Joint Embedding Predictive Architecture with Multimodal Representations

Vladislav Kalinichenko
Innopolis University
Innopolis, Russia
v.kalinichenko@innopolis.university

Polina Korobeinikova
Innopolis University
Innopolis, Russia
p.korobeinikova@innopolis.university

Timur Struchkov
Innopolis University
Innopolis, Russia
t.struchkov@innopolis.university

Alexandra Starikova-Nasibullina
Innopolis University
Innopolis, Russia
a.nasibullina@innopolis.university

I. Project Idea

The project aims to improve I-JEPA, a self-supervised learning model for images that predicts embeddings (abstract representations) of masked parts of an image based on visible context, rather than reconstructing pixels directly [1]. Currently, I-JEPA outputs a single deterministic embedding, which can overlook real-world ambiguity (e.g., a partial animal image could be a cat or fox). We propose modifying it to output a probabilistic distribution of embeddings (it is important to be multi-modal distribution per each latent coordinate), allowing the model to express uncertainty and multiple possible interpretations. This makes the model more robust for tasks requiring world-model representation, while keeping the core benefits of efficient, semantics-focused learning.

II. Method

We will build on the I-JEPA architecture, which includes a context encoder (Vision Transformer, ViT), a target encoder (updated via exponential moving average), and a predictor. The key modification is in the predictor: instead of outputting a single embedding vector, it will predict a probabilistic distribution over embeddings to capture uncertainty. We explore four advanced methods for modeling this distribution, each implemented by a team member, to compare their effectiveness in handling multimodality and preventing collapse.

A. Probabilistic Methods

- **Mixture Density Networks [2] (Timur):** Configuring the predictor as a multilayer perceptron (MLP) to output Gaussian mixture parameters: for K components, weights π_k via softmax, means μ_k, and variances σ_k with diagonal covariance.
- **Autoregression [5] (Polina):** Constructing the predictor to build a conditional distribution sequentially across embedding coordinates: for the i-th coordinate, predicting mixture parameters (π_i, μ_i, σ_i or logistics for heavy tails) dependent solely on previous coordinates $h_1 \dots h_{i-1}$.
- **Normalizing Flows [3] (Vladislav):** Designing the predictor to output parameters of invertible transformations, exemplified by coupling layers with scale s and shift t derived from context. Initiating sampling with base noise $u \sim \mathcal{N}(0, I)$, passing through the flow to yield $h = f(u; \text{params\_from\_context})$.
- **Diffusion [4] (Alexandra):** Developing the predictor as a U-Net-like network ε_ϕ to predict noise ε on a noised embedding $h_t = h_{\text{target}} + \varepsilon$, where t denotes the timestep noise.

III. Dataset Explanation

Utilizing Tiny ImageNet-100, a subset of the Tiny ImageNet dataset, for experiments. Tiny ImageNet, a reduced version of ImageNet, contains 200 classes with approximately 100,000 training images (500 per class), 10,000 for validation (50 per class), and 10,000 for testing. Images are 64x64 pixels in RGB, offering a compact alternative to full ImageNet (typically 224x224). The 100-class subset (Tiny ImageNet-100) includes about 50,000 training images (500 per class), 5,000 for validation, and 5,000 for testing, with classes like "dog", "car", and "airplane" inherited from ImageNet. This variant retains challenges like pose and lighting variability but reduces computational demands, making it ideal for prototyping models like I-JEPA.

IV. Timeline

In 8-week project with four members: Timur (MDN), Polina (Autoregression), Vladislav (Normalizing Flow), Alexandra (Diffusion). Each focuses on their method while collaborating on shared tasks.

- **Weeks 1-2: Preparation and Base Model Setup** General task: Set up environment, download dataset, implement baseline I-JEPA.
  - Timur: Study MDN literature, adapt for embeddings, prepare base predictor code.
  - Polina: Study MADE/RNADE, design masks for autoregressive density estimation by "slices" of PDFs.
  - Vladislav: Study normalizing flows (e.g., RealNVP), run vanilla I-JEPA.

- Alexandra: Study diffusion methods for density estimation, adapt for latent space as in I-JEPA.

- **Weeks 3-4: Method Implementation** General task: Integrate methods into I-JEPA predictor, implement loss functions.
  - Timur: Modify predictor for mixture parameters (π, μ, σ for K components), implement NLL-loss with regularization.
  - Polina: Implement masked network for autoregression.
  - Vladislav: Add flow layers (scale/shift from context), implement forward/inverse and log-det Jacobian, test density estimation.
  - Alexandra: Implement noise model ε_ϕ, add diffusion process by timestep.
- **Weeks 5-6: Training and Evaluation** General task: Train models on Tiny ImageNet-100, evaluate on downstream tasks (classification, kNN-based metrics).
  - Timur: Train MDN variant.
  - Polina: Train autoregressive variant.
  - Vladislav: Train flow model.
  - Alexandra: Train diffusion variant.

- **Weeks 7-8: Integration, Ablations, and Report** General task: Compare all 4 methods (e.g., table by accuracy, uncertainty, compute time), conduct ablations (e.g., number of components/layers), prepare final report. - Timur: Ablate on number of mixtures K, contribute to code/report MDN section.
  - Polina: Ablate on distribution types (Gaussian vs. logistics), integrate into comparison.
  - Vladislav: Ablate on number of flow layers, prepare density visualizations.
  - Alexandra: Ablate on timesteps in diffusion, finalize report with perspectives.

References

[1] 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.

[2] Christopher Bishop. Mixture density networks. Technical Report NCRG/94/004, January 1994. URL https://www.microsoft.com/en-us/research/publication/mixture-density-networks/.

[3] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp, 2017. URL https://arxiv.org/abs/1605.08803.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[5] Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: The real-valued neural autoregressive density-estimator, 2014. URL https://arxiv.org/abs/1306.0186.