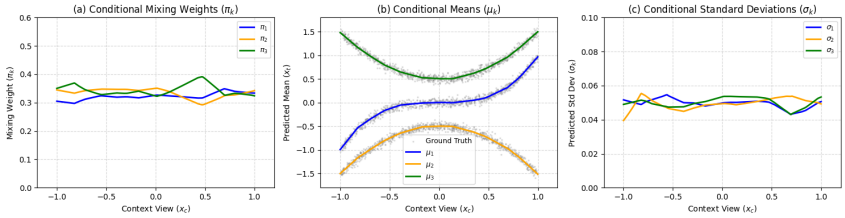


Figure 3: Internal parameter routing of GMJE-MDN on Dataset A. (a) The conditional mixing weights remain close to the ground-truth uniform value of $1/3$. (b) The learned conditional means align closely with the three underlying branch functions. (c) The predicted conditional standard deviations remain near the true Gaussian observation noise level $\epsilon = 0.05$.

4.3 Generative sampling from the latent density

Because GMJE optimizes a normalized density, its target marginal $p(z_t) = \sum_k \pi_k \mathcal{N}(\mu_{t,k}, \Sigma_{tt,k})$ can be sampled without data: draw $k \sim \mathrm{Cat}(\pi)$ and $z_t = \mu_{t,k} + A_k \epsilon$ with $\Sigma_{tt,k} = A_k A_k^\top$ and $\epsilon \sim \mathcal{N}(0, I)$ (Huang, 2025a). Discriminative SSL encoders define no such density, so sampling from them requires a post-hoc model that may place mass in low-density gaps.

5 Experiments

We evaluate (i) recovery of multi-modal conditional structure on synthetic data with known ground truth, (ii) representation quality on CIFAR-10, and (iii) unconditional latent sampling on MNIST. Full settings are given in Section O; additional results are in Section F.

5.1 Ambiguous alignment of synthetic representations

To isolate the conditional modeling problem, we bypass the encoders and treat $x_c \sim \mathcal{U}(-1, 1)$ and $x_t = f(x_c) + \epsilon$, $\epsilon \sim \mathcal{N}(0, 0.05^2)$, as observations, where the branch f is chosen uniformly per sample, in the spirit of Bishop's multi-valued inverse problems (Bishop, 1994). Dataset A has separated branches $\{x_c^2 + 0.5,\ -x_c^2 - 0.5,\ x_c^3\}$, and Dataset B has branches $\{\sin 3x_c,\ -\sin 3x_c,\ 0\}$ that intersect at the origin ($N = 3000$ training pairs each). We compare classic JEPA (MSE), a dual RBF-kernel GJE (GP), GMJE-EM with $K = 1$ and $K = 3$, GMJE-GNG, and GMJE-MDN.

Results. As predicted by squared-loss theory, classic JEPA regresses to the conditional mean and places predictions in the empty regions between branches (Fig. 2a) or near zero on Dataset B. The dual RBF model interpolates smoothly but, being unimodal, covers all branches with one broad band (b); GMJE-EM with $K = 1$ over-smooths the manifold with a single ellipse (c), and $K = 3$ separates modes but only coarsely, owing to rigid global covariances (d). GMJE-GNG tracks the branch topology with adaptively placed prototypes (e), and GMJE-MDN recovers the conditional density most faithfully (f). Its internal parameters confirm this: the conditional means follow the true branches, the predicted standard deviations stay near the true noise level 0.05, and the mixing

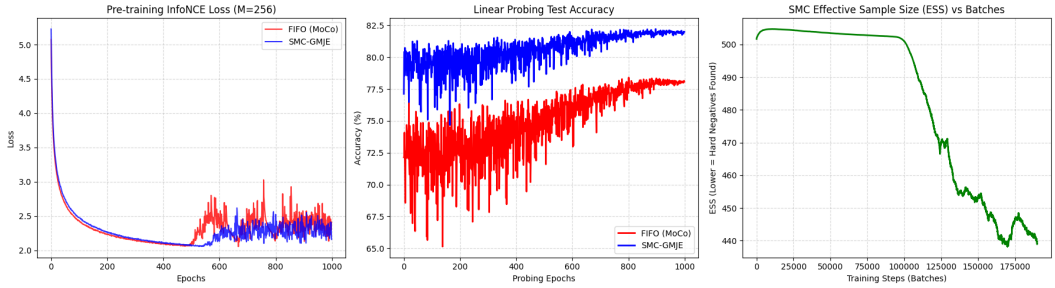


Figure 4: SMC particle bank (SMC-GMJE) versus a FIFO queue (MoCo v2) with memory $M = 256$ over 1000 epochs on CIFAR-10. *Left:* pre-training InfoNCE loss; the FIFO queue becomes volatile from about epoch 500. *Centre:* linear-probe accuracy. *Right:* ESS of the SMC bank, which drops sharply around epoch 500 as mass concentrates on informative particles.

Table 1: Linear-probe top-1 accuracy on CIFAR-10 (ResNet-18, 200 pre-training epochs, 100 probing epochs). These short-horizon results are relative comparisons under limited compute, not fully tuned accuracies.

Method	Architecture type	Accuracy
SimCLR (Chen et al., 2020)	Symmetric, large-batch contrastive	81.50%
BYOL (Grill et al., 2020)	Asymmetric EMA + predictor	79.04%
MoCo v2 (He et al., 2020)	Asymmetric EMA + FIFO queue	79.02%
SMC-GMJE (ours)	Asymmetric EMA + SMC particle bank	74.52%

weights stay near the true value $1/3$ (Fig. 3). The same ordering holds on the intersecting Dataset B (Figs. 8 and 9).

5.2 Representation learning on CIFAR-10

We use a CIFAR-10-adapted ResNet-18, SimCLR augmentations, and linear probing on frozen features (Chen et al., 2020; He et al., 2020). SMC-GMJE (the non-parametric variant of Section 4.2) shares MoCo v2's encoder, augmentations, and EMA momentum, and differs only in the memory-bank update.

SMC versus FIFO under a tight memory budget. With $M = 256$ and 1000 pre-training epochs (Fig. 4), the two banks behave similarly at first, but from about epoch 500 the FIFO baseline becomes unstable, while SMC-GMJE remains stable and reaches a final linear-probe accuracy of $\mathbf{81.99}\%$ versus 78.08% for FIFO ($+3.91$ points). The ESS drops at the same time, indicating that the filter concentrates mass on a smaller set of informative negatives and clones them past their FIFO expiry. A plausible explanation is that, once semantic clusters form, a 256-slot FIFO queue can no longer retain informative negatives for all classes, whereas SMC keeps them.

Comparison with standard baselines. Under a common 200-epoch budget (Table 1), SimCLR, BYOL, and MoCo v2 reach 79–82%, while SMC-GMJE reaches 74.52%. SMC-GMJE thus trains stably and learns useful features, but in this short, untuned regime it does not yet match the mature baselines. The parametric GMJE variants were harder to optimize from scratch in this setting and are not reported (Section O.3). Taken together, the results show that the SMC bank is most beneficial when memory is scarce, and that further tuning is needed before GMJE matches established SSL systems at scale.

5.3 Unconditional generation by latent sampling

On MNIST (padded to 32×32), we train three encoder families from scratch, namely SimCLR, unimodal GJE, and parametric GMJE ($K = 50$); we then freeze each encoder, train a convolutional decoder $D(z)$ to invert its embeddings, discard the data, and decode latent samples drawn from (a) a

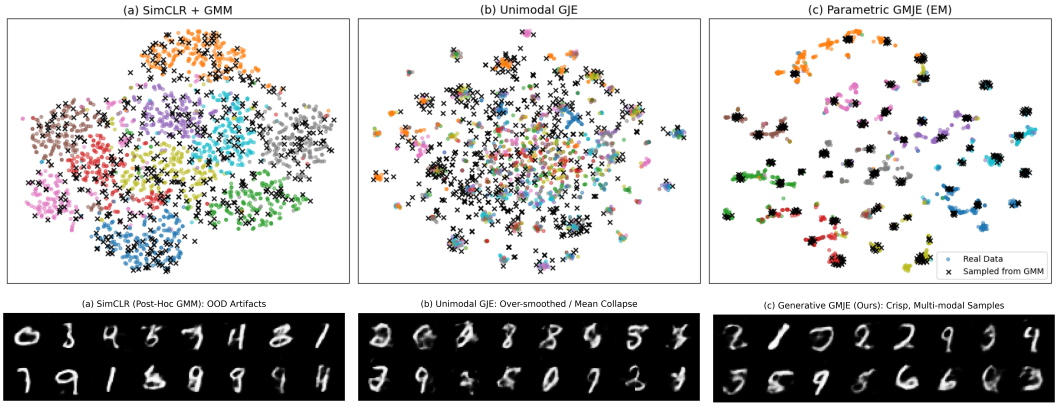


Figure 5: Unconditional latent sampling on MNIST. *Top:* t-SNE of real test embeddings (coloured by class) and synthetic latent samples (black crosses) for (a) SimCLR with a post-hoc GMM, (b) unimodal GJE, and (c) parametric GMJE. *Bottom:* decoded samples. The post-hoc GMM places mass between clusters and yields ambiguous digits; unimodal GJE yields over-smoothed samples with reduced diversity; GMJE samples align with semantic clusters and decode into sharper, more diverse digits.

Table 2: Geometry of the learned latent densities on MNIST ($d = 128$): mean component log-volume, mean component variance, and prototype spread (L^2).

Model	Avg. $\log \lvert\Sigma_k\rvert$	Avg. $\mathrm{Tr}(\Sigma_k)$	Prototype spread
SimCLR (post-hoc GMM)	-878.99	0.85	0.55
Unimodal GJE ($K = 1$)	-636.15	0.99	N/A
Parametric GMJE	$\mathbf{-621.58}$	**1.00**	**1.43**

post-hoc 50-component GMM fitted to SimCLR embeddings, (b) a single Gaussian fitted to GJE embeddings, and (c) GMJE's own learned mixture, projecting samples to the hypersphere.

Results. The post-hoc GMM on SimCLR places substantial mass in inter-cluster gaps (Fig. 5a), and these off-manifold samples decode into blurry, class-ambiguous digits, consistent with its small prototype spread and component volume (Table 2). Unimodal GJE preserves global variance but, with one component, cannot represent the ten-class structure, so its samples are dispersed and decode into over-smoothed averages. Parametric GMJE attains the largest prototype spread with non-degenerate component volume, its samples fall inside the semantic clusters, and it produces the sharpest and most diverse digits. Thus a latent space that is good for discrimination is not automatically a good sampling distribution; learning the multi-modal density during representation learning closes this gap.

6 Discussion and Limitations

A unifying view. GMJE connects the contrastive paradigm, a non-parametric isotropic special case whose FIFO queue we replace with an SMC particle bank, with the predictive paradigm, whose deterministic predictor we replace with closed-form or MDN-based conditional mixtures. In the parametric variants, a covariance-aware objective (Mahalanobis data fit against a log-determinant volume penalty) replaces collapse heuristics, so gradients can flow symmetrically through both encoders; in the dual GJE and in SMC-GMJE, which inherits MoCo's architecture, an EMA target is still used. GMJE-GNG and the non-parametric variant relax the need to choose K in advance.

Limitations. The InfoNCE correspondence holds under the stated normalization, proxy, and low-temperature assumptions. Our vision experiments are small-scale (ResNet-18 on CIFAR-10); full-covariance components scale as $\mathcal{O}(d^3)$ per component and may need regularization to avoid singular

Σ_k. Scaling GMJE to larger backbones and datasets, and extending it to sequential settings such as video and language are important next steps.

7 Conclusion

We introduced Gaussian Joint Embeddings and their multi-modal extension GMJE, which recast self-supervised learning as optimization of a joint density over context and target representations. This yields closed-form conditional inference with uncertainty and covariance-aware regularization. We identified the Mahalanobis Trace Trap of empirical batch covariances and proposed trap-free prototype, MDN, and GNG instantiations; we showed that InfoNCE is a degenerate non-parametric case of GMJE and used this view to replace FIFO queues with SMC particle banks. Experiments show that GMJE recovers multi-modal conditional structure, that SMC banks help under tight memory budgets, and that GMJE's latent density supports unconditional sampling. This generative joint-modeling view helps bridge representation learning and probabilistic density estimation.

AI use statement

AI tools were used to aid and polish the writing of this paper. In addition, parts of the analysis and experimental Python code were developed with the assistance of AI. We have reviewed all AI-assisted text and code, and take full responsibility for the final content of this work.

Ethics statement

This work proposes a probabilistic framework for self-supervised representation learning and evaluates it on synthetic data and on the public CIFAR-10 and MNIST benchmarks; no human subjects or personal data are involved. As with other representation-learning and generative-modeling methods, learned representations or sampled data could inherit biases present in training data; we do not foresee risks specific to this work beyond those of the broader field.

Reproducibility statement

Derivations of the key results are provided in the appendix: joint versus conditional objectives and collapse analyses (Sections G and H), the linear-kernel primal–dual equivalence (Section K), the InfoNCE bridge (Section D), and the Mahalanobis Trace Trap (Section N). Training algorithms are given in Algorithms 1 and 2 and Sections I, J and M. Hardware, data generation, architectures, and all hyperparameters for the three experimental groups are given in Section O. Code will be released upon publication.

References

Mohamed Abdelfattah and Alexandre Alahi. S-jepa: A joint embedding predictive architecture for skeletal action recognition. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XXXII*, pp. 367–384, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73410-6. doi: 10.1007/978-3-031-73411-3_21. URL https://doi.org/10.1007/978-3-031-73411-3_21.

Hugues Van Assel, Mark Ibrahim, Tommaso Biancalani, Aviv Regev, and Randall Balestriero. Joint embedding vs reconstruction: Provable benefits of latent space prediction for self supervised learning, 2025. URL https://arxiv.org/abs/2505.12477.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023. URL https://arxiv.org/abs/2301.08243.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong

Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-jepa 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2019. Curran Associates Inc.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning, 2022. URL https://arxiv.org/abs/2105.04906.

Adrien Bardes, Jean Ponce, and Yann LeCun. Mc-jepa: A joint-embedding predictive architecture for self-supervised learning of motion and content features, 2023. URL https://arxiv.org/abs/2307.12698.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-JEPA: Latent video prediction for visual representation learning, 2024. URL https://openreview.net/forum?id=WFYbBOEOtv.

Christopher M. Bishop. Mixture density networks. Technical Report NCRG/94/004, Neural Computing Research Group, Aston University, February 1994. URL http://www.ncrg.aston.ac.uk/.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, Berlin, Heidelberg, 2006a. ISBN 978-0-387-31073-2.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, Berlin, Heidelberg, 2006b. ISBN 978-0-387-31073-2.

Pierre Blanchard, Desmond J. Higham, and Nicholas J. Higham. Accurate computation of the log-sum-exp and softmax functions, 2019. URL https://arxiv.org/abs/1909.03469.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features, 2019. URL https://arxiv.org/abs/1807.05520.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments, 2021a. URL https://arxiv.org/abs/2006.09882.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers, 2021b. URL https://arxiv.org/abs/2104.14294.

Delong Chen, Mustafa Shukor, Theo Moutakanni, Willy Chung, Jade Yu, Tejaswi Kasarla, Yejin Bang, Allen Bolourchi, Yann LeCun, and Pascale Fung. Vl-jepa: Joint embedding predictive architecture for vision-language, 2026. URL https://arxiv.org/abs/2512.10942.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15745–15753, 2021. doi: 10.1109/CVPR46437.2021.01549.

Nicolas Chopin and Omiros Papaspiliopoulos. *An Introduction to Sequential Monte Carlo*. Springer Series in Statistics. Springer, Springer Cham, 2020. ISBN 978-3-030-47844-5. doi: 10.1007/978-3-030-47845-2.

Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Matthieu Destrade, Oumayma Bounou, Quentin Le Lidec, Jean Ponce, and Yann LeCun. Value-guided action planning with jepa world models, 2025. URL https://arxiv.org/abs/2601.00844.

Arnaud Doucet, Nando de Freitas, and Neil Gordon (eds.). *Sequential Monte Carlo Methods in Practice*. Information Science and Statistics. Springer, New York, 2001. ISBN 978-0-387-95146-1. doi: 10.1007/978-1-4757-3437-9.

Bernd Fritzke. A growing neural gas network learns topologies. In G. Tesauro, D. Touretzky, and T. Leen (eds.), *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL https://proceedings.neurips.cc/paper_files/paper/1994/file/d56b9fc4b0f1be8871f5e1c40c0067e7-Paper.pdf.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent a new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Gregory Gundersen. Random fourier features. https://gregorygundersen.com/blog/2019/12/23/random-fourier-features/, dec 2019.

Manu Srinath Halvagal, Axel Laborieux, and Friedemann Zenke. Implicit variance regularization in non-contrastive ssl, 2023. URL https://arxiv.org/abs/2212.04858.

Wolfgang Härdle. *Kernel Density Estimation*, pp. 43–84. Springer New York, New York, NY, 1991. ISBN 978-1-4612-4432-5. doi: 10.1007/978-1-4612-4432-5_2. URL https://doi.org/10.1007/978-1-4612-4432-5_2.

Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL https://doi.org/10.1038/s41586-020-2649-2.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning, 2020. URL https://arxiv.org/abs/1911.05722.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization, 2019. URL https://arxiv.org/abs/1808.06670.

Yongchao Huang. Classification via score-based generative modelling, 2022. URL https://arxiv.org/abs/2207.11091.

Yongchao Huang. Sampling via gaussian mixture approximations, 2025a. URL https://arxiv.org/abs/2509.25232.

Yongchao Huang. Llm-prior: A framework for knowledge-driven prior elicitation and aggregation, 2025b. URL https://arxiv.org/abs/2508.03766.

Yongchao Huang. Bijepa: Bi-directional joint embedding predictive architecture for symmetric representation learning, 2026a. URL https://arxiv.org/abs/2603.00049.

Yongchao Huang. Vjepa: Variational joint embedding predictive architectures as probabilistic world models, 2026b. URL https://arxiv.org/abs/2601.14354.

Yongchao Huang and Hassan Raza. Knowledge, rules and their embeddings: Two paths towards neuro-symbolic jepa. URL https://zenodo.org/doi/10.5281/zenodo.18777543.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3): 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666, June 2010. ISSN 0167-8655. doi: 10.1016/j.patrec.2009.09.011. URL https://doi.org/10.1016/j.patrec.2009.09.011.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning, 2020. URL https://arxiv.org/abs/2010.01028.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Yann LeCun. A path towards autonomous machine intelligence. Preprint, jun 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf. Version 0.9.2.

Jonathan Q. Li and Andrew R. Barron. Mixture density estimation. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 279–285, Cambridge, MA, USA, November 1999. MIT Press.

Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, pp. 1485–1488, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781605589336. doi: 10.1145/1873951.1874254. URL https://doi.org/10.1145/1873951.1874254.

Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*. Marcel Dekker, 1988.

James Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society A*, 209(441–458):415–446, 1909. doi: 10.1098/rsta.1909.0016.

Pierre Del Moral, Arnaud Doucet, and Ajay Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 68(3):411–436, 2006. ISSN 13697412, 14679868. URL http://www.jstor.org/stable/3879283.

Andriy Norets. Approximation of conditional densities by smooth mixtures of regressions. *The Annals of Statistics*, 38(3), June 2010. ISSN 0090-5364. doi: 10.1214/09-aos765. URL https://projecteuclid.org/journals/annals-of-statistics/volume-38/issue-3/Approximation-of-conditional-densities-by-smooth-mixtures-of-regressions/10.1214/09-AOS765.full. Publisher: Institute of Mathematical Statistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. Curran Associates, Inc., 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.*, 12(null):2825–2830, November 2011. ISSN 1532-4435.

C. Pozrikidis. *An Introduction to Grids, Graphs, and Networks*. Oxford University Press, 2014. ISBN 9780199996735.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.), *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007. URL https://proceedings.neurips.cc/paper_files/paper/2007/file/013a006f03dbc5392effeb8f18fda755-Paper.pdf.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.001.0001. URL https://doi.org/10.7551/mitpress/3206.001.0001.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples, 2021. URL https://arxiv.org/abs/2010.04592.

Walter Rudin. *Fourier Analysis on Groups*. John Wiley & Sons, 1990. ISBN 9780470448199. doi: 10.1002/9781118165621.

Ayumu Saito, Prachi Kudeshia, and Jiju Poovvancheri. Point-jepa: A joint embedding predictive architecture for self-supervised learning on point cloud, 2025. URL https://arxiv.org/abs/2404.16432.

Geri Skenderi, Hang Li, Jiliang Tang, and Marco Cristani. Graph-level representation learning with joint-embedding predictive architectures, 2025. URL https://arxiv.org/abs/2309.16014.

Basile Terver, Tsung-Yen Yang, Jean Ponce, Adrien Bardes, and Yann LeCun. What drives success in physical planning with joint-embedding predictive world models?, 2026. URL https://openreview.net/forum?id=TuYC5Fpp7M.

Yuandong Tian, Xinlei Chen, and Surya Ganguli. Understanding self-supervised learning dynamics without contrastive pairs, 2021. URL https://arxiv.org/abs/2102.06810.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2022. URL https://arxiv.org/abs/2005.10242.

Xiao Wang, Haoqi Fan, Yuandong Tian, Daisuke Kihara, and Xinlei Chen. On the importance of asymmetry for siamese representation learning, 2022. URL https://arxiv.org/abs/2204.00613.

Max A. Woodbury. Inverting modified matrices. Memo Report 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950.

Chuhan Wu, Fangzhao Wu, and Yongfeng Huang. Rethinking infonce: How many negative samples do you need?, 2021. URL https://arxiv.org/abs/2105.13003.

Zhirong Wu, Yuanjun Xiong, Stella Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance-level discrimination, 2018. URL https://arxiv.org/abs/1805.01978.

Zhuoning Yuan, Yuexin Wu, Zi-Hao Qiu, Xianzhi Du, Lijun Zhang, Denny Zhou, and Tianbao Yang. Provable stochastic optimization for global contrastive learning: Small batch does not harm performance, 2022. URL https://arxiv.org/abs/2202.12387.

Zhijian Zhuo, Yifei Wang, Jinwen Ma, and Yisen Wang. Towards a unified theoretical understanding of non-contrastive learning via rank differential mechanism, 2023. URL https://arxiv.org/abs/2303.02387.

A Extended Related Work

Self-supervised learning (SSL) has developed through several major paradigms, including pretext-based prediction, contrastive representation learning, non-contrastive joint-embedding methods, predictive architectures, and classical probabilistic modeling. Our Gaussian Mixture Joint Embeddings (GMJE) framework is most closely related to these latter three directions: contrastive learning, non-contrastive and predictive joint-embedding methods, and probabilistic latent-variable approaches.

Self-Supervised and Contrastive Learning. The modern era of SSL was strongly influenced by Contrastive Predictive Coding (CPC) (van den Oord et al., 2019), which introduced the InfoNCE objective as a tractable contrastive criterion related to mutual information maximization. While CPC provided an influential information-theoretic perspective on representation learning, the tightness of the InfoNCE mutual-information bound generally improves as more negative samples are used, which motivates large effective contrastive dictionaries (van den Oord et al., 2019; Wu et al., 2021). To reduce the dependence on large minibatches, MoCo (He et al., 2020) decoupled dictionary size from minibatch size by introducing a momentum-updated encoder together with a queue-based memory bank, in which the current minibatch is enqueued and the oldest entries are dequeued (i.e. *First-In-First-Out*). This enabled substantially larger sets of negative samples with modest computational overhead, although the replacement rule is based on temporal age rather than sample informativeness. SimCLR (Chen et al., 2020) simplified the contrastive pipeline further by removing the memory bank and instead relying on strong data augmentation, a non-linear projection head, and very large in-batch negative sets. Although architecturally simple and highly effective, SimCLR remains computationally demanding because its performance is closely tied to large-batch training (Chen et al., 2020; Yuan et al., 2022).

Non-Contrastive Learning and Collapse Prevention. An early alternative to pairwise discrimination was DeepCluster (Caron et al., 2019), which alternated between clustering learned representations with k-means and using the resulting assignments as pseudo-labels for representation learning. While effective at capturing global semantic structure, this alternating offline clustering procedure was computationally cumbersome and only loosely coupled to continuous network updates (Caron et al., 2019). To reduce both the reliance on negative sampling and the cost of offline clustering, later methods explored several prominent directions, including asymmetric predictive learning and online clustering. BYOL (Grill et al., 2020) proposed predicting the representation of one augmented view from another using an online network (i.e. a predictor), and a momentum-updated target encoder. Subsequent theoretical analyses showed that architectural asymmetry, especially the predictor and stop-gradient mechanism, plays a central role in preventing representational collapse in such non-contrastive Siamese frameworks (Tian et al., 2021; Halvagal et al., 2023; Wang et al., 2022). In parallel, SwAV (Caron et al., 2021a) transformed the DeepCluster-style paradigm into an online assignment framework by mapping representations to learnable prototypes and computing assignments with the Sinkhorn-Knopp algorithm. This reduced the need for explicit pairwise negatives and was motivated in part by avoiding issues such as class collision, although its balanced assignment mechanism can be restrictive when the underlying semantic frequencies are naturally imbalanced (Caron et al., 2021a). SimSiam (Chen & He, 2021) later demonstrated that stop-gradients alone, without EMA momentum, were theoretically sufficient to prevent collapse in Siamese networks, and subsequent analyses indicated that stop-gradient together with predictor asymmetry can be sufficient to avoid collapse in simplified Siamese settings (Tian et al., 2021; Chen & He, 2021).

Expanding on these asymmetric designs, DINO (Caron et al., 2021b) achieved self-distillation with no labels using a student-teacher architecture with an EMA-updated teacher, together with centering and sharpening, notably without requiring a predictor module. DINO was particularly successful at learning object-centric and semantically meaningful visual features, although its collapse-avoidance mechanisms were initially introduced in a largely empirical form (Caron et al., 2021b). More recently, the Rank Differential Mechanism (RDM) was proposed as a unified theoretical account of several non-contrastive methods (Zhuo et al., 2023). RDM argues that asymmetric designs, including predictors, stop-gradients, and centering, induce a consistent rank difference between the two branches, thereby improving effective dimensionality and alleviating both complete and dimensional feature collapse (Zhuo et al., 2023). Many of these methods therefore rely on asymmetric architectural or optimization mechanisms to stabilize learning. VICReg (Bardes et al., 2022) explored a more symmetric alternative by explicitly regularizing the variance and covariance of the

learned features. However, it does so through feature-level regularization rather than by modeling a full joint generative dependency between context and target representations (Bardes et al., 2022). In contrast, our GMJE framework is designed to provide an explicitly probabilistic alternative: by modeling the full joint covariance structure, it replaces heuristic collapse-prevention mechanisms with a generative objective that directly couples data fit and covariance-aware geometric regularization.

Deterministic and Probabilistic JEPA. Moving away from instance alignment, the Joint Embedding Predictive Architecture (JEPA) (Assran et al., 2023; LeCun, 2022) shifted SSL towards masked latent-space prediction. Rather than reconstructing pixels or tokens, JEPA-style models predict target representations directly in latent space, aiming to capture higher-level semantic structure while filtering out low-level stochastic detail (LeCun, 2022; Assran et al., 2023). This paradigm has since expanded across domains, including image-based JEPA (I-JEPA) (Assran et al., 2023), video-based JEPA (V-JEPA) (Bardes et al., 2024; Assran et al., 2025), motion-content JEPA (MC-JEPA) (Bardes et al., 2023), skeletal JEPA (S-JEPA) (Abdelfattah & Alahi, 2024), Point-JEPA for point clouds (Saito et al., 2025), Graph-JEPA (Skenderi et al., 2025), and vision-language extensions such as VL-JEPA (Chen et al., 2026). Related developments have also extended JEPA toward planning and world modeling, including JEPA World Models (Terver et al., 2026) and Value-Guided JEPA (Destrade et al., 2025).

Despite this growing family of methods, standard JEPA remains based on a deterministic neural predictor, typically optimized with a regression-style objective such as MSE (Assran et al., 2023; Bardes et al., 2024). This design is highly scalable and avoids spending modeling capacity on pixel-level reconstruction (Assran et al., 2023), but deterministic squared-loss prediction is fundamentally limited in genuinely multi-modal settings, since it converges toward the conditional mean and may therefore place predictions in low-density regions (e.g. "empty regimes") between valid modes. To address these limitations, several recent probabilistic and structured JEPA variants have emerged. Variational JEPA (VJEPA) (Huang, 2026b) introduces a probabilistic formulation that learns a predictive distribution over future latent states through a variational objective, linking JEPA-style representation learning with Bayesian filtering and Predictive State Representations (PSRs). Bi-directional JEPA (BiJEPA) (Huang, 2026a) extends the predictive objective to both directions, encouraging cycle-consistent predictability between data segments and capturing semantic information inherent in inverse relationships, although symmetric prediction can amplify optimization instability and may require explicit norm control, such as L^2-normalization, to prevent representation explosion. Most recently, Rule-informed JEPA (RiJEPA) (Huang & Raza) incorporates neuro-symbolic inductive biases through Energy-Based Constraints (EBC), reshaping the latent geometry using structured logical priors to discourage shortcut learning. However, this relies on human-defined logical structure into the latent space. Taken together, these developments address deterministic JEPA from probabilistic, symmetric, and neuro-symbolic directions, yet a fully probabilistic joint-density view of JEPA-style representation learning remains underdeveloped, which motivates our GJE/GMJE framework.

GMM and Probabilistic Inference Mechanisms. The mathematical foundation of our GMJE framework builds on classical probabilistic inference and Gaussian density modeling. Mixture Density Networks (MDNs) (Bishop, 1994) combine neural networks with mixture models to represent conditional probability densities and were originally motivated by multi-valued inverse problems. This makes them a natural starting point for conditional multi-modal prediction, although in our setting a naive symmetric use of MDNs can lead to information leakage across views; GMJE-MDN addresses this through an explicit conditional information bottleneck imposed by the joint-modeling formulation. Growing Neural Gas (GNG) (Fritzke, 1994) provides a topology-preserving mechanism for adaptively inserting prototypes based on local quantization error. Unlike fixed-K mixture fitting, GNG can discover structure incrementally without requiring the number of components (K) in advance, and avoids the local minima of standard Expectation-Maximization (EM), although its traditional implementations are non-differentiable. Our GMJE-GNG adapts this prototype-growth logic for dynamic prototype discovery in continuous representation spaces.

To reduce the computational burden of exact kernel methods, Random Fourier Features (RFF) (Rahimi & Recht, 2007) approximate shift-invariant kernels by projecting inputs into a randomized finite-dimensional Fourier feature space, reducing the computational bottleneck of Gram matrix inversion from $\mathcal{O}(N^3)$ to $\mathcal{O}(ND^2)$. However, as an approximation to dual-space kernel model-

ing, RFF does not by itself resolve the limitations of unimodal conditional prediction under strong multi-modality, which motivates our shift towards explicit primal-space mixture modeling. Finally, probabilistic inference of GMMs can also be approached through sampling-based methods. For example, Sequential Monte Carlo (SMC) (Moral et al., 2006) provides a particle-based framework for sequential weighting, resampling, and posterior approximation, while Gaussian Mixture Approximation (GMA) (Huang, 2025a) introduces a gradient-free optimization-resampling pipeline for approximating unnormalized target densities. Inspired by these sampling and resampling dynamics, we introduce an *SMC-GMJE* variant, which incorporates SMC-style weighting into SSL memory banks in order to adaptively prioritize informative negatives and down-weight stale ones.

B Notation and Extended GJE Formulation

Notation and Terminology. Throughout this work, we use the following mathematical notations.

- *Data and Embeddings:* lowercase letters (x_c, x_t) denote raw individual context and target views (e.g. augmented images). $z_c, z_t \in \mathbb{R}^d$ denote their respective d-dimensional latent representations[1], and $Z = [z_c^T, z_t^T]^T$ denotes the concatenated joint embedding. Uppercase letters (Z_c, Z_t) represent matrices containing finite batches of these embeddings.
- *Probabilistic Parameters:* Σ denotes the general, theoretical population covariance matrix of a multivariate distribution, while μ denotes the mean vector. For mixture models, π_k represents the mixture weight for component k, and $\gamma_k(\cdot)$ represents the data-dependent posterior probability (soft routing responsibility).
- *Structural Capacities (N, M, K):* to strictly distinguish between optimization batches, global memory, and model capacity, we define:
  - N: the *batch size*, i.e. the number of current representations processed in a single forward pass.
  - M: the *memory bank size*, i.e. the massive queue of cached target representations, used in contrastive learning and SMC.
  - K: the *number of components/prototypes* in a Gaussian Mixture Model (GMM).
- *The Dual Meaning of K and Σ:* in Section.3, when discussing dual-GJE, K is exceptionally used to denote the empirical *block Gram matrix* computed over a batch via a positive-definite kernel function (e.g. K_{cc}, K_{tt}). Thus, in Section.3, Σ and K represent the same structural concept, distinguishing between pure probabilistic theory (Σ) and empirical kernel-based computation (K). Everywhere else in the manuscript (Section.4 onwards), scalar K strictly refers to the number of mixture components.

To model the alignment between a context view x_c and a target view x_t, we introduce the *Gaussian Joint Embedding* (GJE) framework. As shown in Fig.1, the input views are passed through a context encoder E_θ and a target encoder $E_{\theta'}$ to produce latent embeddings $z_c = E_\theta(x_c) \in \mathbb{R}^{d_c}$ and $z_t = E_{\theta'}(x_t) \in \mathbb{R}^{d_t}$.

Although the general GJE framework can naturally incorporate optional side information ε (such as physical constraints, action variables in reinforcement learning, or noise parameters) by conditioning the joint distribution, in the following we focus on unconditional representation alignment. Let $z = [z_c^T, z_t^T]^T \in \mathbb{R}^d$, where $d = d_c + d_t$, denote the concatenated joint embedding pair. We assume these representations are drawn from an underlying joint probability distribution $p(z; \theta, \theta')$ parameterized by the neural encoders. The self-supervised objective is then to optimize the encoder parameters by maximizing the expected joint log-likelihood over matching pairs (x_c, x_t) drawn from the dataset $\mathcal{D}$:

$$\max_{\theta,\theta'} \mathbb{E}_{(x_c,x_t)\sim\mathcal{D}} \left[\log p(E_\theta(x_c), E_{\theta'}(x_t))\right] \tag{12}$$

By the chain rule of probability, the joint objective factorizes as

$$\log p(z_c, z_t) = \log p(z_t \mid z_c) + \log p(z_c),$$

[1]Note: in the brief literature review of Mixture Density Networks in Section.C.3.1, we temporarily preserve Bishop's original use of c to denote dimensionality to maintain historical fidelity.

which may be interpreted as a predictive conditional term together with a marginal regularization term. To make this optimization computationally tractable and geometrically meaningful, we impose a Gaussian assumption on the embedding space. Depending on how this Gaussian structure is instantiated, the framework can be developed from two complementary perspectives: the *dual (sample) view*, which focuses on the predictive conditional alignment $p(z_t \mid z_c)$ over a batch of N points, and the *primal (feature) view*, which models the full joint density $p(z_c, z_t)$ natively over the d latent dimensions.

B.1 The Dual Formulation: Sample-Space Gaussian Process Joint Embeddings (GPJE)

In classic JEPA, a deterministic neural predictor maps the context embedding to the target embedding: $\hat{z}_t = g(z_c)$. To generalize this probabilistically, we first formulate the problem in the *dual sample space* by replacing this parametric MLP with a non-parametric probabilistic regression model, which treats the target embedding z_t as a realization of an underlying latent function f evaluated at the context embedding z_c:

$$z_t = f(z_c) + \epsilon, \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I) \tag{13}$$

Here, $f \sim \mathcal{GP}(\mathbf{0}, k(z_c, z_c'))$ is a Gaussian Process defined by a positive-definite kernel k, which dictates that $\text{cov}(f(z_c), f(z_c')) = k(z_c, z_c')$. Rather than modeling the covariance between the d feature channels, this sample-space GP models the similarities between the *individual* data points.

1. Training (The Joint Likelihood). During training, we extract a batch of N context and target embedding pairs, $Z_c \in \mathbb{R}^{N \times d_c}$ and $Z_t \in \mathbb{R}^{N \times d_t}$. We compute the Gram matrix $K_{cc} \in \mathbb{R}^{N \times N}$ over the context embeddings, where each element $K_{cc}[i, j] = k_\phi(z_{c,i}, z_{c,j})$, capturing the non-linear topological distances between every sample in the batch.

Treating the target embeddings $Z_t \in \mathbb{R}^{N \times d_t}$ as the observed function values, we derive the training objective by assuming the d_t latent feature channels are independent Gaussian Processes sharing the same context kernel K_{cc}. For a single feature channel (represented as a column vector $z_{t,j} \in \mathbb{R}^{N \times 1}$), the standard GP negative marginal log-likelihood, assuming negligible observation noise and dropping constant scaling terms, is[2] $\frac{1}{2} z_{t,j}^T K_{cc}^{-1} z_{t,j} + \frac{1}{2} \log |K_{cc}|$. Summing this independent objective across all d_t target dimensions yields the total joint negative log-likelihood. In linear algebra, the summation of these data-fit quadratic forms across all column vectors mathematically condenses into a matrix trace operator ($\sum_{j=1}^{d_t} z_{t,j}^T K_{cc}^{-1} z_{t,j} = \text{Tr}(Z_t^T K_{cc}^{-1} Z_t)$). Meanwhile, the data-independent log-determinant penalty simply accumulates d_t times. Thus, the final Dual-GJE objective used to optimize the encoder weights is:

$$\mathcal{L}_{\text{Dual-GJE}}(\theta, \theta', \phi) = \underbrace{\frac{1}{2} \text{Tr}\left(Z_t^T K_{cc}^{-1} Z_t\right)}_{\text{Data-Fit}} + \underbrace{\frac{d_t}{2} \log |K_{cc}|}_{\text{Complexity Penalty}} \tag{14}$$

This Dual-GJE optimisation objective is a *conditional* NLL[3], $p(Z_t \mid Z_c)$. As this GP marginal likelihood is inherently asymmetric - it penalizes the volume of the context space ($\log |K_{cc}|$) but applies no such expansive regularizer to the target space, optimizing Eq.2 symmetrically would result in catastrophic representation collapse. The target encoder would trivially map $Z_t \to \mathbf{0}$ to minimize the quadratic trace penalty, subsequently causing the log-determinant to collapse the context space[4].

To effectively optimize this dual objective, the target encoder $E_{\theta'}$ must be maintained as an Exponential Moving Average (EMA) (Grill et al., 2020; Chen & He, 2021; Assran et al., 2023; Huang, 2026b) of the context encoder[5], with a strict stop-gradient applied to Z_t. By anchoring the targets,

[2] See Eq.(2.29) and Eq.(2.30) in (Rasmussen & Williams, 2005). We dropped the constant term.

[3] See Eq.(2.29) or Eq.(2.30) in (Rasmussen & Williams, 2005).

[4] A detailed explanation of this learning collapse can be found in Appendix.H.

[5] In standard Joint Embedding Architectures, the target encoder weights θ' are typically updated as an Exponential Moving Average (EMA) of the context encoder weights θ, such that $\theta' \leftarrow \tau\theta' + (1 - \tau)\theta$ where $\tau \in [0, 1)$ is a momentum parameter. This asymmetric weight update, combined with a *stop-gradient* operation on the target branch, is a crucial architectural heuristic in models such as BYOL (Grill et al., 2020) and standard JEPA (Assran et al., 2023) to prevent representation collapse - a failure mode where the network maps all inputs to a trivial constant vector to perfectly minimize prediction error.

Z_t acts as a diverse, fixed topological reference. Only under this asymmetric EMA framework does the trace term correctly function as an expansionary force, compelling the context embeddings to spread out and strictly match the pairwise diversity of the targets, while the log-determinant term acts as an Occam's razor, regularizing the context manifold to remain smooth.

If explicit homoscedastic observation noise $\sigma^2 I$ is modeled, the objective Eq.2 expands to:

$$\mathcal{L}_{\text{JEGP}}(\theta, \theta') = \frac{1}{2}\text{Tr}\left((K_{cc} + \sigma^2 I)^{-1} Z_t Z_t^T\right) + \frac{d_t}{2}\log|K_{cc} + \sigma^2 I| \tag{15}$$

By the cyclic property of the trace operator ($\text{Tr}(AB) = \text{Tr}(BA)$), the data-fit terms in Eq.2 and Eq.15 are mathematically equivalent. Both formulations drive the exact same geometric forces, optimizing the encoders symmetrically.

The Dual-GJE algorithm is presented in Algo.3 in Appendix.I; a scalable Dual-GJE via Random Fourier Features (RFF) is presented in the same appendix.

2. Inference (The Conditional Prediction). Once *the encoders and kernel are trained*, we freeze the training memory bank $Z_t \in \mathbb{R}^{N \times d_t}$ and the training Gram matrix $K_{cc} \in \mathbb{R}^{N \times N}$.

Given a brand new, unobserved test context image x_c^*, we map it to a single embedding $z_c^* \in \mathbb{R}^{1 \times d_c}$. We compute its cross-similarity vector against all N training samples, $k_{*c} \in \mathbb{R}^{1 \times N}$, and its self-similarity scalar $k_{**} \in \mathbb{R}^{1 \times 1}$. Using the Schur complement to condition the joint GP distribution, the predictive distribution for the target $z_t^* \in \mathbb{R}^{1 \times d_t}$ is exactly Gaussian, $\mathcal{N}(\mu_{t|c}, \Sigma_{t|c})$, defined by[6]:

$$\begin{aligned} \mu_{t|c} &= k_{*c} K_{cc}^{-1} Z_t \\ \Sigma_{t|c} &= k_{**} - k_{*c} K_{cc}^{-1} k_{c*} \end{aligned} \tag{16}$$

Notice the dimensions of Eq.16: $[1 \times N] \times [N \times N] \times [N \times d_t] = [1 \times d_t]$. The prediction is a non-parametric, weighted average over (i.e. linear combination of) the entire training memory bank Z_t. *(Note: Because exact inversion requires $\mathcal{O}(N^3)$ compute, GJE supports a highly scalable Random Fourier Feature (RFF) approximation, detailed in Appendix.I.1).*

B.2 The Primal Formulation: Feature-Space Covariance

While the dual formulation evaluates sample similarity, the primal formulation evaluates the geometric shape of the data manifold itself by measuring how the d latent features co-vary.

1. Training and the Mahalanobis Trace Trap. We concatenate the context and target batches into a single joint batch $Z = [Z_c, Z_t] \in \mathbb{R}^{N \times d}$. We assume a joint, zero-mean Multivariate Gaussian generative model $p(z) = \mathcal{N}(\mathbf{0}, C_{joint})$, i.e. each concatenated embedding vector $z_i \in \mathbb{R}^d$ is drawn from this Gaussian distribution:

$$p(z_i) \sim \mathcal{N}(z|\mathbf{0}, C_{joint}) = \frac{1}{\sqrt{(2\pi)^d |C_{joint}|}} \exp\left(-\frac{1}{2} z^T C_{joint}^{-1} z\right) \tag{17}$$

In the context of deep representation learning, this strict zero-mean assumption serves a critical regularizing function: it anchors the latent space at the origin and forces the encoders to delegate all structural learning entirely to the covariance matrix. This prevents the network from trivially minimizing the loss via arbitrary translational mean shifts.

Consequently, the feature covariance matrix C_{joint}, with dimension $d \times d$, can be empirically[7] computed directly from the origin for the N training data points[8]:

$$C_{joint} = \frac{1}{N} Z^T Z \quad \in \mathbb{R}^{d \times d} \tag{18}$$

[6] See Eq.(2.25) and Eq.(2.26) in (Rasmussen & Williams, 2005). If explicit homoscedastic observation noise is modeled (as in Eq.15), the predictive distribution seamlessly adapts by replacing the inverse Gram matrix K_{cc}^{-1} in both equations with the noise-regularized term $(K_{cc} + \sigma^2 I)^{-1}$.

[7] For simplicity, we use the same notation C_{joint} for the population and empirically estimated covariance matrix.

[8] By treating the data matrix $Z \in \mathbb{R}^{N \times d}$ as a stack of N row vectors $Z = [z_1^T, z_2^T, \ldots, z_N^T]^T$, its transpose $Z^T \in \mathbb{R}^{d \times N}$ becomes a matrix of side-by-side column vectors $Z^T = [z_1, z_2, \ldots, z_N]$. Using block matrix multiplication, multiplying the row of blocks by the column of blocks yields the sum of their outer products: $Z^T Z = [z_1, \ldots, z_N][z_1^T, \ldots, z_N^T]^T = \sum_{i=1}^N z_i z_i^T$. Taking the average with N or $N-1$ is trivial.

and it can be partitioned into four blocks:

$$C_{joint} = \begin{bmatrix} C_{cc} & C_{ct} \\ C_{tc} & C_{tt} \end{bmatrix} \tag{19}$$

where $C_{cc} \in \mathbb{R}^{d_c \times d_c}$ and $C_{tt} \in \mathbb{R}^{d_t \times d_t}$ are the auto-covariances, and $C_{ct} \in \mathbb{R}^{d_c \times d_t}$ is the cross-covariance capturing mutual information. Inverting C_{joint} can be done using block matrix inversion and the Schur complement (see Eq.53 in Appendix.G).

Assuming the N samples in the mini-batch are independent and identically distributed, the likelihood of the entire empirical batch is the product of the individual densities, $p(Z) = \prod_{i=1}^{N} p(z_i)$. To optimize the encoder parameters, we formulate the empirical average of the NLL, defined as $-\frac{1}{N} \log p(Z)$:

$$-\frac{1}{N} \log \prod_{i=1}^{N} p(z_i) = -\frac{1}{N} \sum_{i=1}^{N} \log p(z_i) \quad = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} z_i^T C_{joint}^{-1} z_i + \frac{1}{2} \log |C_{joint}| + \frac{d}{2} \log(2\pi) \right)$$

Dropping the constant $\frac{d}{2} \log(2\pi)$ term, which has no effect on the gradient with respect to the network weights, the strict generative objective minimizing the exact NLL over this $d \times d$ space becomes:

$$\mathcal{L}_{\text{Primal-GJE}}(\theta, \theta') = \frac{1}{2N} \sum_{i=1}^{N} z_i^T C_{joint}^{-1} z_i + \frac{1}{2} \log |C_{joint}| \tag{20}$$

From an information-theoretic perspective, minimizing this joint objective natively *maximizes the Mutual Information* between the context and target spaces while regularizing their differential entropy. Further, factoring this joint likelihood shows why conditional-only architectures like classic JEPA require asymmetric heuristics to survive (for formal proofs, see Appendix.G).

It seems from the primal objective Eq.3 that, minimizing the volume penalty $\log |C_{joint}|$ natively enforces compression, while the inverted Mahalanobis distance acts as a repulsive force; however, directly optimizing this exact NLL in the empirical primal space reveals a fatal optimization trap. Because the scalar quadratic form is equal to its own trace, we can apply the cyclic property ($\text{Tr}(AB) = \text{Tr}(BA)$) to the data-fit term:

$$\frac{1}{2N} \sum_{i=1}^{N} \text{Tr}(z_i^T C_{joint}^{-1} z_i) = \frac{1}{2} \text{Tr} \left(C_{joint}^{-1} \left[\frac{1}{N} \sum_{i=1}^{N} z_i z_i^T \right] \right) = \frac{1}{2} \text{Tr}(C_{joint}^{-1} C_{joint}) = \frac{d}{2}$$

The inverted Mahalanobis distance algebraically cancels against the empirical covariance, collapsing the entire data-fit term into a dead constant scalar $\frac{d}{2}$. With a constant data-fit term, the expansive gradient is exactly zero. The optimizer is left solely with the volume penalty ($+\frac{1}{2} \log |C_{joint}|$), which it trivially minimizes by shrinking all representations to the origin, causing catastrophic dimensional collapse (details about The "*Mahalanobis Trace Trap*" see Appendix.N).

Therefore, to maintain stable representations in the primal space, we cannot rely on the native Mahalanobis distance for repulsion. We must structurally invert the regularizer, i.e. *maximizing* the differential entropy $\frac{1}{2} \log |C_{joint}|$ to force the features to span the available volume (Bardes et al., 2022), or alternatively, optimize the cross-covariance directly via the Hilbert-Schmidt Independence Criterion (HSIC).

The Primal-GJE algorithm is presented in Algo.5 in Appendix.J.

2. Inference (The Closed-Form Linear Predictor). After training, we freeze the four $d \times d$ covariance blocks. Given a new test context vector $z_c^* \in \mathbb{R}^{d_c \times 1}$ (formatted as a column vector), the conditional distribution $p(z_t^* \mid z_c^*) = \mathcal{N}(z_t^* | \mu_{t|c}(z_c^*), \Sigma_{t|c})$ is analytically derived via block matrix inversion (Bishop, 2006b):

$$\begin{aligned} \mu_{t|c} &= C_{tc} C_{cc}^{-1} z_c^* \\ \Sigma_{t|c} &= C_{tt} - C_{tc} C_{cc}^{-1} C_{ct} \end{aligned} \tag{21}$$

In Eq.4, the term $(C_{tc} C_{cc}^{-1})$ multiplies a $[d_t \times d_c]$ matrix by a $[d_c \times d_c]$ matrix, resulting in a fixed $[d_t \times d_c]$ weight matrix. Therefore, the primal GJE formulation mathematically derives the *optimal*

linear projection layer directly from the global feature covariances, requiring no training data in memory during inference[9]. It is observed that, the predictive variance is a constant decided by the training set, and it's independent of the test context z_c^*.

B.3 Sample Space vs. Feature Space

The Dual GJE (Section.3) and the Primal GJE (Section.3) both rely on the fundamental Gaussian assumption, but they approach the $z_c \to z_t$ mapping problem from perpendicular geometric perspectives.

Dual GJE (GPJE) operates in the sample space, asking the geometric question: "how similar is Image A to Image B?" By evaluating the $N \times N$ Gram matrix computed via inner products between samples (e.g. $K_{cc} \propto Z_c Z_c^T$), this view does not model individual features, but rather collapses feature vectors into holistic similarity scores. It formulates the predictor as a non-parametric Gaussian Process, assuming the target embeddings Z_t are jointly Gaussian with a covariance structure determined by these kernel similarities of their contexts. The resulting predictive distribution (Eq.16) is dynamic; its conditional mean and variance depend on both the stored training instances and the topological distance of the specific test context z_c^*. However, as an instance-based memory model, evaluating the GPJE objective (Eq.2) requires inverting an $N \times N$ matrix, imposing a severe $\mathcal{O}(N^3)$ computational bottleneck. Further, as the GP marginal likelihood is strictly a *conditional* objective[10] $p(Z_t \mid Z_c)$, it lacks a native volume regularizer for the context space, thereby mandating asymmetric heuristics like EMA target networks to prevent representation collapse.

Primal GJE, conversely, operates in the feature space, asking the perpendicular question: "how similar are the individual latent features to each other across the dataset?" It acts as a parametric model. It assumes a single, unimodal joint Gaussian distribution (Eq.17) over the concatenated feature space. By computing the inner products between the features themselves, its empirical covariance matrix $C_{joint} \propto Z^T Z$ has dimension $d \times d$, which, for typical modern embedding sizes where $d \ll N$, is massively smaller and computationally cheaper to invert ($\mathcal{O}(d^3)$) than the dual Gram matrix. In this primal view, the resulting conditional prediction (Eq.4) yields a mean that linearly depends on the test context z_c^*, but a predictive covariance that is strictly constant, frozen entirely by the global distribution of the training data. Crucially, because it optimizes the full *joint* likelihood[11] $p(z_c, z_t)$, it provides native, symmetric regularizers for both the context and target spaces, eliminating the strict requirement for asymmetric stop-gradients.

The Primal-Dual Equivalence. While they approach the geometry from different dimensions, the two formulations are fundamentally linked. In fact, if the non-parametric Dual-GJE is implemented using a standard linear kernel, the exact $\mathcal{O}(N^3)$ Gaussian Process NLL algebraically collapses entirely into the $\mathcal{O}(d^3)$ Primal-GJE covariance matrix framework (proof see Appendix.K). This strict primal-dual equivalence mathematically proves that, without the use of computationally expensive non-linear kernel approximations (such as RBF or RFF), the dual sample-space optimization locks the latent geometry to the exact same single, rigid density as the primal feature space[12].

Unimodality and Homoscedasticity Issues Regardless of whether GJE is formulated in the non-parametric dual space (Eq.16) or the parametric primal space (Eq.4), both architectures suffer from a shared limitation: *unimodality*. Because both formulations rely on a single joint Gaussian structure, they can only issue a unimodal predictive distribution. Consequently, when faced with diverging semantic branches, i.e. when there are multiple options for z_t^* for a given z_c^*, the predictive mean $\mu_{t|c}$ is mathematically forced to predict the literal average of the modes (slicing through empty space; Bishop, 1994). Neither model can physically route predictions to distinct, isolated target branches (a failure case we will see later in a synthetic example in Section 5).

[9] Training data info is summarised in the matrices $[C_{cc}, C_{ct}, C_{tc}, C_{tt}]$ via Eq.18

[10] see Eq.(2.29) or Eq.(2.30) in (Rasmussen & Williams, 2005).

[11] See Appendix.G a comparison of the joint and conditional NLL opjectives.

[12] The surprise comes as: with a linear kernel, Dual is exactly equal to Primal. Therefore, they both suffer from the exact same Unimodal Gaussian Trap.

Further, the parametric primal-GJE is constrained by *homoscedasticity*. While the dual-GJE yields a dynamic, distance-aware predictive covariance[13] that changes based on the test context z_c^*, the primal-GJE predictive covariance ($\Sigma_{t|c}$ in Eq.4) is a rigid, constant matrix determined entirely by the global training distribution. It contains no reference to the specific input z_c^*. Thus, to blanket the multi-modal spread of ambiguous inputs, the primal-GJE model must globally inflate its static variance across the entire latent space, degrading predictive certainty even for unambiguous samples.

These structural limitations, i.e. the inability to model multiple target branches and the rigid covariance of the parametric space, motivate our transition to Gaussian Mixture Joint Embeddings (GMJE) in the primal feature space.

B.4 The Primal-Dual Paradigm Shift

To conclude our theoretical exposition, it is helpful to contrast the two mathematical perspectives used throughout this work: the Dual (sample-space) view of GJE and the Primal (feature-space) view of GMJE. Together, these complementary perspectives provide a probabilistically grounded framework for multi-modal representation learning. To make this duality concrete, consider a batch of embeddings arranged as a matrix $Z \in \mathbb{R}^{N \times d}$, where each of the N rows corresponds to an individual data sample and each of the d columns corresponds to a latent feature.

- **The Dual "Sample Space" View (Section 3):** the pure GJE framework operates in the dual space, evaluating the $N \times N$ Gram matrix (ZZ^T). This perspective prioritizes *sample similarity*, measuring the relationships between individual images within a batch. Its geometric regularizer ($\frac{1}{2}\log|K_{cc}|$) maximizes *batch diversity*, guarding against standard **Representation Collapse** (where all N samples map to the exact same point). While theoretically elegant for continuous functional mapping, it suffers from an $\mathcal{O}(N^3)$ computational bottleneck (necessitating approximations like RFF) and structurally struggles to isolate discrete multi-modal clusters.

- **The Primal "Feature Space" View (Section 4):** the GMJE framework executes a classical shift into the primal space, evaluating the $d \times d$ feature covariance matrix (Z^TZ). This perspective prioritizes *feature independence*, directly modeling the geometric shape of the data manifold. Its geometric regularizer ($\frac{1}{2}\log|\Sigma_c|$) maximizes the *volume of the latent space*, specifically preventing **Dimensional Collapse** (where latent features become perfectly correlated and lazily flatten the data into a lower-dimensional line or subspace). By operating in the primal space, the computation scales optimally with batch size ($\mathcal{O}(d^3)$ rather than $\mathcal{O}(N^3)$) and natively provides the coordinate framework required to deploy K discrete prototypes for multi-modal semantic routing.

Why Multi-Modality is Difficult to Realize in the Dual Sample Space. To motivate the architectural necessity of GMJE, it is helpful to examine why extending pure GJE to capture multi-modal structure directly in the dual sample space ($N \times N$) is poorly suited to stable semantic modeling. In the primal feature space ($d \times d$), the network operates in a fixed global coordinate system[14], allowing K semantic modes (μ_k) to be represented as stable spatial regions. By contrast, the dual space is defined only through the Gram matrix (ZZ^T), which replaces absolute coordinates with relative pairwise similarities[15].

This creates several structural difficulties for mixture modeling. First, without explicit feature-space axes, any cluster prototype must be represented indirectly, typically as a linear combination of the current data points. Second, in stochastic self-supervised learning, the reference samples in a mini-

[13] Standard Gaussian Process regression is heteroscedastic with respect to distance (the variance grows as the test point moves further from the training data, because k_{*c} goes to zero and we are left with the prior variance k_{**}).

[14] Geometrically, the primal feature space is analogous to a map with fixed coordinate axes, where distinct semantic clusters can be permanently pinned to specific global locations.

[15] Conversely, the dual sample space is analogous to a dynamic similarity table recording only relative affinities between entities, without preserving their absolute global coordinates.

batch change at every optimization step[16], making it difficult to learn persistent global semantic modes across batches and epochs. Third, a mixture model defined directly over the Gram matrix does not naturally correspond to clustering points into distinct spatial semantic categories; instead, it tends to model multiple *similarity-generating patterns* (i.e., predicting that the similarity between two images was drawn from one of K different distributions).

For this reason, the transition to GMJE, which operates through the $d \times d$ feature covariance in primal space, is not merely a computational preference but a more natural and principled formulation for representing stable multi-modal semantic structure.

C GMJE: Extended Formulation and Architectures

C.1 The GMM Formulation for Joint Embeddings

When introducing mixture models, one could natively define a *Conditional GMM*, e.g. a *Mixture Density Network* (Bishop, 1994; Huang, 2025b) where $p(z_t|z_c)$ is directly modeled as a Gaussian mixture with weights predicted by a neural net. However, doing so breaks the theoretical symmetry of GJE: we would lose the explicit modeling of the marginal distribution $p(z_c)$, which, as established in Section.3 (Eq.54 and Eq.55), provides the native geometric regularization preventing representation collapse (and forcing us back into the realm of heuristic stop-gradients[17]).

Instead, we maintain the symmetric paradigm by defining a *Joint GMM* with K components:

$$p(z_c, z_t) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\begin{bmatrix} z_c \\ z_t \end{bmatrix} \middle| \begin{bmatrix} \mu_{c,k} \\ \mu_{t,k} \end{bmatrix}, \begin{bmatrix} \Sigma_{cc,k} & \Sigma_{ct,k} \\ \Sigma_{tc,k} & \Sigma_{tt,k} \end{bmatrix} \right) \tag{22}$$

where π_k are the mixture weights ($\sum \pi_k = 1$).

From this joint formulation, integrating out the context variable z_c yields the marginal distribution of the target representation space $p(z_t)$, which elegantly remains a closed-form Gaussian mixture (and symmetrically so for $p(z_c)$):

$$p(z_t) = \int p(z_c, z_t) dz_c = \sum_{k=1}^{K} \pi_k \mathcal{N}(z_t \mid \mu_{t,k}, \Sigma_{tt,k}) \tag{23}$$

Further, by the properties of joint Gaussians, the conditional distribution $p(z_t|z_c)$ derived from this joint mixture is exactly a GMM with closed-form parameters (see, e.g., Bishop, 2006b):

$$p(z_t|z_c) = \sum_{k=1}^{K} \gamma_k(z_c) \mathcal{N}(z_t|\mu_{t|c,k}, \Sigma_{t|c,k}) \tag{24}$$

where the conditional means $\mu_{t|c,k}$ and covariances $\Sigma_{t|c,k}$ are calculated for each component k exactly as in GJE (cc.Eq.4):

$$\begin{aligned} \mu_{t|c,k} &= \mu_{t,k} + \Sigma_{tc,k} \Sigma_{cc,k}^{-1} (z_c - \mu_{c,k}) \\ \Sigma_{t|c,k} &= \Sigma_{tt,k} - \Sigma_{tc,k} \Sigma_{cc,k}^{-1} \Sigma_{ct,k} \end{aligned} \tag{25}$$

And the new data-dependent mixing weights $\gamma_k(z_c)$ are given by:

$$\gamma_k(z_c) = \frac{\pi_k \mathcal{N}(z_c|\mu_{c,k}, \Sigma_{cc,k})}{\sum_{j=1}^{K} \pi_j \mathcal{N}(z_c|\mu_{c,j}, \Sigma_{cc,j})} \tag{26}$$

[16] For example, if one Gram matrix is formed from similarities among a dog, a car, and a boat, while the next is formed from a plane, a bird, and a frog, then the underlying reference system changes completely across iterations.

[17] The architectural heuristic of applying a stop-gradient to one branch of a *Siamese network* to prevent representation collapse was popularized by *BYOL* (Grill et al., 2020), which utilized an asymmetric exponential moving average (EMA) target network. The fundamental mathematical role of this stop-gradient operation in preventing trivial constant collapse, even without momentum or negative pairs, was subsequently isolated and proven in *SimSiam* (Chen & He, 2021). This asymmetric stop-gradient paradigm was later adapted specifically in I-JEPA (Assran et al., 2023).

This formulation is theoretically profound: the context embedding z_c natively acts as a "router" that smoothly evaluates its own marginal likelihood under each mode's context distribution, dynamically selecting which mixture component (predictor) is most appropriate to generate the target.

An schematic of the GMJE architecture is shown in Fig.6. Note the difference between GMJE and classic JEPA (Fig.1(a)) in that, while standard JEPA relies on a rigid asymmetric design, e.g. utilizing stop-gradients and a lagging Exponential Moving Average (EMA) target network to artificially prevent representation collapse, GMJE operates within a fully symmetric architectural framework. Because the joint probabilistic objective natively penalizes dimensional collapse, gradients can flow freely and simultaneously through both the context and target encoders without requiring any such architectural heuristics.

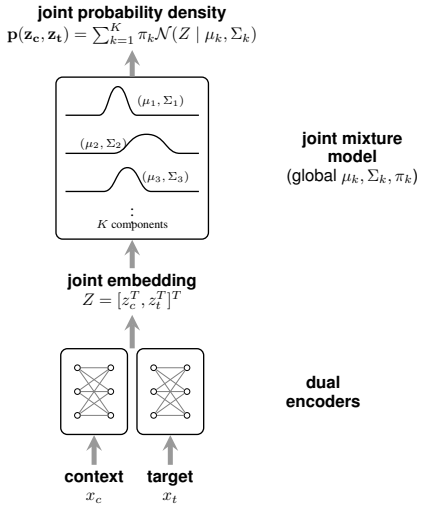


Figure 6: The general Gaussian Mixture Joint Embeddings (GMJE) framework. Dual encoders map the context and target views to a joint embedding Z. This embedding is evaluated against a set of K learnable global mixture parameters (μ_k, Σ_k, π_k) to natively model the joint probability density $p(z_c, z_t)$ within a symmetric architectural framework.

Having established the general GMJE architecture and its corresponding joint probability density, the critical subsequent challenge lies in *effectively optimizing the K mixture components (μ_k, Σ_k, π_k) to accurately map the underlying data manifold*. Because the true representation space in self-supervised learning is highly complex and dynamically changing during training, we cannot rely on trivial clustering initializations. In the following subsections, we introduce diverse, theoretically grounded methods to learn these parameters. We first discuss the **parametric GMJE approach** with K components.

C.2 Scalable Optimization via Learnable Prototypes

To train GMJE efficiently on massive datasets, we introduce a parametric, prototypical approach. We maintain a set of K learnable joint mixture means $\mu_k = [\mu_{c,k}^T, \mu_{t,k}^T]^T$, acting as global cluster prototypes, analogous to those used[18] in *SwAV* (Caron et al., 2021a) or *DeepCluster* (Caron et al., 2019), alongside a shared, parameterized covariance structure Σ.

Rather than relying on a traditional, often unstable Expectation-Maximization (EM) loop, we perform a *differentiable, relaxed EM directly through backpropagation*. We define our objective as minimizing the Negative Log-Likelihood (NLL) of the joint mixture model for an observed concatenated embedding $Z = [z_c^T, z_t^T]^T$.

[18] Specifically, DeepCluster groups representations offline via k-means to generate discrete pseudo-labels and trains the network to predict them via cross-entropy (Caron et al., 2019). SwAV operates online, computing soft assignments (codes) for one augmented view against learnable prototypes using optimal transport, and predicting this code from a differently augmented view (Caron et al., 2021a).

Starting from the joint distribution $p(Z)$ (Eq.6) and assuming a shared[19] covariance Σ across all components for tractability, the NLL expands as:

$$\begin{aligned}\mathcal{L}_{\text{GMJE-Proto}} &= -\log p(Z) \\ &= -\log\left(\sum_{k=1}^{K}\pi_k\mathcal{N}(Z\mid\mu_k,\Sigma)\right) \\ &= -\log\left(\sum_{k=1}^{K}\pi_k\frac{1}{\sqrt{(2\pi)^{2d}|\Sigma|}}\exp\left(-\frac{1}{2}(Z-\mu_k)^T\Sigma^{-1}(Z-\mu_k)\right)\right)\end{aligned} \tag{27}$$

To compute this loss without numerical overflow (a common failure mode when summing small likelihoods), we absorb the mixing weights π_k and the Gaussian normalization constants directly into the exponential using the identity $x = \exp(\log x)$. Ignoring the fixed 2π constant, which does not affect the gradients, the objective simplifies to:

$$\mathcal{L}_{\text{GMJE-Proto}} = -\log\left(\sum_{k=1}^{K}\exp\left(\log\pi_k - \frac{1}{2}(Z-\mu_k)^T\Sigma^{-1}(Z-\mu_k) - \frac{1}{2}\log|\Sigma|\right)\right) \tag{28}$$

By formatting the loss purely as the logarithm of a sum of exponentials, we can optimize it using the *Log-Sum-Exp (LSE) trick*[20] (Blanchard et al., 2019; Huang, 2025a) for numerical stability. Because the covariance Σ is shared across all components, its determinant can be elegantly factored out of the LSE operator:

$$\begin{aligned}\pi_k,\mu_k,\Sigma = \arg\min_{\pi_k,\mu_k,\Sigma}\mathbb{E}_Z\Bigg[&\frac{1}{2}\log|\Sigma| - c(Z) \\ &- \log\sum_{k=1}^{K}\exp\left(\log\pi_k - \frac{1}{2}(Z-\mu_k)^T\Sigma^{-1}(Z-\mu_k) - c(Z)\right)\Bigg] \\ \text{where}\quad c(Z) = \max_{j\in\{1\dots K\}}&\left(\log\pi_j - \frac{1}{2}(Z-\mu_j)^T\Sigma^{-1}(Z-\mu_j)\right)\end{aligned} \tag{29}$$

During the forward pass, the network computes the Mahalanobis distance of the current joint embedding Z to all K prototypes. The optimization of this objective function governs the latent geometry through two distinct, opposing forces:

1. The *Pulling Force* (Data-Fit via Soft-Routing): the Log-Sum-Exp (LSE) function mathematically behaves as a smooth, differentiable approximation of the maximum operator ($\max$). When computing the gradients of the LSE term during backpropagation, the derivative yields the exact softmax probabilities[21], which perfectly correspond to the posterior mixture responsibilities $\gamma_k(Z)$. Consequently, the majority of the gradient signal is routed exclusively to the dominant prototype, i.e. the one with the smallest Mahalanobis distance to Z. This disproportionate gradient update actively pulls the embedding Z and the winning prototype μ_k closer together, effectively performing a probabilistic, soft-clustering alignment.

[19] This assumption can be loosened, though keeping it shared drastically improves tractability and numerical stability.

[20] The Log-Sum-Exp (LSE) trick is a standard technique used to prevent numerical overflow when computing the logarithm of a sum of exponentials. Evaluating $\exp(x_k)$ directly for large values of x_k easily exceeds machine precision, resulting in `NaN` or infinity. By defining $c = \max_k x_k$, we can leverage the algebraic identity $\log\sum_{k=1}^{K}\exp(x_k) = c + \log\sum_{k=1}^{K}\exp(x_k - c)$. This ensures the largest exponent evaluated is exactly $\exp(0) = 1$, safely bounding the sum to prevent overflow while harmlessly allowing negligible values to underflow to zero.

[21] To see this mathematically, consider the Log-Sum-Exp function $f(\mathbf{x}) = \log\sum_{j=1}^{K}\exp(x_j)$. By applying the chain rule, the partial derivative with respect to a specific logit x_k is evaluated as $\frac{\partial f}{\partial x_k} = \frac{1}{\sum_{j=1}^{K}\exp(x_j)}\frac{\partial}{\partial x_k}\sum_{j=1}^{K}\exp(x_j) = \frac{\exp(x_k)}{\sum_{j=1}^{K}\exp(x_j)}$, which is precisely the definition of the standard softmax function.

2. The *Pushing Force* (Geometric Regularization): simultaneously, the model must prevent the trivial collapse of all prototypes to a single point [22]. This defense is natively governed by the shared log-determinant complexity penalty, $\frac{1}{2}\log|\Sigma|$. This term acts as an *information-theoretic regularizer* that strictly penalizes the internal volume of the cluster covariances. To successfully encode a highly diverse, full-rank dataset without inflating Σ (which would incur a massive NLL penalty), the optimization process is mathematically forced to distribute the K prototypes widely across the latent space to cover the data manifold.

The Local Compression vs. Global Spread Trade-off: similar to the GJE joint objective case (Eq.3), ultimately, balance of these two forces creates a profound geometric trade-off. On a local level, minimizing the complexity penalty $\frac{1}{2}\log|\Sigma|$ explicitly *discourages* diversity within *individual* clusters by compressing their covariance volume. However, because the inverse matrix Σ^{-1} dictates the Mahalanobis distance in the data-fit term, this extreme local compression causes penalty for misaligned embeddings to explode. To survive this massive data-fit term without lazily inflating Σ, the network is forced to *encourage* diversity on a *global* level, i.e. to distribute the clusters *widely* in the latent space. It cannot rely on a few massive, overlapping clusters to absorb the dataset; instead, it must spread the discrete prototypes μ_k extensively across the space to cover all training data points. By aggressively compressing the local variance, the objective natively transfers the burden of representing the dataset's total variance onto the global, discrete prototypes, dynamically learning a rich representation space without requiring heuristic uniformity losses.

Extension to Non-Uniform Covariances. While sharing a single covariance matrix Σ across all components drastically improves computational efficiency and numerical stability without losing versatile approximation power (an even stronger realisation of the isotropic approximation property (McLachlan & Basford, 1988; Bishop, 1994)), the GMJE framework natively extends to the fully general case where each mixture component k maintains its own distinct covariance matrix Σ_k. In this unconstrained formulation, the local complexity penalty $\frac{1}{2}\log|\Sigma_k|$ is dependent on the component index k and therefore cannot be factored out of the LSE operator. Following the same derivation, the fully general prototypical loss expands to:

$$\begin{aligned}\mathcal{L}_{\text{GMJE-Proto}} &= -\log\left(\sum_{k=1}^{K}\exp\left(\log\pi_k - \frac{1}{2}(Z-\mu_k)^T\Sigma_k^{-1}(Z-\mu_k) - \frac{1}{2}\log|\Sigma_k|\right)\right)\\ &= -c(Z) - \log\sum_{k=1}^{K}\exp\Bigg(\log\pi_k - \frac{1}{2}(Z-\mu_k)^T\Sigma_k^{-1}(Z-\mu_k)\\ &\qquad - \frac{1}{2}\log|\Sigma_k| - c(Z)\Bigg)\\ \text{where}\quad c(Z) &= \max_{j\in\{1\dots K\}}\left(\log\pi_j - \frac{1}{2}(Z-\mu_j)^T\Sigma_j^{-1}(Z-\mu_j) - \frac{1}{2}\log|\Sigma_j|\right)\end{aligned} \tag{30}$$

Geometrically, this allows the latent space to capture *heterogeneous* data distributions. Certain prototypes can learn to encode tight, highly specific concepts (driving their individual $|\Sigma_k|$ to be small), while others can encode broader, more variable concepts (tolerating a larger $|\Sigma_k|$ to minimize the data-fit term for diverse samples). However, this increased expressiveness comes at an increased computational cost, requiring K independent matrix inversions (Σ_k^{-1}) and log-determinants ($\log|\Sigma_k|$) per optimization step, and may require careful regularization (such as diagonal jitter) to prevent any individual Σ_k from collapsing to singularity (Huang, 2025a).

While optimizing fixed global prototypes via relaxed EM successfully captures macroscopic semantic clusters, it relies on static parameters (μ_k, Σ_k, π_k) that do not adapt to the specific epistemic ambiguity of individual inputs. To achieve true instance-level predictive flexibility while maintaining a fixed number of components K, we explore integrating Mixture Density Networks into the GMJE framework.

[22] If we do not minimize $\frac{1}{2}\log|\Sigma|$, the network would become lazy: it would collapse all prototypes μ_k into the center of the space and inflate Σ to be extremely wide, covering all data points at once (a single giant overlapping blob). By compressing the local variance, the discrete means μ_k are forced to carry the burden of the dataset's global variance.

C.3 GMJE-MDN: MIXTURE DENSITY NETWORK (MDN) FOR GMJE LEARNING

Standard neural networks trained with a sum-of-squares (MSE) or cross-entropy errors are fundamentally limited: they converge to approximate the conditional average of the target data, conditioned on the input (Bishop, 1994). For applications involving multi-valued mappings or complex inverse problems, this conditional average represents a very limited statistical description. Particularly, in a multi-modal space, the average of several correct target values is not necessarily a correct value itself, frequently leading to completely erroneous results.

C.3.1 MDN

To obtain a complete statistical description of the data, Bishop introduced the Mixture Density Network (MDN) (Bishop, 1994), a framework obtained by combining a conventional neural network with a mixture density model. An MDN can, in principle, represent arbitrary conditional[23] probability distributions in the same way that standard networks represent arbitrary functions. Specifically, the conditional probability density of the target data t is represented as a linear combination of K kernel functions (Bishop, 1994):

$$p(t|x) = \sum_{i=1}^{K} \alpha_i(x)\phi_i(t|x) \tag{31}$$

where the mixing coefficients $\alpha_i(x)$ act as prior probabilities conditioned on x. For continuous variables, the kernel functions $\phi_i(t|x)$ are typically chosen to be Gaussians:

$$\phi_i(t|x) = \frac{1}{(2\pi)^{c/2}\sigma_i(x)^c} \exp\left\{-\frac{||t-\mu_i(x)||^2}{2\sigma_i(x)^2}\right\} \tag{32}$$

where c denotes the dimensionality of the target vector $t \in \mathbb{R}^c$, $\mu_i(x)$ represents the center of the i-th kernel, and $\sigma_i(x)^2$ represents a common, scalar variance[24]. Note that, this formulation inherently assumes the components of the target vector are statistically independent *locally within* each individual kernel (i.e. an isotropic covariance structure). While this assumption can be relaxed by introducing full covariance matrices, such a complication is theoretically unnecessary (Bishop, 1994); a Gaussian mixture model utilizing purely *isotropic* kernels can approximate any given density function to arbitrary accuracy, provided the mixing coefficients and Gaussian parameters are correctly chosen (McLachlan & Basford, 1988). Thus, the overall representation remains completely general. In particular, even when constructed purely from isotropic components, the full mixture $p(t|x)$ natively captures global cross-dimensional dependencies through the spatial arrangement of its multiple centers $\mu_i(x)$, avoiding the strict global independence assumed by a single isotropic Gaussian[25].

In the standard MDN architecture, a feed-forward neural network takes the input x and produces raw outputs (logits), denoted here as $h(x)$, that directly parameterize this mixture model. Specifically, the component centers $\mu_i(x)$ are derived directly from the linear network outputs[26]:

$$\mu_i(x) = h_i^{\mu}(x) \tag{33}$$

[23] MDNs model conditional distributions: in an MDN, those parameters are not fixed constants; instead, they depend on, and are dynamic functions of, an input vector x. We feed an input x into a feed-forward neural network, and the network's final layer outputs the specific means $\mu_i(x)$, variances $\sigma_i(x)^2$, and mixing weights $\alpha_i(x)$ to be used for the target variable t. The GMM therefore models the conditional density function $p(t|x)$, i.e. the probability of the target t, conditioned on the fact that we just observed input x (Bishop, 1994).

[24] Note on notation in THIS section: throughout this brief background review, we preserve the original mathematical symbols used by Bishop (Bishop, 1994). For conceptual clarity, Bishop's dynamic mixing weights $\alpha_i(x)$ are directly analogous to the mixture weights (π_k or γ_k) in our GMJE framework, his Gaussian kernel functions $\phi_i(t|x)$ correspond to our component probability densities $\mathcal{N}(\cdot)$, and the generic input-target variables (x, t) map exactly to our context and target embeddings (z_c, z_t). He used c as a notation of dimension, while we used d.

[25] Here is an intuitive example: imagine a 2D space where we want to model a diagonal line (a strong correlation between dimension x and dimension y). A single isotropic Gaussian is just a single perfect circle. It assumes x and y are totally independent globally. It cannot stretch diagonally to fit the line. However, if we take 10 isotropic Gaussians (10 small perfect circles) and line them up diagonally corner-to-corner, the overall shape they create is a diagonal line.

[26] See Eq.(27) in (Bishop, 1994).

The mixing coefficients $\alpha_i(x)$ are constrained by a softmax function applied to their corresponding network outputs to ensure they lie in the range $(0, 1)$ and sum to unity[27]:

$$\alpha_i(x) = \frac{\exp(h_i^\alpha(x))}{\sum_{j=1}^{K} \exp(h_j^\alpha(x))} \tag{34}$$

Importantly, the variances $\sigma_i(x)$ are represented using an exponential function of the network outputs[28]:

$$\sigma_i(x) = \exp(h_i^\sigma(x)) \tag{35}$$

This exponential formulation serves two critical purposes. First, it prevents pathological zero states (since $\exp(\cdot)$ is strictly positive). Second, and more importantly, it allows the variance to be directly and non-linearly parameterized by the input x. By doing so, the MDN breaks free from the *homoscedasticity trap* (constant variance) inherent to classic joint Gaussians. For example, when the neural network observes an input that maps to a noisy or ambiguous region (e.g. $x = 0$), it can dynamically output a large variance ($\sigma = 0.5$). Conversely, when it observes a highly certain input (e.g. $x = 1$), it can tighten the distribution to a very small variance ($\sigma = 0.05$).

The network is then optimized end-to-end by minimizing the negative logarithm of the mixture's likelihood[29] for an observed pattern (x, t):

$$E = -\log\left(\sum_{i=1}^{K} \alpha_i(x)\phi_i(t|x)\right) \tag{36}$$

C.3.2 GMJE-MDN

The GMJE framework integrates the core probabilistic principles of MDN, but fundamentally *re-architects* them for self-supervised representation learning. A standard MDN is designed specifically to model the conditional probability density $p(t|x)$. As discussed in Section.C.1, mapping this directly to our embeddings (i.e. modeling $p(z_t|z_c)$ via a neural network) would break the representational symmetry between the context and target views, losing the explicit modeling of the marginal distribution $p(z_c)$ that provides the native geometric regularization preventing representation collapse.

However, as illustrated in Appendix.L (which was noted later in executing one of our vision benchmarks), if we parameterize the network directly from the full joint embedding $Z = [z_c^T, z_t^T]^T$ introduces a fatal loophole: *Identity Collapse or Information Leakage*. Because the network can observe z_t, it trivially learns the identity function ($\mu_k(Z) = Z$), shrinking the predicted variance to zero and ignoring the semantic manifold entirely.

To resolve this conflict, i.e. preventing both identity collapse and representation collapse, we design a GMJE-MDN architecture which explicitly factorizes the joint objective: $-\log p(z_c, z_t) = -\log p(z_c) - \log p(z_t|z_c)$. As shown in Fig.7, we enforce a strict *Information Bottleneck*: a parameter network is permitted to observe *only* the context embedding z_c, utilizing it to dynamically predict the conditional mixture parameters $(\mu_k(z_c), \Sigma_k(z_c), \pi_k(z_c))$ required to evaluate the target z_t. This design separates prediction and evaluation. To maintain the symmetric joint framework and prevent encoder collapse, this conditional MDN objective ($-\log p(z_t|z_c)$) is coupled with a marginal regularization loss on z_c (e.g. via a global parametric covariance constraint). Consequently, GMJE-MDN learns a dynamic, multi-modal alignment generator while preserving the probabilistic guards necessary for robust self-supervised learning.

Formally, the complete end-to-end training objective for the GMJE-MDN architecture evaluates to:

$$\begin{aligned}
\mathcal{L}_{\text{GMJE-MDN}} &= -\log p(z_c, z_t) \\
&= \underbrace{-\log p(z_c)}_{\text{Marginal Regularization}} \quad \underbrace{-\log p(z_t \mid z_c)}_{\text{Conditional MDN}} \\
&= \left(\frac{1}{2} z_c^T \Sigma_c^{-1} z_c + \frac{1}{2}\log|\Sigma_c|\right) - \log\left(\sum_{k=1}^{K} \pi_k(z_c)\mathcal{N}\Big(z_t \;\Big|\; \mu_k(z_c), \Sigma_k(z_c)\Big)\right)
\end{aligned} \tag{37}$$

[27] See Eq.(25) in (Bishop, 1994).
[28] See Eq.(26) in (Bishop, 1994).
[29] Same as Eq.(29) in (Bishop, 1994).

where Σ_c is a global covariance matrix[30] governing the marginal distribution of the context space, and the conditional parameters $\mu_k(z_c)$, $\Sigma_k(z_c)$, and $\pi_k(z_c)$ are dynamic functions of the context embedding z_c, predicted directly by the parameter network. During training, *the dual encoders and the parameter network are optimized simultaneously*. By minimizing $\mathcal{L}_{\text{GMJE-MDN}}$ via standard gradient descent, gradients flow backward from the conditional mixture evaluation into the target encoder, and through the parameter network into the context encoder, requiring *no alternating optimization phases or heuristic stop-gradients*.

This explicit factorization in Eq.9 is the mathematical key to the architecture's stability. The right-hand term (the MDN objective) acts as a highly expressive, multi-modal *data-fit* routing mechanism. Simultaneously, the left-hand term (the full marginal negative log-likelihood) acts as a compound geometric guard against representation collapse. Similar to GJE (joint loss, Eq.3) and GMJE-Proto (Eq.29), minimizing the log-determinant alone would trivially collapse the space to zero volume. However, because a collapsed Σ_c forces its inverse Σ_c^{-1} to explode, the Mahalanobis distance term ($\frac{1}{2} z_c^T \Sigma_c^{-1} z_c$) fiercely penalizes any degenerate shrinkage. This elegant tension continuously maintains a well-conditioned, full-rank context space, satisfying the symmetric constraints required by self-supervised learning without restricting the target predictions to a rigid linear covariance structure.

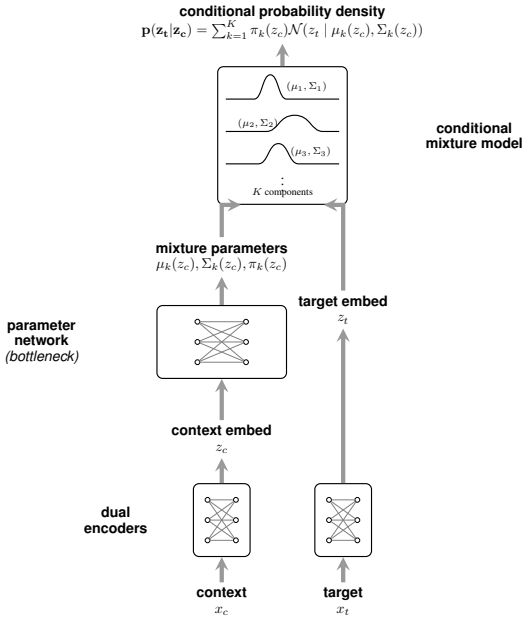


Figure 7: The Information Bottleneck in the GMJE-MDN architecture. To prevent identity collapse, the parameter network must take only the context embedding z_c as input to predict the conditional mixture parameters. These generated parameters are then evaluated against the target embedding z_t inside the conditional mixture model. To prevent representation collapse, this conditional objective is coupled with a marginal loss on z_c.

The GMJE-MDN architecture provides powerful, dynamic conditional routing. However, *both the prototypical EM approach and the MDN approach inherently rely on a fixed, pre-defined K*. In many real-world scenarios, the true number of semantic modes is unknown, and the underlying data manifold features highly irregular, non-linear ridges. To resolve this structural rigidity and to dynamically discover both K and the manifold's topology, we introduce a growing graph-based approach.

[30]This covariance matrix can be tracked via an empirical batch covariance or a moving average during training. If we track Σ_c via EMA, then Σ_{EMA} trails behind the current batch. Because it is a historical average, $\nabla_z(z^T \Sigma_{\text{EMA}}^{-1} z) = 2\Sigma_{\text{EMA}}^{-1} z$. The gradients do not cancel. The expansive Mahalanobis barrier works exactly as intended.

C.4 GMJE-GNG: Dynamic Prototype Discovery via Growing Neural Gas

While the mixture components in the parametric GMJE objective (i.e. the joint context-target embedding distribution, Eq.8) can be optimized through relaxed EM (Eq.29), it relies on a critical structural assumption: *the number of mixture modes, K, must be pre-defined*. Further, standard gradient-based optimization or k-means initializations are notoriously susceptible to local minima, particularly when the underlying joint distribution features highly overlapping or elongated geometric structures (Bishop, 2006a; Jain, 2010).

To resolve the sensitivity to initialization and dynamically discover the optimal number of mixture components, we propose an topological initialization and routing mechanism for learning the GMM components based on *Growing Neural Gas* (GNG) (Fritzke, 1994). Unlike standard clustering, GNG is a *topology-preserving*, *density-matching* algorithm. It treats the GMJE global prototypes as nodes in a dynamic graph. Rather than forcing a predetermined K, GNG organically spawns new prototypes in regions of high latent error, mapping both the modes and the connecting "ridges" (manifold topology) of the joint representation space.

C.4.1 Mathematical Formulation

Let the global prototypes be represented as a set of vertices $\mathcal{V}$ with associated reference vectors (our joint means) $\mu_v \in \mathbb{R}^{2d}$, connected by a set of unweighted edges $\mathcal{E}$. The network is initialized with simply two nodes, $\mathcal{V} = \{v_a, v_b\}$, initialized at random data points.

During the forward pass, we sample a joint embedding $Z = [z_c^T, z_t^T]^T$ from the data distribution. We locate the two nearest prototypes (the "winner" s_1 and "runner-up" s_2) measured by the squared Euclidean distance:

$$\begin{aligned} s_1 &= \arg\min_{v \in \mathcal{V}} \|Z - \mu_v\|^2 \\ s_2 &= \arg\min_{v \in \mathcal{V} \setminus \{s_1\}} \|Z - \mu_v\|^2 \end{aligned} \tag{38}$$

Instead of a harsh winner-takes-all update (which causes local minima), GNG applies a soft topological update. Importantly, rather than relying on a rigid spatial radius to define this neighborhood, GNG determines proximity strictly through the active topological connections (the edge set $\mathcal{E}$). We directly move the winner s_1 and all its direct topological neighbors $\mathcal{N}(s_1) = \{n \in \mathcal{V} \mid (s_1, n) \in \mathcal{E}\}$ towards the observed embedding Z using learning rates ϵ_b and ϵ_n (where $\epsilon_b \gg \epsilon_n$):

$$\begin{aligned} \mu_{s_1} &\leftarrow \mu_{s_1} + \epsilon_b(Z - \mu_{s_1}) \\ \mu_n &\leftarrow \mu_n + \epsilon_n(Z - \mu_n), \quad \forall n \in \mathcal{N}(s_1) \end{aligned} \tag{39}$$

To dynamically grow the mixture model, the winner accumulates a local quantization error $E_{s_1} \leftarrow E_{s_1} + \|Z - \mu_{s_1}\|^2$. After every λ iterations, the algorithm identifies the prototype u with the maximum accumulated error, and its neighbor v with the highest error. A new prototype mode w is organically spawned precisely between them to alleviate the density mismatch:

$$\mu_w = \frac{\mu_u + \mu_v}{2} \tag{40}$$

The edges (u, v) are removed, and new edges (u, w) and (v, w) are created.

C.4.2 The GNG Algorithm for GMJE Component Learning

By sequentially feeding the joint embeddings into this dynamic graph, the continuous data manifold is optimally discretized. The final set of vertices directly yields our K optimal global prototypes μ_k. This explicit procedure is detailed in Algo.2

Algorithm 2 GMJE-GNG: Dynamic GMJE Component Discovery via GNG

Require: Stream of joint embeddings Z, growth interval λ, max nodes K_{max}, learning rates ϵ_b, ϵ_n, max edge age a_{max}, local error decay α, global error decay β.

1: Initialize vertices $\mathcal{V} = \{v_1, v_2\}$ with random Z, and edges $\mathcal{E} = \{(v_1, v_2)\}$.
2: Initialize errors $E \leftarrow \mathbf{0}$.
3: **for** each sampled Z in training step t **do**
4: Find nearest prototype s_1 and second nearest s_2 to Z.
5: Update winner error: $E_{s_1} \leftarrow E_{s_1} + \|Z - \mu_{s_1}\|^2$.
6: Pull s_1 towards Z: $\mu_{s_1} \leftarrow \mu_{s_1} + \epsilon_b(Z - \mu_{s_1})$.
7: Pull topological neighbors towards Z: $\mu_n \leftarrow \mu_n + \epsilon_n(Z - \mu_n)$ for $n \in \mathcal{N}(s_1)$.
8: Create or refresh edge (s_1, s_2) by setting its age to 0.
9: Increment age of all edges connected to s_1.
10: Remove edges with age $> a_{max}$. If any vertices have no remaining edges, remove them from $\mathcal{V}$.
11: **if** $t \pmod{\lambda} == 0$ **and** $|\mathcal{V}| < K_{max}$ **then**
12: Find $u = \arg\max_{v \in \mathcal{V}} E_v$.
13: Find neighbor $v = \arg\max_{n \in \mathcal{N}(u)} E_n$.
14: Insert new prototype w with $\mu_w = \frac{1}{2}(\mu_u + \mu_v)$.
15: Update graph: $\mathcal{V} \leftarrow \mathcal{V} \cup \{w\}$, add edges $(u, w), (v, w)$, remove (u, v).
16: Decay local error: $E_u \leftarrow \alpha E_u$, $E_v \leftarrow \alpha E_v$, $E_w \leftarrow E_u$.
17: **end if**
18: Decay all global errors: $E_v \leftarrow \beta E_v$ for all $v \in \mathcal{V}$.
19: **end for**
20: **Return** Extracted prototypes $\mu_k = \mu_v$ for $v \in \mathcal{V}$.

Note that, we incorporate an edge aging mechanism (Lines 9-10) to accurately map dynamically changing topologies. Because the prototypes μ_k are continuously updated and move through the latent space, initial edges may eventually span empty regions between divergent clusters. By incrementing the age of inactive edges and removing those that exceed a_{max}, the graph organically severs obsolete topological connections. This severing mechanism is critical, as it allows the algorithm to natively break apart and isolate entirely disjoint semantic clusters without enforcing artificial connectivity across the data manifold.

Complexity. Computationally, the per-sample time complexity of Algo.2 is dominated by the search for the two nearest prototypes (Eq.38 and Line 4) and the topological neighbor updates (Line 7). Let $D = 2d$ denote the dimensionality of the joint embedding (z_c, z_t). Both operations scale *linearly* with the maximum number of components, yielding an asymptotic time complexity of $\mathcal{O}(K_{max} \cdot D)$ per step. Memory-wise, the algorithm is exceptionally lightweight; it only requires storing the K_{max} prototype vectors and an adjacency matrix to track edge ages, resulting in a space complexity of $\mathcal{O}(K_{max} \cdot D + K_{max}^2)$. Because K_{max} (the number of semantic concepts) is fundamentally bounded and typically orders of magnitude smaller than the massive memory banks (whose size is often denoted as M, discussed next in Section.4.1) required by non-parametric contrastive frameworks, this mechanism adds negligible overhead compared to their heavy $\mathcal{O}(M \cdot d)$ memory footprint.

The methods discussed thus far comprise the **parametric branch** of GMJE, focusing on learning a highly compressed, finite set of structural modes (whether globally static, conditionally dynamic, or topologically grown). But what if we discard the concept of a pre-specified or estimated K entirely, and instead treat every single observed data point as its own mixture component? This question bridges GMJE directly into the realm of *non-parametric density estimation*, with connection to modern *Contrastive Learning*.

D Contrastive Learning as Non-Parametric GMJE: Full Derivation

The GMJE framework provides a unified mathematical perspective on modern representation learning; instead of learning K fixed global prototypes, suppose we utilize a non-parametric formulation

where we treat every previously encoded data pair in a memory bank[31] (or current batch) of size M, denoted $\{(z_c^{(m)}, z_t^{(m)})\}_{m=1}^M$, as an *equiprobable* mode[32].

This is equivalent to performing *Kernel Density Estimation*[33] (KDE (Härdle, 1991)) on the joint distribution. We define the joint distribution as a *uniform* mixture of M joint Gaussians *centered precisely on these data points*:

$$p(z_c, z_t) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{N}\left(\begin{bmatrix} z_c \\ z_t \end{bmatrix} \middle| \begin{bmatrix} z_c^{(m)} \\ z_t^{(m)} \end{bmatrix}, \Sigma \right) \tag{41}$$

We note the connection of this approach to standard Contrastive Learning. Contrastive methods inherently assume that, *conditional on being generated by a specific instance* m, the augmentations (the positive pairs, e.g. two crops of the same image) forming the context and target views are statistically independent and uniformly spherical (Wu et al., 2018; Hjelm et al., 2019; Bachman et al., 2019; He et al., 2020; Chen et al., 2020; Wang & Isola, 2022). Mathematically, this corresponds to restricting the component covariance Σ to an isotropic, block-diagonal matrix governed by a scalar variance (temperature) hyperparameter τ:

$$\Sigma = \begin{bmatrix} \tau I & \mathbf{0} \\ \mathbf{0} & \tau I \end{bmatrix}$$

Because the off-diagonal cross-covariance blocks are strictly zero, the joint Gaussian component factorizes exactly into the product of two independent marginals:

$$p(z_c, z_t) = p(z_c) \times p(z_t) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{N}(z_c \mid z_c^{(m)}, \tau I) \mathcal{N}(z_t \mid z_t^{(m)}, \tau I) \tag{42}$$

Again, if we wish to evaluate the conditional NLL, $-\log p(z_t \mid z_c)$, of a true matching pair (z_c, z_t) under this non-parametric mixture, we first apply Bayes' theorem. By integrating the joint distribution over z_t, the marginal distribution trivially evaluates to $p(z_c) = \frac{1}{M} \sum_{j=1}^{M} \mathcal{N}(z_c \mid z_c^{(j)}, \tau I)$. The conditional distribution is thus:

$$p(z_t \mid z_c) = \frac{p(z_c, z_t)}{p(z_c)} = \frac{\sum_{m=1}^{M} \mathcal{N}(z_c \mid z_c^{(m)}, \tau I) \mathcal{N}(z_t \mid z_t^{(m)}, \tau I)}{\sum_{j=1}^{M} \mathcal{N}(z_c \mid z_c^{(j)}, \tau I)} \tag{43}$$

To simplify this into the standard *contrastive* format, we apply two structural assumptions. *First, we assume all embeddings are L^2-normalized* ($\|z\|^2 = 1$). Expanding the squared Euclidean distance inside the isotropic Gaussian density yields:

$$\begin{aligned} \mathcal{N}(z_c \mid z_c^{(m)}, \tau I) &= \frac{1}{(2\pi\tau)^{d/2}} \exp\left(-\frac{1}{2\tau} \|z_c - z_c^{(m)}\|^2 \right) \\ &= \frac{1}{(2\pi\tau)^{d/2}} \exp\left(-\frac{1}{2\tau} (\|z_c\|^2 - 2 z_c^T z_c^{(m)} + \|z_c^{(m)}\|^2) \right) \\ &= \underbrace{\frac{e^{-1/\tau}}{(2\pi\tau)^{d/2}}}_{C_0} \exp\left(\frac{z_c^T z_c^{(m)}}{\tau} \right) \end{aligned} \tag{44}$$

[31] The memory bank is essentially a dictionary storing the target embeddings for all recent images.

[32] Note that, adding a new data point (i.e. treating a sample as a new mode) increases the number of mixture components M, but it does not alter the dimensionality of the underlying probability distribution. The GMM continues to operate strictly within the fixed $2d$-dimensional joint continuous space, consistent with the primal, feature space view.

[33] KDE is a *non-parametric mixture model*. It places a Gaussian bump on every single data point and sums their probabilities: $p(z) \propto \Sigma \mathcal{N}(z|z_i, \sigma^2)$. Because it is a sum of Gaussians, it is multi-modal.

Second, in instance-level discrimination[34], the context prototype for mode m is effectively represented by its paired target view in the memory bank[35], implying $z_c^{(m)} \approx z_t^{(m)}$. Substituting this and the expanded Gaussian density (Eq.44) into the denominator of Eq.43, the marginal distribution simplifies to $p(z_c) = \frac{C_0}{M} \sum_{j=1}^{M} \exp(z_c^T z_t^{(j)}/\tau)$.

For the numerator in Eq.43, we evaluate the joint probability $p(z_c, z_t) = \frac{1}{M} \sum_{m=1}^{M} \mathcal{N}(z_c \mid z_c^{(m)}, \tau I)\mathcal{N}(z_t \mid z_t^{(m)}, \tau I)$. Suppose the continuous observation (z_c, z_t) is a true matching pair generated by mode m^*. By definition[36], its target embedding z_t perfectly coincides with its own memory bank entry $z_t^{(m^*)}$. The target Gaussian for this specific matching mode evaluates exactly at its peak density, as the distance is zero:

$$\mathcal{N}(z_t \mid z_t^{(m^*)}, \tau I) = \mathcal{N}(z_t \mid z_t, \tau I) = \frac{1}{(2\pi\tau)^{d/2}} \exp(0) = (2\pi\tau)^{-d/2} \equiv C_1$$

Conversely, for all non-matching modes $m \neq m^*$, the distance $\|z_t - z_t^{(m)}\|^2 \gg 0$. Under the standard contrastive assumption where the variance τ is extremely small (the "low temperature" limit), the exponential term $\exp(-\|z_t - z_t^{(m)}\|^2/2\tau)$ decays to practically zero. Consequently, the summation over the M joint components is overwhelmingly dominated by the single matching mode m^*, allowing us to drop the remaining terms:

$$p(z_c, z_t) \approx \frac{1}{M}\mathcal{N}(z_c \mid z_c^{(m^*)}, \tau I)\mathcal{N}(z_t \mid z_t^{(m^*)}, \tau I) = \frac{1}{M}\mathcal{N}(z_c \mid z_c^{(m^*)}, \tau I) \cdot C_1$$

Applying our previous L^2-expansion (Eq.44) to the context Gaussian, and substituting our proxy assumption $z_c^{(m^*)} \approx z_t^{(m^*)} = z_t$, the density for the context view evaluates to $C_0 \exp(z_c^T z_t/\tau)$.

Substituting this simplified numerator, alongside the denominator we derived for $p(z_c)$, back into Bayes' theorem (Eq.43) yields the final conditional probability:

$$p(z_t \mid z_c) = \frac{p(z_c, z_t)}{p(z_c)} \approx \frac{\frac{1}{M} C_0 \exp(z_c^T z_t/\tau) \cdot C_1}{\frac{1}{M} C_0 \sum_{j=1}^{M} \exp(z_c^T z_t^{(j)}/\tau)} = C_1 \frac{\exp(z_c^T z_t/\tau)}{\sum_{m=1}^{M} \exp(z_c^T z_t^{(m)}/\tau)}$$

Taking the negative natural logarithm, the scaling constants separate cleanly into an additive constant $C = -\log C_1$, and the log-likelihood factorizes into:

$$-\log p(z_t \mid z_c) = -\log \frac{\exp(z_c^T z_t/\tau)}{\sum_{m=1}^{M} \exp(z_c^T z_t^{(m)}/\tau)} + C \tag{45}$$

Remarkably, Eq.10 is precisely the **InfoNCE loss** (van den Oord et al., 2019) heavily utilized in state-of-the-art Contrastive Learning frameworks (e.g. SimCLR (Chen et al., 2020), MoCo (He et al., 2020)). This connection between contrastive learning and our GMJE constitutes a major theoretical result of this work: *standard Contrastive Learning is mathematically proven to be a*

[34] In standard instance-level discrimination (like SimCLR (Chen et al., 2020) or MoCo (He et al., 2020)), $z_c^{(m)}$ and $z_t^{(m)}$ are just two different augmented views of the exact same underlying image (instance m). The entire contrastive objective is explicitly designed to make the network invariant to these augmentations, pulling these two vectors together. Over the course of training, they naturally converge to represent roughly the exact same point on the hypersphere.

[35] In practical implementations like MoCo, we do not store both a context bank and a target bank. We only cache the target embeddings (the "keys" from the momentum encoder) in a single memory queue. When computing the denominator (the negative samples), the anchor z_c is compared directly against these stored target keys $z_t^{(j)}$. Therefore, the stored target key $z_t^{(m)}$ is forced to act as the single proxy prototype for the entire instance.

[36] In the context of instance-level discrimination, the current observation (z_c, z_t) represents the embeddings of two augmented views from a specific image (instance m^*) in the current forward pass. Because z_t is exactly the target representation generated for this instance, it is mathematically identical to the target key $z_t^{(m^*)}$ stored in the memory bank for that specific slot. Consequently, the distance $\|z_t - z_t^{(m^*)}\|^2$ is strictly zero, causing the Gaussian density to evaluate at its absolute peak, which elegantly simplifies the numerator.

special, degenerate case of the GMJE framework. Specifically, it is estimating a density using a *non-parametric*, *isotropic* GMM.

Establishing this mathematical equivalence required two structural assumptions which reveal the theoretical shortcomings of standard contrastive methods and articulates the advantages of our fully generalized GMJE approach:

- *The Isotropic, Independent Covariance Assumption:* by restricting the covariance Σ of the instance-based joint distribution (Eq.41) to an isotropic block-diagonal matrix (τI), contrastive methods inherently assume all latent dimensions vary equally and independently. More critically, they force the cross-covariance block (Σ_{ct}) to strictly zero. Mathematically, setting $\Sigma_{ct} = \mathbf{0}$ breaks all relational links between the context and target spaces, partitioning them into disconnected universes. To force alignment despite this mathematical flaw, contrastive frameworks must rely on a brute-force architectural workaround: they mandate strict *Siamese weight sharing* ($\theta = \theta'$, or slow copying) and define positive pairs via dot-product maximization (Eq.10), which physically forces the representations to occupy the exact same coordinate on the unit hypersphere ($z_c \approx z_t$). Consequently, contrastive learning acts strictly as an *invariance learner*: it minimizes loss by systematically erasing any dynamic features that make the two views distinct (e.g. erasing the semantic difference between a dog "sitting" vs "running"). In contrast, by evaluating a full covariance structure, GMJE preserves the cross-covariance block (Σ_{ct}) which natively models the transformation rules between the two states. This allows z_c and z_t to reside at distinct coordinates (preserving rich, view-specific descriptive features) while flawlessly aligning them via the learned cross-view mutual information.
- *The Non-Parametric, Instance-Level Assumption:* by treating every individual data point as an independent mode (the non-parametric KDE formulation), contrastive learning ignores global semantic clustering. Because the objective operates strictly at the instance level without any semantic awareness, it inherently suffers from the "class collision" (or "false negative") problem (Khosla et al., 2020; Chuang et al., 2020). In standard instance-level discrimination, any two distinct images in a batch or memory queue are mathematically forced to be negative samples and are repelled in the latent space, even if they represent the exact same underlying concept (e.g. two different images of a golden retriever). This forces the network to expend representational capacity memorizing arbitrary, instance-level granularities to keep semantically identical images apart, rather than learning generalized class features. Further, accurately estimating the marginal distribution (the denominator) using an unweighted KDE requires an enormous number of modes M, necessitating computationally expensive, massive memory banks. Transitioning to the *parametric* GMJE (Sections.4 to 4) resolves these class collision and memory bottlenecks by replacing the M instances with $K \ll M$ learnable semantic prototypes (i.e. the parametric GMJE approach).

Empirical Optimization and the Mahalanobis Trace Trap. While the theoretical advantages of a full-covariance generative objective are profound, realizing them empirically introduces a critical optimization vulnerability. As detailed in Appendix N, attempting to optimize the pure joint distribution using a dynamically computed empirical batch covariance triggers the "*Mahalanobis Trace Trap*". This algebraic cancellation mathematically eradicates the exact cross-view pulling force (C_{ct}^{-1}) we wish to preserve, causing representations to disconnect and dimensionally collapse. Consequently, the three advanced formulations proposed in this work, i.e. Parametric Decoupling via Prototypes (Section 4), Conditional Factorization via MDN (Section 4), and Dynamic Non-Parametric Density via SMC (Section 4.2), are not merely alternative design choices; they are carefully engineered structural remedies designed specifically to break this algebraic cancellation, safely restoring the generative pulling force while explicitly maximizing entropy.

While the fully parametric GMJE framework bypasses the limitations of instance discrimination entirely, one may still wish to leverage the non-parametric M-mode formulation typical of contrastive learning. However, our theoretical bridge exposes a fundamental operational flaw in this standard setup: it relies on a naive, uniform, un-informative prior FIFO queue to manage its massive number of modes. To resolve this uniform-prior limitation, the rigid queue must be replaced by a dynamic weighting mechanism to mitigate redundant repelling. We address this next by transforming the memory bank into a probability-weighted particle system via *Sequential Monte Carlo*.

E SMC MEMORY BANKS: DETAILS

In Section.4.1, we established a profound theoretical equivalence: standard contrastive learning is mathematically equivalent to estimating a density using a *Non-Parametric Gaussian Mixture Model*. Under this non-parametric GMJE framework, we can mathematically view every single one of the M negative keys in a contrastive memory bank as the center (μ_m) of a tiny, fixed-variance Gaussian component. The memory bank itself *is* the Gaussian Mixture Model representing the marginal distribution $p(z_c)$.

Architecturally, frameworks such as MoCo (He et al., 2020) implement this memory bank via a *First-In-First-Out* (FIFO) queue to manage representation staleness. However, from a probabilistic standpoint, this FIFO queue implicitly assumes a *rigid, uniform prior*: it assumes every single component (key) in the mixture has the exact same mixing weight ($\pi_m = \frac{1}{M}$). When a new batch of embeddings is generated, the oldest embeddings are deterministically discarded regardless of their informational value.

This unweighted FIFO mechanism is theoretically suboptimal. It frequently discards highly informative representations (e.g. rare classes or "hard negatives") simply because they are temporally old, while simultaneously retaining redundant, uninformative samples (e.g. continuous streams of easy background images). To resolve this, we propose abandoning the uniform prior and formulating the memory bank as a dynamic, probability-weighted particle system using *Sequential Monte Carlo* (SMC) (Doucet et al., 2001; Chopin & Papaspiliopoulos, 2020), commonly known as *Particle Filtering*.

If the memory bank is a non-parametric GMM, forcing a uniform prior ($\pi_m = \frac{1}{M}$) via a FIFO queue is mathematically restrictive and suboptimal. SMC perfectly instantiates the GMJE solution by relaxing this constraint:

- *The Particles are the Mixture Components:* in SMC, the particles are not merely data points; they are explicitly the M components of the non-parametric GMM.
- *Dynamically Updating the Weights:* instead of a uniform prior, SMC mathematically computes the true posterior mixing weights (π_m) of those components using their likelihoods under the current batch.
- *Resampling:* it dynamically reallocates the mixture components to cluster densely around the hardest, most complex parts of the data manifold, preventing redundant uniform coverage.

E.0.1 THE MEMORY BANK AS A PARTICLE SYSTEM

Under the SMC framework, we treat the M stored embeddings as a set of discrete particles $\{Z^{(m)}\}_{m=1}^{M} = \{(z_c^{(m)}, z_t^{(m)})\}_{m=1}^{M}$ used to approximate the continuous marginal distribution of the latent space. Importantly, each particle is assigned a dynamic importance weight $W^{(m)}$, constrained such that $\sum_{m=1}^{M} W^{(m)} = 1$.

The instance-based joint distribution (Eq.41) is thus generalized into a weighted Kernel Density Estimate:

$$p(z_c, z_t) = \sum_{m=1}^{M} W^{(m)} \mathcal{N}\left(\begin{bmatrix} z_c \\ z_t \end{bmatrix} \middle| \begin{bmatrix} z_c^{(m)} \\ z_t^{(m)} \end{bmatrix}, \Sigma \right) \tag{46}$$

By carrying these weights through the exact Bayes' theorem derivation previously established, the conditional contrastive objective (Eq.10) elegantly upgrades into an *SMC-weighted InfoNCE loss*:

$$-\log p(z_t \mid z_c) = -\log \frac{\exp(z_c^T z_t/\tau)}{\sum_{m=1}^{M} W^{(m)} \exp(z_c^T z_t^{(m)}/\tau)} + C \tag{47}$$

E.0.2 SEQUENTIAL IMPORTANCE WEIGHTING AND RESAMPLING

To dynamically maintain the optimal set of particles, the SMC memory bank updates via two principled steps during each training iteration t:

1. Importance Weight Update: as a new batch of query embeddings Z_{query} arrives, we evaluate *how well our existing particles explain this new data*. Particles that consistently yield high likelihoods under the current queries (i.e. those that act as "hard negatives" by lying densely close to the current manifold) should be amplified. The unnormalized weight $\tilde{W}_t^{(m)}$ of each particle is updated recursively based on its similarity to the current batch:

$$\tilde{W}_t^{(m)} = W_{t-1}^{(m)} \times \mathbb{E}_{z_c \in Z_{query}} \left[\exp(z_c^T z_t^{(m)} / \tau) \right] \tag{48}$$

the weights are then normalized to sum to one: $W_t^{(m)} = \tilde{W}_t^{(m)} / \sum_j \tilde{W}_t^{(j)}$.

2. Sequential Importance Resampling (Replacing FIFO): In a standard Sequential Monte Carlo particle filter, the system will inevitably suffer from weight degeneracy over time, where a few highly relevant particles accumulate all the probability mass. To prevent this, we execute a *resampling* step at every iteration, effectively replacing the naive deterministic FIFO drop mechanism:

- We form a combined pool of size $M + B$ (the existing memory bank plus the incoming batch).
- We sample exactly M particles from this combined pool with replacement, drawing with probabilities proportional to their normalized posterior weights $W_t^{(m)}$.
- Highly weighted particles (valuable hard negatives) are naturally retained or even duplicated (cloned), while particles with vanishing weights (redundant or useless easy negatives) are probabilistically discarded.
- All weights in the newly sampled memory bank of size M are then reset to a uniform $1/M$ to begin the next cycle.

To monitor the health of this dynamic sampling process, we track the Effective Sample Size (ESS $= 1/\sum(W^{(m)})^2$) over the combined pool before resampling. A dropping ESS correctly indicates that the filter is aggressively concentrating its capacity on dense, hard-negative regions of the manifold rather than maintaining uniform coverage.

Dual Optimization Dynamics. We have dual processes here: optimizing the neural network parameters of the two encoders and maintaining the memory bank. The SMC weight update and resampling operations are purely algebraic, gradient-free processes (Eq.11). Simultaneously, the encoder networks are optimized via standard backpropagation using the SMC-weighted InfoNCE loss (Eq.47), wherein the particle weights $W^{(m)}$ act as constant scalars scaling the contrastive pushing force. Therefore, *the InfoNCE Loss + Backprop creates better representations, while the SMC Update creates a better memory bank to contrast those representations against.*

Theoretical Advantages. Integrating SMC provides a mathematically principled solution to "hard negative mining" - a practice that typically relies on messy, ad-hoc heuristics in standard contrastive learning (Robinson et al., 2021; Kalantidis et al., 2020). Because hard negatives inherently yield higher likelihoods under the current batch's distribution, they organically accumulate higher importance weights $W^{(m)}$, dominating the denominator of Eq.47 and providing stronger, more informative gradient signals. Further, by selectively retaining high-value representations across many epochs, SMC dramatically increases the informational density of the memory bank, potentially allowing for much smaller queue sizes M without sacrificing representation quality.

E.0.3 Algorithm: SMC-GMJE for Dynamic Memory Bank Optimization

The complete procedure for dynamically updating the particle-based memory bank during contrastive training is detailed in Algo.1. By caching the dot-products already calculated during the forward pass, the weight update adds negligible overhead to standard instance-level discrimination.

Complexity. Computationally, the time complexity of the SMC-GMJE memory update is strictly bounded. The importance weight update (Line 12) requires the dot products between the B context queries and the $M + B$ pooled memory bank particles. However, because these exact dot products $\exp(z_c^T z_t^{(m)}/\tau)$ are largely already computed to evaluate the InfoNCE denominator in the forward pass, they can be cached and re-used. Thus, the weight update only requires an $\mathcal{O}(M)$ summation overhead. The resampling step (Line 15), implemented via systematic resampling or the alias

method, operates in linear $\mathcal{O}(M)$ time. Therefore, the overall time complexity remains $\mathcal{O}(B \cdot M \cdot d)$ per step, asymptotically identical to standard contrastive methods such as MoCo (He et al., 2020). Space complexity is similarly preserved; storing the weights adds a trivial $\mathcal{O}(M)$ scalars to the $\mathcal{O}(M \cdot d)$ memory bank footprint.

It is important to emphasize that Algo.1 is designed specifically for the non-parametric, isotropic special case of GMJE, which we established in Section.4.1 is mathematically equivalent to standard contrastive learning. This is evident from the mathematical formulation of the algorithm: (1) *The Loss Function* (Line 5) explicitly optimizes the SMC-weighted InfoNCE loss (Eq.47, same as used in contrastive learning); (2) *The Covariance Assumption* (Line 12) utilizes the dot-product similarity $\exp(z_c^T z_t^{(m)}/\tau)$, which only emerges when assuming a strictly isotropic covariance $\Sigma = \tau I$ over L^2-normalized embeddings; and (3) *The Components* (Line 11) iterate over discrete particles in a memory bank, treating every individual data instance as a fixed mode rather than optimizing global, learnable prototypes. Therefore, SMC-GMJE is a direct, drop-in upgrade for contrastive learning frameworks such as MoCo (He et al., 2020). It takes the standard contrastive architecture but replaces the heuristically uninformative FIFO queue with a probabilistically rigorous particle filter.

SMC Beyond Contrastive Learning: Generalizing SMC to general GMJE. While Algo.1 specifically formulates SMC-GMJE as a non-parametric drop-in replacement for standard contrastive learning (relying on the InfoNCE objective and isotropic, fixed-variance assumptions), the SMC mechanism is theoretically universal. It is not fundamentally restricted to the special contrastive case. As detailed in Appendix.M, SMC can be directly generalized to optimize the full, parametric GMJE objective (Algo.6). In this generalized setting, the particles are no longer restricted to isotropic spherical bounds; instead, each particle m can be evaluated and dynamically re-weighted using the exact general joint NLL formulated in Eq.8. Under this generalized SMC-GMJE formulation, the network organically maintains a probability-weighted particle system of full-covariance components, seamlessly bridging the gap between massive non-parametric memory banks and rich, multi-modal density estimation.

E.1 Generative GMJE: Details

A direct consequence of the probabilistic GMJE framework is that, by explicitly optimizing the NLL of the embedding space, the network does not merely learn a discriminative mapping function $x \to z$; it natively learns a continuous, closed-form generative probability density function over the latent manifold[37].

Recall that the architecture explicitly models the $2d$-dimensional joint distribution of the context and target embeddings, $Z = [z_c^T, z_t^T]^T$, as a GMM (Eq.6):

$$p(z_c, z_t) = \sum_{k=1}^{K} \pi_k \mathcal{N}\left(\begin{bmatrix} z_c \\ z_t \end{bmatrix} \middle| \begin{bmatrix} \mu_{c,k} \\ \mu_{t,k} \end{bmatrix}, \begin{bmatrix} \Sigma_{cc,k} & \Sigma_{ct,k} \\ \Sigma_{tc,k} & \Sigma_{tt,k} \end{bmatrix} \right)$$

In standard downstream generative tasks (e.g. unconditional image synthesis or latent data augmentation), we typically require a standard d-dimensional representation, rather than a concatenated $2d$-dimensional relational pair. To obtain this, we must marginalize the joint distribution. By the fundamental properties of multivariate Gaussians, integrating out the context variable z_c trivially yields the marginal distribution of the target representation space $p(z_t)$, which elegantly remains a closed-form GMM:

$$p(z_t) = \int p(z_c, z_t) dz_c = \sum_{k=1}^{K} \pi_k \mathcal{N}(z_t \mid \mu_{t,k}, \Sigma_{tt,k}) \tag{49}$$

Because we possess this explicit, marginalized generative formulation, we can *unconditionally generate novel, mathematically valid embeddings* without requiring any real input data x. Using e.g. Gaussian Mixture Approximation (GMA) sampling (Huang, 2025a), we can draw *synthetic*, meaningful representations z_t via a highly efficient, gradient-free two-step sampling procedure:

[37] From a Bayesian perspective, any valid probability distribution over the data or latent space natively constitutes a generative model. Modern generative frameworks (e.g. GANs, VAEs, or Score-Based models (Huang, 2022)) implicitly or explicitly learn these underlying distributions or their gradient fields.

1. *Discrete Mode Selection:* sample a component index k from the categorical distribution defined by the learned mixing weights, $k \sim \text{Categorical}(\pi_1, \ldots, \pi_K)$.
2. *Continuous Feature Generation:* given the selected mode k, draw a continuous d-dimensional embedding vector z_t from its corresponding marginalized Gaussian component, $z_t \sim \mathcal{N}(\mu_{t,k}, \Sigma_{tt,k})$, computed practically via the Cholesky decomposition of the target covariance matrix $\Sigma_{tt,k} = A_k A_k^T$ such that $z_t = \mu_{t,k} + A_k \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, I)$.

In contrast, standard self-supervised frameworks (e.g. JEPA, SimCLR, SwAV) only train *discriminative* encoders. Because they lack a normalized probability density function $p(z)$ and do not penalize "empty space" between representations, attempting to sample random coordinates from their latent spaces invariably yields out-of-distribution (OOD) garbage vectors[38]. By contrast, the GMJE latent space is a strict generative manifold. The ability to sample directly from it opens the door to powerful downstream applications, e.g. generative replay for continual learning, latent data augmentation, and unconditional image synthesis, which we evaluate later in Section.5.

Summary of the GMJE Framework. This concludes the mathematical formulation of the GMJE paradigm. To summarize the structural organization of this generalized framework:

- **Sections 4 (Reduced EM), 4 (GMJE-MDN), and 4 (GMJE-GNG)** constitute the **Parametric GMJE** branch. Here, the architecture learns K highly compressed, full-covariance prototypes (whether globally static, conditionally dynamic, or topologically grown) to map the semantic manifold, and does not require a memory bank. The learned GMM can be used for downstream discriminative or generative tasks.
- **Sections 4.1 (the InfoNCE bridge) and 4.2** (SMC-GMJE) constitute the **Non-Parametric GMJE** branch. Here, the architecture treats M encoded data instances as fixed, isotropic modes, which fundamentally requires a dynamic memory bank to estimate the marginal distribution (the contrastive denominator).

F Additional Experimental Results and Discussion

Motivation: multi-modal inverse prediction. A classical difficulty in regression arises when the underlying mapping is *multi-valued*. Bishop (Bishop, 1994) illustrated this using the inverse kinematics of a two-joint robot arm: while the forward problem is deterministic, the inverse problem is ambiguous, since multiple joint configurations can produce the same end-point coordinate. When such a problem is trained with squared loss, a deterministic neural predictor converges to the conditional mean[39] rather than to the distinct valid solutions. In multi-modal settings, this average may lie between modes, in regions unsupported by the true data distribution.

The same issue appears naturally in self-supervised representation learning. Predicting a target view from a heavily degraded or partially observed context view is often an inverse problem: a single context may be compatible with several plausible target completions. For example, if x_c contains only a partial view of a dog, then the corresponding target x_t may plausibly depict different breeds, poses, or body configurations. A deterministic predictor trained with MSE cannot represent such ambiguity explicitly and instead tends to average over the valid modes. This motivates a generative formulation capable of routing predictions toward multiple conditional possibilities.

Objective. One of the central claims of this work is that deterministic predictive architectures (e.g. classic JEPA) and unimodal generative models (e.g. single-Gaussian GJE) are limited when the true conditional distribution $p(x_t \mid x_c)$ is strongly multi-modal. To examine this claim in a controlled setting, we construct synthetic ambiguous-alignment tasks inspired by inverse problems (Bishop, 1994). Because the ground-truth data-generating process (DGP) is known exactly, these experiments allow us to compare how different architectures recover the underlying conditional structure, both visually and probabilistically.

[38] An exception is the RiJEPA architecture (Huang & Raza), which learns a manifold, reshaped by rule constraints, in the embedding space via a shared predictor; it can then be used to generate, e.g., new rule sets.

[39] A predictor trained with squared loss converges to the conditional mean $\mathbb{E}[y \mid x]$ under standard risk minimization (Bishop, 2006b).

Task design. To isolate the predictive and generative mechanisms from representation learning itself, we bypass the dual encoders in this experiment and treat x_c and x_t as direct observations. This lets us focus exclusively on the conditional modeling problem.

To examine this last point more closely, we inspect the internal parameter routing of GMJE-MDN in Fig. 3 and Fig. 9. Several patterns are evident. First, the learned conditional means in panel (b) closely follow the ground-truth branch functions. Second, the predicted conditional standard deviations in panel (c) remain near the true observation noise level $\epsilon = 0.05$, indicating that the model captures the local dispersion without artificially inflating uncertainty. Third, the conditional mixing weights in panel (a) remain close to the ground-truth uniform value of $1/3$, which is consistent with the fact that the generating process selects each branch equiprobably. Taken together, these results suggest that GMJE-MDN recovers the synthetic conditional density substantially more faithfully than the alternative baselines.

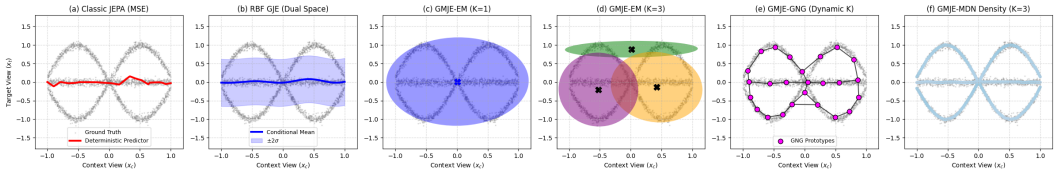


Figure 8: Predictive distributions on Dataset B (Intersecting Branches). The overall trends are consistent with Dataset A but the intersection structure makes the task more challenging. (a) Classic JEPA again collapses toward the conditional average near zero. (b) The dual-space RBF kernel baseline remains unimodal and cannot isolate the intersecting branches. (c–d) The fixed-covariance parametric GMJE baselines capture part of the multi-modal structure but remain limited by global geometric rigidity. (e) GMJE-GNG provides an adaptive topological approximation of the manifold. (f) GMJE-MDN most accurately captures the intersecting conditional density among the compared methods.

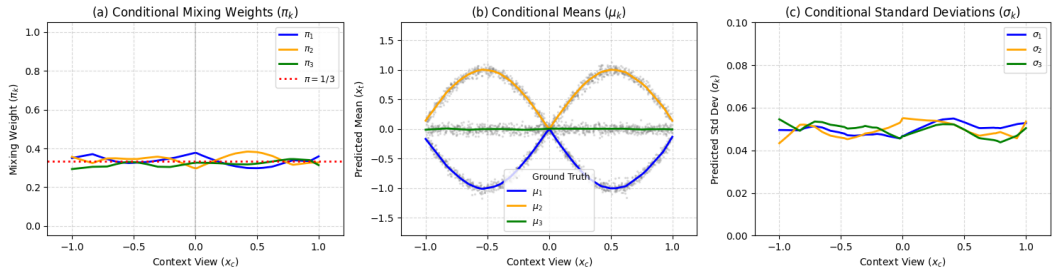


Figure 9: Internal parameter routing of GMJE-MDN on Dataset B. The learned means track the three intersecting target branches, the standard deviations remain close to the true observation noise, and the mixing weights remain near the ground-truth value of $1/3$ across the context domain.

Summary of synthetic experiments. Across both ambiguous-alignment tasks, the experiments support the theoretical picture developed in the preceding sections. Deterministic prediction collapses towards the conditional mean and therefore fails to model multi-valued targets. Unimodal dual-space and single-Gaussian primal-space models capture only coarse averages of the conditional structure and cannot represent several branches simultaneously. Fixed-K parametric mixtures improve mode separation but remain limited by rigid global covariance geometry. In contrast, the non-parametric GMJE-GNG variant preserves the manifold topology more effectively, while the instance-conditional GMJE-MDN delivers the most faithful recovery of the underlying multi-modal conditional density. These results provide a clear qualitative demonstration that generative joint modeling is substantially better suited than deterministic prediction for ambiguous semantic alignment.

F.1 VISION BENCHMARKS: EXTENDED ANALYSIS

Analysis. Fig.4 shows that the two memory-bank strategies behave similarly during the early training phase, but diverge substantially later. The FIFO baseline begins to exhibit loss instability after approximately 500 epochs, whereas SMC-GMJE remains more stable and achieves a higher final linear-probe accuracy. A plausible explanation is that, once the latent space begins to stabilize and becomes more structured, and distinct semantic clusters (e.g. dogs, cars, ships) are forming, a small FIFO queue ($M = 256$) is increasingly unable to preserve sufficiently informative negatives across all semantic regions (insufficient to cover all 10 CIFAR-10 classes simultaneously). Because FIFO deterministically overwrites the oldest representations regardless of their informational value, it frequently purges rare classes or "hard negatives" (e.g. overwriting the only "ship" embeddings in the queue simply because they expired). This leaves the network temporarily blind to certain semantic boundaries, resulting in severe loss volatility and a suppressed final linear probing accuracy of 78.08%.

By contrast, the SMC bank adaptively reweights particles according to their current utility. This is reflected in the ESS curve: during the early warmup stage the ESS remains high, indicating relatively uniform importance across particles; however, precisely when the semantic clusters begin to form (epoch 500), the ESS decreases as probability mass concentrates on a smaller subset of more informative negatives (hard-negative particles are becoming scarce but informative). Rather than uniformly exploring, it assigns them massive posterior weights and clones them during the sequential resampling step, keeping them alive in the 256-slot memory bank long past their temporal expiration date. Under this severe memory constraint, the resulting contrastive signal is more stable and yields stronger downstream performance (**81.99%** final accuracy, which is +3.91% over FIFO), which empirically proves that transitioning from a deterministic queue to a probabilistic particle system dramatically increases the informational density of the memory bank, yielding stable, rich gradient signals even under extreme hardware limitations.

Analysis. Table 1 shows that SimCLR, BYOL, and MoCo v2 all reach approximately 79-82% linear-probe accuracy after 200 epochs pre-training, whereas SMC-GMJE reaches 74.52%. Thus, under this short training budget, the non-parametric GMJE variant is competitive but does not yet surpass the strongest heuristic baselines. This is an important complement to the long-horizon memory experiment above: the 1000-epoch $M = 256$ study isolates the benefits of SMC under severe queue constraints, while the 200-epoch benchmark evaluates overall representation quality in a broader SSL comparison.

SMC-GMJE trains stably and improves steadily, suggesting that the non-parametric particle-bank formulation is viable in practice: by replacing the rigid FIFO queue with a dynamic SMC particle filter, the memory bank successfully tracks the marginal distribution[40], and provides stable, informative gradients without succumbing to weight degeneracy or representation collapse. However, the gap to SimCLR, BYOL, and MoCo indicates that the current implementation remains *under-optimized* relative to those mature baselines. This is not surprising: the heuristic baselines have benefited from extensive design refinement, whereas SMC-GMJE is a first principled instantiation of the non-parametric GMJE view.

Summary of vision benchmarks. Taken together, the vision experiments support two conclusions. First, *under strong memory constraints*, dynamic particle reweighting is a viable and often more effective alternative to a rigid FIFO queue, yielding a more informative contrastive memory bank and more stable long-horizon training. Second, when evaluated as a practical SSL method on CIFAR-10, the current non-parametric GMJE implementation learns competitive representations but does not yet outperform the strongest heuristic baselines under short pre-training. These findings support the usefulness of the GMJE perspective while also indicating that further optimization and larger-scale tuning are needed for it to fully match mature contrastive and predictive SSL systems.

[40] In contrastive learning, the memory bank serves as an empirical approximation of the marginal distribution $p(z_c)$ over the latent space, which constitutes the denominator of the InfoNCE objective. Geometrically, this represents the overall density and shape of the dataset's manifold. While a rigid FIFO queue indiscriminately discards older representations—thereby losing track of true density peaks, the SMC particle filter dynamically re-weights and resamples particles based on their likelihoods. This ensures the memory bank continuously and accurately maps the high-density regions and hard-negative ridges of the underlying latent landscape.

We also tested SwAV, VICReg, Parametric-GMJE, and GMJE-MDN in the 200-epoch CIFAR-10 setting. As they were unstable or under-tuned when trained from scratch in this short regime, they are not reported in Table 1; they likely require stronger initialization, longer schedules, or more carefully matched objectives.

F.2 Generative sampling: extended analysis

Results and analysis. By jointly examining the geometric statistics in Table 2, the latent topology in Fig. 5, and the decoded synthetic samples in Fig. 5, we observe three distinct generative behaviors.

- *Post-hoc density fitting on SimCLR latents.* SimCLR learns useful discriminative representations, but its latent space is not trained as an explicit unconditional generative density. When a post-hoc GMM is fitted to the frozen embeddings, the resulting mixture places non-negligible probability mass in low-density regions between semantic clusters. This is visible in Fig. 5(a), where sampled points frequently appear in inter-cluster regions rather than concentrating inside the class manifolds. After decoding, these off-manifold samples lead to blurrier and more class-ambiguous digits, as shown in Fig. 5(a). The relatively small prototype spread reported in Table 2 is consistent with this weak macroscopic separation.
- *Over-smoothing in unimodal GJE.* The unimodal GJE baseline maintains substantial latent variance and volume (Table 2), but a single Gaussian component is too restrictive to model the multi-class structure of MNIST. As a result, unconditional samples cover the latent space broadly but lack explicit multi-modal separation. This is reflected in Fig. 5(b), where sampled points are dispersed without clear cluster structure, and in Fig. 5(b), where the decoded images appear over-smoothed and less diverse. In this sense, the unimodal model captures global density but not class-conditional structure.
- *Structured multi-modality in GMJE.* Parametric GMJE combines explicit latent density modeling with multiple separated mixture components. Table 2 shows that it achieves the largest prototype spread while preserving non-degenerate component volumes and overall variance. In Fig. 5(c), latent samples drawn from the learned mixture align much more closely with clustered semantic regions than either of the two baselines. Correspondingly, the decoded samples in Fig. 5(c) are sharper and more visually diverse. Taken together, these results suggest that GMJE provides a substantially better latent generative model than either post-hoc density fitting on a discriminative space or a single-Gaussian latent approximation.

Summary. These results highlight an important distinction between *discriminative* and *generative* latent geometry. A representation space that is effective for downstream discrimination does not automatically define a good unconditional sampling distribution. Post-hoc density fitting can partially recover latent structure, but it remains mismatched to an encoder that was not trained with a generative objective. Likewise, a single Gaussian latent model captures only coarse global geometry. By contrast, GMJE directly learns a multi-modal latent density during representation learning itself, which leads to both better-aligned sampled embeddings and more plausible decoded images.

Overall, the MNIST latent-sampling experiment supports the claim that GMJE is not only a representation learner but also a meaningful latent generative model. Compared with a post-hoc GMM on SimCLR and a unimodal GJE density, GMJE produces better-separated latent samples and the strongest qualitative unconditional generations among the tested models. This provides empirical evidence that explicit multi-modal latent density modeling is a useful mechanism for bridging self-supervised representation learning and generative sampling.

F.3 Extended discussion

Unifying SSL Paradigms. Prior to this work, the SSL landscape was largely developed through separate design traditions. In this work, we show that GMJE provides a common probabilistic perspective that connects several of these fragmented directions:

- *The Contrastive Paradigm (Non-Parametric GMJE):* we show that standard contrastive learning methods (e.g. SimCLR, MoCo) can be interpreted as a degenerate non-parametric

special case of the GMJE framework (Section 4.1). This perspective highlights a limitation of standard FIFO memory queues, namely that sample replacement is based on age rather than informativeness. To address this, we introduce an *SMC Particle Bank* (Section 4.2), which adaptively reweights memory samples so that informative hard negatives are emphasized while stale or redundant samples are downweighted.

- *The Predictive Paradigm (Parametric GMJE):* we address the multi-modal alignment difficulty that arises in inverse-style augmentation tasks by replacing deterministic predictors (classic JEPA) with dynamic conditional mixtures (*GMJE-MDN*). In addition, to reduce the rigidity of fixed mixture cardinality K, we introduce a topology-preserving *Growing Neural Gas* variant (*GMJE-GNG*) which adaptively adds prototypes according to local quantization error.

The Elimination of Asymmetric Heuristics. Many empirically successful non-contrastive predictive methods, including BYOL and related Siamese architectures, rely on asymmetric architectural or optimization mechanisms, e.g. EMA target encoders, stop-gradients, or predictor asymmetry, to mitigate representation collapse (Grill et al., 2020; Chen & He, 2021; Zhuo et al., 2023). In contrast, the GMJE formulation is designed to mitigate dimensional collapse through an explicit covariance-aware geometric regularizer, including the log-determinant term $\frac{1}{2}\log|\Sigma|$. Under our probabilistic interpretation, the resulting tension between the Mahalanobis data-fitting term and the covariance volume penalty promotes a non-degenerate, full-rank representation geometry. As a result, gradients can be propagated symmetrically through both encoders without relying on the asymmetric dampers or rank-differential mechanisms that are commonly used in prior non-contrastive SSL methods.

Generative vs. Discriminative Manifolds. This work connects discriminative representation learning and explicit generative modeling in latent space. Standard predictive and contrastive architectures do not natively define a continuous probabilistic density over the representation space. As demonstrated in our generative experiments (Section.5.3), applying post-hoc density estimation to a uniformly scattered contrastive space (e.g. SimCLR) inevitably samples from low-density gaps, producing out-of-distribution chimeric artifacts. Conversely, strictly unimodal constraints collapse generative sampling into an over-smoothed, ambiguous average. In contrast, GMJE equips the latent space with an explicit Gaussian-mixture density model during representation learning itself. By balancing local variance compression with global prototype spread, it constructs a mathematically valid generative manifold capable of crisp, unconditional sampling without sacrificing discriminative utility.

Dynamic Component Selection and GMJE-GNG. A persistent challenge in parametric density estimation is selecting the number of mixture components, K, in advance. In our Parametric GMJE experiments, K was fixed empirically (e.g. $K = 50$ in Section 5.3) as a rough approximation to the semantic granularity of the dataset. In realistic, uncurated settings, however, the true number of latent modes is rarely known beforehand. Overestimating K can lead to fragmented semantic classes and inactive or redundant prototypes, whereas underestimating K can force several distinct modes to be absorbed into overly broad components, thereby blurring semantic boundaries. Traditional model-selection criteria such as AIC or BIC are mainly post-hoc and are difficult to integrate into end-to-end deep representation learning.

Fortunately, our non-parametric formulation (*SMC-GMJE*, or in general *DaM-GMJE*) inherently bypasses this limitation. By dynamically tracking the marginal distribution using a SMC particle filter, it maps the latent manifold without ever requiring an a priori declaration of K. Within a purely parametric architecture, we have *GMJE-GNG* as well. Rather than relying on a fixed K, GMJE-GNG represents the marginal distribution through a dynamic topological graph. By monitoring localized representation error and the evolving density of the latent manifold, the model can recruit new Gaussian prototypes in regions requiring additional capacity and prune dormant components when they are no longer useful. In this way, the generative memory bank can adapt its structural complexity to the observed topology of the data, reducing the need to hand-tune K in advance.

Future work. Future extensions of this framework will investigate the integration of GMJE into sequential and autoregressive settings, such as video prediction and Large Language Models

(LLMs), where predictive ambiguity is often inherently multi-modal. Another promising direction is to combine the continuous density modeling of GMJE with the discrete neuro-symbolic constraints (e.g. RiJEPA (Huang & Raza)), with the aim of learning more structured and causally informed generative manifolds.

G PRIMAL-GJE LEARNING OBJECTIVE: JOINT VS. CONDITIONAL LIKELIHOOD

In Primal-GJE (Section.3), by modeling the context/target representations as jointly Gaussian, our optimization objective can be formulated from two distinct mathematical perspectives: the *generative* (joint) perspective and the *predictive* (conditional) perspective.

G.1 THE GENERATIVE OBJECTIVE (JOINT NLL)

To learn the optimal encoder weights θ and θ' directly, we minimize the Negative Log Likelihood (NLL) of the joint distribution. Let $z_i = [z_{c,i}^T, z_{t,i}^T]^T \in \mathbb{R}^d$ represent a single concatenated context-target embedding pair from the batch. The probability density function (pdf) of our zero-mean joint Gaussian is:

$$p(z_{c,i}, z_{t,i}) = \frac{1}{\sqrt{(2\pi)^d |C_{joint}|}} \exp\left(-\frac{1}{2} \begin{bmatrix} z_{c,i} \\ z_{t,i} \end{bmatrix}^T C_{joint}^{-1} \begin{bmatrix} z_{c,i} \\ z_{t,i} \end{bmatrix}\right) \tag{50}$$

Taking the negative natural logarithm and dropping the constant term, the joint loss function evaluated at this single data point becomes:

$$\begin{aligned} \mathcal{L}_{\text{joint}}^{(i)}(\theta, \theta') &= \underbrace{\frac{1}{2} z_i^T C_{joint}^{-1} z_i}_{\text{data-fit term}} + \underbrace{\frac{1}{2} \log |C_{joint}|}_{\text{regularizer}} \\ &= \frac{1}{2} \begin{bmatrix} z_{c,i} \\ z_{t,i} \end{bmatrix}^T \begin{bmatrix} C_{cc} & C_{ct} \\ C_{tc} & C_{tt} \end{bmatrix}^{-1} \begin{bmatrix} z_{c,i} \\ z_{t,i} \end{bmatrix} + \frac{1}{2} \log \left| \begin{bmatrix} C_{cc} & C_{ct} \\ C_{tc} & C_{tt} \end{bmatrix} \right| \end{aligned} \tag{51}$$

The second term, i.e. the log-determinant complexity penalty $\log |C_{joint}|$, serves as *the geometric regularizer* which prevents the volume of the embeddings from expanding infinitely to trivially minimize the data-fitting term. Note that this single-sample loss is exactly the summand of the full empirical batch objective $\mathcal{L}_{\text{Primal-GJE}}$ (Eq.3). It is this strict *generative joint objective* $p(z_c, z_t)$ that mathematically underpins the Primal-GJE formulation. This stands in contrast to the Dual-GJE (GPJE) formulation derived earlier in Section 3, which optimizes the Gaussian Process marginal likelihood - a strictly *conditional* distribution[41] $p(Z_t \mid Z_c)$ that models the target features given the context inputs, thereby lacking a native marginal regularizer for the context space.

G.2 THE PREDICTIVE OBJECTIVE (CONDITIONAL NLL)

Alternatively, as z_c and z_t are jointly Gaussian, the conditional distribution of the target given the context, $p(z_t|z_c)$, is guaranteed to be Gaussian. Using the Schur complement (Bishop, 2006b), the conditional distribution is exactly $\mathcal{N}(\mu_{t|c}, \Sigma_{t|c})$ with[42]:

$$\begin{aligned} \mu_{t|c} &= C_{tc} C_{cc}^{-1} z_c \\ \Sigma_{t|c} &= C_{tt} - C_{tc} C_{cc}^{-1} C_{ct} \end{aligned} \tag{cc.Eq.4}$$

This allows us to formulate a conditional objective, $\mathcal{L}_{\text{cond}} \propto -\log p(z_t|z_c)$, i.e. taking the negative log likelihood of $p(z_t|z_c)$:

$$\mathcal{L}_{\text{cond}}(\theta, \theta') = \frac{1}{2}(z_t - \mu_{t|c})^T \Sigma_{t|c}^{-1} (z_t - \mu_{t|c}) + \frac{1}{2} \log |\Sigma_{t|c}| \tag{52}$$

This formulation is interesting: it reveals that the optimal "predictor" in our architecture is not a black-box neural network, but the closed-form conditional mean, accompanied by a dynamic covariance matrix that quantifies epistemic uncertainty.

[41] Se Eq.(2.29) and Eq.(2.30) in the GP for ML book (Rasmussen & Williams, 2005).

[42] Note that a joint Gaussian distribution is strictly *unimodal* at any conditional slice.

Further, it mathematically bridges our framework with classic JEPA. If one assumes the conditional covariance is a fixed identity matrix ($\Sigma_{t|c} = I$) and drops the resulting constant regularization term, Eq.52 trivially degenerates into a standard Mean Squared Error (MSE) objective: $\mathcal{L}_{\text{MSE}} = ||z_t - g(z_c)||^2$, which is exactly the heuristic objective of standard JEPA. Thus, dual-GJE is a strict, probabilistically grounded generalization of JEPA.

G.3 Joint NLL: Closed-Form Inference via Block Inversion

Once the empirical covariance matrix is computed from the training representations, the optimal "predictor" falls out for free. The matrix blocks $[C_{cc}, C_{ct}, C_{tc}, C_{tt}]$ are fixed. Given a new test context z_c^*, we wish to find $p(z_t^* \mid z_c^*)$. By the definition of conditional probability for joint Gaussians, we can derive this using block matrix inversion and the Schur complement.

We partition the inverse covariance matrix (precision matrix) $\Lambda = C_{joint}^{-1}$ as[43]:

$$\begin{aligned} C_{joint}^{-1} = \begin{bmatrix} C_{cc} & C_{ct} \\ C_{tc} & C_{tt} \end{bmatrix}^{-1} &= \begin{bmatrix} \Lambda_{cc} & \Lambda_{ct} \\ \Lambda_{tc} & \Lambda_{tt} \end{bmatrix} \\ &= \begin{bmatrix} C_{cc}^{-1} + C_{cc}^{-1}C_{ct}(C_{tt} - C_{tc}C_{cc}^{-1}C_{ct})^{-1}C_{tc}C_{cc}^{-1} & -C_{cc}^{-1}C_{ct}(C_{tt} - C_{tc}C_{cc}^{-1}C_{ct})^{-1} \\ -(C_{tt} - C_{tc}C_{cc}^{-1}C_{ct})^{-1}C_{tc}C_{cc}^{-1} & (C_{tt} - C_{tc}C_{cc}^{-1}C_{ct})^{-1} \end{bmatrix} \\ &= \begin{bmatrix} (C_{cc} - C_{ct}C_{tt}^{-1}C_{tc})^{-1} & -(C_{cc} - C_{ct}C_{tt}^{-1}C_{tc})^{-1}C_{ct}C_{tt}^{-1} \\ -C_{tt}^{-1}C_{tc}(C_{cc} - C_{ct}C_{tt}^{-1}C_{tc})^{-1} & C_{tt}^{-1} + C_{tt}^{-1}C_{tc}(C_{cc} - C_{ct}C_{tt}^{-1}C_{tc})^{-1}C_{ct}C_{tt}^{-1} \end{bmatrix} \end{aligned} \tag{53}$$

The quadratic form $z^T C_{joint}^{-1} z$ in the joint generative exponential (Eq.50) can be expanded algebraically into $z_c^{*T}\Lambda_{cc}z_c^* + 2z_t^{*T}\Lambda_{tc}z_c^* + z_t^{*T}\Lambda_{tt}z_t^*$. By isolating the terms involving the target z_t^* and completing the square (see, e.g., Bishop, 2006b), we find the conditional distribution is exactly Gaussian, $p(z_t^* \mid z_c^*) = \mathcal{N}(\mu_{t|c}, \Sigma_{t|c})$, with mean and covariance (Eq.4):

$$\begin{aligned} \mu_{t|c} &= C_{tc}C_{cc}^{-1}z_c^* \\ \Sigma_{t|c} &= C_{tt} - C_{tc}C_{cc}^{-1}C_{ct} \end{aligned}$$

This elegantly establishes an exact, mathematically optimal linear projection for the prediction ($\mu_{t|c}$), permanently coupled with calibrated, dataset-driven epistemic uncertainty ($\Sigma_{t|c}$).

G.4 Combating Collapse: Why Joint Optimization is Required

While the conditional objective ($\mathcal{L}_{\text{cond}}$) elegantly maps to predictive tasks, optimizing a deterministic variant of it in isolation is vulnerable to representation collapse. By the laws of probability, the joint likelihood factors exactly into the conditional and marginal likelihoods:

$$-\log p(z_c, z_t) = \underbrace{-\log p(z_t|z_c)}_{\mathcal{L}_{\text{cond}}} \underbrace{-\log p(z_c)}_{\text{marginal regularization}} \tag{54}$$

This factorization provides a profound theoretical symmetry: the marginal term ($-\log p(z_c)$) prevents *context* representation collapse, while the conditional term ($\mathcal{L}_{\text{cond}}$) prevents *target* representation collapse. If we bring in $\mathcal{L}_{\text{cond}}$ from Eq.52 and expand $\log p(z_c)$ in Eq.54:

$$\begin{aligned} \mathcal{L}_{\text{joint}} &= \mathcal{L}_{\text{cond}} + \left(\frac{1}{2}z_c^T C_{cc}^{-1} z_c + \frac{1}{2}\log|C_{cc}|\right) \\ &= \left(\frac{1}{2}(z_t - \mu_{t|c})^T\Sigma_{t|c}^{-1}(z_t - \mu_{t|c}) + \frac{1}{2}\log|\Sigma_{t|c}|\right) + \left(\frac{1}{2}z_c^T C_{cc}^{-1} z_c + \frac{1}{2}\log|C_{cc}|\right) \end{aligned} \tag{55}$$

This reveals two competing forces driving the loss: the *data-fit terms* and the *geometric regularizers*. First, the data-fit terms prevent representation collapse across both branches. If the context or target encoders collapse their respective embeddings, the corresponding auto-covariances (C_{cc} or C_{tt}) approach singularity. The polynomial explosion of their inverses (C_{cc}^{-1} in the marginal distribution,

[43] This equation isn't actually calculating the inverse; it simply defines the block variables. Letting the Schur complement be $S = C_{tt} - C_{tc}C_{cc}^{-1}C_{ct}$, the blocks evaluate to: $\Lambda_{tt} = S^{-1}$, $\Lambda_{ct} = -C_{cc}^{-1}C_{ct}S^{-1}$, $\Lambda_{tc} = -S^{-1}C_{tc}C_{cc}^{-1}$, and $\Lambda_{cc} = C_{cc}^{-1} + C_{cc}^{-1}C_{ct}S^{-1}C_{tc}C_{cc}^{-1}$. See, e.g., Bishop (2006b) for block matrix inversion.

and the resulting $\Sigma_{t|c}^{-1}$ in the conditional distribution) drives the loss to infinity, strictly guarding against trivial collapsed solutions.

Conversely, the complexity penalty terms, i.e. $\frac{1}{2}\log|\Sigma_{t|c}|$ and $\frac{1}{2}\log|C_{cc}|$, act as a *geometric regularizer* bounded by information-theoretic principles. They directly regularize the *differential entropy* of the learned representations[44], preventing the embeddings from infinitely expanding their variance to artificially minimize the data-fit term. Further, minimizing this joint log-determinant natively forces the network to maximize the *Mutual Information* between the context and target spaces, provided the marginal volumes do not collapse.

This also exposes the fundamental flaw in standard predictive architectures. Classic JEPA heuristically optimizes only a *degenerate*, *deterministic* version of the conditional loss (MSE), dropping the conditional covariance entirely, and it completely discards the marginal distribution of the context space ($-\log p(z_c)$). By throwing away the covariance-dependent terms, classic JEPA loses the native defense against entropy collapse and severs the link to mutual information maximization. Consequently, it is forced to rely on asymmetric stop-gradients and EMA updates to survive. By optimizing the full joint objective ($\mathcal{L}_{\text{joint}}$), Primal-GJE natively preserves these probabilistic and information-theoretic guards, ensuring a diverse, full-rank embedding space across both branches purely through the laws of probability.

H Representation Collapse in Symmetric Dual GJE

As discussed in Section.3, unlike our primal feature-space objectives, the dual objective of GPJE cannot be optimized symmetrically. In this appendix, we provide the formal mathematical proof demonstrating why optimizing the Dual-GJE objective via symmetric gradient descent instantly leads to catastrophic representation collapse, formally justifying the necessity of EMA target networks (Grill et al., 2020; Assran et al., 2023;?).

Consider the exact Dual-GJE negative marginal log-likelihood objective (Eq.2):

$$\mathcal{L} = \frac{1}{2}\text{Tr}\left(Z_t^T K_{cc}^{-1} Z_t\right) + \frac{d_t}{2}\log|K_{cc}| \tag{cc.Eq.2}$$

If the weights of both the context encoder E_θ and the target encoder $E_{\theta'}$ are updated symmetrically *without stop-gradients*, the optimization dynamics follow a deterministic two-step collapse trajectory.

Step 1: The Target Space Collapses to Zero. First, consider the gradients flowing into the target encoder $E_{\theta'}$. The only term in cc.Eq.2 containing the target embeddings Z_t is the data-fit term (the trace term). Because the inverse Gram matrix K_{cc}^{-1} is strictly positive-definite by definition, this trace operation evaluates to a pure, lower-bounded quadratic penalty. To minimize this term to its absolute theoretical minimum (0), the target encoder trivially learns to map all inputs to the origin. Consequently, $Z_t \to \mathbf{0}$, and the trace term becomes exactly 0.

Step 2: The Context Space Collapses to a Constant. Once the target space has collapsed ($Z_t = \mathbf{0}$), the data-fit expansive force is entirely neutralized. The objective degenerates completely to the Complexity Penalty:

$$\mathcal{L}_{\text{degenerated}} = 0 + \frac{d_t}{2}\log|K_{cc}| \tag{56}$$

Now, analyzing the gradients flowing into the context encoder E_θ, the optimizer seeks to minimize $\log|K_{cc}|$. This is equivalent to shrinking the differential volume of the context Gram matrix to zero. To achieve this, the context encoder maps every input image to the exact *same constant embedding vector*. As a result, K_{cc} degenerates into a singular matrix of identical values (up to the ϵI jitter), driving the log-determinant toward $-\infty$.

Thus, the system achieves a loss of $-\infty$ by learning a completely uninformative, degenerate latent space ($Z_c \to$ constant, $Z_t \to \mathbf{0}$).

[44] For Gaussian distributions, the differential entropy $H(\mathbf{x}) \propto \log|\Sigma|$ (**?**).

H.1 The Root Cause: The Asymmetry of the Conditional Likelihood

This catastrophic failure stems from the inherent directional asymmetry of Gaussian Processes. A standard GP models the conditional[45] distribution of outputs given inputs ($X \to Y$). As established in Section 3, the GP *marginal likelihood* used in Dual-GJE mathematically evaluates a strictly *conditional* Negative Log-Likelihood, $p(Z_t \mid Z_c)$.

In traditional machine learning paradigms, this asymmetry is perfectly sound because the targets Y are fixed, ground-truth observations. The conditional objective natively penalizes the volume and complexity of the *input* space ($\log |K_{cc}|$) to enforce smoothness, but it applies no corresponding volume penalty to the *output* space because empirical outputs are inherently immutable.

In the self-supervised representation learning setting, however, *the "targets" Z_t are not fixed*; they are dynamically learned parameters. Because the conditional GP formulation ($p(Z_t \mid Z_c)$) inherently lacks the target-space marginal regularizer ($-\log p(Z_t)$) that would be present in a full joint likelihood formulation, the target embeddings feel no expansive force. This fundamental absence of output regularization leads directly to the Step 1 collapse, mathematically mandating the use of artificial target anchors like EMA.

H.2 The Structural Resolution: EMA and Stop-Gradients

This mathematical vulnerability formally validates the necessity of the architectural heuristics popularized by BYOL (Grill et al., 2020) and JEPA (Assran et al., 2023). To successfully optimize the Dual-GJE objective, we must artificially enforce the assumption of the Gaussian Process: the targets must be treated as fixed observations. By maintaining the target encoder as an EMA of the context encoder and applying a strict stop-gradient to Z_t, the target embeddings act as a diverse, static topological anchor. Because the optimizer is mathematically barred from shifting $Z_t \to \mathbf{0}$, the trace penalty cannot be trivially bypassed. Instead, the trace term correctly functions as an expansive force, compelling the context embeddings Z_c to spread out and strictly match the fixed pairwise diversity of the target memory bank, safely preventing system collapse.

Importantly, this vulnerability is entirely circumvented when transitioning the framework to the symmetric *primal feature space*. Because primal models (like GMJE and HSIC) evaluate the volume of the feature covariance matrices directly, and natively incorporate symmetric regularizers that penalize the collapse of both spaces, they eliminate the need for asymmetric EMA architectures entirely.

I Training Dual-GJE: Kernel Optimization and EMA

In standard Gaussian Process regression, as detailed in classic texts such as Rasmussen & Williams (Rasmussen & Williams, 2005), training the GP model is equivalent to hyperparameter optimization. Given a covariance function k_ϕ parameterized by hyperparameters ϕ (e.g. length-scale ℓ, observation noise σ^2), the optimal parameters ϕ^* are found by maximizing the log marginal likelihood[46] of the observed targets given the inputs.

In the Dual-GJE framework, we extend this principle to deep representation learning. The context encoder weights θ act as the ultimate "hyperparameters" of the kernel, dynamically shaping the input space $Z_c = E_\theta(X_c)$ upon which the Gram matrix is computed. Thus, training the Dual-GJE model involves taking the gradients of the conditional Negative Log-Likelihood (NLL) with respect to both the kernel parameters ϕ and the context encoder weights θ.

As proven in Appendix.H, optimizing the target encoder symmetrically under this conditional objective causes catastrophic collapse. Therefore, the target encoder $E_{\theta'}$ is detached from the computational graph and updated exclusively via an Exponential Moving Average (EMA). The exact $\mathcal{O}(N^3)$ procedure is detailed in Algo.3.

[45] See Eq.(2.29) or Eq.(2.30) in (Rasmussen & Williams, 2005)

[46] Eq.(2.29) or Eq.(2.30) in (Rasmussen & Williams, 2005).

Algorithm 3 Training Exact Dual-GJE Model

Require: Dataset $\mathcal{D}$, Context Encoder E_θ, Target Encoder $E_{\theta'}$. Kernel function k_ϕ with parameters ϕ. Batch size N, Learning rate η, EMA momentum τ.

while network has not converged **do**
  Sample a batch of N augmented view pairs $\{(x_c^{(i)}, x_t^{(i)})\}_{i=1}^N \sim \mathcal{D}$
  // 1. Forward Pass
  Compute context embeddings: $Z_c = E_\theta(X_c) \in \mathbb{R}^{N \times d_c}$
  No-grad: Compute target anchors: $Z_t = \text{stop_grad}(E_{\theta'}(X_t)) \in \mathbb{R}^{N \times d_t}$
  // 2. Exact Dual Covariance Matrix & Loss
  Compute $N \times N$ Gram matrix K_{cc} over Z_c using kernel k_ϕ.
  Add jitter for numerical stability: $K_{cc} \leftarrow K_{cc} + \epsilon I_N$
  Compute conditional NLL loss (Eq.2):
    $\mathcal{L}(\theta, \phi) = \frac{1}{2}\text{Tr}(Z_t^T K_{cc}^{-1} Z_t) + \frac{d_t}{2} \log |K_{cc}|$
  // 3. Backward Pass (Asymmetric Optimization)
  Compute gradients: $\nabla_{\theta,\phi}\mathcal{L}$
  Update context network and kernel using optimizer (e.g. Adam):
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
    $\phi \leftarrow \phi - \eta \nabla_\phi \mathcal{L}$
  // 4. Target Network EMA Update
    $\theta' \leftarrow \tau\theta' + (1 - \tau)\theta$
end while

Computational Complexity. The overall computational complexity of the exact Dual-GJE algorithm evaluates to $\mathcal{O}(N^2 d_c + N^2 d_t + N^3)$. In Step 2, computing the $N \times N$ empirical Gram matrix K_{cc} across the batch requires $\mathcal{O}(N^2 d_c)$ operations. Subsequently, evaluating the conditional NLL loss necessitates computing both the inverse (K_{cc}^{-1}) and the log-determinant ($\log |K_{cc}|$) of this Gram matrix. These operations, typically implemented via a Cholesky decomposition, impose a severe $\mathcal{O}(N^3)$ computational bottleneck. Further, calculating the trace of the data-fit term involves matrix multiplications that scale as $\mathcal{O}(N^2 d_t)$. Because self-supervised learning inherently requires massive batch sizes (N) to provide a sufficiently diverse topological landscape for the contrastive and predictive forces, the cubic $\mathcal{O}(N^3)$ term overwhelmingly dominates the computational footprint. This strict intractability for large N heavily motivates the $\mathcal{O}(ND^2)$ scalable Random Fourier Features (RFF) approximation detailed below, and the foundational pivot to the $\mathcal{O}(d^3)$ Primal-GJE formulation.

I.1 Scalable Dual-GJE via Random Fourier Features (RFF)

As established above, formulating GJE in the dual sample space requires computing the inverse and log-determinant of an $N \times N$ Gram matrix. This imposes a strict $\mathcal{O}(N^3)$ computational bottleneck, rendering exact dual optimization intractable for large batch sizes. To resolve this, we utilize Random Fourier Features (RFF) (Rahimi & Recht, 2007) to approximate the kernel mapping, reducing the complexity to $\mathcal{O}(ND^2)$. This section details the theoretical roots of Kernel methods to the explicit application of RFF for the dual GJE objective.

I.1.1 Kernel Machines and Bochner's Theorem

By *Mercer's theorem* (Mercer, 1909), any continuous, symmetric, positive-definite kernel $k(x, y)$ can be expressed as an inner product in a high-dimensional feature space: $k(x, y) = \langle \varphi(x), \varphi(y) \rangle_\mathcal{V}$. To circumvent the $\mathcal{O}(N^3)$ Gram matrix bottleneck associated with this kernel trick, Rahimi and Recht (Rahimi & Recht, 2007) proposed approximating $k(x, y)$ using a randomized, low-dimensional feature map $z : \mathbb{R}^d \mapsto \mathbb{R}^D$ (where $D \ll N$), such that $k(x, y) \approx z(x)^T z(y)$.

This approximation relies on *Bochner's Theorem* (Rudin, 1990), which states that a continuous, shift-invariant kernel $k(x, y) = k(x - y)$ is positive-definite if and only if $k(\Delta)$ is the Fourier transform of a non-negative probability measure $p(\omega)$. Thus, the kernel can be expressed as an expectation:

$$k(x - y) = \mathbb{E}_{\omega \sim p(\omega)} \left[e^{i\omega^T (x-y)} \right] \tag{57}$$

By discarding the imaginary component via Euler's formula and introducing a random phase shift $b \sim \mathcal{U}(0, 2\pi)$ (Gundersen, 2019), we can draw D samples of frequencies $\omega_1, \ldots, \omega_D \sim p(\omega)$ to construct the explicit D-dimensional feature vector:

$$\psi(x) = \sqrt{\frac{2}{D}} \begin{bmatrix} \cos(\omega_1^T x + b_1) \\ \vdots \\ \cos(\omega_D^T x + b_D) \end{bmatrix} \in \mathbb{R}^D \tag{58}$$

Taking the inner product of two such feature vectors elegantly yields the Monte Carlo approximation of the kernel expectation: $\psi(x)^T \psi(y) \approx k(x, y)$.

I.1.2 Applying RFF to the Dual GJE Matrix

For a batch of N context embeddings, let $\Psi \in \mathbb{R}^{N \times D}$ be the matrix where each row corresponds to the evaluated Fourier features $\psi(z_{c,i})^T$. The empirical $N \times N$ Gram matrix can thus be approximated as a low-rank linear inner product: $K_{cc} \approx \Psi \Psi^T$.

To optimize the dual GJE loss, we must evaluate the inverse and log-determinant of the jitter-regularized Gram matrix $K_\epsilon = \Psi \Psi^T + \epsilon I_N$. By factoring the intractable $N \times N$ matrix into the product of these low-dimensional Ψ matrices, we invoke the **Woodbury Matrix Identity** (Woodbury, 1950) to compute the inverse:

$$(\Psi \Psi^T + \epsilon I_N)^{-1} = \epsilon^{-1} I_N - \epsilon^{-1} \Psi (\Psi^T \Psi + \epsilon I_D)^{-1} \Psi^T \tag{59}$$

Similarly, we apply the **Weinstein-Aronszajn Identity** (Pozrikidis, 2014) to evaluate the log-determinant:

$$\log |\Psi \Psi^T + \epsilon I_N| = \log |\Psi^T \Psi + \epsilon I_D| + (N - D) \log \epsilon \tag{60}$$

By structurally rewriting the objective entirely in terms of the $D \times D$ primal feature covariance matrix $C = \Psi^T \Psi + \epsilon I_D$, we reduce the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(ND^2)$, enabling strictly linear scaling with respect to the batch size N. The implementable training algorithm is detailed in Algo.4.

Algorithm 4 Training Dual-GJE with Random Fourier Features (GJE-RFF)

Require: Dataset $\mathcal{D}$, Context Encoder E_θ, Target Encoder $E_{\theta'}$. RFF dimension D, Batch size N, Learning rate η, EMA momentum τ. Kernel parameters ϕ determining $p(\omega)$.

while network has not converged **do**
  Sample a batch of N augmented view pairs $\{(x_c^{(i)}, x_t^{(i)})\}_{i=1}^N \sim \mathcal{D}$
  // 1. Forward Pass (Asymmetric)
  Compute context embeddings: $Z_c = E_\theta(X_c) \in \mathbb{R}^{N \times d_c}$
  No-grad: Compute target anchors: $Z_t = \text{stop_grad}(E_{\theta'}(X_t)) \in \mathbb{R}^{N \times d_t}$
  // 2. Scalable RFF Approximation
  Sample frequencies $\Omega \in \mathbb{R}^{d_c \times D} \sim p(\omega)$ and biases $b \in \mathbb{R}^{1 \times D} \sim \mathcal{U}(0, 2\pi)$.
  Compute Fourier features over context: $\Psi = \sqrt{\frac{2}{D}} \cos(Z_c \Omega + b) \in \mathbb{R}^{N \times D}$
  Compute $D \times D$ feature covariance: $C = \Psi^T \Psi + \epsilon I_D$
  // 3. Fast Inverse and Log-Determinant Computation
  Use Woodbury identity for inverse: $K_{cc}^{-1} \approx \epsilon^{-1} I_N - \epsilon^{-1} \Psi C^{-1} \Psi^T$
  Use Weinstein–Aronszajn identity for log-det: $\log |K_{cc}| \approx \log |C| + (N - D) \log \epsilon$
  Compute approximated conditional NLL loss:
    $\mathcal{L}(\theta, \phi) \approx \frac{1}{2} \text{Tr}(Z_t^T K_{cc}^{-1} Z_t) + \frac{d_t}{2} \log |K_{cc}|$
  // 4. Backward Pass
  Compute gradients $\nabla_{\theta, \phi} \mathcal{L}$ and update context/kernel parameters.
  Update target network via EMA: $\theta' \leftarrow \tau \theta' + (1 - \tau) \theta$
end while

Note on computational trick: by reordering the matrix multiplications inside the trace operator via cyclic permutation in Step 3, evaluating Tr$(\Psi C^{-1} \Psi^T Z_t Z_t^T)$ as Tr$((\Psi^T Z_t)(Z_t^T \Psi) C^{-1})$, we avoid ever materializing the $N \times N$ matrix in memory.

J TRAINING PRIMAL-GJE: SYMMETRIC OPTIMIZATION AND THE ENTROPY FIX

In contrast to the Dual space, the Primal-GJE framework formulates the geometry in the $d \times d$ feature space using the exact joint likelihood $p(z_c, z_t)$. Because this generative formulation provides symmetric marginal regularization for both the context and target spaces, *it mathematically eliminates the need for an EMA target network or stop-gradients*. The gradients flow synchronously through both encoders, providing a symmetric representation learning architecture.

However, as mentioned in Section.3 and detailed in Appendix.N, attempting to directly minimize the exact NLL using the empirically estimated batch covariance C_{joint} triggers the "*Mahalanobis Trace Trap*", algebraically collapsing the repulsive data-fit gradients to zero.

To rescue the optimization dynamics while maintaining the symmetric structure, the empirical implementation of Primal-GJE structurally inverts the regularizer. Instead of minimizing the log-determinant (volume penalty) against a dead Mahalanobis term, we explicitly *maximize* the differential entropy ($\frac{1}{2}\log|C_{joint}|$). By setting our loss function to minimize the negative of this entropy ($\mathcal{L} = -\frac{1}{2}\log|C_{joint}|$), this formulation actively forces the joint feature distribution to span the available geometry of the unit hypersphere, safely guarding against dimensional collapse. The complete symmetric training procedure is detailed in Algo.5.

Algorithm 5 Training Primal-GJE Model (Symmetric Feature Covariance)

Require: Dataset $\mathcal{D}$, Context Encoder E_θ, Target Encoder $E_{\theta'}$. Batch size N, Learning rate η.

while network has not converged **do**
  Sample a batch of N augmented view pairs $\{(x_c^{(i)}, x_t^{(i)})\}_{i=1}^N \sim \mathcal{D}$
  // 1. Forward Pass (Symmetric)
  Compute context embeddings: $Z_c = E_\theta(X_c) \in \mathbb{R}^{N \times d_c}$
  Compute target embeddings: $Z_t = E_{\theta'}(X_t) \in \mathbb{R}^{N \times d_t}$
  Optional: Apply L^2-normalization to bound the embeddings to a unit hypersphere.
  Concatenate joint embeddings (feature-wise): $Z = [Z_c, Z_t] \in \mathbb{R}^{N \times d}$
  // 2. Feature Covariance Matrix & Empirical Loss
  Compute the $d \times d$ empirical covariance matrix: $C_{joint} = \frac{1}{N} Z^T Z$
  Add jitter for numerical stability: $C_{joint} \leftarrow C_{joint} + \epsilon I_d$
  Compute the explicit entropy maximization loss (inverted geometric regularizer):
    $\mathcal{L}(\theta, \theta') = -\frac{1}{2}\log|C_{joint}|$
  // 3. Backward Pass (Synchronous Optimization)
  Compute gradients: $\nabla_{\theta,\theta'}\mathcal{L}$
  Update parameters synchronously using an optimizer (e.g. Adam):
    $\theta \leftarrow \theta - \eta\nabla_\theta\mathcal{L}$
    $\theta' \leftarrow \theta' - \eta\nabla_{\theta'}\mathcal{L}$
  Note: No EMA target network or stop-gradients are applied.
end while

Computational complexity. The overall computational complexity of the symmetric Primal-GJE algorithm evaluates to $\mathcal{O}(Nd^2 + d^3)$. In Step 2, computing the $d \times d$ empirical feature covariance matrix C_{joint} via the inner product $Z^T Z$ requires $\mathcal{O}(Nd^2)$ operations. Subsequently, evaluating the log-determinant geometric regularizer ($\log|C_{joint}|$) necessitates a Cholesky decomposition of this covariance matrix, imposing an $\mathcal{O}(d^3)$ computational cost. In standard self-supervised representation learning paradigms, the batch size N (e.g. 4096) is typically much larger than the embedding dimension d (e.g. 512). Under the condition $d \ll N$, the $\mathcal{O}(d^3)$ term becomes negligible, and the algorithm scales strictly linearly with respect to the batch size ($\mathcal{O}(N)$). This fundamentally resolves the intractable $\mathcal{O}(N^3)$ matrix inversion bottleneck inherent to the exact Dual-GJE (Sample Space) formulation (Algo.3 in Appendix.I), allowing the primal network to scale gracefully to massive batch sizes without requiring randomized kernel approximations.

K EXACT DUAL GJE WITH A LINEAR KERNEL: THE PRIMAL-DUAL EQUIVALENCE

As mentioned in Section 3, with a linear kernel, Dual-GJE is exactly equal to Primal-GJE. In this appendix, we mathematically prove that, if this exact dual GJE formulation is implemented using a standard *Linear Kernel*, it algebraically collapses into a single-component Gaussian Mixture Model ($K = 1$ GMM) operating in the *primal feature space*.

This primal-dual equivalence proves that computing the joint likelihood using an $N \times N$ linear Gram matrix over the samples is mathematically identical to computing the likelihood using the $d \times d$ feature covariance matrix, thereby validating the unimodal ($K = 1$) visualization baseline utilized in our synthetic experiments (Section 5.1).

K.1 SETUP AND THE EMPIRICAL LOSS

Consider a batch of joint embeddings represented by the matrix $Z \in \mathbb{R}^{N \times d}$, where N is the batch size and $d = d_c + d_t$ is the combined dimensionality of the context and target features. To ensure positive-definiteness and invertibility, we introduce a standard jitter term $\epsilon > 0$.

We define the corresponding covariance matrices in the two perpendicular spaces:

- **The Dual (Sample) Gram Matrix:** $K = ZZ^T + \epsilon I_N \quad \in \mathbb{R}^{N \times N}$
- **The Primal (Feature) Covariance Matrix:** $C = Z^T Z + \epsilon I_d \quad \in \mathbb{R}^{d \times d}$

When evaluating the exact Negative Log-Likelihood (NLL) over a multi-dimensional batch Z, the Mahalanobis distance sum across all d independent feature channels analytically translates into the matrix trace operator. The empirical dual loss function to be minimized is (Eq.3):

$$\mathcal{L}_{\text{Dual}} = \underbrace{\frac{1}{2}\text{Tr}(Z^T K^{-1} Z)}_{\text{Data-Fit (Trace)}} + \underbrace{\frac{1}{2}\log|K|}_{\text{Regularizer (Log-Det)}} \tag{61}$$

We will now mathematically project both the *Data-Fit term* and the *Regularizer term* from the $\mathbb{R}^{N \times N}$ sample space into the $\mathbb{R}^{d \times d}$ feature space.

K.2 PART 1: THE DATA-FIT TERM VIA THE PUSH-THROUGH IDENTITY

To project the $N \times N$ inverse Gram matrix K^{-1} into the primal space, we leverage the *Push-Through Identity*, a direct corollary of the Woodbury Matrix Identity, which dictates how matrices "push through" inverses: $A(I + BA)^{-1} = (I + AB)^{-1}A$.

Applying this structural property to our matrices:

$$Z^T(\epsilon I_N + ZZ^T)^{-1} = (\epsilon I_d + Z^T Z)^{-1} Z^T$$

We multiply both sides of this equation by Z on the right:

$$Z^T(\epsilon I_N + ZZ^T)^{-1} Z = (\epsilon I_d + Z^T Z)^{-1} Z^T Z \tag{62}$$

Notice that the left side is exactly our dual inner term $Z^T K^{-1} Z$, and the right side is formulated entirely in terms of the primal feature matrix $C = Z^T Z + \epsilon I_d$. We can substitute C into the right-hand side. Recognizing that $Z^T Z = C - \epsilon I_d$, we evaluate the matrix product:

$$\begin{aligned} Z^T K^{-1} Z &= C^{-1}(Z^T Z) \\ &= C^{-1}(C - \epsilon I_d) \\ &= C^{-1}C - \epsilon C^{-1} \\ &= I_d - \epsilon C^{-1} \end{aligned} \tag{63}$$

Finally, taking the trace of both sides yields the exact primal equivalence for the data-fit term:

$$\boxed{\text{Tr}(Z^T K^{-1} Z) = \text{Tr}(I_d) - \epsilon\text{Tr}(C^{-1}) = d - \epsilon\text{Tr}(C^{-1})} \tag{64}$$

K.3 Part 2: The Geometric Regularizer via Weinstein-Aronszajn

Next, we project the $N \times N$ log-determinant volume penalty, $\log|K|$, into the primal space. We apply the *Weinstein-Aronszajn Identity* (also known as Sylvester's Determinant Theorem), which states that for any matrices $A \in \mathbb{R}^{N\times d}$ and $B \in \mathbb{R}^{d\times N}$, the determinant identity $|I_N + AB| = |I_d + BA|$ holds true.

We begin by factoring the scalar jitter ϵ out of the dual Gram matrix determinant. Because K is an $N \times N$ matrix, factoring out ϵ raises it to the power of N:

$$|K| = |ZZ^T + \epsilon I_N| = \left|\epsilon\left(I_N + \frac{1}{\epsilon}ZZ^T\right)\right| = \epsilon^N \left|I_N + \left(\epsilon^{-1}Z\right)Z^T\right| \tag{65}$$

We now apply the Weinstein-Aronszajn identity, setting $A = \epsilon^{-1}Z$ and $B = Z^T$, which swaps the multiplication order and cleanly reduces the identity matrix dimension from N to d:

$$|K| = \epsilon^N \left|I_d + Z^T\left(\epsilon^{-1}Z\right)\right| = \epsilon^N \left|I_d + \epsilon^{-1}Z^TZ\right| \tag{66}$$

To reconstruct the primal feature covariance matrix $C = Z^TZ + \epsilon I_d$ inside the determinant, we must absorb a factor of ϵ back into the $d \times d$ matrix. Absorbing ϵ into a d-dimensional determinant scales the outside multiplier by ϵ^{-d}:

$$|K| = \epsilon^N\epsilon^{-d}\left|\epsilon(I_d + \epsilon^{-1}Z^TZ)\right| = \epsilon^{N-d}|Z^TZ + \epsilon I_d| = \epsilon^{N-d}|C| \tag{67}$$

Taking the natural logarithm of both sides transforms the exponentiation into linear scalar addition:

$$\boxed{\log|K| = \log|C| + (N-d)\log\epsilon} \tag{68}$$

K.4 Conclusion: The Unimodal Constraint

By substituting our two derived primal equivalencies back into the original dual loss function (Eq.61), we obtain:

$$\begin{aligned}\mathcal{L}_{\text{Dual}} &= \frac{1}{2}\left[d - \epsilon\text{Tr}(C^{-1})\right] + \frac{1}{2}\left[\log|C| + (N-d)\log\epsilon\right] \\ &= \frac{1}{2}\left(\log|C| - \epsilon\text{Tr}(C^{-1})\right) + \text{Constant}\end{aligned} \tag{69}$$

Equation 69 reveals a profound structural truth: *if one utilizes a linear kernel, the exact $\mathcal{O}(N^3)$ Gaussian Process NLL is mathematically driven entirely by the $d \times d$ feature covariance matrix C.* It possesses absolutely no non-linear flexibility.

Because the loss gradients are algebraically locked to the exact same $d \times d$ global covariance structure, optimizing the dual sample-space GJE yields the exact same weights as fitting a single, global multivariate Gaussian ($K = 1$ GMM) in the primal feature space. This fundamentally restricts the learned representations to a strict linear geometry. To wrap around non-linear, multi-modal semantic branches (as illustrated in our synthetic datasets), one is forced to either utilize highly expensive non-linear approximations (e.g. Random Fourier Features for an RBF kernel) to bend this single density in an infinite-dimensional RKHS, or more naturally, pivot directly into the primal feature space and deploy multiple discrete prototypes via Gaussian Mixture Joint Embeddings (GMJE).

L The Identity Collapse Trap in Dynamic Parameterization (MDN)

During our first experiments, we noted that, when extending the Gaussian Mixture Joint Embeddings (GMJE) framework to utilize dynamic, instance-specific parameters via a neural network (as in the GMJE-MDN architecture), one might intuitively attempt to parameterize the joint distribution directly from the joint embedding itself, i.e. dropping all info $Z_k = [z_c^T, z_t^T]_k^T$ of instance k into the neural network to obtain an amortized inference of $[\mu, \Sigma, \pi]_k$ which can be used to parameterize the subsequent GMM. In this appendix, we mathematically demonstrate why feeding the full joint vector $Z = [z_c^T, z_t^T]^T$ into the parameter network leads to a pathological failure state known as *Identity Collapse* (or *Information Leakage*), and why introducing a strict *conditional information bottleneck* is theoretically required.

L.1 The Naive Joint Parameterization

Suppose we construct a parameter network f_ϕ that takes the entire joint embedding Z as input to predict the mixture parameters for that exact same joint space: $f_\phi(Z) \to \{\pi_k(Z), \mu_k(Z), \Sigma_k(Z)\}_{k=1}^K$. The network is optimized to minimize the Negative Log-Likelihood (NLL) of the joint mixture:

$$\mathcal{L} = -\log \sum_{k=1}^{K} \pi_k(Z) \mathcal{N}\Big(Z \;\Big|\; \mu_k(Z), \Sigma_k(Z)\Big)$$

L.2 The Pathological Optimum (Identity Collapse)

Because the neural network takes Z as its input, it has full deterministic access to the exact target variable it is attempting to evaluate in the Gaussian density. To maximize the Gaussian likelihood (and thus minimize the NLL loss), the network discovers a trivial "cheat code": it simply learns the *Identity Function*.

By setting the predicted mean of every component exactly equal to the input, $\mu_k(Z) = Z$ for all $k \in \{1 \dots K\}$, the Mahalanobis distance term inside the Gaussian exponent evaluates to exactly zero:

$$-\frac{1}{2}(Z - \mu_k(Z))^T \Sigma_k(Z)^{-1} (Z - \mu_k(Z)) = 0$$

Consequently, the exponential term reaches its absolute maximum, $\exp(0) = 1$. The network can then trivially drive the loss toward $-\infty$ by shrinking the determinant of the predicted covariance matrices, $|\Sigma_k(Z)| \to 0$, causing the probability density to approach infinity.

L.3 Empirical Symptoms

When this identity collapse occurs, the network fails to learn the underlying semantic branches or multi-modal structure of the data manifold. Instead, it merely memorizes the exact noise profile of the specific input data points.

Further, because every component k perfectly cheats by outputting exactly $\mu_k(Z) = Z$, the network has no mathematical incentive to prefer one component over another. As a result, the dynamic mixing weights degenerate into a flat, uniform distribution ($\pi_k(Z) \approx 1/K$ everywhere). The network completely loses its ability to perform intelligent, semantic density routing based on ambiguity.

L.4 The Resolution: The Conditional Information Bottleneck

To successfully achieve dynamic multi-modal routing, we must enforce a strict *Information Bottleneck*. As derived in the GMJE-MDN formulation (Section.4), the parameter network must operate conditionally: it is only permitted to observe the context view z_c, and must predict the mixture parameters for the target view z_t:

$$\mathcal{L} = -\log \sum_{k=1}^{K} \gamma_k(z_c) \mathcal{N}\Big(z_t \;\Big|\; \mu_{t|c,k}(z_c), \Sigma_{t|c,k}(z_c)\Big) \tag{70}$$

Under this strict conditional formulation, the network does not have access to z_t. When presented with an ambiguous context embedding z_c (e.g. the intersection point of multiple valid branches, as in Fig.8), it cannot rely on the identity function. To survive the loss penalty, it is mathematically forced to distribute its K components to cover the distinct, valid possibilities of z_t that could arise from that specific z_c. This physically forces the network to learn the true underlying manifold and dynamically route its probability mass (γ_k) to the appropriate branches, ensuring a robust and mathematically sound generative alignment space.

M Generalizing SMC to the Full GMJE Objective

In Section 4.2, we introduced SMC-GMJE (Algo.1) as a direct upgrade to standard contrastive learning frameworks (e.g. MoCo). However, by substituting the InfoNCE loss with the exact parametric

GMJE Negative Log-Likelihood (NLL) objective (Eq.8), the Sequential Monte Carlo particle filter natively generalizes to manage full-covariance Gaussian components.

In this generalized formulation, the memory bank does not merely store target embeddings z_t; it stores the concatenated joint embeddings $Z = [z_c^T, z_t^T]^T$, which act as the M component means ($\mu^{(m)}$) of a dynamically weighted Gaussian Mixture Model. Simultaneously, the neural network learns a shared, full-rank covariance matrix Σ via gradient descent to capture the non-linear dependencies and multi-modal shape of the manifold.

M.1 The Generalized Weight Update

Instead of relying on the isotropic dot-product similarity ($\exp(z_c^T z_t^{(m)}/\tau)$), the particle likelihoods are evaluated using the true Mahalanobis distance governed by Σ. For a batch of B incoming joint queries Z_{query}, the unnormalized importance weight for particle m recursively updates as:

$$\tilde{W}_t^{(m)} = W_{prior}^{(m)} \times \frac{1}{B} \sum_{Z \in Z_{query}} \exp\left(-\frac{1}{2}(Z - \mu^{(m)})^T \Sigma^{-1} (Z - \mu^{(m)})\right) \tag{71}$$

Note that the constant scaling factors and the log-determinant of Σ from the standard Gaussian density function elegantly cancel out when normalizing the weights to sum to one, rendering the weight update mathematically clean and strictly dependent on the learned metric geometry of Σ^{-1}.

To understand why this update is computationally efficient, consider the exact probability density function of the multivariate Gaussian. The true unnormalized importance weight, utilizing the exact likelihood, is:

$$\hat{W}_t^{(m)} = W_{prior}^{(m)} \times \frac{1}{B} \sum_{Z \in Z_{query}} \underbrace{\frac{1}{\sqrt{(2\pi)^D |\Sigma|}}}_{C(\Sigma)} \exp\left(-\frac{1}{2}(Z - \mu^{(m)})^T \Sigma^{-1} (Z - \mu^{(m)})\right) \tag{72}$$

where $D = d_c + d_t$ is the total dimensionality of the joint embedding, and $C(\Sigma)$ is the standard Gaussian normalization constant.

Because the GMJE formulation utilizes a *shared* global covariance matrix Σ across all components, $C(\Sigma)$ is identical for every particle m. Consequently, it acts as a global scalar that can be factored out of the summation:

$$\hat{W}_t^{(m)} = C(\Sigma) \times \underbrace{\left[W_{prior}^{(m)} \times \frac{1}{B} \sum_{Z \in Z_{query}} \exp\left(-\frac{1}{2}(Z - \mu^{(m)})^T \Sigma^{-1} (Z - \mu^{(m)})\right) \right]}_{\tilde{W}_t^{(m)}} \tag{73}$$

When normalizing the posterior weights across the pool of particles to ensure they sum to one, this shared constant $C(\Sigma)$ elegantly factors out of the denominator and cancels with the numerator:

$$W_{pool}^{(m)} = \frac{\hat{W}_t^{(m)}}{\sum_{j=1}^{M+B} \hat{W}_t^{(j)}} = \frac{C(\Sigma)\tilde{W}_t^{(m)}}{C(\Sigma) \sum_{j=1}^{M+B} \tilde{W}_t^{(j)}} = \frac{\tilde{W}_t^{(m)}}{\sum_{j=1}^{M+B} \tilde{W}_t^{(j)}} \tag{74}$$

This cancellation mathematically proves that evaluating the computationally expensive log-determinant ($|\Sigma|$) and scaling factors is strictly unnecessary for the SMC routing step. The normalized posterior weights are determined entirely by the unnormalized exponent $\tilde{W}_t^{(m)}$, rendering the dynamic particle update mathematically clean and strictly dependent on the learned metric geometry of the precision matrix Σ^{-1}.

M.2 Algorithm: General SMC-GMJE

The complete procedure for the generalized full-covariance SMC-GMJE is detailed in Algo.6. During the forward pass, the objective acts exactly as the relaxed EM optimization formulated in Section 4, but it utilizes the dynamic posterior particle weights $W^{(m)}$ as the mixing priors π_k.

Algorithm 6 General SMC-GMJE: Full-Covariance Particle Filter

Require: Stream of query batches $Z_{query} = \{Z_b\}_{b=1}^{B}$ generated by encoders (where $Z_b = [z_c^T, z_t^T]_b^T$), memory bank size M. Learnable shared covariance Σ.

Initialize memory bank $\mathcal{M}$ with M random joint embeddings acting as means $\mu^{(m)}$, and uniform weights $W \leftarrow 1/M$.
for each training step t with incoming batch Z_{query} **do**
  // Loop A: Parametric Optimization (Requires Gradients)
  Forward Pass: Compute the Log-Sum-Exp objective (Eq.8) for the batch Z_{query} against the memory bank means $\mu^{(m)}$, utilizing the current particle weights $W^{(m)}$ as the fixed mixing priors π_m, and the learnable covariance Σ.
  Backward Pass: Compute gradients and update the dual encoder networks' parameters *and* the shared covariance matrix Σ via gradient descent.

  // Loop B: General SMC Optimization (Gradient-Free)
  Combine: Pool $\mathcal{M}_{pool} \leftarrow \mathcal{M} \cup Z_{query}$, resulting in size $M + B$.
  Prior Weights: Set $W_{prior}^{(m)} \leftarrow \frac{1}{M+B}$ for the new batch components, and scale old memory bank weights by $\frac{M}{M+B}$.
  Importance Update: For each particle $m \in \{1 \dots M + B\}$:
    $\tilde{W}_t^{(m)} \leftarrow W_{prior}^{(m)} \times \frac{1}{B} \sum_{Z \in Z_{query}} \exp\left(-\frac{1}{2}(Z - \mu^{(m)})^T \Sigma^{-1} (Z - \mu^{(m)})\right)$
  Normalize: $W_{pool}^{(m)} \leftarrow \tilde{W}_t^{(m)} / \sum_{j=1}^{M+B} \tilde{W}_t^{(j)}$
  // Execute Sequential Importance Resampling (SIR)
  Sample M particles from $\mathcal{M}_{pool}$ with replacement, proportional to their new posterior weights $W_{pool}^{(m)}$.
  Replace the memory bank $\mathcal{M}$ with these sampled particles.
  Reset all memory bank weights to uniform: $W \leftarrow 1/M$.
end for

N THE "MAHALANOBIS TRACE" TRAP IN EMPIRICAL BATCH OPTIMIZATION

The "Mahalanobis Trace" Trap is a mathematically elegant but dangerous linear algebra quirk that frequently catches researchers off guard when implementing probabilistic loss functions or geometric regularizers in deep learning frameworks. In our context, this trap is not limited to unimodal distributions (e.g. Primal-GJE); it infects multi-modal Gaussian Mixtures (GMJE) with equal severity. To understand this trap, one must carefully distinguish between theoretical population mathematics and empirical mini-batch operations.

In generative theory, the Mahalanobis distance $z^T \Sigma^{-1} z$ measures how far a point z is from the center of a distribution, scaled by a *fixed population covariance* Σ. If the representation z moves during optimization, its distance relative to the fixed Σ changes, providing a valid non-zero gradient to the neural network encoders.

However, in empirical self-supervised learning, one typically does not have access to a fixed Σ. Instead, one computes an empirical covariance matrix C dynamically from the current mini-batch Z, and then immediately evaluates the exact same mini-batch Z against C^{-1}. The following mathematical proofs demonstrate why doing this collapses the entire calculation into a static constant for both unimodal and multi-modal settings, catastrophically killing the data-fit gradients.

N.1 CASE 1: THE TRAP IN UNIMODAL GAUSSIAN (PRIMAL-GJE)

Let $Z \in \mathbb{R}^{N \times d}$ be a batch of N centered embedding vectors, where each row $z_i \in \mathbb{R}^d$ is a single image's latent representation. The empirical batch covariance matrix C is defined as $C = \frac{1}{N} \sum_{i=1}^{N} z_i z_i^T$.

The average Mahalanobis distance across the batch, which fundamentally provides both the *expansive barrier* (preventing volume compression) and the *cross-view attractive force*, is formulated as:

$$\mathcal{L}_{\text{Mahalanobis}} = \frac{1}{N}\sum_{i=1}^{N} z_i^T C^{-1} z_i \tag{75}$$

Because the quadratic form $z_i^T C^{-1} z_i$ results in a single scalar value, wrapping it in a trace operator leaves its value unchanged. We apply the trace cyclic property, $\text{Tr}(ABC) = \text{Tr}(CAB)$, and push the linear summation inside the operator:

$$\begin{aligned}\mathcal{L}_{\text{Mahalanobis}} &= \frac{1}{N}\sum_{i=1}^{N} \text{Tr}(C^{-1} z_i z_i^T) \\ &= \text{Tr}\left(C^{-1}\left[\frac{1}{N}\sum_{i=1}^{N} z_i z_i^T\right]\right)\end{aligned} \tag{76}$$

The bracketed term is exactly the mathematical definition of the empirical covariance matrix C. By directly substituting C into the equation, the variables gracefully cancel to yield the Identity matrix (I_d), the trace of which is strictly the constant intrinsic dimensionality d:

$$\mathcal{L}_{\text{Mahalanobis}} = \text{Tr}(C^{-1}C) = \text{Tr}(I_d) = d \tag{77}$$

N.2 Case 2: The Trap in GMM (GMJE)

One might assume that partitioning the space into K discrete components via a Gaussian Mixture Model resolves this issue. However, standard Expectation-Maximization (EM) batch updates trigger the exact same cancellation.

For a specific mixture component k, let γ_{ik} be the posterior responsibility (soft assignment) of data point z_i to cluster k. The effective number of points in the cluster is $N_k = \sum_{i=1}^{N} \gamma_{ik}$, and the empirical component covariance is evaluated as:

$$\Sigma_k = \frac{1}{N_k}\sum_{i=1}^{N} \gamma_{ik}(z_i - \mu_k)(z_i - \mu_k)^T \tag{78}$$

When evaluating the Mahalanobis data-fit term for this specific cluster using the same batch, weighted by the responsibilities, we apply the exact same trace trick and cyclic permutation:

$$\begin{aligned}\mathcal{L}_{\text{data-fit},k} &= \sum_{i=1}^{N} \gamma_{ik}(z_i - \mu_k)^T \Sigma_k^{-1}(z_i - \mu_k) \\ &= \text{Tr}\left(\Sigma_k^{-1}\left[\sum_{i=1}^{N} \gamma_{ik}(z_i - \mu_k)(z_i - \mu_k)^T\right]\right)\end{aligned} \tag{79}$$

Observe the bracketed term. It is exactly equal to $N_k\Sigma_k$. Substituting this yields:

$$\mathcal{L}_{\text{data-fit},k} = \text{Tr}\left(\Sigma_k^{-1}[N_k\Sigma_k]\right) = N_k\text{Tr}(I_d) = N_k \cdot d \tag{80}$$

Thus, the data-fit gradient for every single cluster k algebraically collapses to a constant. The network feels absolutely zero gradient pulling the representations toward the cluster center μ_k.

To intuitively grasp why evaluating this term as a constant is fatal, one must first examine the geometric dynamics of the factorized joint distribution objective. By the laws of probability, the joint negative log-likelihood factors elegantly into two opposing forces: $-\log p(z_c, z_t) = -\log p(z_t \mid z_c) - \log p(z_c)$. The conditional term, $-\log p(z_t \mid z_c)$, serves as the *Attractive (Pulling) Force*. It acts as a mathematical rubber band connecting two augmented views, pulling the target embedding z_t toward the specific semantic coordinate or cluster generated by the context z_c. Conversely, the marginal term, $-\log p(z_c)$, governs the *Repulsive (Pushing) Force*. When structurally inverted to explicitly maximize the differential entropy (via the log-determinant, which measures the geometric volume), it acts as a universal repulsive force that pushes all independent samples in the batch apart.

When these forces fall out of balance, the latent space suffers from distinct geometric failure modes. If the repulsive force is entirely absent, the attractive force wins completely, resulting in *Complete Collapse* (or *Instance Collapse*), where the network lazily maps every single image in the dataset to the exact same coordinate, reducing the variance to zero. More insidiously, if the geometric repulsion is misaligned, weak, or improperly evaluated, the network suffers from *Dimensional Collapse* (or *Feature Collapse*). In this state, the embeddings spread out enough to distinguish instances, but they collapse onto a strictly lower-dimensional subspace (e.g. squashing a 128-dimensional hypersphere onto a 1D line or 2D plane). Mathematically, the feature covariance matrix Σ becomes highly ill-conditioned, dominated by zero eigenvalues. Geometrically, the network wastes its representational capacity; by failing to utilize orthogonal dimensions, independent semantic concepts (e.g. "color" and "shape") become highly entangled into a single metric, destroying the expressiveness of the joint embedding.

N.3 The Dual Consequences of the Trace Trap

In deep learning frameworks, one might intuitively implement the exact NLL penalty as a complex batched tensor operation (e.g. evaluating the sum of dot products between the embeddings and their inverse covariance). However, because C was derived directly from Z, linear algebra perfectly cancels the moving variables, evaluating the entire data-fit term as the constant scalar d. Because $\nabla_Z(d) = \mathbf{0}$, **no gradients ever flow backward through this term**.

Because the exact data-fit term mathematically governs both the expansive barrier and the cross-view attraction, its algebraic death triggers a simultaneous, dual geometric failure:

Consequence 1: Loss of Repulsion (Dimensional Collapse). Without the data-fit term acting as an *expansive barrier*, the complementary half of the exact joint loss function, i.e. the geometric penalty $\frac{1}{2}\log|C|$, is left completely unopposed. Driven solely to minimize this unconstrained log-determinant, the optimizer aggressively shrinks the volume of the space to zero, causing the catastrophic *dimensional collapse* frequently observed in naive implementations of joint covariance regularization.

Remedy 1: Explicit Entropy Maximization. To rescue the optimization dynamics, the regularizer must be inverted. By discarding the dead Mahalanobis term entirely and flipping the sign of the log-determinant (minimizing $-\frac{1}{2}\log|C|$), we explicitly instruct the optimizer to *maximize* the differential entropy. This actively expands the variance of the embeddings, creating a flawless repulsive force that covers the available spherical space.

Consequence 2: Eradication of the Pulling Force. While explicit entropy maximization successfully pushes representations apart to fill the hypersphere, *it cannot restore view alignment*. The dead data-fit term containing the inverse cross-covariance block (C_{ct}^{-1}) is responsible for pulling z_c and z_t together. Without it, the network will scatter the embeddings across the hypersphere to maximize entropy, but matched pairs z_c and z_t will arbitrarily drift into completely random, disconnected coordinates.

Remedy 2: GMJE Architectural Interventions. GMJE survives this trap not because Gaussian Mixtures are magically immune to linear algebra, but because our specific architectural designs explicitly decouple the parameters from the empirical batch calculations. The three structural remedies introduced in this work act as the literal "escape hatches" from the GMM trap:

1. *Parametric Decoupling (Global Prototypes):* as utilized in Section 4, we do not calculate Σ_k empirically from the batch. Instead, Σ is initialized as a standalone learnable parameter (or treated as a fixed constant like temperature τ). Because Σ_{param} is updated via gradient descent over many epochs, it never equals the instantaneous batch covariance ($\Sigma_{\text{param}} \neq \Sigma_{\text{batch}}$). Consequently, the algebraic cancellation fails, and the network physically feels the gradients of the Mahalanobis distance, allowing the Log-Sum-Exp objective to pull z_c and z_t together.
2. *Conditional Factorization (MDN):* in Section 4, the covariance $\Sigma_k(z_c)$ is predicted dynamically by a neural network based solely on the context view. It is a generated function of

the input, not an empirical average of the batch. Therefore, the cancellation cannot occur, and the conditional formulation natively insulates the pulling force.

3. *Dynamic Non-Parametric Density (SMC):* in the non-parametric branch (Section 4.2), the "components" are historical representations sitting in a memory bank, and the covariance is fixed (τI). The current batch evaluates its distance against the historical bank, not against itself. By isolating the target representations from the current batch gradient graph, the pulling force (the numerator of the InfoNCE objective) remains perfectly intact and highly informative.

N.4 Beyond the Trace Trap: The Log-Determinant Cheat

Even though the architectural interventions of Parametric GMJE successfully bypass the Mahalanobis Trace Trap by decoupling the parameters, the optimization dynamics remain vulnerable to a secondary failure mode if the exact Negative Log-Likelihood (NLL) is used.

Because the neural network controls both the geometric parameters and the representations (Z) simultaneously, optimizing the exact NLL exposes a trivial shortcut. The network realizes that mapping all representations Z to the origin ($\mathbf{0}$) forces the empirical variance to zero, which in turn allows it to shrink $\Sigma \to 0$. As Σ collapses, the log-determinant regularizer drives $\log|\Sigma| \to -\infty$, netting a massively negative loss. Consequently, optimizing the exact NLL directly induces catastrophic Dimensional Collapse.

To prevent this "cheat", one could apply a hard boundary constraint to the representations. For instance, VicReg (Bardes et al., 2022) utilizes a heuristic hinge loss ($\max(0, 1-\sigma)$) to explicitly constrain the variance of each dimension to be at least 1. However, in generative representation learning, such constraints are suboptimal because they create a "huddled" latent space: the network will shrink the representations until they hit the exact floor of the constraint, and then it will stop, doing the bare minimum.

This geometric reality fundamentally motivates **Explicit Entropy Maximization** (Remedy 1). By explicitly flipping the sign of the regularizer to minimize $-\frac{1}{2}\log|\Sigma_c|$, we replace a rigid floor constraint with a continuous, outward repulsive force. The network actively pushes the representations apart, forcing them to span the entire available volume of the hypersphere uniformly. This mathematical tension seamlessly guarantees a rich, full-rank generative embedding space without requiring manually tuned constraint hyperparameters.

O Experimental Details

To support reproducibility, this appendix summarizes the shared computing environment, data processing steps, model architectures, and hyperparameters used in the three experimental groups reported in the paper: (i) synthetic ambiguous alignment, (ii) vision benchmarks on CIFAR-10, and (iii) unconditional latent sampling on MNIST.

O.1 Shared Computing Environment

All experiments were executed in Google Colab on a Linux-based virtualized instance (KVM) with the following hardware configuration:

- *CPU:* Intel(R) Xeon(R) Platinum 8481C CPU at 2.70 GHz, with 13 physical cores and 26 logical threads.
- *Memory (RAM):* 247.01 GB.
- *Storage:* 253.06 GB disk space.
- *GPU:* a single NVIDIA H100 80GB HBM3 GPU (Driver Version 580.82.07, CUDA Version 13.0).
- *Software stack:* Python 3.x, PyTorch (Paszke et al., 2019), Torchvision (Marcel & Rodriguez, 2010), Scikit-Learn (Pedregosa et al., 2011), NumPy (Harris et al., 2020), and Matplotlib (Hunter, 2007). The synthetic experiments additionally used Scikit-Learn's *GaussianMixture* and *GaussianProcessRegressor*.

O.2 Experiment 1: Synthetic Ambiguous Alignment

O.2.1 Random Seeds

For both Dataset A and Dataset B, NumPy and PyTorch were seeded with `111`.

O.2.2 Data Generation

Both synthetic datasets used the same basic generation protocol:

- *Sample size:* $N = 3000$ training pairs (x_c, x_t).
- *Context distribution:* $x_c \sim \mathcal{U}(-1, 1)$.
- *Branch probabilities:* uniform over the three branches, i.e. $\pi_1 = \pi_2 = \pi_3 = 1/3$.
- *Observation noise:* additive Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.05^2)$ applied to x_t only.
- *Evaluation grid:* 300 evenly spaced test points over $[-1, 1]$.

The two target-generation rules were:

- **Dataset A (Separated Branches):**
$$f(x_c) \in \left\{x_c^2 + 0.5,\ -x_c^2 - 0.5,\ x_c^3\right\}.$$
- **Dataset B (Intersecting Branches):**
$$f(x_c) \in \{\sin(3x_c),\ -\sin(3x_c),\ 0\}.$$

O.2.3 Models and Hyperparameters

Classic JEPA (MSE). A 3-layer MLP was used:

$$\texttt{Linear}(1, 64) \to \texttt{ReLU} \to \texttt{Linear}(64, 64) \to \texttt{ReLU} \to \texttt{Linear}(64, 1).$$

The model was optimized with Adam using learning rate 10^{-2} for 1200 full-batch epochs under MSE loss.

Dual-space kernel baseline (RBF GJE). The dual baseline was implemented with Scikit-Learn's `GaussianProcessRegressor` using

$$\texttt{RBF(length_scale=0.5)} + \texttt{WhiteKernel(noise_level=0.1)}.$$

The internal optimizer was disabled (`optimizer=None`) to keep the kernel fixed on the strongly multi-modal tasks.

GMJE-EM ($K = 1$) and GMJE-EM ($K = 3$). Both were implemented using Scikit-Learn's `GaussianMixture` with:

- `covariance_type='full'`,
- `init_params='kmeans'` for the $K = 3$ model,
- standard EM fitting on the 2D joint data $Z = [x_c, x_t]^T$.

GMJE-GNG. Growing Neural Gas was initialized with two nodes sampled from the training data and then updated online with:

- maximum nodes $K_{\max} = 25$,
- maximum edge age $a_{\max} = 50$,
- winner learning rate $\epsilon_b = 0.2$,
- neighbor learning rate $\epsilon_n = 0.01$,
- insertion interval $\lambda = 100$,
- local error decay $\alpha = 0.5$,
- global error decay $\beta = 0.995$.

GMJE-MDN. The conditional MDN used $K = 3$ mixture components. Its architecture consisted of a shared 2-layer MLP backbone,

$$\texttt{Linear}(1, 64) \to \texttt{ReLU} \to \texttt{Linear}(64, 64) \to \texttt{ReLU},$$

followed by three linear heads for the mixture weights, conditional means, and conditional standard deviations. The weights were passed through a softmax, the means were unconstrained, and the standard deviations were parameterized as

$$\sigma(x_c) = \exp(\cdot) + 10^{-5}.$$

Training used Adam with learning rate 10^{-2} for 1200 full-batch epochs, minimizing the exact conditional Gaussian-mixture negative log-likelihood.

O.3 Experiment 2: Representation Learning on Vision Benchmarks (CIFAR-10)

O.3.1 Random Seeds

The CIFAR-10 script set NumPy and PyTorch seeds to `42`.

O.3.2 Data Pipeline and Augmentations

All CIFAR-10 experiments used the standard SSL augmentation pipeline:

- `RandomResizedCrop(32)`,
- `RandomHorizontalFlip()`,
- `ColorJitter(0.4, 0.4, 0.4, 0.1)` applied with probability 0.8,
- `RandomGrayscale(p=0.2)`,
- channel normalization with CIFAR-10 statistics.

For linear probing, the supervised augmentation pipeline used random crop with padding 4 and random horizontal flip.

O.3.3 Shared Backbone

All CIFAR-10 models used a modified ResNet-18 backbone adapted to 32×32 images:

- the original 7×7 stride-2 convolution was replaced by a 3×3 stride-1 convolution with padding 1,
- the initial max-pooling layer was removed,
- global average pooling produced a 512-dimensional feature vector,
- a projection head mapped $512 \to 512 \to 128$ via

$$\texttt{Linear}(512, 512) \to \texttt{BatchNorm1d} \to \texttt{ReLU} \to \texttt{Linear}(512, 128).$$

O.3.4 Experiment 2a: SMC Memory Banks vs. FIFO

This long-horizon experiment compared MoCo v2 and SMC-GMJE under a severely constrained bank:

- *Training duration:* 1000 epochs.
- *Batch size:* 256.
- *Memory-bank size:* $M = 256$.
- *EMA momentum:* $m = 0.999$.
- *Temperature:* $\tau = 0.1$.
- *Backbone and augmentations:* identical between MoCo v2 and SMC-GMJE.

The goal of this experiment was to isolate the effect of the bank update rule. SMC-GMJE replaced FIFO replacement with likelihood-based particle reweighting and multinomial resampling.

O.3.5 Experiment 2b: 200-Epoch Benchmark Comparison

The shorter benchmark comparison used a common 200-epoch pre-training budget, followed by 100 epochs of linear probing. Shared settings were:

- *Batch size:* 256.
- *Pre-training epochs:* 200.
- *Linear probing epochs:* 100.

Baseline-specific pre-training hyperparameters.

- **SimCLR:** SGD, learning rate 0.03, momentum 0.9, weight decay 10^{-4}, temperature $\tau = 0.1$.
- **MoCo v2:** queue size $M = 4096$, EMA momentum $m = 0.999$, temperature $\tau = 0.1$, with the same SGD optimizer settings as above.
- **BYOL:** EMA momentum $m = 0.99$. The predictor head was

$$\texttt{Linear}(128, 256) \to \texttt{BatchNorm1d} \to \texttt{ReLU} \to \texttt{Linear}(256, 128).$$

- **SwAV:** $K = 1000$ prototypes, temperature $\tau = 0.1$, three Sinkhorn iterations.
- **VICReg:** variance, invariance, and covariance coefficients $(25.0, 25.0, 1.0)$. Unlike the contrastive-style models, VICReg operated on raw Euclidean features without L^2 normalization. The appendix draft indicates LARS with learning rate 0.3 where available, or SGD with learning rate 0.1 as fallback.
- **Parametric GMJE:** $K = 1000$ global prototypes, temperature $\tau = 0.1$, and an EMA-tracked covariance buffer with momentum 0.99 for the explicit marginal regularizer.
- **GMJE-MDN:** $K = 5$ dynamic components. The parameter network used

$$\texttt{Linear}(128, 256) \to \texttt{BatchNorm1d} \to \texttt{ReLU} \to \texttt{Linear}(256, K \times 128 + K + K).$$

  The covariance outputs were constrained with $\texttt{Softplus} + 0.05$.
- **SMC-GMJE:** queue size $M = 4096$, EMA momentum $m = 0.999$, temperature $\tau = 0.1$, with likelihood-based particle reweighting and multinomial resampling at each update.

Linear probing. After pre-training, the encoders were frozen and a linear classifier

$$\texttt{Linear}(512, 10)$$

was trained for 100 epochs using SGD with learning rate 10.0, momentum 0.9, and a cosine annealing schedule.

O.4 Experiment 3: Generative GMJE via Unconditional Latent Sampling on MNIST

O.4.1 Random Seeds

The combined vision/generative script explicitly sets global NumPy and PyTorch seeds to `42`. In addition, the repeated unconditional sampling phase resets both seeds to `1000+i` for the i-th generated sample grid, for $i = 0, \ldots, 9$.

O.4.2 Data Pipeline

MNIST images were padded from 28×28 to 32×32. Two data pipelines were used:

- *SSL manifold-learning pipeline:* `RandomResizedCrop` with scale range $[0.8, 1.0]$, `RandomRotation` up to $15°$, conversion to tensor, and normalization with mean 0.5 and standard deviation 0.5.
- *Plain reconstruction pipeline:* deterministic padding, conversion to tensor, and the same normalization, used for decoder training and latent extraction.

O.4.3 Encoder and Decoder Architectures

CNN Encoder. All three encoder families (SimCLR, Unimodal GJE, Parametric GMJE) used the same CNN backbone:

- Conv2d$(1, 32, 3,$ stride $2,$ padding $1)$ + BatchNorm2d + ReLU,
- Conv2d$(32, 64, 3,$ stride $2,$ padding $1)$ + BatchNorm2d + ReLU,
- Conv2d$(64, 128, 3,$ stride $2,$ padding $1)$ + BatchNorm2d + ReLU,
- Conv2d$(128, 256, 3,$ stride $2,$ padding $1)$ + BatchNorm2d + ReLU,
- flatten,
- `Linear`$(256 \times 2 \times 2, 512)$ + BatchNorm1d + ReLU,
- `Linear`$(512, 128)$.

CNN Decoder. The decoder inverted the latent code back to image space:

- `Linear`$(128, 512)$ + ReLU,
- `Linear`$(512, 1024)$ + ReLU,
- reshape to $256 \times 2 \times 2$,
- four transposed convolutions:

$$256 \to 128 \to 64 \to 32 \to 1$$

  each using kernel size 3, stride 2, padding 1, output padding 1,
- BatchNorm2d + ReLU after each intermediate layer,
- final `Tanh` activation.

O.4.4 Training and Sampling Protocol

Phase 1: encoder pre-training. SimCLR, Unimodal GJE, and Parametric GMJE ($K = 50$) were each trained for 50 epochs with batch size 256 using Adam with learning rate 10^{-3}.

Phase 2: decoder training. The encoders were frozen, and one decoder per encoder family was trained for 50 epochs using MSE reconstruction loss and Adam with learning rate 10^{-3}. During decoder training, latent codes were L^2-normalized before decoding.

Phase 3: latent density extraction.

- *SimCLR:* a post-hoc full-covariance GMM with $K = 50$ components was fit to 10,000 training embeddings using EM with a maximum of 20 iterations.
- *Unimodal GJE:* the empirical mean and covariance were estimated from the full latent distribution.
- *Parametric GMJE:* the mixture means were taken directly from the learned prototype matrix, normalized to the hypersphere, and the covariance was extracted from the EMA-tracked marginal-regularizer buffer. The component weights were set uniformly as $\pi_k = 1/50$ in the released script.

Phase 4: unconditional latent sampling. Synthetic embeddings were sampled directly from the corresponding latent densities, projected back onto the unit hypersphere by L^2 normalization, and decoded into image space. The script generated 10 repeated sample grids (Sample 2 was used in Fig.5 for demonstration purpose), each using 16 synthetic latent points per method, as well as a separate t-SNE diagnostic using 500 synthetic embeddings and 2000 real test embeddings.