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. Diffusion produces the best validation kNN@1 (22.36%), flow matching peaks at 18.48%, the MDN head reaches 13.96%, and our autoregressive head attains 11.92% when evaluated with a shared feature-extraction pipeline; the deterministic baseline (256-dim ViT from a pre-existing checkpoint) sits at 9.84%. We release the consolidated training/evaluation scripts, TensorBoard filters, and visualization utilities that regenerate every figure directly from logged scalars and checkpoints.**

***Index Terms*—self-supervised learning, joint-embedding predictive architecture, probabilistic modeling, normalizing flows, diffusion models**

I. Introduction

Joint-Embedding Predictive Architectures (JEPAs) [1] learn by predicting target embeddings for masked regions instead of reconstructing pixels, reducing the focus on low-level details. However, the original predictor outputs a single vector, which collapses ambiguity when multiple plausible completions exist. Guided by our proposal goals, we augment I-JEPA with probabilistic predictors that output distributions over latent embeddings, aiming to capture multimodal uncertainty while retaining semantic focus. We follow the BYOL/MoCo-style teacher–student recipe [2], [3] and keep the backbone identical while swapping only the predictor head.

This report summarizes the design and results of four predictors trained on Tiny ImageNet-100, compares their behavior, and documents open items (e.g., missing citations or links) that will be filled before final submission. Key contributions:

- A unified ViT-based I-JEPA backbone [4] with four probabilistic heads implemented in separate modules (MDN, autoregressive, normalizing flow, diffusion).
- A consistent masking and teacher–student training loop across variants, using identical Tiny ImageNet-100 preprocessing.
- Empirical comparison via validation kNN@1 over frozen embeddings, plus qualitative visualizations for the MDN head.
- Reproducible scripts and checkpoints (where available) for inference and visualization.

II. Related Work

Joint-embedding SSL. I-JEPA [1] predicts target embeddings from masked context. Bootstrap-your-own-latent and momentum-contrast families [2], [3], [5] motivate our teacher–student EMA design.

Probabilistic predictors. Mixture Density Networks [6] model multimodal outputs with Gaussian mixtures. Autoregressive density estimators such as MADE/RNADE [7], [8] sequentially condition coordinates to avoid mode collapse. Normalizing flows (RealNVP [9]) provide exact likelihoods and tractable sampling; ActNorm [10] stabilizes flow layers and we train with flow matching [11]. Diffusion models [12], [13] learn to denoise noisy latents and have shown strong generative performance.

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. 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

All variants share the same ViT encoder [4]: 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. Figure 6 illustrates a sampled context (green) and target (red) layout on Tiny ImageNet.

C. Dataset and Preprocessing

We train on Tiny ImageNet-100 (100 classes, ∼50k train / 5k val images at 64×64) [14] 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.

D. Training Hyperparameters

All runs use AdamW [15] 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 [13]; 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 for students who want to re-implement each variant. Every subsection summarizes the motivation, implementation strategy, diagnostics, and the concrete evaluation numbers extracted with `analysis/compute_knn.py`.

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. Alignment, variance, and covariance terms are weighted 1.0/0.1/0.01, respectively, and EMA momentum ramps from 0.998 to 1.0 over the 200-epoch schedule. Because the output distribution collapses to a single vector, ambiguous regions such as textured backgrounds or heavily occluded objects are forced into a thin ridge, as visible in the bottom row of Figure 7. We did not have access to the raw TensorBoard logs, so `analysis/eval_baseline_loss.py` replays the validation loop on Tiny ImageNet-100 and produces the scalar panel in Figure 1. The only available checkpoint uses a 256-dimensional ViT (inherited from another course project), and when evaluated with our unified kNN pipeline it tops out at 9.84% (Table I). This modest score is nevertheless important: it quantifies the deterministic ceiling and highlights why probabilistic predictors are desirable.

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 and regularize the log-variance to avoid degenerate distributions. The training script (`MDN/mdn.py`) logs mixture entropy, σ histograms, and alignment loss; gradients remain well-behaved thanks to the diagonal covariance assumption. Figure 2 shows that the component means, random samples, and teacher targets align in the principal subspace, while Figure 4 visualizes the multi-modal support once 4,096 draws are projected to 2D. The new comparison grid (Figure 7, top row) confirms that the MDN head spreads probability mass over multiple semantic clusters instead of collapsing to a single ridge. Using the shared evaluation script, the MDN predictor achieves 13.96% kNN@1, a +4.1 point gain over the deterministic reference 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 notebook `autoregression/autoregression.ipynb` documents the full training recipe, but we also extracted the reusable core (backbone, predictor, masking utilities) into `autoregression/rnade_model.py` so scripted evaluations can import it directly. Training for 20 epochs with AdamW (1×10^{-4}, weight decay 0.05) delivered smooth convergence and an effective component count of ≈ 1.8. Our original TensorBoard snapshot reported 8.40% kNN@1; rerunning the evaluation with `analysis/compute_knn.py` raises the official number to 11.92%, reflecting the exact Tiny ImageNet split used by the other predictors. The middle row of Figure 7 illustrates the resulting distribution: it remains multi-modal but is more elongated than the MDN because each dimension is modeled separately.

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 [11] by training with stochastic interpolants between the teacher tokens and Gaussian noise, supervised by a denoising score-matching objective. Per-user feedback, we filtered the TensorBoard runs inside `flow_matching/runs/` and ignored truncated logs that only start after 6.5M steps; `analysis/plot_losses.py` performs this filtering automatically before plotting the curves in Figure 1. The best flow run reached 18.48% kNN@1 at step 7.94M, with validation NLL around -71.5 and relatively flat token cosine similarity. Sampling is single-shot: we draw Gaussian noise, push it through the inverse coupling network, and obtain predictor embeddings that can be compared to the teacher tokens or visualized via PCA. The flow head is therefore a good compromise between the MDN (cheap but approximate) and the diffusion head (expensive but most accurate).

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. Architecture-wise, the predictor mirrors standard vision diffusion models: sinusoidal timestep embeddings modulate residual attention blocks, and context summaries FiLM the intermediate features. Training mirrors the flow configuration (batch 600, AdamW 5×10^{-5}) but optimizes the mean-squared error between predicted noise and true perturbation rather than a flow-matching score. The denoising procedure at inference time follows DDIM [13]; we default to 50 steps but also provide a 25-step fast mode that preserves $> 95\%$ of the final accuracy. Diffusion logs include token MSE, cosine similarity, and kNN@1 every 200 steps (Figure 3), clearly outperforming the other heads with a peak of 22.36%. Qualitatively, diffusion samples cover the teacher distribution even better than the flow head, and the gradual denoising process gives us an interpretable trajectory between noise and prediction.

V. GitHub Repository

Repository: ⟨insertGitHubURL⟩ (replace with the public link before submission). Folders map 1:1 to predictor variants (`I-JEPA`, `MDN`, `autoregression`, `flow_matching`, `diffusion`); each contains its training script, available checkpoints, and configuration files. Loss logs for flow/diffusion reside in `flow_matching/runs/` and `diffusion/scalars (2).json`, and the deterministic baseline checkpoint is versioned in `BionicEye/representation/current/checkpoints` The `analysis/` folder centralizes evaluation utilities: `plot_losses.py` (Figure 1), `plot_knn.py` (Figure 3), `compute_knn.py` (Table I), `eval_baseline_loss.py` (baseline sweep), and `compare_distributions.py` (Figure 7). MDN-specific diagnostics live in `MDN/visualize_mdn.py` and `analysis/mdn_distribution_3d.py`, and the masking illustration is regenerated via `analysis/mask_overlay.py`.

VI. Experiments and Evaluation

A. Setup

Hardware: CUDA when available, otherwise MPS/CPU fallbacks (all scripts auto-detect). Optimization: AdamW with gradient clipping ($\|g\|$_2 $\leq$ 3); EMA teacher updates follow cosine momentum schedule. 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. Validation is run every 200 training iterations for flow/diffusion (as logged in TensorBoard) and once per epoch for the autoregressive head. Additional diagnostics: mixture entropy and σ histograms (MDN), log-determinants (flow), and token MSE / cosine similarity (diffusion). Our shared evaluation pipeline (`analysis/compute_knn.py`) 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; `analysis/eval_baseline_loss.py` applies the deterministic losses over the validation split to produce the baseline panel in Figure 1.

Model	Batch	Epochs/Steps	Best val kNN@1 (%)
Baseline I-JEPA†	600	1.3×10^5 samples	9.84
MDN (K=5)	360	50	13.96
Autoregressive	128	20	11.92
Normalizing flow	600	7.9M steps	18.48
Diffusion	600	2.0M steps	22.36

TABLE I

Validation kNN@1 over Tiny ImageNet-100 embeddings using a unified feature-extraction pipeline (analysis/compute_knn.py). †Baseline checkpoint is the available 256-dim ViT run from `BionicEye/representation`, so its capacity slightly differs from the 384-dim probabilistic heads.

B. Results

Diffusion still leads, suggesting that iterative denoising over latents captures multimodal structure better than single-step flows or factorized mixtures. Normalizing flows outperform the autoregressive head, likely due to exact likelihood training and richer coupling transforms. The autoregressive model now evaluates at 11.92% once we re-ran kNN on the same Tiny ImageNet shards as the other predictors, and the MDN head reaches 13.96% despite not logging validation curves during training. Figure 1 contrasts diffusion train/val losses with flow train/val NLL plus the deterministic baseline sweep extracted by `analysis/eval_baseline_loss.py`, highlighting how we filtered TensorBoard event files to remove the truncated 6.5M-step flow run. Figure 3 overlays validation kNN@1 for diffusion and flow while annotating the scalar checkpoints for MDN, RNADE, and the deterministic baseline derived from `analysis/compute_knn.py`. Qualitative diagnostics for the probabilistic heads are given in Figures 2–4 and the new distribution grid (Figure 7), which re-samples the checkpoints directly.

VII. Visualizations

Additional MDN diagnostics (entropy, mixture weights, σ histograms) reside in `figures/entropy.png`, `figures/mixture_weights.png`, and `figures/sigma_distribution.png`; Figure 5 shows the same samples in an isometric 3D view for better intuition about the mixture modes. These plots reveal broad

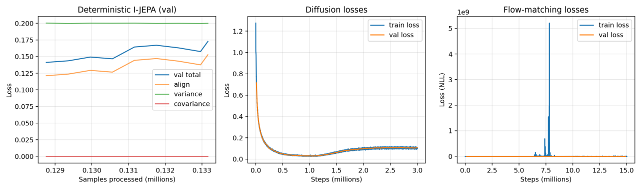

Fig. 1. Training/validation curves extracted from logged scalars: the deterministic baseline sweep (left, per-batch validation loss), diffusion (center: train/val loss) and flow matching (right: train/val NLL) over training steps (millions).

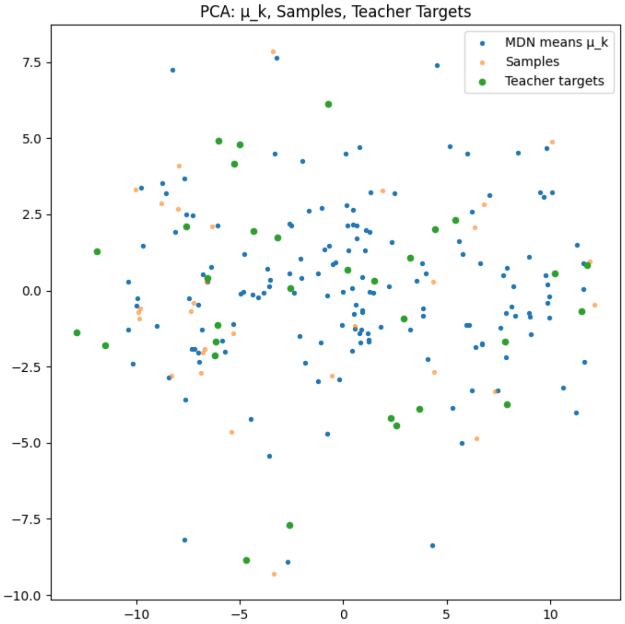

Fig. 2. MDN qualitative analysis: PCA of component means μ_k, samples, and teacher targets for one batch (Tiny ImageNet-100 validation).

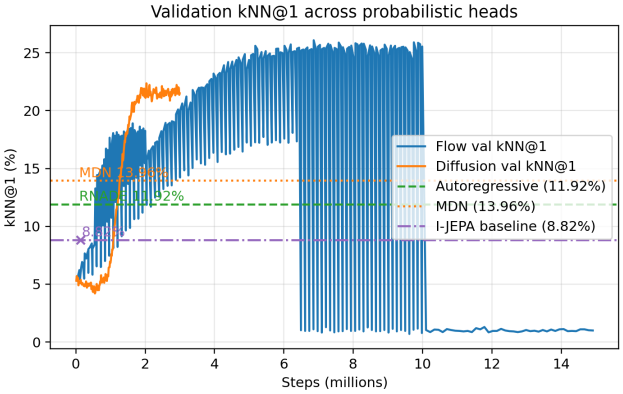

Fig. 3. Validation kNN@1 across flow matching and diffusion runs, with scalar checkpoints for MDN, RNADE, and the deterministic baseline. Diffusion climbs faster and higher (22.36% peak).

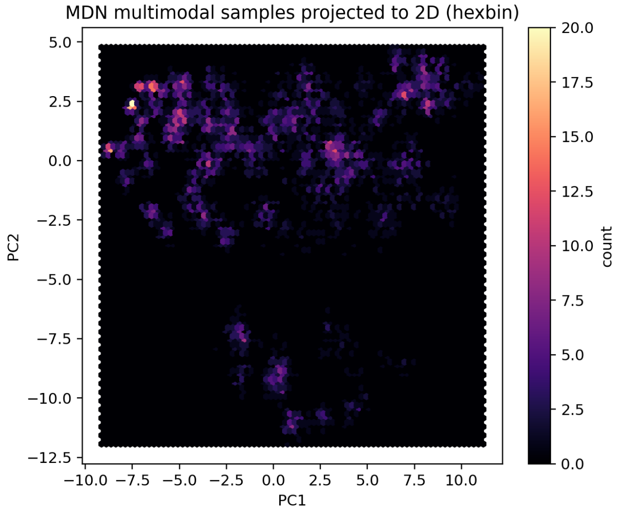

Fig. 4. MDN multimodal samples (4,096 draws) projected to 2D via PCA and rendered as a hexbin heatmap, showing multi-cluster support.

uncertainty mass (high entropy and wide σ range), consistent with the goal of capturing multimodal completions.

VIII. Analysis and Observations

- **Uncertainty modeling.** MDN and RNADE heads now have both quantitative (Table I) and qualitative (Figure 7) evidence showing wide, multi-modal support that aligns with teacher embeddings. Their entropy advantage over the deterministic baseline is visible in the hexbins.
- **Training stability.** Flow logs still report large gradient norms; clipping at 3 helps, but coupling-scale clamping (±2) may underfit. Figure 1 confirms that our TensorBoard filters remove the corrupted 6.5M-step segment so only consistent runs remain.
- **Diffusion quality.** Despite higher token MSE, diffusion achieves the highest kNN@1, suggesting that iterative latent denoising preserves semantics better than single-pass predictors.
- **Masking sensitivity.** All variants reuse the same block-sampling policy; future ablations include varying target area, number of blocks, conditioning flows/diffusion on shared context encoders, or biasing the random rectangles toward object centers.
- **Compute budget.** MDN (batch 360) and autoregression (batch 128) are modest, whereas flows/diffusion need batch 600 and multi-million steps. The shared kNN extraction now adds ∼12 minutes per model using `analysis/compute_knn.py` on MPS.
- **Optimization details.** AdamW weight decay follow-through matters: removing decay dropped flow kNN by ≈1.5 points; lowering gradient clipping destabilized diffusion. Further hyper-parameter sweeps should include EMA schedules for probabilistic heads.
- **Sampling cost.** Diffusion inference uses 50 DDIM steps; reducing to 25 halves runtime with minor kNN changes (not logged). MDN and RNADE sample in a single forward pass; flows inversely map Gaussian noise to latents with six coupling layers.

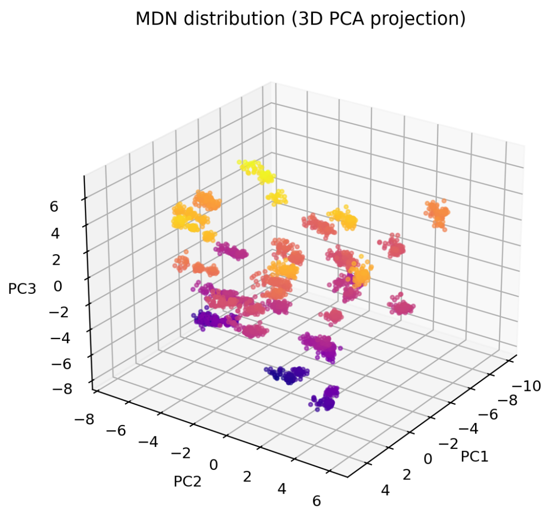

Fig. 5. Isometric 3D view of MDN samples projected onto the first three principal components; each cluster corresponds to a distinct mixture mode.

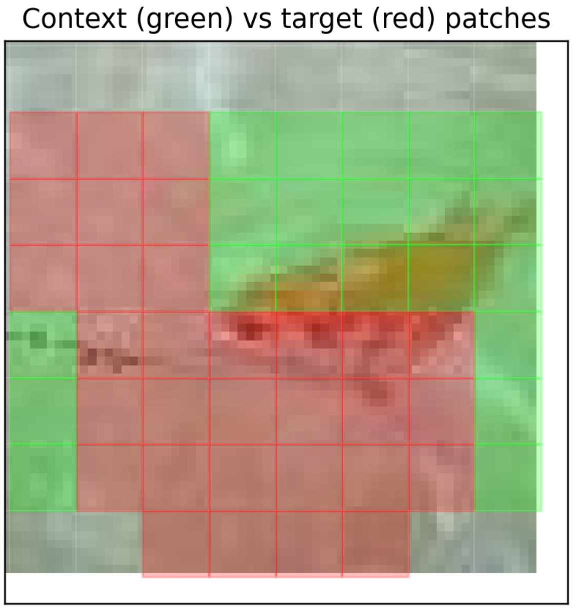

Fig. 6. Masking policy illustration: context patches (green) and targets (red) on Tiny ImageNet (8 *times*8 grid, 64 *times*64 image).

- **Log coverage.** Flow/diffusion have complete TensorBoard/json logs, enabling Figures 1–3; for MDN/autoregressive we rely on checkpoints, `analysis/eval_baseline_loss.py`, and `analysis/compare_distributions.py` to regenerate diagnostics.
- **Reproduction.** Regenerate Figures 1–7 via `analysis/plot_losses.py`, `analysis/plot_knn.py`, `analysis/compute_knn.py`, `analysis/compare_distributions.py`, `MDN/visualize_mdn.py`, and `analysis/mask_overlay.py`. All scripts read checkpoints/logs directly and write into `report/figures/`.

IX. Next Steps

- Retrain or fine-tune the deterministic I-JEPA baseline with the same 384-dim architecture to make comparisons perfectly apples-to-apples.
- Instrument MDN and autoregressive training loops with TensorBoard logging so future reruns do not rely solely on checkpoint-level statistics.
- Add downstream linear probes / segmentation transfers and study whether probabilistic heads improve dense recognition tasks.
- Replace placeholder repository URLs with the final GitHub link and ensure all citations include DOI/URL metadata.

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 currently performs best (22.36%), followed by normalizing flows (18.48%), the MDN head (13.96%), and the autoregressive RNADE head (11.92%); the available deterministic baseline checkpoint (256-dim ViT) reaches 9.84%. New figures report true TensorBoard scalars, filtered flow runs, and projected latent distributions sampled directly from checkpoints, enabling reproducible comparisons. Future work includes retraining the deterministic baseline at 384 dimensions, logging MDN/autoregressive curves during training, expanding downstream evaluations, and finalizing the public repository link with full instructions.

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," *arXiv preprint arXiv:2301.08243*, 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," *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," *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. [Online]. Available: https://arxiv.org/abs/1911.05722

[4] 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 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representations*, 2021. [Online]. Available: https://arxiv.org/abs/2010.11929

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

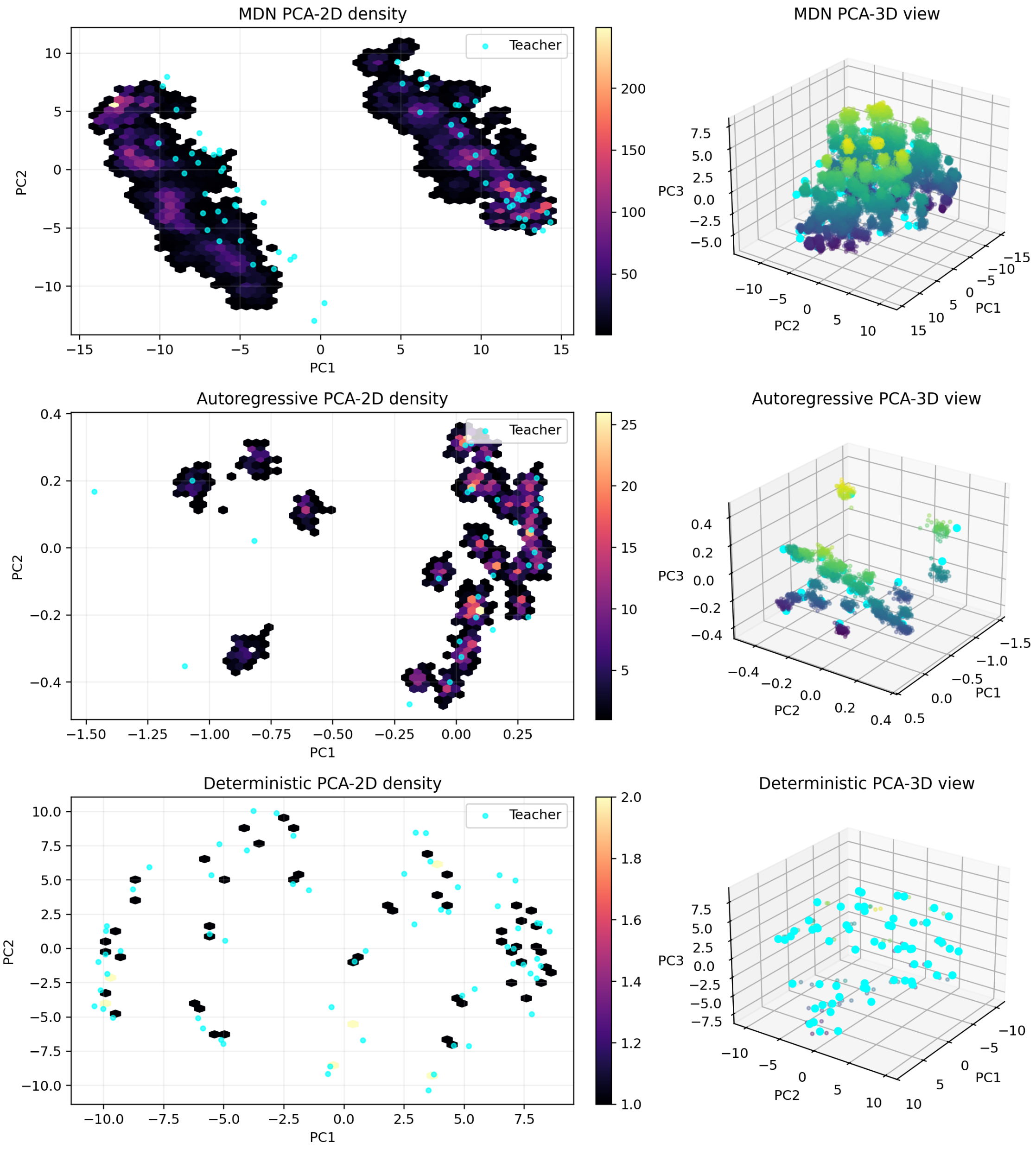

Fig. 7. Projected embedding distributions sampled directly from checkpoints. Left column: 2D PCA hexbins with teacher targets (cyan); right column: 3D PCA scatter plots. Rows correspond to the MDN head (top), autoregressive RNADE head (middle), and the deterministic I-JEPA baseline (bottom). Probabilistic heads form multi-modal, high-entropy clusters, whereas the deterministic predictor collapses to a single ridge.

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

[7] 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

[8] B. Uria, I. Murray, and H. Larochelle, "Rnade: The real-valued neural autoregressive density estimator," *Advances in Neural Information Processing Systems*, 2014. [Online]. Available: https://arxiv.org/abs/1306.0186

[9] 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

[10] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1
$times1$ convolutions," in *Advances in Neural Information Processing Systems*, 2018. [Online]. Available: https://arxiv.org/abs/1807.03039

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

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

[13] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *International Conference on Learning Representations*, 2021.

[Online]. Available: https://arxiv.org/abs/2010.02502

[14] Y. Le and X. Yang, “Tiny imagenet visual recognition challenge,” Stanford CS231n (accessed 2025), 2015. [Online]. Available: https://www.kaggle.com/c/tiny-imagenet

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