Appendix[10]

A. Training Details

In this section, we provide more details about our implementation and a full list of the hyperparameters for reproducibility.

A.1. Architecture

We build our system on top of (DINO)'s [9] github implementation. We use the small Vision Transformer model variant (ViT-S/16) in all our experiments (unless otherwise stated) as it provides the best trade-off between throughput and accuracy and has comparable number of parameters to the baselines with which we compare our method. However, our implementation also supports the larger model variants which are evaluated in [9] (*e.g.* ViT-S/8 , ViT-B/16, and ViT-B/8).

On top of the ViT-S backbone `[CLS]` token output representation $\boldsymbol{z}_* \in \mathbb{R}^{384}$, we attach our projection MLP g_{θ_*} (*see* Fig. 1) comprising 3 linear layers with 2048 hidden dimension each followed by a GELU activation except for the last layer which uses an output bottleneck dimension of 256 and a linear activation. Subsequently, we connect our two heads to the output of the MLP $\boldsymbol{q}_*$: 1) the language MLP head ω_{θ_*} comprises 2 weight-normalized linear layers with GELU activation and a hidden dimension 2048 and output dimension 768; 2) a linear classifier comprising a single weight-normalized linear layer followed by a temperature sharpened softmax (the student network temperature differs from the teacher network temperature). Please refer to Tab. 6 for a list of hyperparameters.

A.2. LAVA Training Summary

LAVA training comprises a pretraining stage on a source dataset and an adaptation/transfer stage to a target dataset. During source pretraining, LAVA first learns self-supervised representations by minimizing DINO [9] loss $\mathcal{L}_{ssl}$. Then it uses the labels to learn a mapping (*i.e.* language MLP) between the frozen self-supervised representations and the label language embeddings using our hinge loss $\mathcal{L}_{sem}$(Eqn. 2). During target training, LAVA uses a modified variant of the DINO self-supervised objective (described in detail in next section) to adapt its source representations to the target domain without any use of target labels. Subsequently, LAVA employs a hybrid supervised/unsupervised loss to further train on the target labelled/unlabelled instances: the hinge loss on the labelled instances and our novel multi-crop pseudo-label loss on the unlabelled ones. Importantly, the language MLP is transferred directly from source to target without the need to be reinitialised (as opposed to a linear classifier head in vanilla transfer settings). That is because it predicts a fixed size embedding in a dataset-independent semantic space (the pretrained language model space). Concisely, LAVA loss can be summarised as per: $\mathcal{L} = \mathcal{L}_{ssl} + \mathcal{L}_{sem} + \mathcal{L}_{pl}$ with modulation coefficients to control the contribution of each term to the different stages of the training pipeline.

[10]All section, table, and figure references are following the original paper.

	clipart	quickdraw	painting
KNN before adaptation	51.09	35.28	56.49
KNN after adaptation ($\gamma = 0.996$)	65.11	58.31	59.68
KNN after adaptation ($\gamma = 0.95$)	69.08	62.29	61.58

Table 5: KNN validation accuracy for different values of teacher momentum (γ).

A.3. DINO Target Adaptation

After source DINO pretraining, we adapt to target dataset by fine-tuning the source representations on target unlabelled instances using the same DINO objective. In all our experiments, we found that 50 epochs are sufficient to adapt to the target dataset. Empirically, we found that there are two crucial factors to enable the success of this procedure: first, it is important to load the source pretrained weights of the MLP and the DINO head in conjunction with the backbone weights and not just the backbone as commonly done when fine-tuning. The second factor that highly impacts the adaptation procedure is the teacher EMA momentum (γ) which controls the speed with which the teacher model weights follow the student. We found that allowing the teacher network to update its parameters faster than it did during the source pretraining helps to better adapt to the target dataset. More specifically, instead of using the default value of $\gamma = 0.996$ used in [9], we use $\gamma = 0.95$ during adaptation. In Tab. 5, we present a comparison between the different options for teacher momentum while adapting to various datasets. We report the KNN accuracy on 3 datasets of DomainNet before adaptation (*i.e.* using ImageNet pretrained model) and after adaptation with two different values for the teacher momentum. We observe that adaptation with lower value of γ significantly improves the learnt representations.

B. Coupling DINO and LAVA

We exploit the fact that DINO and LAVA share a similar self-distillation backbone and only differ in the projection heads architecture, their training procedure and the hyperparameters. Accordingly, to allow training both DINO and LAVA using the same codebase, we simply attached an additional head to LAVA's MLP g_{θ_*} projection with an output space dimension matching that of DINO's (they use 65536 output dimension by default). Subsequently, one can switch between DINO pretraining and LAVA training by ad-

Hyperparameter	Value
batch size	256
learning rate	0.0005
optimizer	adam
minimum learning rate	0.000001
warmup learning rate	True
scheduler	cosine decay
weight decay start/end	0.04/0.4
num small crops	8
num large crops	2
global crops scale	(0.4, 1.0)
local crops scale	(0.05, 0.4)
spatial augmentations	random flip, color jitter, gaussian blur, solarization.
out dim (c)	65536
student softmax temperature	0.1
teacher softmax temperature	0.07
warmup teacher temperature	True
teacher temperature start	0.04
teacher momentum start (γ)	0.996/0.95
teacher momentum end	1.0

Table 6: DINO pretraining default hyperparameters.

Hyperparameter	Value
learning rate	0.000025
num small crops student	6
num small crops teacher	0
num large crops student	2
num large crops teacher	2
student softmax temperature	0.1
teacher softmax temperature	0.04
warmup teacher temperature	False
teacher momentum start (γ)	0.99
teacher momentum schedule	cosine
language model	*mpnet-base-v2*
language model latent dimension (d)	768
hinge loss margin (η)	0.4

Table 7: LAVA default hyperparameters.

justing the running configuration to match the respective setup. An additional benefit for such coupling is that it allows us to add DINO proposed loss as an auxiliary loss to our model objective during adaptation and explore if it introduces any benefits to LAVA. The intuition is that since we use DINO for LAVA source pretraining and target fine-tuning, it might be useful to continue applying its loss as an auxiliary loss so as to prevent damaging or "forgetting" the learnt self-supervised representations. However, we only found marginal benefits of doing so in the very limited label regimes (*e.g.* 1 and 2-shot experiments in SSL) but such benefits diminish once we have more labelled data.

classification	semantic	pseudo-label	MSCOCO (FSL)	Clipart (SSL)
	✓		66.41	43.39
✓			54.55	42.74
	✓	✓	67.68	48.89
✓		✓	57.25	48.57
✓	✓	✓	66.41	48.65

Table 8: LAVA's performance with different loss settings in SSL and FSL regimes.

C. Ablation Study

Here, we are interested to examine the effect of key design choices on LAVA's performance.

C.1. Semantics for FSL

Loss Ablations. In Tab. 1, we demonstrated a marginal benefit for our proposed semantic loss under the SSL setting and we concluded that the semantic loss is tailored specifically to address generalisation to new classes. To evaluate such claim, we conduct another experiment in the FSL setting on MSCOCO dataset: starting from LAVA's base learner, described in Sec. 4, we run 100 test episodes of MSCOCO (using the same random seed for the episode generation) while ablating over three different losses: standard classification loss using one-hot labels as targets, our proposed semantic loss using language semantics as targets, and our proposed multi-crop pseudo-labelling loss. In Tab. 8, we report the ablation results of both SSL and FSL regimes side-by-side for comparison. First, we observe that pseudo-label loss helps in both cases but its role is more evident in SSL as expected. In contrast, we observe that the semantic loss plays an important role in FSL: when combined with pseudo-label loss, it achieves a 10% boost in accuracy when compared with the standard classification loss (67.68% vs 57.25%) while the difference between the same cases in SSL is marginal (48.89% vs 48.57%). This confirms the usefulness of rich semantics to generalise to unseen classes as conjectured earlier. Finally, we observe that using semantic and pseudo-labeling losses (LAVA standard setting), we obtain the best performance for both SSL and FSL cases.

Source of semantics. Here, we examine different alternatives to obtain the class label embeddings (Ω). The first choice as in the seminal work of [16] is to simply use the word embeddings of the class labels (such as Word2Vec [30] or Glove [36]) to ground the semantic relations between classes. Word embeddings are learnt in an unsupervised manner to capture contextual relations between words based on their co-occurence statistics in a large corpus of text. This results in vectors which capture contextual similarity but not necessarily visual similarity. Alternatively, Nassar *et al.* [32] suggested to use knowledge graphs [48] to adjust word embeddings in a way which also correlates with visual similarity. While the two methods work reasonably well, they both suffer from a cov-

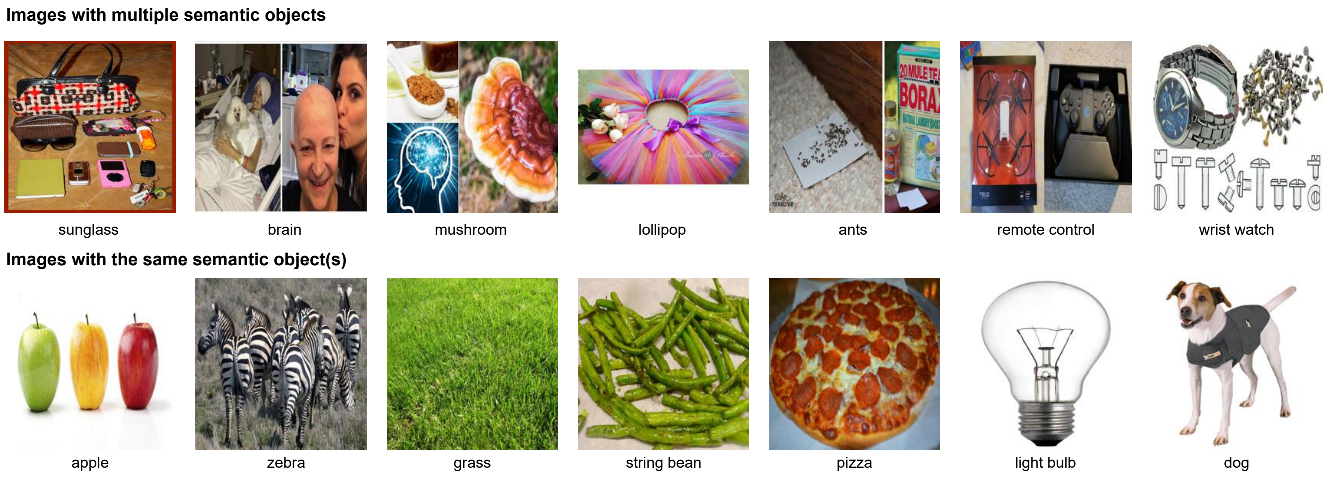

Figure 6: We display a subset of the top (top) and least (bottom) ranked images based on their large crops pseudo-label disagreement rate. We observe that the instances with high disagreement rate are those which contain multiple semantic objects, while instances with the same semantic object tend to suffer less from inconsistencies.

erage issue because they use a predefined set of vocabulary and so it is common that some of the target class labels do not exist in their vocabulary. Accordingly, we suggest in our work to use a pretrained language model using sub-word tokenization such as BERT [12] and its variants. Such models alleviate out-of-vocabulary problems by operating on parts of words (*i.e.* subwords) instead of words. Hence, to examine these different alternatives, we compare LAVA's performance when using each of them. We fix the learning task to the MSCOCO FSL task where we run 100 FSL test episodes with different choices of embeddings: 1) Glove word embeddings [36], 2) Knowledge graph embeddings [32], 3) A multilingual BERT-based sentence encoder[11], and 4) a praphrase language model [47]. We respectively obtain the following top-1 average performance on the 100 FSL MSCOCO episodes (using the same random seed for the episode generation): 63.52%, 64.12%, 66.67%, and 67.68%. Accordingly, we select the paraphrase model as the default option for LAVA.

C.2. Design alternatives for Multi-crop pseudo-labelling

As discussed in Sec. 3.2, we explored different design choices to aggregate the multi-crop losses. We present a comparison between the different design choices in Tab. 9. Specifically, $\mathcal{L}^i_{multi}$ can be calculated as *pair-wise average soft* pseudo-label loss as in our method (in the main text), or as the *pair-wise average hard* pseudo-label loss, *i.e.* by replacing $(\boldsymbol{p}^j_t)$ with $\arg\max(\boldsymbol{p}^j_t)$. Alternatively, a *single average soft* pseudo-label can be obtained based on all the teacher crops then used as a soft target against all the student crops predictions, *i.e.*

$$\mathcal{L}^i_{multi} = \frac{1}{|\mathcal{S}^i|} \sum_{\tilde{\boldsymbol{u}^j}_s \in \mathcal{S}^i} -\boldsymbol{p}^i_t \log \boldsymbol{p}^j_s, \tag{2}$$

$$\text{where,} \quad \boldsymbol{p}^i_t = \frac{1}{|\mathcal{T}^i|} \sum_{\tilde{\boldsymbol{u}^j}_t \in \mathcal{T}^i} -\boldsymbol{p}^j_t \tag{3}$$

or a *single average hard* pseudo-label, *i.e.* by replacing $(\boldsymbol{p}^i_t)$ with $\arg\max(\boldsymbol{p}^i_t)$ in Eqn. 2, or finally a *single majority hard* pseudo-label by applying a majority vote on the hard predictions of all the teacher crops, *i.e.* using $\boldsymbol{p}^i_t = \text{majority}(\arg\max(\boldsymbol{p}^j_t)) | \tilde{\boldsymbol{u}^j}_t \in \mathcal{T}^i$ instead of Eqn. 3.

D. Details of Analysis Experiments and Additional Examples

In this section we provide more details about the experimental setup for the analysis experiments in Sec. 5, as well as additional qualitative examples to provide more intuition.

D.1. Multi-crop pseudo-labelling analysis

Here, we elaborate on the experimental setup for the experiment presented in Sec. 5 - Fig. 4. We train LAVA using the *clipart* 2-shot SSL setting and while training, we capture the teacher and student predictions $\boldsymbol{p}_t$ and $\boldsymbol{p}_s$ for each of the large and small scale crops for every training iteration. Upon convergence (20 epochs of training), we apply argmax on all the captured values to obtain the most dominant "pseudo-label" as viewed by the student/teacher based on each of the crops. Subsequently, we use the ground truth labels of the SSL unlabelled instances to calculate the true top-1 accuracy associated with each of the crops and we average it over each epoch. Additionally, we calculate the dis-

[11] distiluse-base-multilingual-cased-v1 in `https://www.sbert.net/docs/pretrained_models.html`

Aggregation strategy	real 2-shot	clipart 2-shot	Small crops count	real 2-shot	clipart 2-shot	Momentum	real 2-shot	clipart 2-shot
pair-wise average soft	58.79	48.65	0	54.26	46.97	0	27.84	38.44
pair-wise average hard	57.22	46.25	4	56.12	47.9	0.9	51.02	46.02
single average soft	55.89	45.98	6	58.57	48.57	0.95	55.38	46.86
single average hard	54.12	45.12	8	57.77	48.68	0.99	58.67	48.57
single majority hard	55.95	46.43	10	57.97	48.42	0.999	57.90	30.81

Table 9: **Further Ablations.** Left: multi-crop loss aggregation strategy. Middle: number of small crops seen by the student model. Right: the effect of teacher momentum (γ).

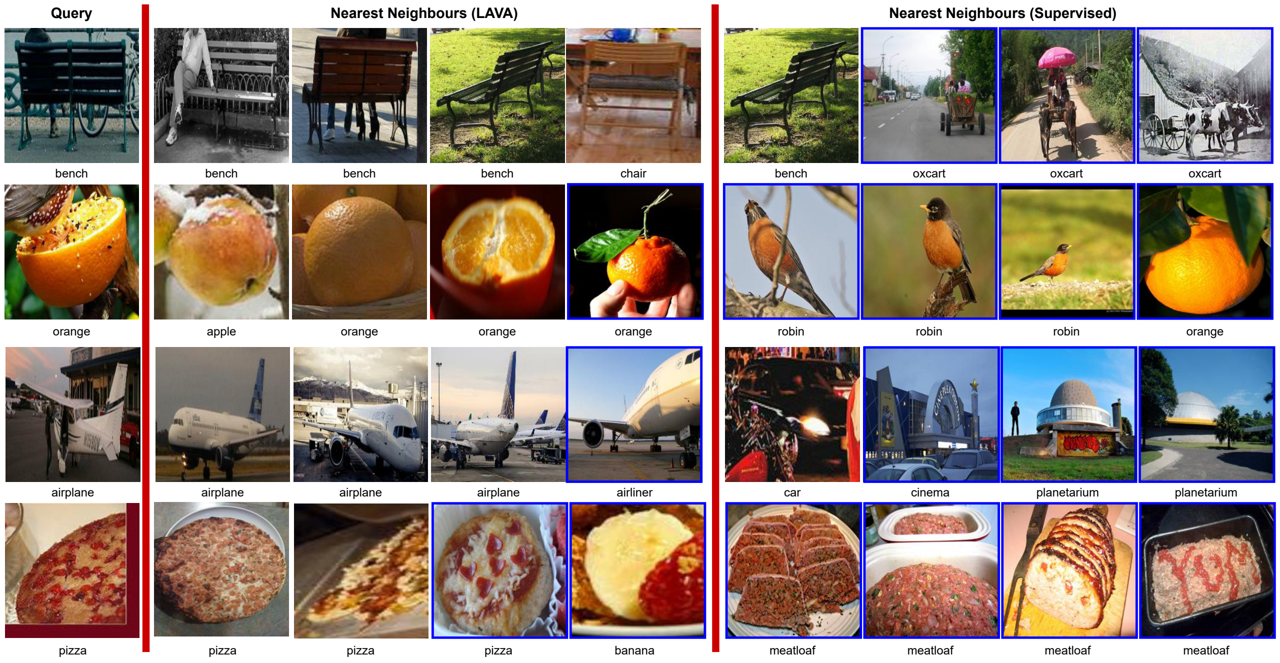

Figure 7: Additional examples of class collapse when transferring from ImageNet to MSCOCO dataset.

agreement rate among the small/large crops as the ratio between the number of unique pseudo-label classes obtained for small/large crops to the total number of small/large crops used. For example using 6 small crops, if the pseudo-labels obtained were (*dog*, *dog*, *dog*, *cat*, *squirrel*, *mouse*) then the disagreement rate is $= \frac{4}{6} = 0.667$. In Fig. 4, we report the accuracy based on one of the large crops seen by the student and teacher (denoted as *student* and *teacher* in the legend) together with two small crops seen by the student (denoted as *small1* and *small2*). Even though, we use 6 small crops and 2 large crops during training, we only display the above mentioned subset to avoid clutter. Moreover, we report the disagreement rate among the small crops and the large crops (denoted as *disagree_small*, *disagree_large*).

Finally, we rank all the training instances based on the disagreement rate among their large crops, averaged over all the training iterations, to examine which images suffer the most and the least from pseudo-labelling inconsistencies due to cropping. We display an additional subset of the top and least ranked images in Fig. 6. We observe that the instances with high disagreement rate are those which contain multiple semantic objects; which confirms our intuition about the necessity of the proposed multi-crop pseudo-labelling strategy.

D.2. Collapse Analysis

Here we provide further details about the transfer collapse described in Fig. 2. Doersch *et al.* [13] originally suggested that collapse happens as a result of the supervised pretraining used by most recent FSL methods when training their base learner. Essentially, since the learner is trained to merely classify images into one of a predefined set of classes, the learner encodes information which is useful to discriminate such training classes but discard any other information including that which is useful to generalise to new classes or domains. We further categorise supervision collapse into two types: class and domain collapse as we illustrate below. To visually examine such problem, we follow a protocol inspired by [13]: first, we train a supervised base learner (ViT) on all the instances of the 712 Meta-dataset ImageNet train classes using standard classification cross-entropy loss and following [51] training procedure. Then, we randomly select 100 instances per each of the 712 train classes together with 1000 query instances from a given target dataset (*e.g*. *MSCOCO* or *clipart*). Subsequently, we use the supervised base learner and LAVA to obtain the latent representations (*i.e*. z_t) for each of the sampled images (712 x 100 train + 1000 test query images); and we retrieve the 10 nearest neighbours (based on cosine similarity) for each of the 1000 query images with respect to such representations. Note that the 1000 images are never seen by either of the models during training and hence we hope that a general visual learner would be able to retrieve, for each query image, neighbours which are at least semantically related. If the learner retrieves a majority of neighbours which belong to one of the train classes for a given query image, it is said that its representations have collapsed to that training class. In Fig. 2 and Fig. 7, we display collapse examples from *MSCOCO* dataset, where we observe that *e.g*. a query "bench" instance has collapsed into "oxcart" and an "orange" instance has collapsed into a "robin" in the supervised learner case, while LAVA retrieves plausible neighbours.

To further investigate collapse in a different visual domain, we conduct a similar experiment but using *clipart* as the target dataset. Interestingly, we witness two types of collapse when the domain also differs, which we denote as "class collapse" (Fig. 8) and "domain collapse" (Fig. 9). The former is similar to what was described earlier where the majority of retrieved neighbours belongs to a semantically different train class which shares superficial similarity with the query image. While the latter is when the retrieved neighbours belong to a semantically similar train class albeit in the training visual domain rather than the target domain. To elaborate, examining Fig. 8, we see "class collapse" where a "monalisa" clipart instance collapses to "book jacket" ImageNet class (top section); and a "wheel" instance collapses to "dome" (third section from top). Whereas in Fig. 9, we witness a "domain collapse" where *e.g*. "hamburger" clipart instances collapses to "cheese burger" which is semantically related but belongs to the "real" visual domain rather than the target "clipart" domain.

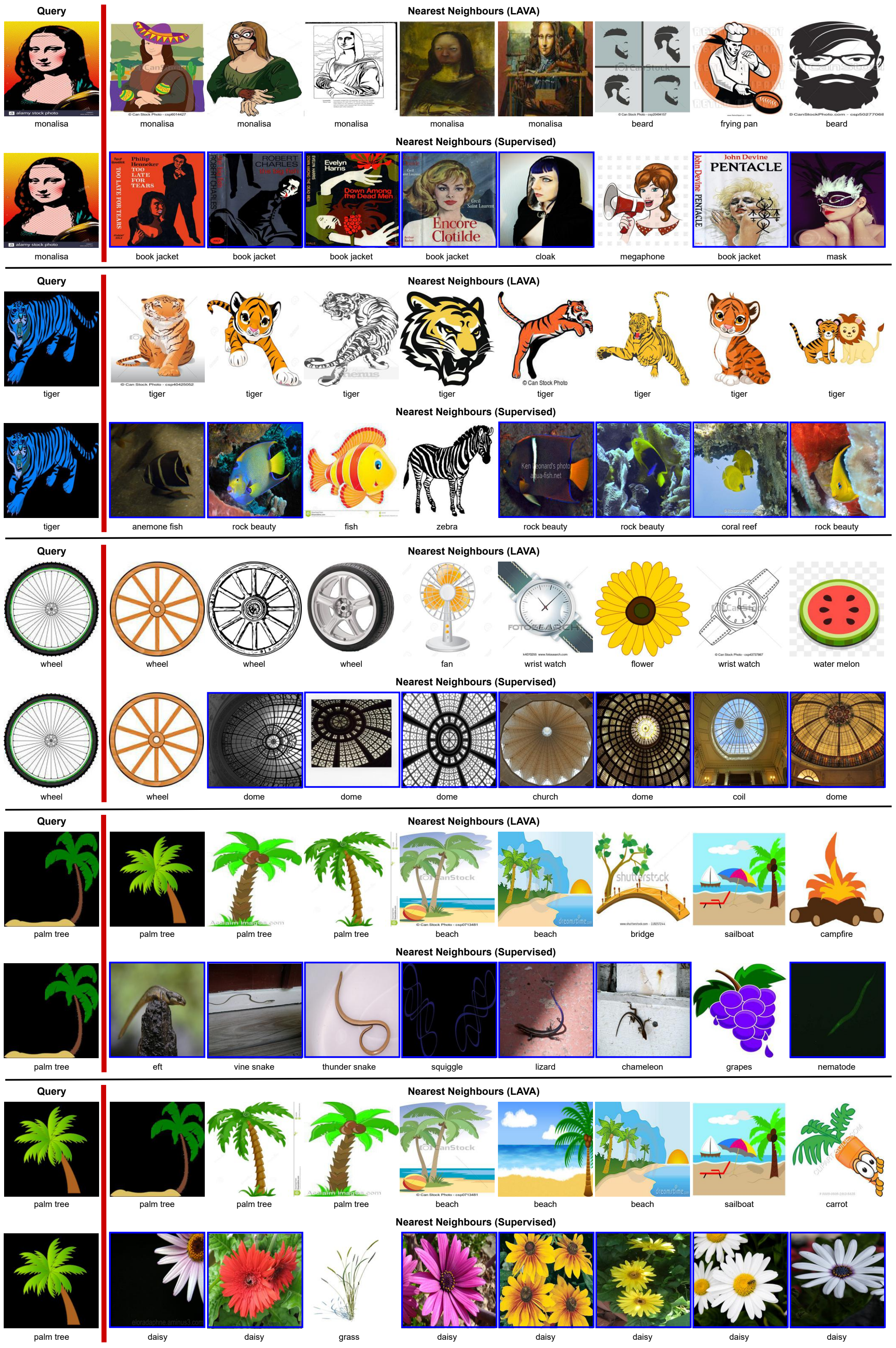

Figure 8: Class collapse examples when transferring from ImageNet to *clipart* dataset.

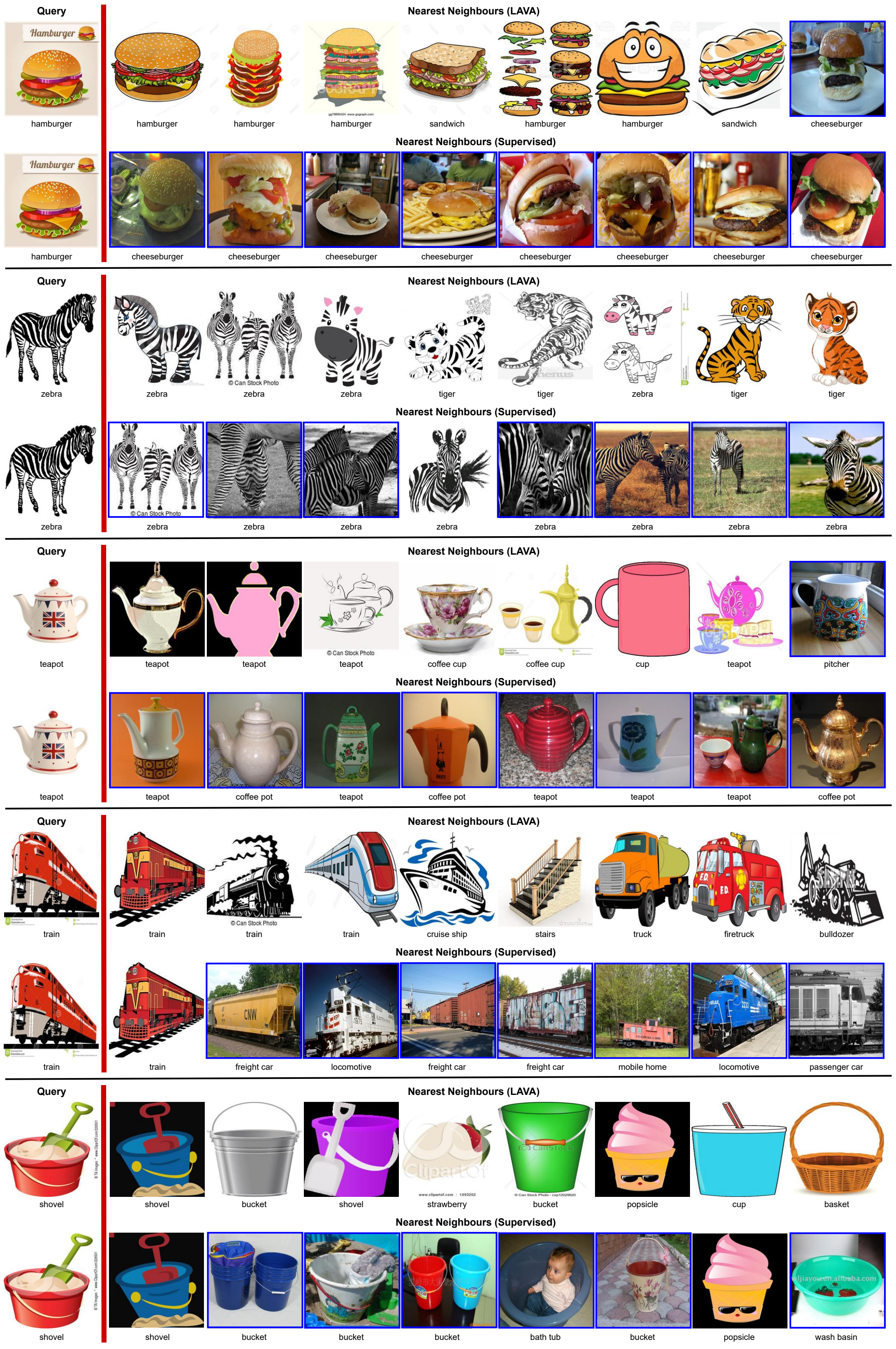

Figure 9: Domain collapse examples when transferring from ImageNet to *clipart* dataset.