Elucidating the Role of Feature Normalization in IJEPA

Adam Colton
Harmony AI
adam@harmony.com.ai
https://harmony.com.ai/

Abstract

In the standard image joint embedding predictive architecture (IJEPA), features at the output of the teacher encoder are layer normalized (LN) before serving as a distillation target for the student encoder and predictor. We propose that this feature normalization disrupts the natural energy hierarchy of visual tokens, where high-energy tokens (those with larger L2 norms) encode semantically important image regions. LN forces all features to have identical L2 norms, effectively equalizing their energies and preventing the model from prioritizing semantically rich regions. We find that IJEPA models trained with feature LN exhibit loss maps with significant checkerboard-like artifacts. We propose that feature LN be replaced with a DynTanh activation as the latter better preserves token energies and allows high-energy tokens to greater contribute to the prediction loss. We show that IJEPA trained with feature DynTanh exhibits a longer-tailed loss distribution and fixes the checkerboard artifacts in the loss map. Our empirical results show that our simple modification improves ImageNet linear probe accuracy from 38% to 42.7% for ViT-Small and reduces RMSE by 0.08 on NYU Depth V2 monocular depth estimation. These results suggest that preserving natural token energies is crucial for effective self-supervised visual representation learning.

1 Introduction

IJEPA has emerged as an successful self-supervised framework for learning discriminative features without relying on labeled data [2]. IJEPA models learn to encode image patches into features that are useful for predicting masked regions, a task very similar to masked image modeling (MIM) techniques such as masked autoencoders (MAEs) [8]. However, instead of predicting ground truth pixel values as done in MAE, the model predicts layer normalized features that are obtained from a momentum teacher encoder. This alternate strategy is motivated by the observation that learning to predict latent representations is more effective for encoding discriminative features than learning to predict convincing pixel values [16, 7, 4].

We aim to tackle a critical yet underexplored aspect of IJEPA: how the encoder's features are processed before serving as contexts and targets for the prediction model. In the standard IJEPA implementation, hidden states at the output of the student encoder are processed by a LN before being passed to the prediction model. The prediction model is then tasked with predicting masked tokens that are standardized along the hidden dimension. The encoder's LN can be seen as serving two roles. Firstly, it stabilizes the training loss by squashing the magnitude of outliers. Secondly, as we show in our experiments, it reduces the variance of the loss over many samples.

While LN shows strong empirical results, we argue that it introduces a fundamental limitation. LN dilutes the energy hierarchy by equalizing the statistical properties of every token. This flat energy space conflicts with our goal of obtaining an encoder that can understand which image regions contain more discriminative information. We hypothesize that feature LN reduces the surprise of semantically meaningful tokens and that replacing feature LN with a magnitude preserving DynTanh will improve feature learning.

We propose a modified IJEPA architecture where LN is replaced by a simple DynTanh activation function. DynTanh provides similar stabilizing effects of LN while preserving vital information about relative token magnitude [17]. We validate our modification by comparing IJEPA models trained with LN at the output of the encoder, to IJEPA models trained with DynTanh at the output of the encoder. Across different diverse setups we observe that this simple modification improves downstream classification and dense prediction tasks.

2 Related Work

Various approaches propose different methods for processing teacher-generated features before using them as training targets. These methods can be broadly categorized based on how they transform raw features into a self-distillation objective.

Works such as iBOT [16], DINOv2 [10], and DoRA [15] employ an MLP projection followed by a temperature-scaled softmax to convert teacher features into soft probabilities. This essentially treats each image token as a classification problem, where the ground truth labels are obtained from a teacher model. A similar approach is used by MSN [1] where discrete probabilities are generated by comparing target features to a set of trainable prototypes.

Another family of approaches treat feature tokens as continuous vectors in a regularized geometric space. BYOL projects features tokens onto the unit sphere, using mean squared error between unit feature vectors [7]. This approach preserves directional information while discarding magnitude, implicitly weighting all tokens equally regardless of their magnitude. IJEPA uses a similar token-wise normalization, employing a learnable LayerNorm layer to normalize each token at the output of the encoder. We hypothesize that the token-wise normalization common to these previous works introduces an artificial uniformity that may hinder optimal representation learning.

Most similar to our work, BYOL ablates the usage of L2 normalization at the output of the projector. They show that removing L2 normalization decreases Imagenet linear probe accuracy from 72.5% to 67.4%. Replacing L2 normalization with batch normalization decreases the accuracy further to 65.3% [1]. Similarly, iBOT's distribution centering is critical for learning high quality representations, without which the linear probe accuracy drops from 74.2% to 63.5%. One notable characteristic is apparent across different works; all successful postprocessing methods use token-wise operations to project each token to a regularized space.

3 Methods

We describe our modified IJEPA training regime. We make several key modifications that allow us to better utilize our limited compute resources. We use the same core principles as IJEPA, patchifying images and assigning a proportion of patches as either context or targets. Differing from vanilla IJEPA, we leverage the technique introduced by Patch N' Pack [5] to vary both mask rates and image sizes within a batch, significantly improving training efficiency. Additionally, we do away with the block masking scheme employed by IJEPA, instead using a simple window masking approach. Our modified regime allows us to quickly test architectural modifications and better understand the important role of feature normalization.

3.1 Image Processing Pipeline

Our image processing pipeline consists of the following steps:

1. **Dynamic Resolution Scaling**: An input image $\mathbf{y} \in \mathbb{R}^{\mathbf{H}_{\text{orig}} \times \mathbf{W}_{\text{orig}} \times \mathbf{C}}$ is downscaled using a random scaling factor $s \sim \mathcal{U}(0.1, 1)$, resulting in an image of size $H \times W \times C$ where

[1] The Relevant experiment can be found in Section F.6, Table 20 of [7]

$H = s \cdot H_{\text{orig}}$ and $W = s \cdot W_{\text{orig}}$. We enforce the resized height and width to be no smaller than a minimum height and width, and the resized height and width to be divisible by the model's patch size.

2. **Patch Extraction**: The scaled image is divided into non-overlapping square patches, yielding a set of tokens $y \in \mathbb{R}^{H_p \times W_p \times (P^2 \cdot C)}$, where P is the patch size, and $H_p = H/P$ and $W_p = W/P$ represent the number of patches along height and width dimensions.
3. **Context-Target Partitioning**: We sample a random context capacity $c \sim \mathcal{U}(0.25, 0.5)$ and assign this proportion of image patches to set A, $x_A \in \mathbb{R}^{n \times d_{\text{in}}}$, with the remaining patches assigned to set B, $x_B \in \mathbb{R}^{m \times d_{\text{in}}}$. Here, $d_{\text{in}} = P^2 \cdot C$, $n = \lfloor H_p \cdot W_p \cdot c \rfloor$, and $m = H_p \cdot W_p - n$.
   Rather than partitioning image patches at the unit of a single patch, we treat image patches as square windows. This changes the unit of masking from patches into windows. Example outputs of this partitioning can be found in Figure 5.
4. **Dual Sequence Packing**: We implement an online dual-stream greedy bin packing algorithm to efficiently batch variable-length sequences. Given a batch size B, we allocate two buffers: one of shape $B \times N$ for packing context sequences and another of shape $B \times M$ for packing target sequences, where N and M represent the maximum sequence lengths for context and target, respectively. For each sample, we place its context-target pair in the first available buffer position that can accommodate both sequences simultaneously, ensuring they occupy the same batch index. We track each token's provenance using sample and position identifiers, which are used to compute an attention mask that prevents tokens from different samples from attending to each other. This approach, similar to the packing strategy in [5], maximizing computational efficiency by maintaining high occupancy rates in both streams.

3.2 IJEPA Pretraining

The core design of our approach maintains the key principles of the original IJEPA framework [2]. A teacher encodes both the context and target regions into features. A predictor uses the student's features to try and predict the teacher's features given only the positions of these target features.

3.2.1 Formal definition

After the patch partitioning described in the previous section, the student encoder $f(\cdot)$ processes patches from the context set A, producing raw features:

$$\tilde{S}_x = f(x_A) \in \mathbb{R}^{N \times d} \tag{1}$$

where d is the hidden dimension of the model. Concurrently, the teacher encoder $f'(\cdot)$ processes the complete set of patches (both context and target), yielding raw features:

$$\tilde{S}_{y_{AB}} = f'(x_A \oplus x_B) \in \mathbb{R}^{(N+M) \times d} \tag{2}$$

where $\oplus$ denotes concatenation.

We then take only the tokens originating from the target patches, to obtain our raw target features:

$$\tilde{S}_y \in \mathbb{R}^{M \times d} \tag{3}$$

Both representation undergo layer normalization with learnable affine parameters:

$$S_x = \text{LN}_{student}(\tilde{S}_x), S_y = \text{LN}_{teacher}(\tilde{S}_y) \tag{4}$$

We apply two additional operations:

1. Batch repetition by factor r to increase effective batch size of the predictor. In our experiments we set $r = 4$.
2. Random token dropout both sequences, using a fixed rate p_{drop} from both S_x and S_y. In our experiments we set $p_{\text{drop}} = 0.75$

The predictor network $g(\cdot)$ then transforms the student features to predict the teacher's features at target locations:

$$\hat{S}_y = g(S_x, \mathcal{P}) \in \mathbb{R}^{M \times d} \tag{5}$$

where $\mathcal{P}$ represents the positions of target patches.

The training objective minimizes the smooth L1 distance between predicted and actual teacher features.

$$\mathcal{L} = \text{SmoothL1Loss}(\hat{S}_y, \text{SG}(S_y)) \tag{6}$$

where $\text{SG}(\cdot)$ denotes the stop-gradient operation that prevents gradient flow through the teacher.

After updating the student and the predictor using the gradient of the loss, we update the teacher using an exponential moving average of the student's parameters.

3.2.2 Training Efficiency

Training on variable resolutions offers two significant advantages. First, it improves convergence speed and helps models generalize to resolutions unseen during training [5, 3]. Second, it substantially increases sample throughput.

Our parallel data loading pipeline, inspired by Patch N' Pack, allows workers to resize, pack, and patch images concurrently. With a nominal batch size of 256, our approach effectively processes approximately $1,200$ images per batch and $2,000$ images per second, increasing image throughput by a factor of $4\times$[2]. Unlike the original IJEPA, where the sequence length of the batch supplied to the student and teacher varies between steps, our packed batch has a static sequence length. Our method can make the most of `torch.compile` because we do not need to compile the computational graph with dynamic sizes. Our method allows a machine equipped with a single RTX3090 GPU to train a 236m parameter ViT-small and 52m parameter predictor at a training throughput of 2,000 images per second. Compare this to the throughput of the original IJEPA on the same machine and model, which can train at only 500 images per second.

This efficiency gain makes state-of-the-art self-supervised visual representation learning accessible for researchers with limited computational resources. Additionally, sequence packing allows us to have mask rates that differ across different samples in a mini-batch, whereas in the original IJEPA the mask rate is fixed for all samples in a minibatch.

4 Experiments

We train two small IJEPA models, one using layer normalization at the output of the encoder, and one with DynTanh at the output of the encoder, denoted as IJEPA-LN, and IJEPA-Tanh respectively. IJEPA-Tanh simply replaces the LN in Equation 4 with a DynTanh nonlinear activation. We use DynTanh without affine parameters, squashing features into the range [-1, 1] [17].

$$S_x = \tanh(a_{student}\tilde{S}_x), S_y = \tanh(a_{teacher}\tilde{S}_y) \tag{7}$$

Where $a_{student}$ is a learnable vector, and $a_{teacher}$ is a slow moving EMA copy of the student's vector.

We train both models for 600 epochs on Imagenet1K. We compute the self supervised losses over 8,000 validation images and collect and analyze the distribution of all unreduced loss values.

5 Results

We observe in Figure 1 that IJEPA-Tanh exhibits a longer tailed loss distribution than IJEPA-LN. We believe that the regularization introduced by DynTanh introduces a synergy where high energy tokens are more surprising and thus exhibit higher loss scores. IJEPA-LN on the other hand, cannot greatly accentuate differences in semantic importance, because the activations of every token are normalized to the standard normal distribution.

[2]We compare image throughput by training an identical model using the original IJEPA code. We reduce the batch size to 224 to avoid OOM. The original IJEPA achieves (224 images per batch / 0.45 seconds per batch) = 500 images per second.

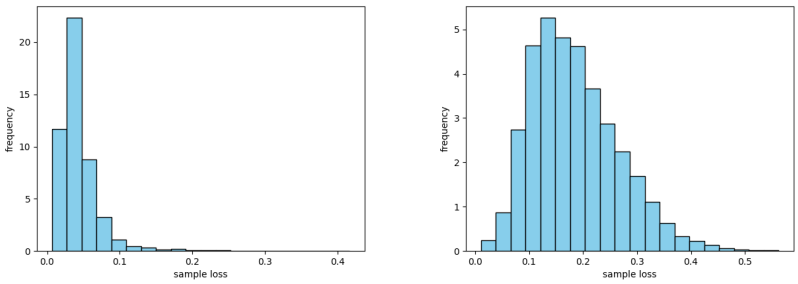

Figure 1: Left and right plots show the loss distributions of IJEPA-Tanh and IJEPA-LN respectively. Losses are measured over 8,000 validation images. IJEPA-Tanh exhibits a longer tail distribution, whereas IJEPA-LN more resembles a shorter tailed distribution.

To better compare IJEPA-Tanh and IJEPA-LN we visualize the spatial distribution of the self supervised losses. Figure 2 visualizes the self supervised losses for several images, averaged over 8,000 different random context-target masks. This demonstrates the regions in the image that have higher losses. As expected, for both IJEPA-Tanh and IJEPA-LN, the foreground has the highest losses.

We make an interesting observation - the loss maps of IJEPA-LN exhibit a checkerboard pattern. This artifact presents as a grid of high losses persisting without regards to the semantic content in an image. Perhaps even more intriguing is that this checkerboard pattern is better seen when looking at losses averaged across many samples. We compute IJEPA losses across 6,500 samples, computing a mean loss image for both IJEPA-Tanh and IJEPA-LN. We average the losses to produce a smoothed loss map. As seen in Figure 3, the checkerboard artifact in IJEPA-LN contrasts starkly with the smooth distribution of losses in IJEPA-Tanh.

We measure the validation performance of the two models on classification and monocular depth estimation. We observe that IJEPA-Tanh obtains consistent gains in validation performance. We encode features from the training set of Imagenet1k, and encode them using the teacher encoder. We mean pool the features across the sequence dimension to obtain a single vector embedding per image. We train a linear probe for 50 epochs, then evaluating the performance on the held-out validation set. We find that IJEPA-Tanh boosts the Imagenet1K classification accuracy of a linear probe from 38.021% to 42.71%.

We measure the quality of both model's features for the dense prediction task of monocular depth estimation. We train a DPT-head[11] on the test set of NYU-depth-V2 for 10 epochs, then testing the performance on the validation set. We find that IJEPA-Tanh decreases the root squared mean error on NYU-Depth-v2 from 0.6273 to 0.6163 compared to IJEPA-LN.

6 Limitations

We find that simply replacing the encoder's penultimate LN with DynTanh results in smoother loss maps and better downstream performance. However, IJEPA-Tanh still uses LN in the intermediate transformer layers. While we would have liked to test further modifications, our current test setup requires training the model fully for several hundred epochs only then revealing the differences in performance. We believe that future work remains to improve the intermediate insights into the features that the model is learning. Compared to MAE, which directly predicts pixel values, IJEPA models learn features that are difficult to comprehend. We hope for future IJEPA architectures that are natively trained to produce human-interpretable visualizations that encompass the thoughts and reflections that the model is making.

7 Conclusion

We demonstrate the importance of preserving the relative energies of features learned by IJEPA models. Our key finding is that the standard LN applied to encoder features disrupts the natural energy hierarchy of visual tokens, where high-energy tokens encode semantically important image regions.

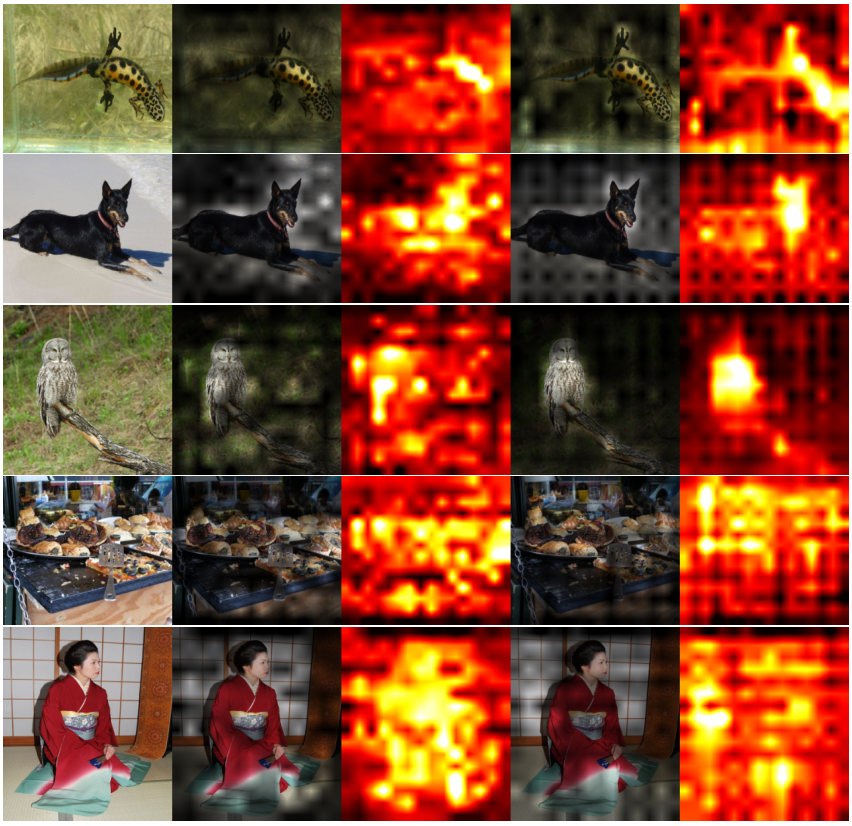

Figure 2: IJEPA losses computed from 8,000 random masks, each row showing a different image. For each row from left to right: Original image, multiplied loss map of IJEPA-Tanh, loss heatmap of IJEPA-Tanh, multiplied loss map of IJEPA-LN, loss heatmap of IJEPA-LN.

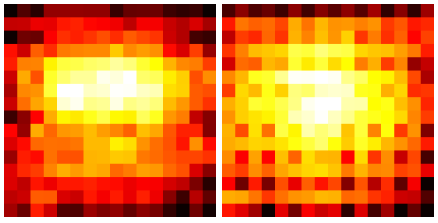

Figure 3: Loss heatmaps averaged over 6,500 samples. Left, IJEPA-Tanh, right: IJEPA-LN. The checkerboard artifact can be seen in IJEPA-LN's loss heatmap.

By replacing layer normalization with DynTanh activation, we preserve these energy relationships while maintaining training stability.

Our empirical results show that this simple modification yields consistent improvements across diverse tasks: ImageNet linear probe accuracy increases from 38% to 42.7% for ViT-Small, and monocular depth estimation RMSE decreases by 0.08 on NYU Depth V2. Notably, IJEPA-Tanh exhibits a longer-tailed loss distribution and eliminates the checkerboard artifacts present in loss maps of IJEPA-LN models, suggesting that the model learns to focus on semantically meaningful regions rather than being constrained by artificially imposed uniformity.

These findings suggest that feature normalization strategies in self-supervised learning deserve more careful consideration. While normalization techniques like LN provide training stability, they may inadvertently constrain the model's ability to learn hierarchical representations. Our work opens avenues for future research into energy preserving architectures that preserve semantic structure while maintaining optimization stability. The consistent improvements across classification and dense prediction indicate that preserving natural token energies is a fundamental principle for effective self-supervised visual representation learning.

References

[1] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Michael Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning, 2022.

[2] 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.

[3] Daniel Bolya, Po-Yao Huang, Peize Sun, Jang Hyun Cho, Andrea Madotto, Chen Wei, Tengyu Ma, Jiale Zhi, Jathushan Rajasegaran, Hanoona Rasheed, Junke Wang, Marco Monteiro, Hu Xu, Shiyu Dong, Nikhila Ravi, Daniel Li, Piotr Dollár, and Christoph Feichtenhofer. Perception encoder: The best visual embeddings are not at the output of the network, 2025.

[4] Xinlei Chen, Zhuang Liu, Saining Xie, and Kaiming He. Deconstructing denoising diffusion models for self-supervised learning, 2024.

[5] Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim Alabdulmohsin, Avital Oliver, Piotr Padlewski, Alexey Gritsenko, Mario Lučić, and Neil Houlsby. Patch n' pack: Navit, a vision transformer for any aspect ratio and resolution, 2023.

[6] Quentin Garrido, Randall Balestriero, Laurent Najman, and Yann Lecun. Rankme: Assessing the downstream performance of pretrained self-supervised representations by their rank, 2023.

[7] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning, 2020.

[8] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[9] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[10] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2024.

[11] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction, 2021.

[12] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer, 2020.

[13] Minglei Shi, Ziyang Yuan, Haotian Yang, Xintao Wang, Mingwu Zheng, Xin Tao, Wenliang Zhao, Wenzhao Zheng, Jie Zhou, Jiwen Lu, Pengfei Wan, Di Zhang, and Kun Gai. Diffmoe: Dynamic token selection for scalable diffusion transformers, 2025.

[14] Vimal Thilak, Chen Huang, Omid Saremi, Laurent Dinh, Hanlin Goh, Preetum Nakkiran, Joshua M. Susskind, and Etai Littwin. Lidar: Sensing linear probing performance in joint embedding ssl architectures, 2023.

[15] Shashanka Venkataramanan, Mamshad Nayeem Rizve, João Carreira, Yuki M. Asano, and Yannis Avrithis. Is imagenet worth 1 video? learning strong image encoders from 1 long unlabelled video, 2024.

[16] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer, 2022.

[17] Jiachen Zhu, Xinlei Chen, Kaiming He, Yann LeCun, and Zhuang Liu. Transformers without normalization, 2025.

Figure 4: Random samples with mask window size set to two by two patches

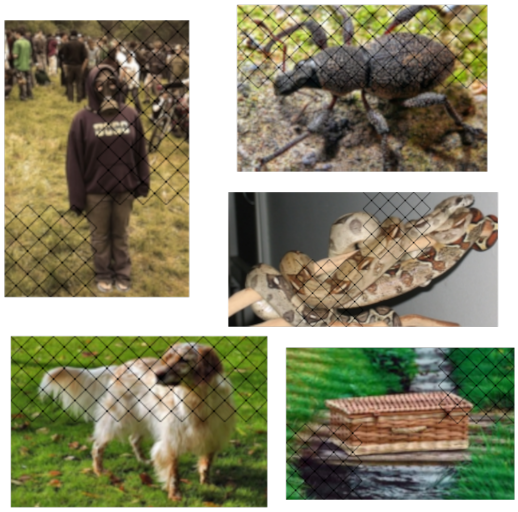

Figure 5: Random samples with mask window size set to four by four patches

8 Supplementary Material

8.1 Training and Architecture

We use a modified ViT architecture for the encoder and the predictor. Instead of normalizing the input RGB values along the channel dimension, we simply scale them to be in the range [-1,1]. We use a patch size of 16 and use RoPe2D for position information. We use DiffMOE as a drop in replacement for the MLP in the transformer block [13]. Both reported models (IJEPA-Tanh and IJEPA-LN) use 16 experts. We use QK normalization in all attention layers. The architecture and training hyperparameters of IJEPA-Tanh and IJEPA-LN are identical, except that we replace the LN at the end of the encoder with a DynTanH layer.

We train both ViT-small models on a single machine at a batch size of 256. We start with a learning rate of 1e-4, increasing it linearly for 10,000 warmup steps until it reaches 5e-4, then keeping it constant for the remainder of training. We use the AdamW optimizer with betas of (0.9,0.95) and weight decay of 0.05. We warmup the EMA β value for the teacher. We start it at 0.95, linearly increasing for 1,000 steps until it reaches 0.999, then linearly increasing it for 300,000 steps until it reaches 0.9995, where it remains constant for the remainder of training.

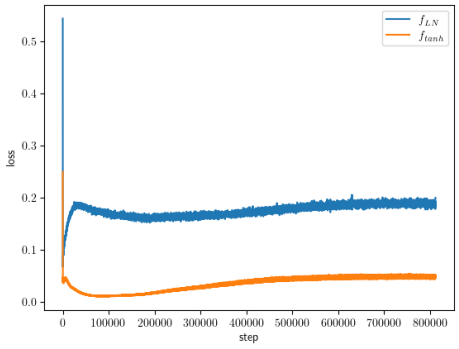

Figure 6: Loss during pretraining.

We train on a preprocessed training set of Imagenet1k [3]. We resize images that have a side length larger than 256 so that their new maximum side length is 256. Images with a maximum side length smaller than 256 are not resized. This preserves the aspect ratio of the original image while significantly increasing the dataloading speed.

We record the RankMe score [6] and LiDAR score [14] during pretraining. The different curves can be seen in Figure 9. Both RankMe and LiDAR proved to be poor oracles, with IJEPA-Tanh exhibiting smaller RankMe and LiDAR scores than IJEPA-LN. LiDAR and RankMe scores plateaued after 150,000 training steps despite both models showing continued improvements in linear probe accuracy past this point.

We observed that the model's training dynamics are sensitive to the shard shuffling strategy. Our first implementation shuffled shards without resampling. In this setup, each image is seen exactly once at a random point in a training epoch. When we resumed training from a checkpoint saved in the middle of an epoch we observed a loss spike and a drop in the RankMe score. This is probably due to the model overfitting to the order of images observed during an epoch. We employed a simple fix: simply shuffling shards with replacement fixed the loss spike when resuming training. All reported models were trained with shuffling with replacement.

8.2 Linear Classification

We evaluate the discriminative performance of the encoder by training a linear probe to predict the labels of Imagenet. We do not use any data augmentation, simply square cropping all images to the minimum of their original (H,W), and then resizing to the a constant resolution of (256,256). We extract the hidden states from each layer of the encoder. Notably, we do not apply the final LN or DynTanh to the features from the last layer, we found that using the encoder's final LN or DynTanh did not yeild better linear classification performance. We mean pool the hidden states along the sequence dimension. We train the probe on the imagenet training split with a constant learning rate of 1e-3 and a batch size of 2048 for 50 epochs, using the Adam optimizer. We then evaluate the accuracy on the validation set and report the maximum accuracy across all feature depths. Results can be seen in Figure 10. We observed that the performance degraded significantly at the last layer of the encoder, peaking somewhere between the fourth to last and second to last layer.

8.3 Monocular Depth Estimation

While our models are good at learning highly discriminative features, we also evaluate them on the more granular task of depth prediction. We train a DPT-head [11] over frozen encoder features from the third to last layer. The DPT-head produces a scale and shift invariant depth map, which is compared to a ground truth depth map that has been scaled and shifted. We train the DPT-head for 10 epochs on the training split of NYU Depth V2 [9] using the losses defined in [12]. For validation, we scale and and shift the model's predicted depth map using the scale and shift computed from the

[3] https://huggingface.co/datasets/adams-story/imagenet1k-256-wds

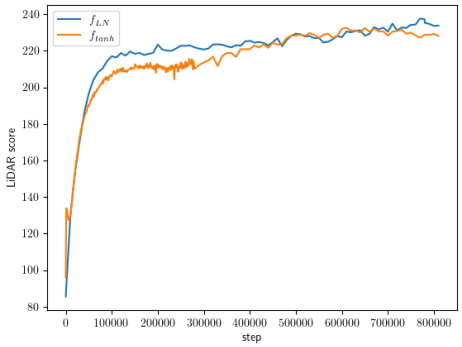

Figure 7: LiDAR score [14] measured during pretraining.

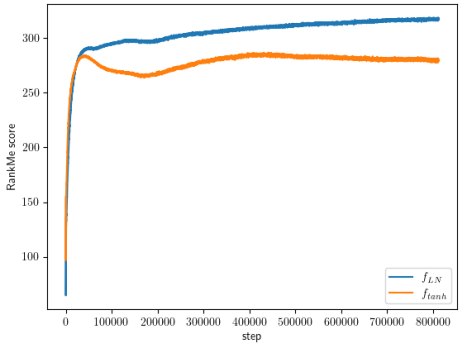

Figure 8: RankMe score [14] measured during pretraining.

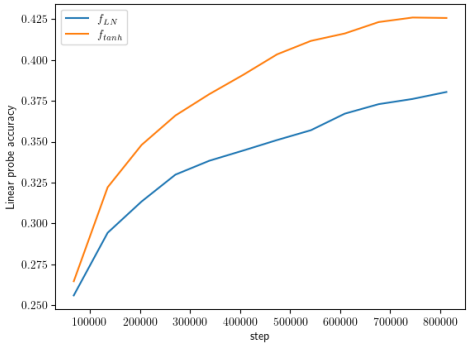

Figure 9: Linear probe accuracy measured during pretraining.

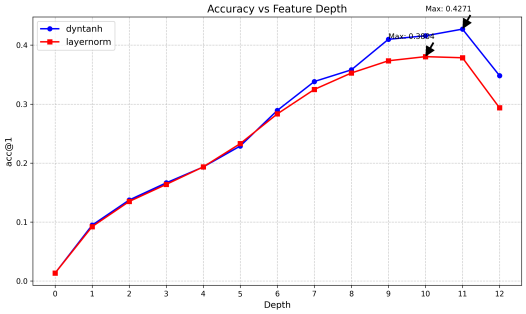

Figure 10: Linear probe accuracy on Imagenet1k across mean-pooled features extracted from different layers of the teacher encoder.

Figure 11: Depth estimation outputs for several validation samples from NYU-Depth-V2. From the left to right: original image, ground truth depth map, IJEPA-Tanh depth map, IJEPA-LN depth map

ground truth depth map. We report the RMSE between the model's scaled and shifted depth map and the raw ground truth depth map averaged over all samples in the validation split of NYU Depth V2.

We visualize our model's predicted depth map for samples from the validation split of NYU Depth V2. We scale depth maps to fit in the range [0,1], and use the 'hot' colormap. Results can be seen in Figure 11.

8.4 Additional Results

In addition to the two main models reported in the main body of this paper, we report several other training runs with different configurations in Table 1. Notably, across all of these configurations, the models trained with DynTanh at the end of the encoder perform better than models trained with LN

at the end of the encoder. Some models we train we evaluate without the use of dynamic experts, where the capacity predictor [13] is discarded.

We show the strength of our pretraining regimen by comparing a 100 epoch training run of our DynTanh model, to a 100 epoch training run using the code from IJEPA. We train identical models apart from replacing the final LN with a DynTanh. Our method drastically increases linear probe accuracy from 12.578% to 32.216%.

Method	**Feature Norm**	**Samples per Batch**	**EMA Schedule**	**LR**	**LR Cool-down**	**Grad Scaler**	**Dynamic Ex-perts**	**Experts**	**Mask Window (px)**	**Epoch**	**Wall Time (h)**	**ImageNet Acc@1**	**Depth RMSE**[9]
Ours	DynTanh	1200	Constant 0.996	5e-4	no	no	no	8	32	280	40	**39.956%**	0.61886
Ours	LayerNorm	1200	Constant 0.996	5e-4	no	no	no	8	32	280	40	34.61%	**0.60804**
Ours	DynTanh	1200	Increasing 0.996→0.9999	1e-3	40k	yes	no	8	64	350	50	**39.49%**	0.63489
Ours	LayerNorm	1200	Increasing 0.996→0.9999	1e-3	40k	yes	no	8	64	350	50	37.588%	**0.61457**
Ours	DynTanh	950	Increasing 0.999→0.9995	5e-4	no	yes	yes	16	32	600	116	**42.71%**	0.6163
Ours	LayerNorm	950	Increasing 0.999→0.9995	5e-4	no	yes	yes	16	32	600	116	38.021%	**0.6273**
Ours	DynTanh	470	Increasing 0.999→0.9995	5e-4	no	yes	yes	32	32	600	172	**45.08%**	-
Ours	DynTanh	950	Increasing 0.999→0.9995	5e-4	no	yes	yes	16	32	100	20	**32.216%**	-
IJEPA [2]	LayerNorm	224	Increasing 0.995→0.9999	5e-4	no	yes	yes	16	Na	100	73	12.578%	-

Table 1: Additional Experimental results comparing different configurations

8.5 Feature Visualization

We use a similar technique as [3] to visualize the encoder's features. We resize images to a random side length between 256 and 512 before encoding them. We smooth the features using a gaussian blur kernel. Then we take the top three PCA components. We then upscale feature patches using bilinear interpolation to increase the spatial size of the features to the input resolution. We treat these top three PCA components as hue, saturation, and value. We scale these values to fit in the range [0,1] before converting them to RGB pixels.

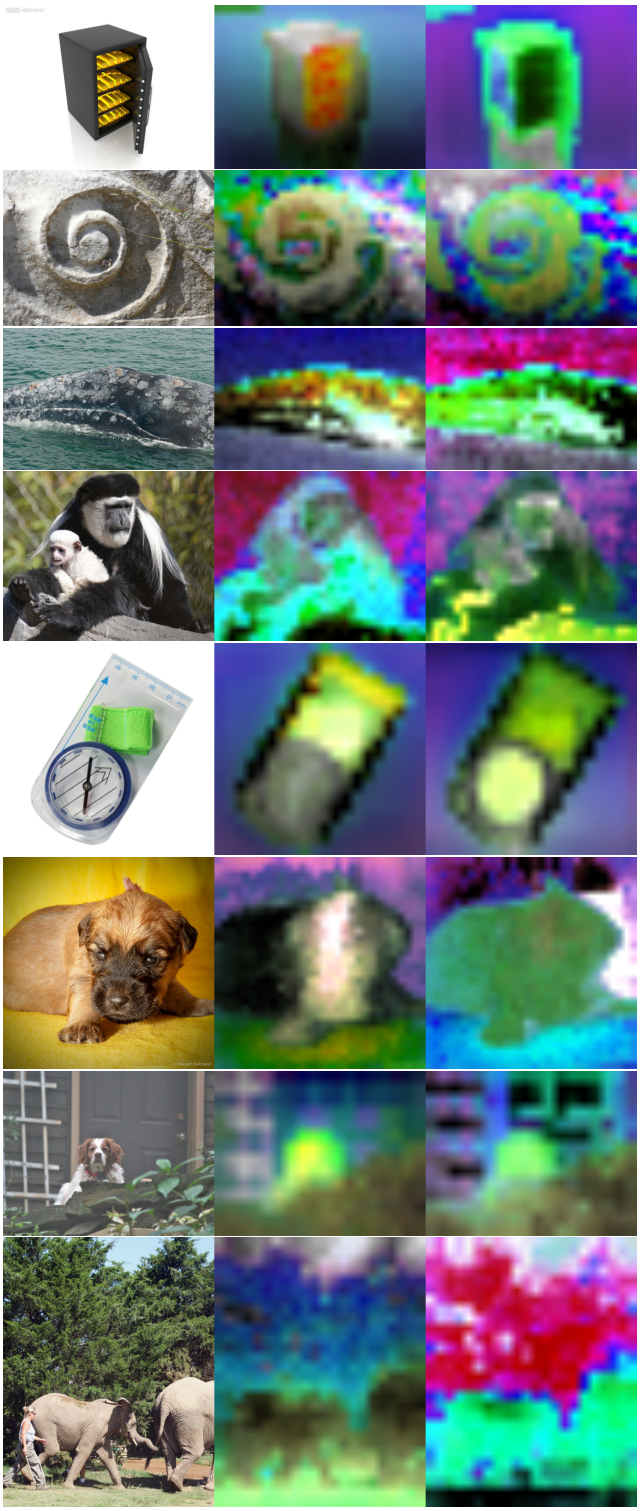

Figure 12: From left to right, original image, IJEPA-Tanh features, IJEPA-LN features