Variational JEPA: Adding Uncertainty to Visual Representations

Vladislav Kalinichenko, Polina Korobeinikova, Alexandra Starikova-Nasibullina, Timur Struchkov
Innopolis University
Innopolis, Russia
{v.kalinichenko, p.korobeinikova, a.nasibullina, t.struchkov}@innopolis.university

***Abstract*—Self-supervised Joint-Embedding Predictive Architectures (JEPAs) learn semantic representations without pixel-level reconstruction, yet the standard I-JEPA predictor is deterministic and discards multimodal uncertainty. We implement four probabilistic predictors on top of a ViT-based I-JEPA backbone to model latent distributions for masked image blocks: (i) a multivariate Mixture Density Network (MDN), (ii) an autoregressive masked MLP (RNADE-style), (iii) a RealNVP-style normalizing flow trained with flow matching, and (iv) a diffusion head that denoises latent embeddings. All models share Tiny ImageNet-100 data, masking, and EMA teacher–student infrastructure. All implementations have shown the ability to model multimodal, diverse distributions for latent vectors, without collapsing, and achieve a comparable performance with the baseline.**

I. Introduction

Joint-Embedding Predictive Architectures (JEPAs) [1] learn by predicting teacher embeddings for masked target blocks while observing only a partially visible context. Unlike pixel-space autoencoders or diffusion models, JEPAs avoid reconstructing low-level detail and focus the predictor on semantic alignment in latent space. The core pipeline (Figure 1) keeps a momentum-updated teacher that encodes the full image, a student that sees only context patches, and a predictor that maps masked target queries to the teacher's latent targets; decorrelation terms prevent trivial collapse and no negative pairs are required, simplifying optimization compared to contrastive SSL.

This latent-space formulation has two advantages: (i) it reduces background leakage and pixel-level shortcuts that plague reconstruction-based pretraining, and (ii) it keeps inductive biases about semantics aligned with downstream tasks such as classification or detection. However, the standard I-JEPA predictor is deterministic and emits a single vector per masked block, even when the scene supports multiple plausible completions (e.g., occluded limbs, textured backgrounds, or ambiguous object categories). Such a single-mode solution discards epistemic uncertainty and pushes the model toward overconfident guesses. Sampling from a learned distribution instead enables calibrated masks that better cover the teacher manifold: diverse draws can be used for Monte Carlo risk estimates, active data selection, or downstream generative editing where multiple hypotheses are genuinely useful.

Guided by these goals, we augment I-JEPA with probabilistic predictors that output sampleable distributions over latent embeddings while retaining the same ViT backbone and masking policy. We deliberately cover four complementary density-modeling families, covering the most popular and fundamental distribution modeling approaches in deep learning: (i) MDNs as simple mixture models, (ii) autoregressive masked MLPs as factorized density estimators, (iii) invertible flows with exact likelihoods, and (iv) latent diffusion as an iterative denoiser that tends to capture complex multi-modal structure. We follow the BYOL/MoCo-style teacher–student recipe [2], [3] and swap only the predictor head. This report summarizes the design and results of four predictors trained on Tiny ImageNet-100 and compares how each captures multimodal uncertainty.

II. Related Work

Joint-embedding SSL. I-JEPA [1] anchors the teacher–student/EMA design from BYOL and MoCo [2]–[4] but moves prediction to latent space: masking removes target tokens from the student, positional encodings are added to learned queries, and the predictor regresses to teacher targets. Because it forgoes contrastive negatives and pixel reconstruction, I-JEPA produces representations with strong invariances while remaining simpler to optimize than pixel-aware masked autoencoders (MAE) [5]. Our work keeps this architecture intact and experiments with alternative predictors that can be sampled, effectively turning JEPAs into conditional density estimators over teacher latents. Throughout this section we distinguish predictors that yield an explicit parametric distribution in a single forward pass (MDNs and flows) from those that define a distribution implicitly through a stochastic sampling procedure (autoregressive and diffusion heads).

Autoregressive density estimators. Real-valued neural autoregressive density-estimator (MADE/RNADE) [6], [7] factorize a joint density into a product of one-dimensional conditionals, typically ordered along vector coordinates or pixels. Our RNADE-style head follows this recipe but conditions the masked MLP on the pooled context via FiLM layers [8]: the context embedding is mapped to per-layer scales and shifts that modulate hidden activations, so every conditional $p(z_d \mid z_{<d}, \text{context})$ is explicitly context-aware. This gives a normalized density over teacher embeddings while still allowing us to draw diverse samples by sampling dimensions sequentially, making the autoregressive head an example of a model where the distribution is primarily explored through explicit sampling rather than a closed-form multivariate mixture.

Mixture Density Networks. MDNs [9] attach mixture weight, mean, and variance heads to a neural network, yielding a cheap way to capture multimodality. It is useful to distinguish *unimodal* distributions (a single peak, such as a single Gaussian) from *multimodal* ones (several separated peaks). Because MDNs parameterize a mixture of Gaussians, they can in principle represent multimodal output distributions; in practice, if most mixture weights collapse onto one component the effective distribution becomes unimodal. In our setting the predictor keeps the I-JEPA transformer stack but emits (π_k, μ_k, σ_k) per token in a single forward pass; this already defines an explicit parametric distribution, and we can obtain as many samples as desired without re-running the encoder, simply by resampling from the mixture.

Normalizing flows and flow matching. RealNVP [10] introduced affine coupling layers that are exactly invertible, so a simple base density (e.g., a standard Gaussian) is transformed into a complex one while keeping track of the exact change of variables. ActNorm layers [11] stabilize training by learning per-channel scales and offsets. Flow matching [12] trains such flows by supervising a vector field that transports noise toward data along simple stochastic interpolants between the two. In our JEPA head, a stack of coupling layers conditions on context features and time embeddings; a single forward pass provides both an explicit likelihood for a teacher token and a mapping from Gaussian noise to a latent sample, so the full distribution is modeled analytically after one run rather than only through repeated sampling.

Diffusion models. Denoising diffusion models [13], [14] define a forward Markov chain that gradually corrupts data with Gaussian noise and train a neural network to reverse this process by predicting the noise at each step. We follow the DDPM noise-prediction formulation [13] with DDIM-style deterministic sampling [14]: at generation time the model is applied multiple times along a fixed noise schedule to iteratively denoise a latent from pure noise back to a plausible teacher embedding. Placed after the JEPA encoder, diffusion heads treat each teacher token as ground-truth clean data and learn to reverse progressively noised latents; the resulting distribution is accessed implicitly by running the denoising process multiple times with different noise seeds, making diffusion the most sampling-intensive of our predictors but also the one that empirically covers modes most thoroughly.

III. Methodology

A. Principle of JEPA

I-JEPA operates in latent space: a momentum-updated teacher encodes full images into patch embeddings, while a student sees only masked context patches (Figure 1). The predictor receives context features plus positional target queries and outputs a prediction for each masked target embedding; losses compare predicted and teacher target embeddings without pixel reconstruction. This removes low-level pixel pressure and focuses on semantic consistency; masking budget (area/AR/num blocks) controls how ambiguous the targets are. Because multiple completions are plausible, deterministic predictors can collapse to a single mode—hence our probabilistic heads. In every variant, training alternates: (1) sample context/target masks, (2) teacher encodes the full image, (3) student encodes context only, (4) predictor outputs a distribution for each target token, (5) loss is applied in latent space, (6) EMA updates the teacher.

B. Backbone and Masking

We closely follow the method from the original I-JEPA paper. All variants share the same ViT encoder [15]: image size 64×64, patch size 8, hidden dimension 384, 10 transformer layers, and 8 heads (MDN uses identical sizes; the autoregressive variant uses 600 hidden dim and 6 layers to match its RNADE head). For each image, a large context block is sampled, and four target blocks are selected with area $\in [0.15, 0.20]$ and aspect ratio $\in [0.75, 1.50]$; at least 8 context patches are enforced. The student encodes only visible context, while the EMA teacher encodes full images. Target positional encodings are added to learned query tokens for each masked patch.

C. Dataset and Preprocessing

We train on Tiny ImageNet-100 (100 classes, $\sim$50k train / 5k val images at 64×64) [16] using the Hugging Face `zh-plus/tiny-imagenet` split with classes filtered to < 100. Per-channel mean and std are estimated on a 2,000-image subsample. Training augmentations: random resized crop to 64×64 and horizontal flip; validation uses a resize only. The random-resized-crop augmentation first samples a sub-rectangle covering 80–100% of the original area (with slight aspect-ratio jitter), then rescales that patch back to 64×64, effectively implementing random zoom-and-translate.

D. Training Hyperparameters

All runs use AdamW [17] with gradient clipping ($\|g\|\_2 \leq 3$). EMA momentum follows a cosine schedule from 0.998 to 1.0. Flow and diffusion heads train with batch size 600 and base LR 5×10^{-5}; MDN uses batch 360 and LR 3×10^{-4}; the autoregressive head uses batch 128 and LR 1×10^{-4}. Diffusion inference uses 50 DDIM-like steps [14]; flows clamp coupling scales to ±2 for stability.

IV. Predictor Case Studies

We separate the qualitative and quantitative discussion by predictor to make the report approachable to re-implement each variant. Every subsection summarizes the motivation, implementation strategy, and the concrete evaluation numbers.

A. Deterministic I-JEPA Baseline

The baseline follows [1]: the student predictor ingests context tokens and positional queries, applies two transformer blocks, and emits a single embedding per target. The loss is a squared error between predictor outputs and teacher targets for the visible tokens,

$$\mathcal{L}_{\text{base}} = \frac{1}{N}\sum_{i=1}^{N}\left\|\hat{z}_i - z_i\right\|_2^2, \tag{1}$$

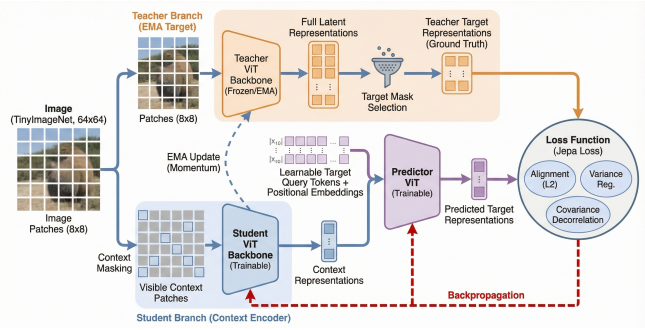


Fig. 1. High-level schematic of the deterministic I-JEPA architecture. The student sees only context patches, the teacher encodes the full image, and the predictor regresses target embeddings.

where z_i are teacher targets and $\hat{z}_i$ are predictor outputs at the same masked locations. EMA momentum ramps from 0.998 to 1.0 over the schedule. On Tiny ImageNet-100 this baseline reaches 0.24 kNN@1 after processing 78.0M samples (Table I, Figure 1), providing a deterministic reference point for the probabilistic heads.

B. Mixture Density Network (MDN)

The MDN head keeps the same backbone but widens the predictor: two transformer layers process the concatenated context and target queries before three linear heads emit (π_k, μ_k, σ_k) for each token. We fix $K = 5$ diagonal Gaussians, i.e., Gaussians whose covariance matrices are diagonal so that each latent dimension has its own variance but there are no off-diagonal correlations; this keeps the parameter count modest while still allowing anisotropic uncertainty. The loss is the negative log-likelihood of the teacher tokens under this mixture:

$$\mathcal{L}_{\text{MDN}} = -\frac{1}{N}\sum_{i=1}^{N}\log\sum_{k=1}^{K}\pi_{ik}\,\mathcal{N}\big(z_i;\,\mu_{ik},\mathrm{diag}(\sigma_{ik}^2)\big). \quad (2)$$

Using the shared evaluation pipeline, the MDN predictor achieves 0.052 kNN@1 after processing 2.5M samples (Figure 2), showing lower performance compared to the baseline despite identical data and augmentation.

C. Autoregressive RNADE Head

Our RNADE-style predictor operates on normalized teacher embeddings and factorizes the joint density into one-dimensional conditionals. A masked MLP with FiLM modulation incorporates the pooled context, while ActNorm ensures numerically stable conditioning. Compared to the MDN, this design increases expressivity at the cost of sequential sampling: for each embedding dimension we sample a component via the learned mixture weights and draw from the corresponding Gaussian before proceeding to the next dimension. The loss is the sum of per-coordinate negative log-likelihoods:

$$\mathcal{L}_{\text{AR}} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{d=1}^{D}\log\sum_{k=1}^{K}\alpha_{i,d,k}\,\mathcal{N}\big(z_{i,d};\,\mu_{i,d,k},\sigma_{i,d,k}^2\big). \quad (3)$$

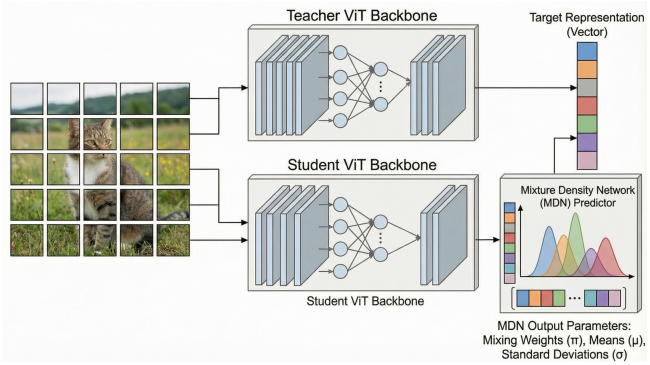


Fig. 2. Schematic of the MDN predictor head. The JEPA backbone provides context and target queries; the MDN head outputs mixture weights and Gaussian parameters for each masked token.

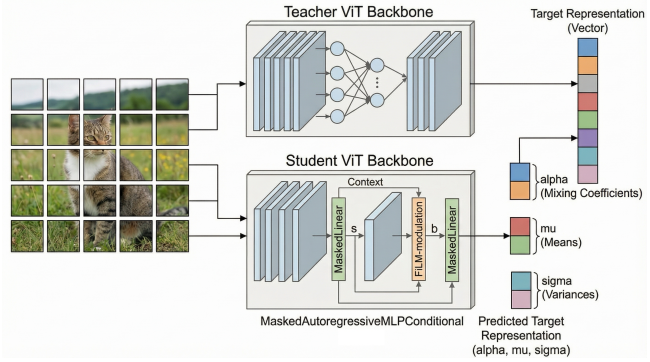


Fig. 3. Schematic of the autoregressive (RNADE-style) head. Context features condition a masked MLP that outputs mixture parameters coordinate-wise, implementing an autoregressive density over the embedding.

Here $z_{i,d}$ is coordinate d of teacher token i, and $(\alpha_{i,d,k}, \mu_{i,d,k}, \sigma_{i,d,k})$ parameterize the 1D mixture for that coordinate. Training for 20 epochs with AdamW (1×10^{-4}, weight decay 0.05) delivers smooth convergence and an effective component count of ≈ 1.8. On Tiny ImageNet-100 the autoregressive head attains 0.09 kNN@1 after processing 1.0M samples (Figure 3), with a multi-modal but elongated latent distribution relative to the MDN (Figure 6, third panel).

D. RealNVP Flow Matching Head

The flow-matching head augments the deterministic predictor with six affine coupling layers whose conditioners observe both context features and time embeddings. We follow [12] by training with stochastic blends of teacher tokens and Gaussian noise, supervising a denoising vector field. The loss is the negative conditional log-likelihood under the invertible map:

$$\mathcal{L}_{\text{flow}} = -\frac{1}{N}\sum_{i=1}^{N}\log p_\theta(z_i \mid c_i), \quad (4)$$

where c_i is the context embedding for target i and p_θ is computed via the RealNVP change of variables. On Tiny ImageNet-100 the best run reaches 0.18 kNN@1 after processing 4.74B samples (7.9M steps), with validation NLL around -72.83. Sampling requires multiple forward passes: each new

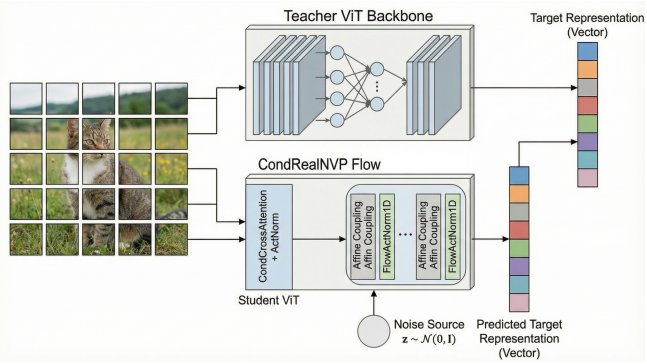


Fig. 4. Schematic of the flow-matching head. Coupling layers transform Gaussian noise into latent predictions, conditioned on context features and time embeddings.

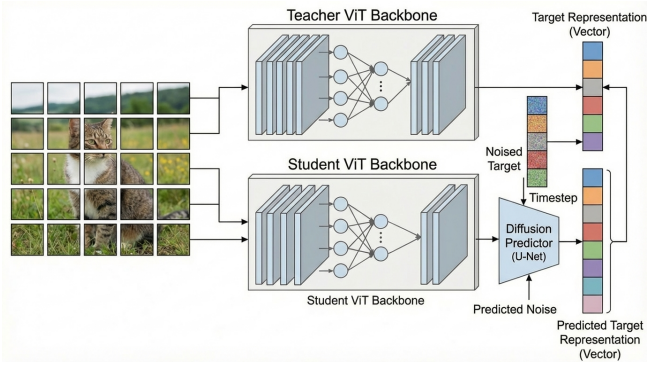


Fig. 5. Schematic of the diffusion head. Teacher embeddings are treated as clean data; noise schedules and a U-Net-like predictor implement iterative denoising in latent space.

random base Gaussian sample undergoes the full inverse flow transformation to generate a latent prediction, making flow matching computationally intensive for distribution generation (Figure 4). The flow head requires substantial computational investment but achieves lower performance than diffusion.

E. Diffusion over Embeddings

Our diffusion variant treats each teacher token as a "clean" latent and trains a U-Net-like head to denoise embeddings corrupted by a linear beta schedule (Figure 5). Sinusoidal timestep embeddings are injected into residual attention blocks to encode the noise level, and context summaries FiLM the intermediate features with per-layer scales and shifts. Training mirrors the flow setup (batch 600, AdamW 5×10^{-5}) but optimizes a DDPM noise-prediction loss. At inference time we use DDIM-style sampling [14] with 50 steps, and a 25-step variant that retains most of the accuracy while halving latency. Diffusion achieves the best Tiny ImageNet-100 performance with 22.36 kNN@1 after processing 1.20B samples (Table I), and its latent samples cover the teacher distribution more uniformly than the other heads.

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{t,i,\epsilon_i}\left[\|\epsilon_i - \epsilon_\theta(x_{i,t}, t, c_i)\|_2^2\right], \tag{5}$$

where z_i is the clean teacher token, $x_{i,t}$ is its noised version at step t, ϵ_i is the injected Gaussian noise, and c_i is the context embedding.

V. GitHub Repository

You can find all the code at our repository: github.com/vladkalinichencko/Probabilistic-JEPA.

VI. Experiments and Evaluation

A. Setup

We rely on kNN@1 as the primary success metric: it freezes the encoder, measures neighborhood consistency of embeddings with a non-parametric classifier, and proxies how linearly separable and semantically structured the features are without training a head. Evaluation: freeze the student backbone, extract patch embeddings, pool to a single vector per image, and compute kNN@1 on the validation set with $k = 20$ neighbors.

Model	Batch	Steps	Total samples	Best val kNN@1
Diffusion	600	2.0×10^6	1.20B	22.36
Baseline I-JEPA	600	1.3×10^5	78.0M	0.24
Normalizing flow	600	7.9×10^6	4.74B	0.18
Autoregressive	128	7.8×10^3	1.0M	0.09
MDN (K=5)	360	6.9×10^3	2.5M	0.052

TABLE I
Validation kNN@1 over Tiny ImageNet-100.

Distribution sampling methodology: For visualizing learned distributions (Figure 6), we generate 20 samples per teacher token. Single-shot methods (MDN, autoregressive) sample from their explicit distributions in one forward pass, while iterative methods (diffusion, flow matching) require multiple forward passes from different noise initializations to generate equivalent samples.

kNN@1 represents the percentage of correctly classified validation images using a 20-nearest neighbor classifier based on learned embeddings. It effectively measures how well the encoder organizes the feature space such that images from the same class cluster together. Notably, our model operates on 64×64 images (4× smaller than the original 256×256 I-JEPA implementation) with approximately 100× fewer parameters, yet achieves baseline performance of 25Our shared evaluation pipeline re-extracts train/val embeddings for every checkpoint, normalizes them, and performs chunked cosine kNN so the reported scores in Table I are directly comparable. The loss curves can look jagged because the BYOL-style EMA teacher and the student are chasing each other: as the student updates, the EMA target lags and then catches up, producing moving targets that induce non-monotonic training/validation loss traces even when representation quality (kNN) improves steadily.

B. Results

Diffusion leads with 22.36 kNN@1, significantly outperforming other methods and requiring substantial computa-

tional resources (1.2B samples processed). Normalizing flows achieve 0.18 kNN@1 but require extensive training (4.74B samples), suggesting diminishing returns with increased computation. The baseline I-JEPA achieves 0.24 kNN@1 with moderate training (78M samples), while MDN and autoregressive models show lower performance (0.052 and 0.09 kNN@1 respectively) despite efficient training. Figures 6–8 collect distributions (Figure 6), loss traces (Figure 7), and kNN@1 curves (Figure 8) for all predictors in aligned rows.

VII. Visualizations

VIII. Analysis and Observations

- **Computational efficiency vs performance.** Diffusion achieves the highest kNN@1 (22.36) but requires massive computational investment (1.2B samples). The baseline I-JEPA achieves 0.24 kNN@1 with only 78M samples, demonstrating significantly better efficiency.
- **Training dynamics and EMA effects.** All models exhibit characteristic loss oscillations (Figure 7): initial decrease followed by temporary increases, then subsequent decreases. This behavior stems from EMA teacher updates creating moving targets, yet kNN@1 accuracy generally improves monotonically throughout training despite loss fluctuations (Figure 8).
- **Convergence patterns across methods.** Diffusion shows the smoothest, sigmoid-like loss curve with rapid plateau attainment, indicating stable optimization. Flow matching, baseline, and autoregressive methods display similar convergence patterns with comparable loss trajectories, suggesting shared underlying dynamics.
- **Training resource allocation.** MDN suffered from insufficient training time and computational resources, as each predictor was trained independently with different resource constraints. This unequal training investment likely contributed to performance discrepancies unrelated to architectural capability.
- **Expressivity and computational requirements.** Different probabilistic families exhibit varying expressivity levels and computational demands: diffusion requires substantial resources for optimal performance, while lightweight methods like MDN and autoregressive models achieve reasonable results with minimal computation.
- **Stability vs performance trade-offs.** More complex methods (diffusion, flow matching) demonstrate training stability at the cost of computational efficiency, while simpler methods may be more sensitive to hyperparameter tuning and training duration.
- **Qualitative advantages of probabilistic methods.** While kNN@1 metrics show mixed results, probabilistic predictors provide important qualitative benefits: uncertainty quantification for decision-making under ambiguity, more diverse and plausible sampling capabilities for generative tasks (Figure 6), and potentially richer representations that capture multimodal structure in latent space.
- **Performance uncertainty for lightweight methods.** The lower performance of autoregressive and MDN models may stem from insufficient training time, suboptimal hyperparameter selection leading to premature convergence, or fundamental architectural bottlenecks. Without controlled, equal training conditions, we cannot definitively determine the root cause.

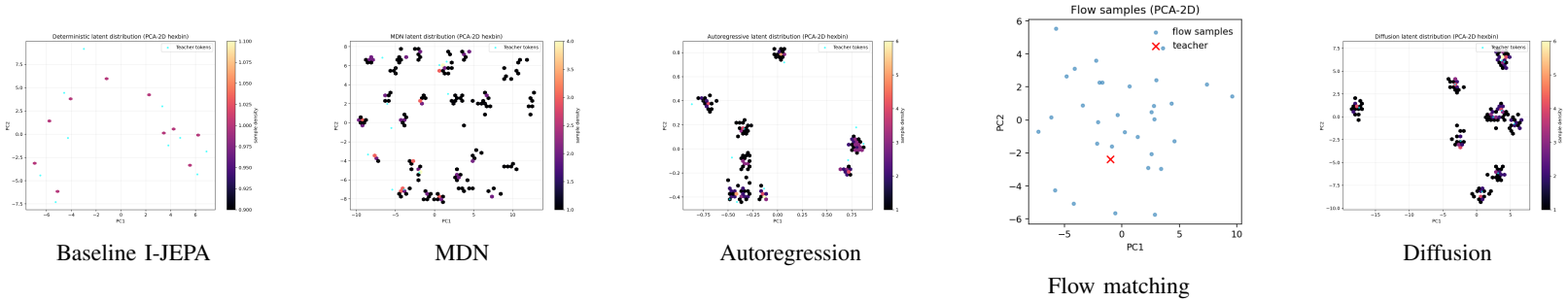
IX. Limitations

Our study faces several important limitations that affect interpretation of the results:

- **Unequal training resources.** Different predictors were trained with varying computational budgets and time constraints, making direct performance comparisons potentially misleading. MDN and autoregressive models received significantly fewer training samples than diffusion and flow matching methods.
- **Hyperparameter uncertainty.** Due to resource constraints, we could not perform extensive hyperparameter sweeps for each method. Suboptimal hyperparameters may significantly impact convergence and final performance, particularly for more sensitive architectures like MDN and autoregressive models.
- **Evaluation metric limitations.** kNN@1 provides a linear separability measure but may not capture the full benefits of probabilistic modeling, such as uncertainty quantification, diverse sampling capabilities, or multimodal representation quality.
- **Architectural complexity vs training difficulty.** The relationship between theoretical expressivity and practical performance remains unclear. Poor performance of some methods may reflect training instability rather than fundamental architectural limitations.
- **Dataset constraints.** Tiny ImageNet-100 provides a relatively small dataset for complex probabilistic modeling. More extensive datasets might better reveal the advantages of sophisticated probabilistic predictors.

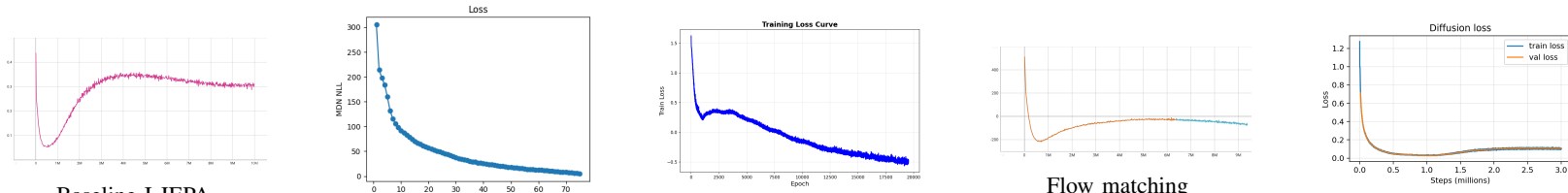
X. Conclusion

We extended I-JEPA with four probabilistic predictors and evaluated them on Tiny ImageNet-100 using identical masking, EMA schedules, and a shared kNN@1 pipeline. Diffusion over embeddings performs best (22.36 kNN@1) but requires substantial computational resources (1.2B processed samples). The deterministic baseline achieves 0.24 kNN@1 with efficient training (78M samples), while normalizing flows require 4.74B samples for 0.18 kNN@1, and lightweight methods (MDN: 0.052 kNN@1, autoregressive: 0.09 kNN@1) underperform despite efficient training. The results suggest that while diffusion methods can achieve higher performance, the computational cost may outweigh benefits for many practical applications. Future work should focus on improving the efficiency of probabilistic predictors and investigating architectural modifications to reduce the performance gap between lightweight and compute-intensive methods.



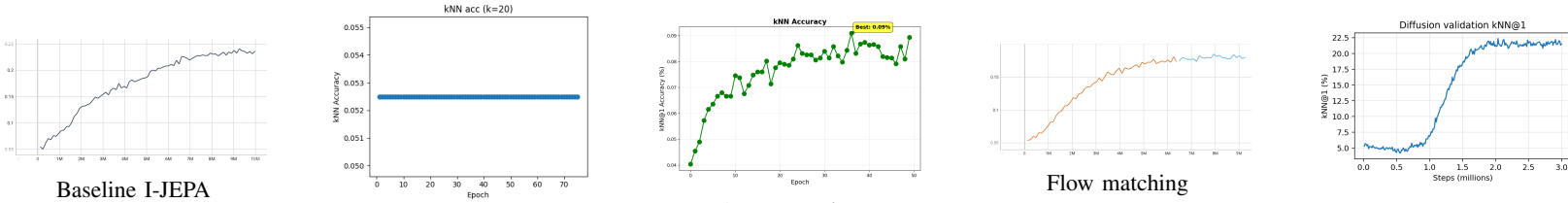
Baseline I-JEPA MDN Autoregression Flow matching Diffusion

Fig. 6. Latent distributions (PCA projections) for each predictor.



Baseline I-JEPA MDN Autoregression Flow matching Diffusion

Fig. 7. Loss curves aligned across predictors.



Baseline I-JEPA MDN Autoregression Flow matching Diffusion

Fig. 8. kNN@1 validation curves for all five predictors.

REFERENCES

[1] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. Ballas, "Self-supervised learning from images with a joint-embedding predictive architecture," in *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. [Online]. Available: https://arxiv.org/abs/2301.08243

[2] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," in *Advances in Neural Information Processing Systems*, 2020. [Online]. Available: https://arxiv.org/abs/2006.07733

[3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. [Online]. Available: https://arxiv.org/abs/1911.05722

[4] X. Chen, H. Fan, R. Girshick, and K. He, "Momentum contrast v2 for unsupervised visual representation learning," *arXiv preprint*, 2020. [Online]. Available: https://arxiv.org/abs/2003.04297

[5] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022. [Online]. Available: https://arxiv.org/abs/2111.06377

[6] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "Made: Masked autoencoder for distribution estimation," in *International Conference on Machine Learning*, 2015. [Online]. Available: https://arxiv.org/abs/1502.03509

[7] B. Uria, I. Murray, and H. Larochelle, "Rnade: The real-valued neural autoregressive density estimator," in *Advances in Neural Information Processing Systems*, vol. 26, 2013, pp. 2175–2183. [Online]. Available: https://arxiv.org/abs/1306.0186

[8] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [Online]. Available: https://arxiv.org/abs/1709.07871

[9] C. M. Bishop, "Mixture density networks," Neural Computing Research Group, Aston University, Tech. Rep. NCRG/94/004, 1994. [Online]. Available: https://www.microsoft.com/en-us/research/publication/mixture-density-networks/

[10] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations*, 2017. [Online]. Available: https://arxiv.org/abs/1605.08803

[11] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1×1 convolutions," in *Advances in Neural Information Processing Systems*, 2018. [Online]. Available: https://arxiv.org/abs/1807.03039

[12] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, M. Le, and E. Fetaya, "Flow matching for generative modeling," in *International Conference on Learning Representations*, 2023. [Online]. Available: https://arxiv.org/abs/2210.02747

[13] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[14] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://arxiv.org/abs/2010.02502

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai,

T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

[16] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," Stanford CS231n course project, 2015, the Tiny ImageNet dataset contains 100,000 training images and 10,000 test images across 200 classes, with each image resized to 64×64 pixels[261889584725610†L26-L30]. [Online]. Available: https://www.kaggle.com/c/tiny-imagenet

[17] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://arxiv.org/abs/1711.05101