Does JEPA really learn high level image features ? An investigation through generative modelling

Kartik Khandelwal (kartikkh), Vishwajeet Agrawal (vishwaja)

1 Introduction

Joint-Embedding Predictive Architectures (JEPA's) [2, 3] learn by predicting features in the latent space (Fig. 1) and have shown promising results on learning image and video representations with must less compute and data compared to other self-supervised learning methods like masked autoencoders [7] that are trained to predict masked pixels. For instance, on linear probing evaluation on ImageNet, I-JEPA [2] on VIT-H/14 backbone achieves an accuracy of approx 79% compared to 77% for MAE [7], while only pre-trained on ImageNet for 300 epochs compared to 1600 epochs for MAE.

Authors of I-JEPA claim that by learning to predict in the latent space, the model focuses on learning high level semantic features of the image that are easy to predict while ignoring fine pixel level details that are difficult to predict and are not semantically useful for downstream tasks. This enables not only faster learning but also learning of more useful features. While the claim sounds intuitive, it has not been systematically analyzed what kind of features are actually learnt, and if I-JEPA actually learn high level features while ignoring low level ones.

To investigate this, we analyze the quality of a generative model trained to model the distribution over the JEPA embeddings. The idea is that if JEPA embeddings indeed only captures high level features and ignore low level ones, it should be easier to learn a generative model since it now does not has to model all the pixel level details. The sampled images may not be as high in quality in terms of fine details but should be semantically better, i.e. capture the distribution of image semantics (e.g. class) and follow the prompt in the case of text conditioned generation. Comparing with MAE, we indeed find that while FID score (that captures the overall distribution) is roughly similar or slightly lower for I-JEPA , metrics that captures semantics like Inception score, distribution over generated classes, and semantic correctness of the generated image (in case of text / class conditioned), I-JEPA outperforms MAE by a clear margin. On the qualitative side, we find that samples for the model trained on I-JEPA embeddings often ignore background details, almost giving the image in a potrait mode while model trained on MAE show fine background details but do not capture as good semantically relevant features. We find that the model trained on I-JEPA embeddings learn these features much faster achieving a low FID score and high quality images compares to model trained on MAE emebddings that take very long to generate good quality images.

Owing to the capability of JEPA to model semantically relevant features, we further observe that this allows for a new class of foundation image generative models that produces images capturing the essential semantic details by only modelling the distribution in the latent space instead in the image space, thus enabling access to a much smaller generative model capturing rich semantics. This can be particularly useful in scenarios such as an artist / illustrator iterating over image ideas while not caring about fine photographic details. A downstream model that adds fine details such as super-resolution could then be used on top for the finalized image.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

2 Literature review

Joint-Embedding Predictive Architectures[2, 3] are trained in a self-supervised manner to optimize the reconstruction loss in the feature space. This is in contrast to other self-supervised learning methods for image reconstruction which directly try to predict the pixel values. At a high level, the model tries to predict representation of part of an image given representation of some non-overlapping parts of the image as input. It has 3 parts: a context encoder, a predictor and a target encoder. The image context $\mathbf{x}$ is passed through the Context encoder to generate embeddings $\mathbf{s}_x$. Similarly the target context is passed through the target encoder to generate $\mathbf{s}_y$. Then the predictor takes as input the context embeddings $\mathbf{s}_x$ and positional mask embedding z to output $\hat{\mathbf{s}}_y$, an estimate of the target embedding $\mathbf{s}_y$. Finally, an L_1 or L_2 loss between $\hat{\mathbf{s}}_y$ and $\mathbf{s}_y$ is minimized to train the model. In particular, to avoid collapsed representations, the loss is only propagated through the context encoder while for the target encoder, an exponential moving average of the previous iterations of context encoders is used to compute the target embeddings. This procedure of self supervised learning in the latent space was introduced in BOYL [6]. The key innovation in I-JEPA is that, while in BOYL, they minimized loss between augmented views of the image in the latent space, I-JEPA additionaly has a prediction model that predicts target embeddings, i.e. embeddings on nearby patches given other patches of the same image, and minimizes loss between predicted and target. Thus I-JEPA does not require hand-crafted augmentations.

JEPA is also thought of as an energy based model, where energy can be written as,

$$E(x, y) = \|P_\phi(\text{Enc}_\theta(x)) - \text{Enc}_\theta(y)\|_2 \tag{1}$$

where x and y are different patches of the same image.

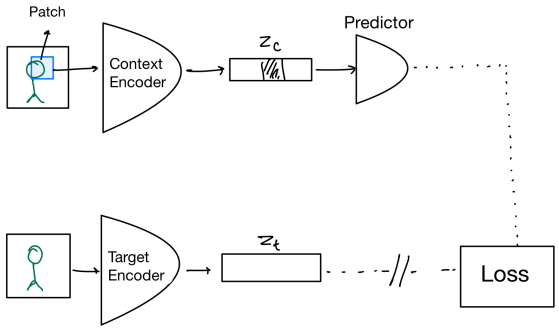


Figure 1: Training of an image encoder through JEPA. Figure adapted from [1].

Diffusion Models [9] are probabilistic models that operate by learning to denoise a noisy image to learn high dimensional distribution of the original images. They do it by modelling it as reverse process of a Markov Chain. Latent diffusion models [12] build upon this concept, but operate on the latent space of a pre-trained autoencoder rather than directly in pixel space. By leveraging the compressed latent representation, latent diffusion models can achieve high-resolution image synthesis with significantly reduced computational requirements compared to traditional pixel-based diffusion models.

Masked Autoencoders [7] model the image distribution by predicting masked patches from unmasked ones. Around 75% of the image is masked and the remaining patches are passed through a ViT encoder. Outputs of encoder (tokens for unmasked patches) and tokens for masked tokens are then passed through the decoder and a reconstruction loss is minimized. During inference, the decoder is then discarded and the encoder is used to extract the full image representations.

3 Method

Our goal is to learn the distribution over the image embeddings given by a pretrained encoder (I-JEPA or MAE model) for the images in the dataset. For this, we use Latent Diffusion models [12]. Our

method can be divided into 3 steps i) Compressing the encoder embeddings into low dimensional latents ii) Train a diffusion model to model these latents iii) Train a decoder model to reconstruct the image from the latents.

Compressing Encoder embeddings using VAE: The encoder Enc_θ I-JEPA or MAE have a VIT backbone [5] which splits an image $\mathcal{I} \in \mathbb{R}^{N\times N}$ into K^2 patches $p_{ij} \in \mathbb{R}^{N/K\times N/K}$. For each patch, it produces a $d-$dimensional embeddings, thus giving an encoding $\text{Enc}_\theta(\mathcal{I}) \in \mathbb{R}^{K\times K\times D}$ for the whole Image. This encoding can be very high dimension. Initially, we contemplated the option of averaging the embeddings for all the patches, as done by the authors in [2], to evaluate their performance in downstream discriminative tasks. However, we decided against this approach due to the potential loss of information required for generating high-fidelity images. Thus, as a first step, we train a Variational autoencoder [10] to compress the image encoding into lower dimensional latent vector z_L. Specifically, given an image encoding $\text{Enc}_\theta(\mathcal{I}) \in \mathbb{R}^{K\times K\times d}$, the variational AE encoder compresses it to latent embeddings $z_L \in \mathbb{R}^{K\times K\times d}$ with $d < D$.

Learning the Diffusion Model: Having access to low-dimensional latents from the VAE encoder, we train a latent diffusion model over the compressed latents. We follow the same recipe as [12] which uses a time-conditional UNet backbone [13] for the diffusion model. For conditional image generation like class/text conditioned generation, a domain specific encoder τ_ϕ is simultaneously trained to encode the conditional information y which is then passed to the intermediate layers of the Diffusion UNet via a cross-attention layer on the intermediate layer activations of the UNet and $\tau_\phi(y)$. The loss function for the class conditioned model is given as,

$$L_{\text{LDM}} := \mathbb{E}_{\mathcal{E}(x),\varepsilon\sim\mathcal{N}(0,1),t}\left[\|\varepsilon - \epsilon_\theta(\mathbf{z}_t, t, y)\|_2^2\right] \tag{2}$$

where $\mathcal{E}$ is the VAE encoder learned in step 1, that compresses the input embeddings, $\mathbf{z}_t$ is the noisy version for the input for time step t, and y is the class label for the input. The neural backbone ϵ_θ is realized through a time conditioned U-Net.

Learning Image Decoder for Visualization: Since the original VAE in section 3 is trained to decode the latent embeddings into the pretrained encoder Enc embeddings, we cannot directly visualize the quality of the distribution learnt or estimate image generation quality metrics. Thus, we learn an additional decoder model that projects latent embeddings into the image space. The decoder model is trained using a combination of perceptual loss and adversarial objective in a similar way as [12].

4 Experiments

We train diffusion models under two different settings i) unconditional generation ii) class conditional generation. We use the AFHQ [4] dataset for our experiments. It contains about $15,000$ high quality images of animal faces with three classes namely, dogs, cats and wild(contains images of tigers, lions, wolfs etc) with 5000 images per class.

4.1 Quality Metrics

To measure the quality of the images generated by a generative model, we use the following metrics.

Inception score: It was introduced in [14] is based on the classification of a pretrained InceptionV3 model, it is high when the classes predicted by the model are evenly distributed (indicating diversity) and the model outputs are low in entropy (indicating confident outputs, thus sharp, distinct images). Given a classification model m, the score is given by

$$\text{Score}(\theta; m) = \mathbb{E}_{x\sim p_\theta}[m(x) \cdot (\log m(x) - \log \mathbb{E}[m(x)])] \tag{3}$$

where expectation is taken with respect to trained generative model p_θ.

ResNet score: We use the above score calculated on a pretrained ResNet18 model finetuned on the same dataset (AFHQ). The trained model achieves an accuracy of 99.2% on the validation set thus indicating near perfect classifier, and thus better suited for the semantic score calculation (Eq. 3).

TV Distance: We also calculate a TV (total variation) distance between classes of the images generated by the diffusion model and images in the dataset as discriminated by our ResNet18 model. The lower the TV-distance the closer the distribution learnt by the model to the dataset distribution

(uniform over classes).

$$\text{TV}(p_\theta; m) = \sum_{j=1}^{C} 1/2 \cdot (\mathbb{E}_{x \sim p_\theta}[\mathbf{I}(m(x) = j)] - 1/C) \tag{4}$$

where C is the number of classes and real images are distributed uniformly over classes.

Precision & Recall: In the context of image generation, precision measures the fraction of generated images that are realistic and high-quality, recall measures the fraction of training data manifold that has been convered by the image generator. We use the method of [11], where they calculate it with respect to a manifold defined on the feature vectors as calculated by a pretrained image model of real and generated samples. In particular they use the following function to estimate if a vector ϕ lies in the manifold defined by a set of vectors $\mathbf{\Phi}$,

$$f(\phi, \mathbf{\Phi}) = \mathbf{I}(\exists \phi' \in \mathbf{\Phi} : \|\phi - \phi'\| < \|\phi' - \text{NN}_k(\phi', \mathbf{\Phi})\|) \tag{5}$$

where NN_k is the k''th nearest neighbor. Let $\mathbf{\Phi_g}$ denote the set of generated images, and $\mathbf{\Phi_r}$ the set or real images, the precision and recall is estimated as,

$$\text{precision}(\mathbf{\Phi_r}, \mathbf{\Phi_g}) = \frac{1}{|\mathbf{\Phi_g}|} \sum_{\phi_g \in \mathbf{\Phi_g}} f(\phi_g, \mathbf{\Phi_r}) \qquad \text{recall}(\mathbf{\Phi_r}, \mathbf{\Phi_g}) = \frac{1}{|\mathbf{\Phi_r}|} \sum_{\phi_r \in \mathbf{\Phi_r}} f(\phi_r, \mathbf{\Phi_g}) \tag{6}$$

In words, precision is measured by querying for each sampled image, whether the image lies in the estimated manifold of real images, and recall is measured by querying for each real image, whether the image lies in the manifold of sampled images.

FID Score: It was introduced in [8] and is based on comparing the distributions by fitting a multivariate gaussian distribution over the latent representations of real and sampled images. We use the activations of InceptionV3 model for the latent representation.

4.2 Training Details

For both I-JEPA and MAE, we used the VIT-H/14 architecture which has been pretrained on ImageNet for 300 and 1600 epochs respectively. While training the VAE model in Section 3, we used images of size $224 \times 224 (N = 224)$ with patches of size 14×14 leading to encoder embeddings of size $16 \times 16 \times 1280 (D = 1280)$. We use a latent embedding dimension of $16 \times 16 \times 16 (d = 16)$ for the VAE which we found to be a good trade-off between compression and embedding quality. We train the VAE with a very small KL loss weight of 0.05 to prevent biasing the latents too much towards the standard normal and retain maximum amount of original distribution information.

Architecture Details: We use a roughly 37M parameter diffusion model using a UNet backbone with 4 upsampling (and downsampling) channels. Also, we employ a lightweight 3.3M parameter decoder model to convert the latent embeddings into images. The Diffusion models were trained using an initial learning rate of 5×10^{-6} and a lambdaLinearScheduler with a 1000 warm-up steps. The decoder models were trained with a fixed learning rate of 4×10^{-6}. The full training configuration can be found on the GitHub repository.

4.3 Sampling Details

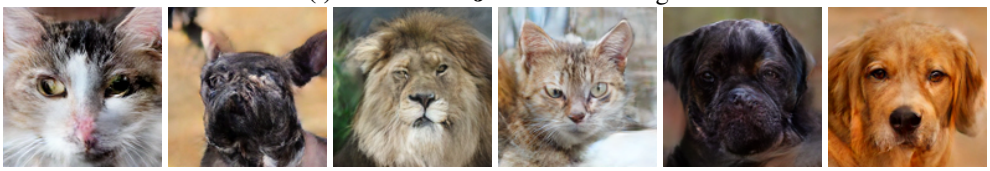
To estimate the quality metrics for the learned diffusion models, we generate $5,000$ samples of embeddings using DDIM sampler [15], and pass it through the decoder to generate images. We take 200 steps of the sampler to generate each sample.

5 Results

5.1 Unconditional Generation

We show results for the quality of images generated unconditionally by diffusion models as they are trained for different epochs. Figure 3 show plots of various quality metrics with training epochs. Table 1a, 1b show numbers for 50 and 150 epochs respectively.

(i) Trained on **I-JEPA** embeddings.



(ii) Trained on **MAE** embedings.

(a) Trained for **50** epochs.

(i) Trained on **I-JEPA** embeddings.

(ii) Trained on **MAE** embeddings.

(b) Trained for **150** epochs.

Figure 2: Unconditional sampling of diffusion models trained for different epochs.

model	Inception	ResNet	TV	Recall	Precision	FID
MAE	7.85	2.59	0.05	**0.30**	0.50	20.66
I-IJEPA	**9.09**	**2.87**	**0.03**	0.15	**0.89**	**13.57**

(a) Training for 50 epochs.

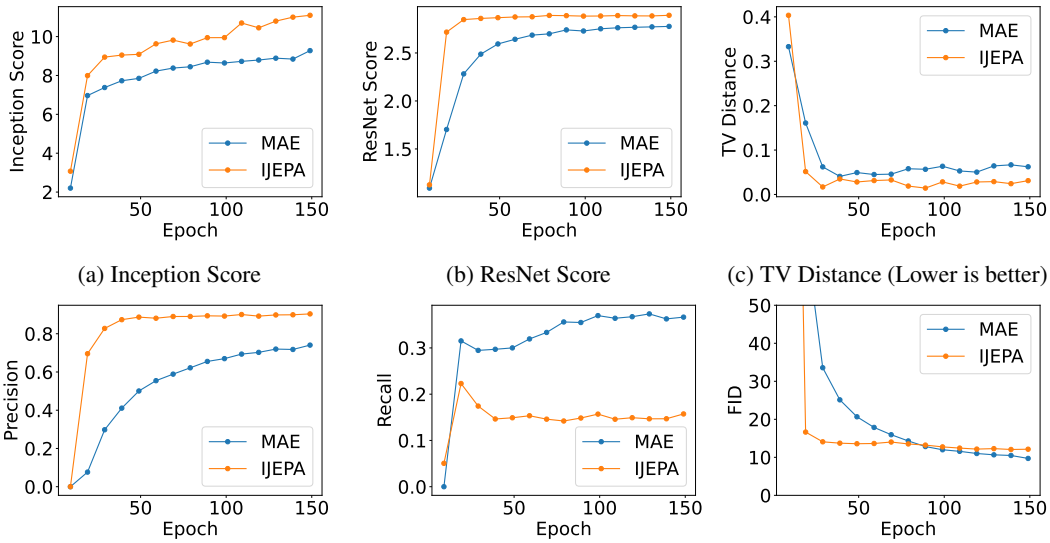
Model	Inception	ResNet	TV	Recall	Precision	FID
MAE	9.28	2.78	0.06	**0.37**	0.74	**9.71**
I-JEPA	**11.10**	**2.89**	**0.03**	0.16	**0.90**	12.13

(b) Training for 100 epochs.

Table 1: Metrics for samples generated by conditional diffusion model on AFHQ dataset.

Semantic Scores: On the Inception and Resnet score, and TV distance that measure semantic quality and diversity of the images, we see the model trained on I-JEPA doing much better than model trained on MAE embeddings.

Precision & Recall: Compared to MAE , we see very high precision for I-JEPA generative model, indicating that the images generated by I-JEPA are much closer to real images. On the other hand recall is much better for MAE , indicating that MAE model is able to capture the whole distribution present in the dataset whereas I-JEPA model does not learn the full distribution as expected since it purpotedly only captures the high level features.



(a) Inception Score (b) ResNet Score (c) TV Distance (Lower is better)

(d) Precision (e) Recall (f) FID score

Figure 3: Metrics with training evolution of diffusion models on unconditional generation on AFHQ dataset.

FID Score: The FID score for I-JEPA model quickly drops within a few tens of epochs while for MAE it takes much longer indicating that I-JEPA model learns the distribution much faster. Finally the FID score for MAE model is lower, this is expected since MAE captures all the details of the training distribution including background features.

Together these metrics indicate that model trained on MAE embeddings may capture the full distribution of the dataset when trained for long enough indicated by low FID and high recall, potentially also learning background features, but also generate images that are far from real images as indicated by low precision and semantic scores. On the other hand, model trained on I-JEPA embeddings do not capture the full distribution as expected but learn better features faster as indicated by high precision and low FID within 50 epochs as shown in table 1a, and better capture semantic quality and diversity as indicated by high semantic scores of Inception and ResNet classification models and lower value of TV distance between distribution over classes in generated images and distribution over classes in real images.

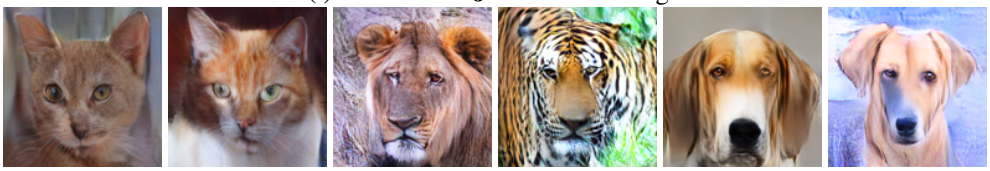
Efficiency: Tables 1a, 1b show metrics for models trained on 50 and 150 epochs respectively. We see that I-JEPA model learns much faster reaching a precision of 0.89 and and FID of 13.57 compared to precision of 0.5 and FID of 20.66 for MAE model within 50 epochs. This can also be seen in the plots 3.

Image Visualization: Figure 2 shows examples of sampled images for I-JEPA and MAE when trained on 50 and 150 epochs. Within 50 epochs, we can see that I-JEPA diffusion model learns to generate images with good high level features whereas MAE model struggles. In particular, we see that MAE model tries to learn low level pixel details first they are not very semantically useful, the high level features are not good (skewed eyes and face), whereas for I-JEPA it seems to blur out the background and low level pixel details while learning high quality high level features.

5.2 Class Conditional Generation

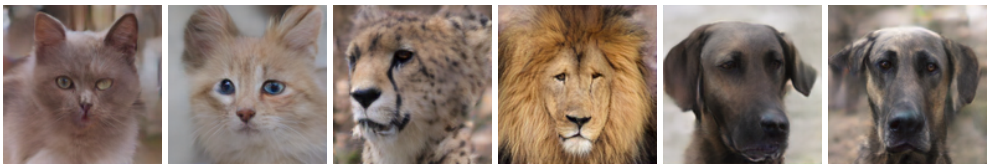
We trained a diffusion model to generate class conditioned images for the AFHQ dataset. Some class conditioned samples from the I-JEPA and MAE diffusion models are shown in 4. Evaluation metrics can be found in Tables 2a, 2b. We can see that the I-JEPA images are better than MAE images, especially for lower number of epochs where MAE seems to struggle a lot. Also, The images are of somewhat inferior quality compared to unconditioned generation. We suspect this might be because of insufficient training and training for more epochs should improve the quality of generated images.

(i) Trained on **I-JEPA** embeddings.

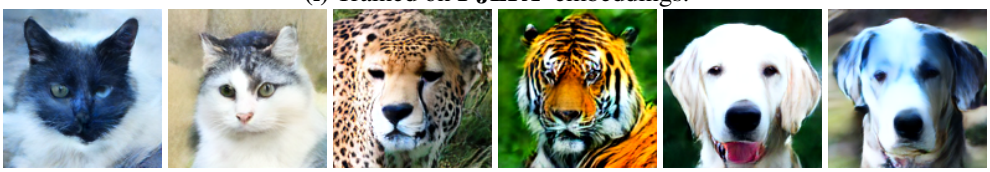


(ii) Trained on **MAE** embedings.

(a) Trained for **50** epochs.



(i) Trained on **I-JEPA** embeddings.



(ii) Trained on **MAE** embeddings.

(b) Trained for **100** epochs.

Figure 4: class-conditioned samples from I-JEPA and MAE diffusion models trained for different epochs. The first 2 are for class Cat, next 2 for class Wild and final 2 for class Dog

5.3 Image Decoder Training Evolution

Figure 10 shows the reconstructed images of the training data at different training time steps of the training of decoder. We make some interesting observations, the for I-JEPA embeddings, model starts with first learning the high level features and only much later it learns the low level features such as color information. For MAE embedings on the other, model seems to learn everything simultaneously, and not particularly favoring high level features.

6 Code

The code for running the experiments is given in https://github.com/Kartik14/latent-diffusion. For implementing diffusion and autoencoder models, we used the public github repo, https://github.com/CompVis/latent-diffusion. For evaluating metrics like Inception Score, FID, precision and recall, we used the public github repo https://github.com/openai/guided-diffusion.

7 Discussion

We find evidence to the claim that I-JEPA indeed learns high level semantic features of images while ignoring low level ones, thus compared to masked autoencoders, the I-JEPA embeddings capture overall less but semantically more useful information, enabling to learn a good generative model in a smaller number of epochs.

model	Inception	Precision	Recall	FID
MAE/CAT	2.54	0.15	0.51	67.08
MAE/DOG	2.26	0.00	0.00	178.53
MAE/WILD	3.31	0.00	0.00	218.13
I-JEPA /CAT	1.81	0.36	0.17	**60.93**
I-JEPA /DOG	2.74	0.24	0.01	**135.66**
I-JEPA /WILD	4.01	0.06	0.46	**148.70**

(a) Training for 50 epochs.

model	Inception	Precision	Recall	FID
MAE/CAT	2.10	0.27	0.47	50.26
MAE/DOG	2.84	0.01	0.01	183.21
MAE/WILD	3.87	0.00	0.00	202.97
I-JEPA /CAT	2.45	0.69	0.29	**40.68**
I-JEPA /DOG	2.25	0.42	0.01	**142.75**
I-JEPA /WILD	3.91	0.37	0.27	**46.64**

(b) Training for 100 epochs.

Table 2: Metrics for samples generated by diffusion model on class conditional generation on AFHQ dataset.

A further investigation could make a comparison by varying the model size (number of parameters in the diffusion model) and the number of training samples. We suspect that since the model learns distribution I-JEPA learns very quickly, it is simple enough to be captured by a smaller diffusion model with less number of training samples.

This highlights the utility of learning a foundation generative model that captures the distribution over image semantics.

References

[1] V-JEPA: The next step toward yann lecun's vision of advanced machine intelligence (AMI). https://ai.meta.com/blog/v-jepa-yann-lecun-ai-model-video-joint-embedding-predictive-architecture/. Accessed: 2010-09-30.

[2] 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 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023.

[3] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. V-jepa: Latent video prediction for visual representation learning. 2023.

[4] Y. Choi, Y. Uh, J. Yoo, and J.-W. Ha. Stargan v2: Diverse image synthesis for multiple domains, 2020.

[5] 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, 2021.

[6] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[7] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners, 2021.

[8] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[9] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020.

[10] D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2022.

[11] T. Kynkäänniemi, T. Karras, S. Laine, J. Lehtinen, and T. Aila. Improved precision and recall metric for assessing generative models, 2019.

[12] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models, 2022.

[13] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans, 2016.

[15] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models, 2022.

8 Appendix

8.1 Data Memorization by Diffusion Model

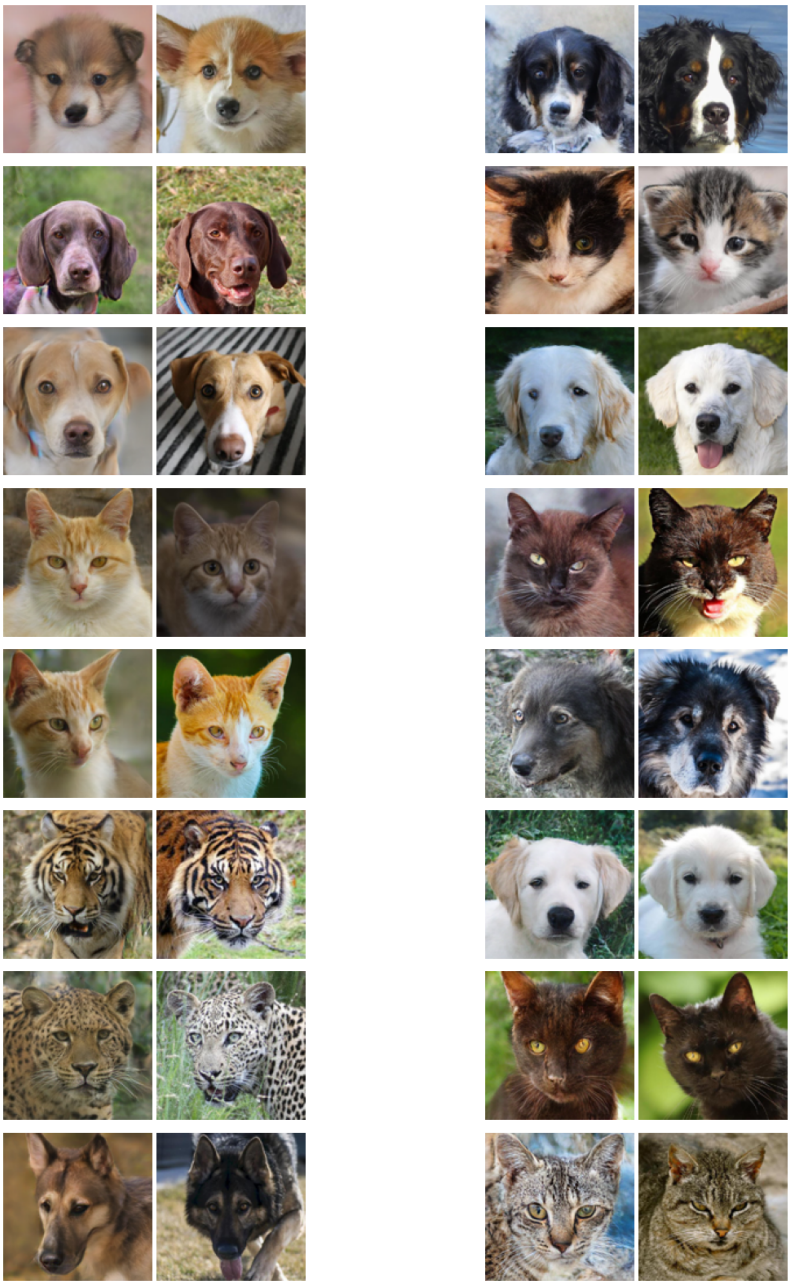


Figure 5: Closest image in the training set for I-JEPA (left) and MAE (right) diffusion models. Sampled images on the left and train set images on the right. Distance between VGG features from the second last layer are used to calculate the closet image.

8.2 Image Decoder Training Evolution

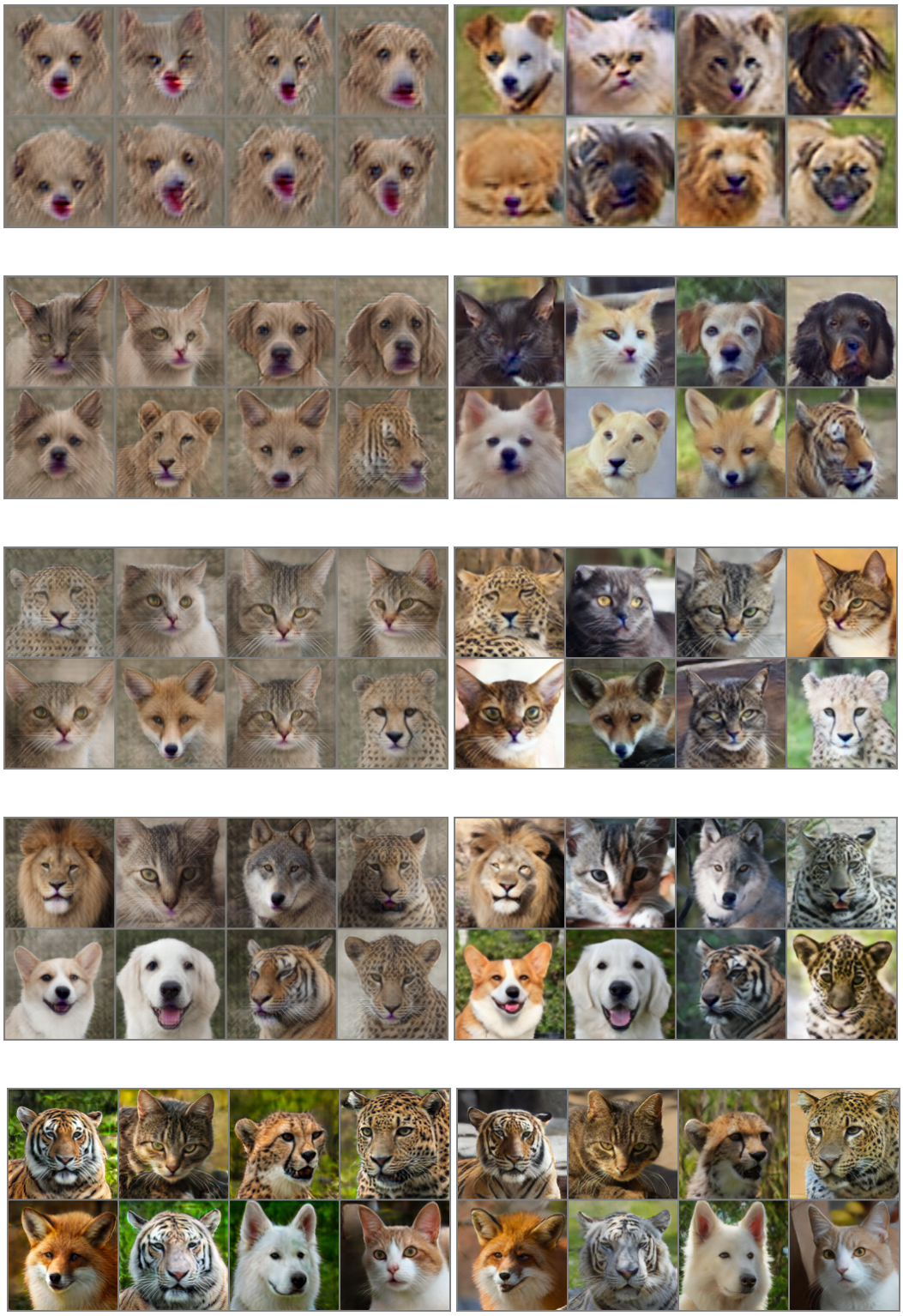


Figure 10: Reconstructed images while training the image deocder for I-JEPA (left) and MAE(right) for $T = 64, 256, 1000, 4000, 7000$ training steps. Notice how the I-JEPA decoder learns to reconstructed the main object first and then filling in the finer details like background details, color higlighting that the I-JEPA encoder only captures the high level semantic details