Make it SING: Analyzing Semantic Invariants in Classifiers

Supplementary Material

Contents

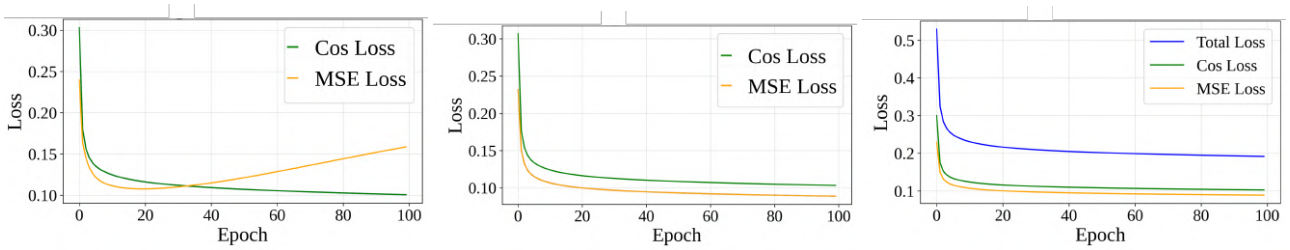
1. Setup and reproducibility

1.1. Translator training

As described in the paper, each translator trained on a specific classifier and its task is to map features from the penultimate layer $f \in \mathbb{R}^d$ to a CLIP image feature $e \in \mathbb{R}^{d_e}$. Nonlinear translators were trained directly in PyTorch [16], while linear translators were fitted by ridge regression using scikit-learn [17] and then ported to PyTorch for unified inference. The hyperparameters were chosen using sweeps logged in Weights & Biases [2]. We compared three training objectives:

1. Mean squared error (MSE) loss:
$$\mathcal{L}_{\text{MSE}}(f, e) = \|T_\theta(f) - e\|_2^2 . \tag{1}$$
2. Cosine similarity loss:
$$\mathcal{L}_{\cos}(f, e) = 1 - \frac{T_\theta(f) \cdot e}{\|T_\theta(f)\|_2 \, \|e\|_2}. \tag{2}$$
3. MSE + Cosine loss

For all three cases, we applied L_2 regularization. In practice, minimizing $\mathcal{L}_{\text{MSE}}$ alone proved sufficient to achieve high cosine similarity, whereas optimizing $\mathcal{L}_{\cos}$ alone does not reliably reduce MSE, suggesting an asymmetric relationship between the two objectives. This trend is illustrated in Figure 1.



(a) Cosine-only loss $\mathcal{L}_{\cos}$. (b) MSE-only loss $\mathcal{L}_{\text{MSE}}$. (c) Joint loss $\mathcal{L}_{\text{joint}}$.

Figure 1. Training losses for the different translator objectives. Minimizing the MSE loss also improves cosine similarity, whereas cosine-only training leaves the MSE substantially higher.

Our baseline translator is a linear map chosen for stability. To compare linear and non-linear translators, we evaluate three additional. As for Nonlinear architectures, we tried the following combinations:

1. A 3-layer MLP with blocks of the form LayerNorm–GELU–Dropout-FC [1, 8, 21]
2. A 4-layer MLP with the same block.
3. A residual MLP with one residual blocks and one projection layer.

All nonlinear translators were optimized with AdamW [13]. with learning rate 1×10^{-4} and weight decay $\lambda = 0.1$.

We report validation results over a 2,000-image subset in Table 1 and Figure 2 showing no significant advantage of any non-linear variant over the linear translator.

Table 1. Validation results for different translator architectures on a 2 000-image validation subset from 16 classes. None of the non-linear architectures shows a significant advantage over the linear translator.

Architecture	Mean cosine similarity	Validation MSE
3-layer MLP	0.9049	0.082246
Residual MLP	0.9045	0.082366
4-layer MLP	0.9023	0.084346
Linear	0.8946	0.091355

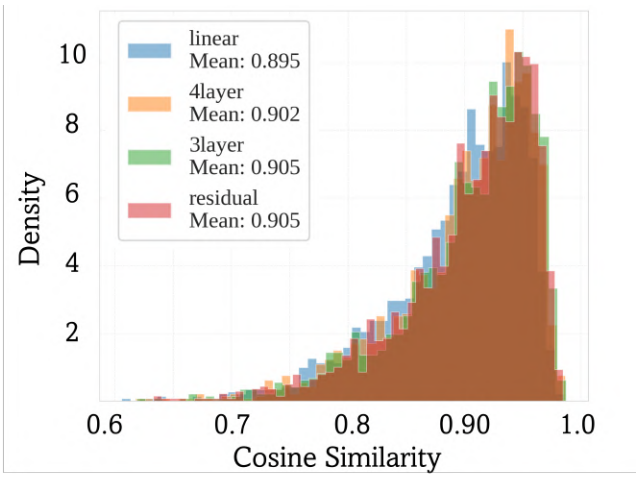


Figure 2. Cosine similarity distribution of different architectures between the translated and the original CLIP features, over 2k ImageNet features from 16 classes. All the histograms are leaned towards high correlation

2. Null space validation

Let f denote the penultimate classifier feature and $\ell(f) \in \mathbb{R}^C$ the corresponding vector of logits for C classes. We define the logit change induced by a perturbation δ as

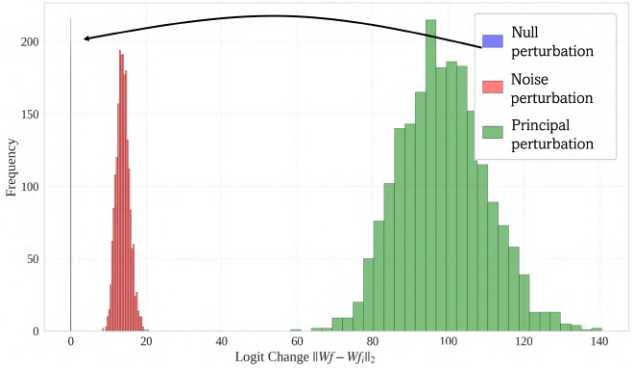
$$\Delta_\ell(f, \delta) = \|\ell(f + \delta) - \ell(f)\|_2 . \tag{3}$$

We compare three types of perturbations with matched ℓ_2-norm: (i) a null perturbation δ_{null} in the approximate null space of the classifier head, satisfying

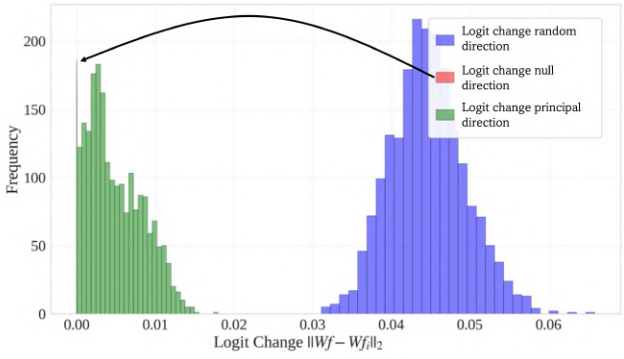
$$W \, \delta_{\text{null}} \approx 0, \tag{4}$$

where W are the head weights; (ii) a random perturbation δ_{rand} sampled from an isotropic Gaussian and rescaled to the same norm; and (iii) a principal perturbation $\delta_{\text{principal}}$ chosen along a direction that strongly affects the logits (e.g. a leading sensitive direction for the predicted class) rescaled as well to the null perturbation magnitude. For each type we compute the logit change in L2-norm over a validation set and summarize the distribution in Figure 3.

As expected, null-space perturbations produce negligible logit changes, while random and principal perturbations have a noticeable shifts. In Figure 7 we illustrate the corresponding UnCLIP generations for a single feature under these three perturbations and multiple seeds.



(a) Logit change under null vs. random perturbations.



(b) Logit change under principal vs. random perturbations.

Figure 3. Distribution of logit changes $\Delta_\ell(f, \delta)$ for null-space, random, and principal perturbations. Null-space perturbations leave logits almost unchanged, whereas principal perturbations induce large logit shifts.

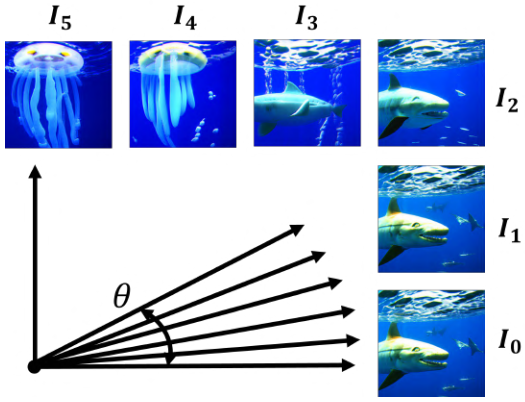


Image	AS (°)	IS (°)
I_0	0.00	0.0
I_1	1.58	4.0
I_2	3.80	10.8
I_3	4.70	23.0
I_4	9.48	29.0
I_5	11.29	36.2

Figure 4. Example images with different attribute scores (AS) and image scores (IS), illustrating the relationship between angular distance and perceived semantic change. Small angles correspond to nearly identical images, while larger angles reflect more significant semantic changes.

3. Image-level and visualization details

3.1. Angle visual interpretation

For the readers convenience, we provide a visual interpretation of the angles we measured along the paper. For two non-zero vectors u and v we define the angle in degrees

$$\theta(u, v) = \arccos\left(\frac{u \cdot v}{\|u\|_2 \, \|v\|_2}\right) \cdot \frac{180}{\pi}. \tag{5}$$

The attribute score (AS) and image score (IS) used in the main paper are instances of $\theta(\cdot, \cdot)$ applied in CLIP image-embedding space. Figure 4 provides a concrete mapping between AS/IS values and visual changes for a single example. Angles below roughly $3°$ in AS and $10°$ in IS correspond to barely perceptible changes, while larger angles produce clear semantic differences such as pose or shape variations.

3.2. Visualization with UnCLIP

UnCLIP is a two-stage image generator: a *prior* maps text to a CLIP image embedding, and a diffusion-based *decoder* with super-resolution modules synthesizes the corresponding image [19]. CLIP encoders normalize image and text embeddings to unit length and compare them using cosine similarity, so semantic information is primarily encoded in the angular component on the unit hypersphere [18].

We use trained translators T_Θ to map classifier features f and their perturbed variants $\tilde{f}$ into the CLIP image-embedding space. Given a feature and its equivalent feature set translated to CLIP, $T_\Theta(f)$ and $T_\Theta(\tilde{f})$, we rescale the translated equivalent feature to match the norm of the original:

$$\hat{T}_\Theta(\tilde{f}) = T_\Theta(\tilde{f}) \, \frac{\|T_\Theta(f)\|_2}{\left\|T_\Theta(\tilde{f})\right\|_2}. \tag{6}$$

This preserves the angular relationships while restoring the radial component, preventing distortions in the visualizations due to radial drift.

To ensure that observed visual differences are solely attributable to changes in the classifier feature f, we remove the stochasticity in the diffusion sampling process. We fix the random seed, draw a single Gaussian noise tensor with `randn_tensor`, scale it by the scheduler's `init_noise_sigma`, and reuse this tensor for all images in the batch and for both the decoder and super-resolution stages. For a fixed CLIP image embedding, this procedure yields deterministic outputs.

Our implementation uses the **Karlo-v1.0.alpha** UnCLIP model [5], which follows the original OpenAI framework [19]. The system includes frozen CLIP text and image encoders, a projection layer into the decoder space, a `UNet2DConditionModel` decoder, two `UNet2DModel` super-resolution networks, and `UnCLIPScheduler` instances for both stages. A generation example of the same feature translated by different translators is shown in Figure 5.

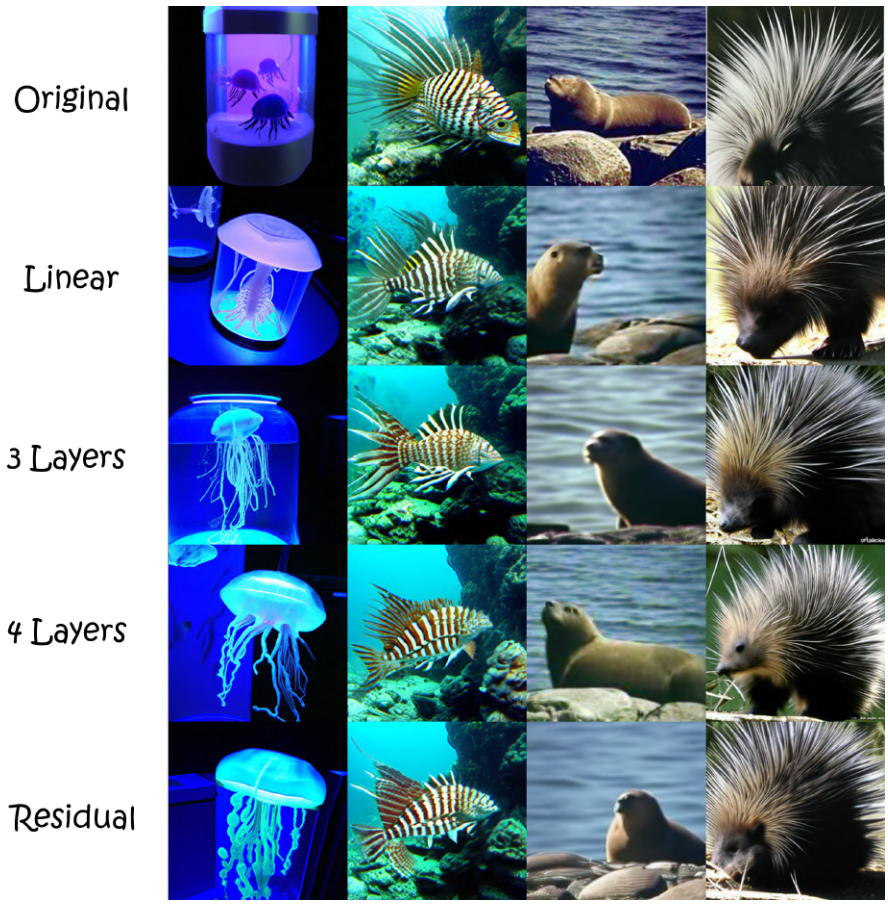
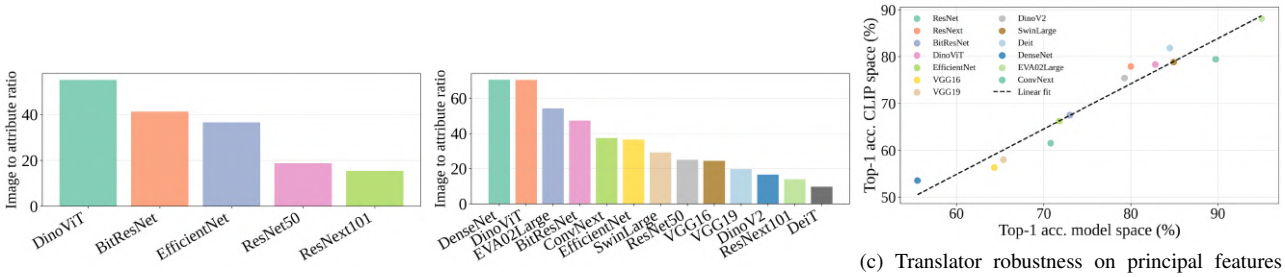


Figure 5. UnCLIP generations from a single classifier feature translated by different translator architectures. Despite small quantitative differences in cosine similarity, the resulting visualizations are qualitatively consistent.



(a) 5-model ratio comparison in EVA02 space. (b) Extended ratio comparison over 12 models. (c) Translator robustness on principal features (Pearson 0.972).

Figure 6. Additional model-level validations used in the camera-ready update.

4. Model-level result extensions

We include three additional checks requested during rebuttal integration. First, we repeat the 5-model ratio comparison with EVA02 as the target multimodal space. Second, we expand the ratio comparison from 5 models to 13 models pretrained on ImageNet [4] to increase architectural variety. the list of all models can be found in 2. Third, we evaluate translator robustness by training classifier heads on 500k principal features before and after translation to CLIP space, and computing the model-wise Pearson correlation of classification accuracy. 6. The resulting Pearson score is 0.972, indicating strong consistency between the original-principal and translated-principal feature spaces.

Table 2. List of models from CNNs to ViTs that we used as our test subjects.

Model	ImageNet Top-1 Acc (%)
VGG-16 [20]	71.6
VGG-19 [20]	72.4
DenseNet-121 [9]	74.4
ResNet50 [7]	76.1
DinoViT [3]	84.0
EfficientNet-B0 (NS) [23]	78.7
BiT-ResNet (M-R50x1) [10]	80.4
ResNeXt-101 32x8d (WSL) [14]	82.6
ConvNeXt-Base [12]	85.8
Swin-L [11]	86.3
DINOv2-L [15]	86.5
DeiT-3-L/16 [22]	87.7
EVA-02-L [6]	89.9

5. Class-level analyses

We provide violin plots for all models that participated in our experiments (Figures 8a to 8c). Each violin summarizes the distribution of semantic angle changes (in degrees) under null-space perturbations for a given class. Figures 9 and 10 provide extended open-vocabulary concept lists used in the class analyses of the "Arabian Camel" and "Jellyfish" classes in

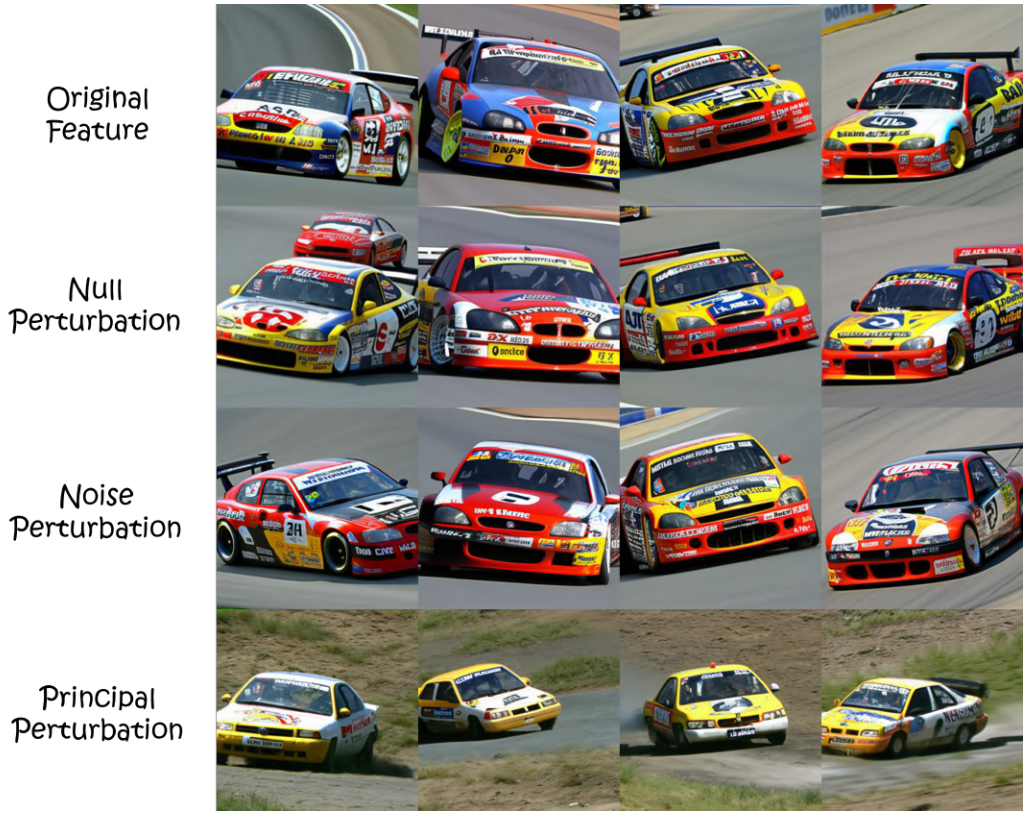
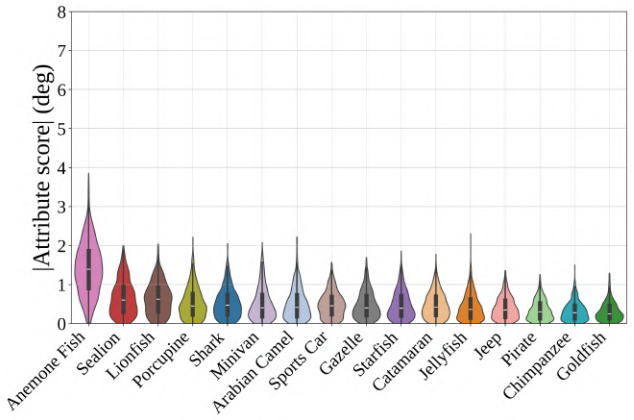
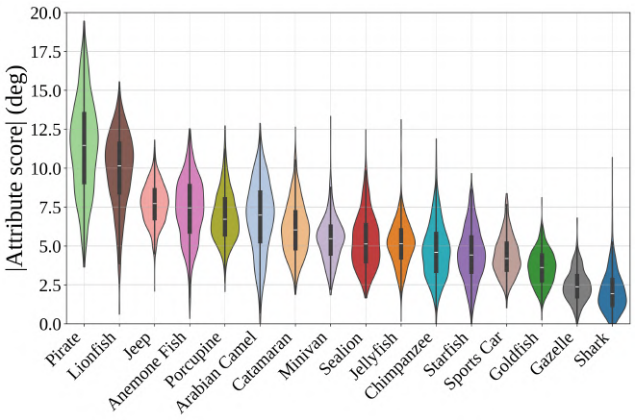


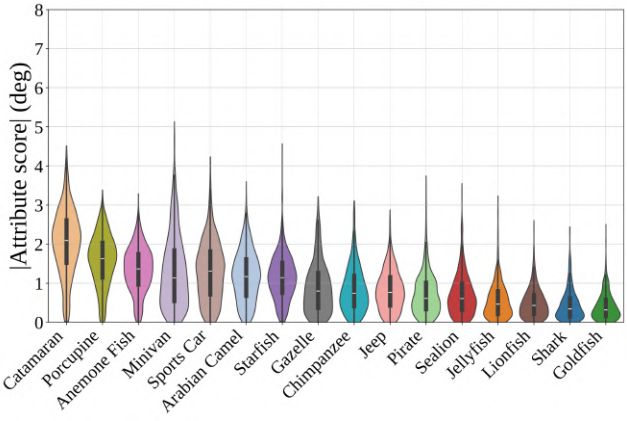
Figure 7. UnCLIP generations of a single feature under three perturbation types (null, random, principal) across four random seeds. Null-space perturbations preserve the global class semantics, while random and principal perturbations produce more noticeable semantic changes.



(a) BiT-ResNet [10].



(b) ResNeXt [14].



(c) EfficientNet [23].

Figure 8. Per-class distribution of null-space semantic angle changes across three architectures. Each violin corresponds to a single class; narrow distributions around zero indicate classes largely invariant to null-space perturbations. The consistent pattern across architectures with different inductive biases confirms the generality of our observations.

DinoViT. Nodes correspond to text prompts and the target class, and edge strengths reflect CLIP similarity between image and text embeddings. These plots show that the concepts we highlight in the main paper are representative of broader open-vocabulary neighborhoods.

6. DinoViT feature wrapper

We use a wrapper around a pre-trained DinoViT backbone [3] to expose the penultimate feature f and the classifier head weights W. We extract the sequence of tokens from the layer immediately before the classifier head (denoted `"encoder.ln"` in our implementation), take the class token as f, and apply the original head to obtain logits $\ell(f) = Wf$.

```
class SelectClassToken(nn.Module):
    def __init__(self, f):
        super().__init__()
        self.f, self.B = f, 1
    def forward(self, x):
        # x: (B * num_tokens, f); reshape and select class token (index 0)
        return x.reshape(self.B, -1, self.f)[:, 0, :]
    def set_B(self, B=1):
        self.B = B

class DinoHookable(nn.Module):
    def __init__(self, base: nn.Module, extractor, feature_dim=1024):
        super().__init__()
        self.extractor = extractor
        self.fc = base.heads.head        # classifier head; weights are W^T
        self.penultimate = SelectClassToken(f=feature_dim)
    def forward(self, x: torch.Tensor) -> torch.Tensor:
        self.penultimate.set_B(x.size(0))
        # token sequence before the classifier head
        x = self.extractor.extract(x, "encoder.ln")
        # penultimate feature f (class token)
        x = self.penultimate(x)
        # logits; penultimate feature f is available for analysis
        return self.fc(x)
```

This wrapper allows us to reuse the original DinoViT classifier while directly accessing the feature space in which we construct translators and null-space perturbations.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 2

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from https://www.wandb.com/. 2

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 6, 7

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 5

[5] Lee Donghoon, Kim Jiseob, Choi Jisu, Kim Jongmin, Byeon Minwoo, Baek Woonhyuk, and Kim Saehoon. Karlo-v1.0.alpha on coyo-100m and cc15m. https://github.com/kakaobrain/karlo, 2022. 4

[6] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA-02: A visual representation for neon genesis. *arXiv preprint arXiv:2303.11331*, 2023. 6

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[8] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 6

[10] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020. 6, 7

[11] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, 2021. 6

[12] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

[13] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 2

[14] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018. 6, 7

[15] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. 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 Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024. 6

[16] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, Adam Lerer, James Massa, Tete Liskovich, Wojciech Chmiel, Roman Serdyuk, Mengjia Yang, Marcin Kopacz, Piotr Sal Pietrek, Franz Zesch, Jonas Schick, Jeff Dearing, Alban Bhargava, Kai Wu, Wojciech Zaremba, David Killeen, Jie Sun, Yang Liu, Ye Wang, Peizhao Ma, Rong Huang, Vaibhav Pratap, Ying Zhang, Abhishek Kumar, Ching-Yi Yu, Cong Zhu, Chang Liu, Jeremy Kahn, Mirco Ravanelli, Peng Sun, Shinji Watanabe, Yang Shi, Tao Tao, Raphael Scheibler, Stephen Cornell, Sanghyun Kim, and Stavros Petridis. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32:8024–8035, 2019. 2

[17] F. Pedregosa, G. Varoquaux, A. Gramfort, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 2

[18] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, and Jack Clark. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 4

[19] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022. 4

[20] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2

[22] Hugo Touvron, Matthieu Cord, and Hervé Jégou. DeiT III: Revenge of the ViT. In *Computer Vision – ECCV 2022*, pages 516–533. Springer, Cham, 2022. 6

[23] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020. 6, 7

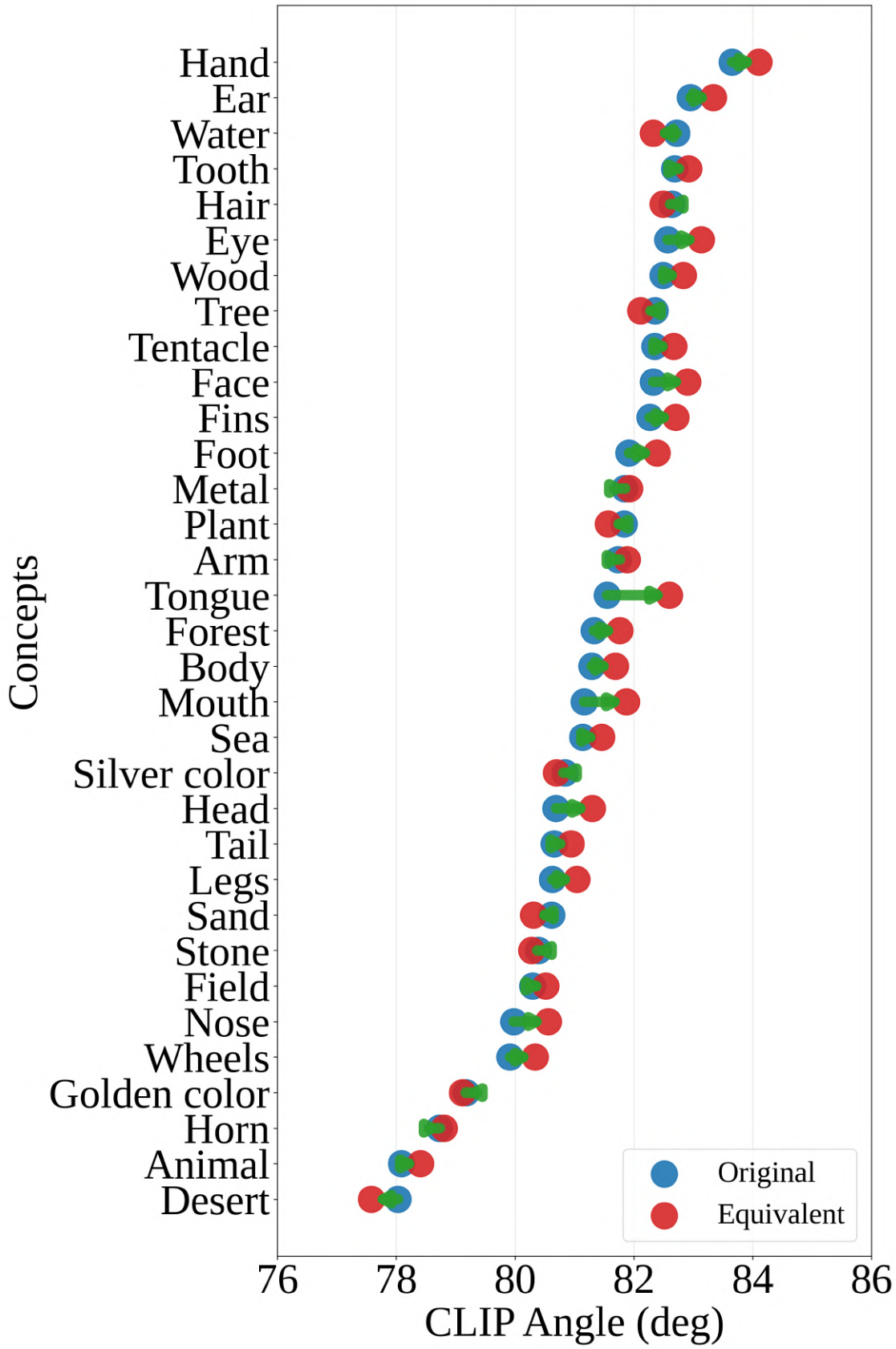


Figure 9. Open-vocabulary analysis for the “Arabian Camel” class in DinoViT. We show a larger set of prompts and their CLIP similarities to the class, illustrating the semantic neighborhood used in our analysis.

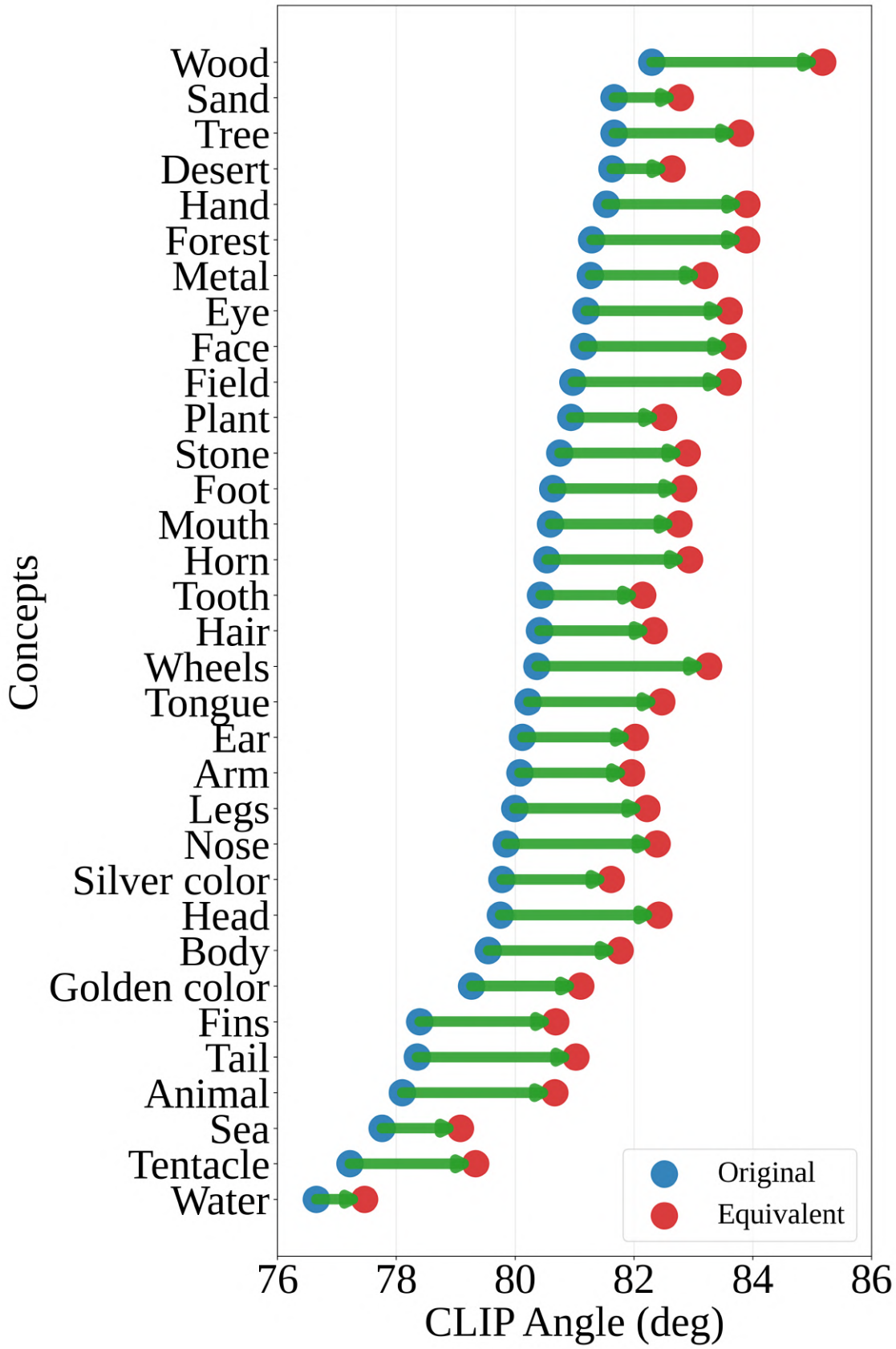


Figure 10. Open-vocabulary analysis for the "Jellyfish" class in DinoViT. The graph highlights related concepts and their CLIP similarities, showing that the concepts discussed in the main paper are part of a consistent semantic cluster.