A Additional Quantitative Results and Clarifications

A.1 Comparison of Different Pre-trained Visual Models

In our proposed pipeline (Figure 1), CLIP's image encoder is not the only candidate of pre-trained models. To further justify the generalizability of CPP, we evaluate the clustering performance of CPP leveraging visual representations from MAE (He et al., 2022) and DINO (Caron et al., 2021). For each model, CPP significantly improved the clustering performance and NMI score when compared with KMeans. Empirically, we find that both DINO and CLIP give good initializations for CPP; while MAE features are not suitable for image clustering, i.e. 12.3% accuracy via KMeans and 24.2% accuracy via CPP pipeline on ImageNet-1k, which is aligned with the discussion by Oquab et al. (2023) that MAE as a backbone are great for finetuning with labels, while less competent directly learn a discriminative representation.

		CPP Pipeline		KMeans	
pre-train	Backbone	ACC	NMI	ACC	NMI
MAE	ViT L/16	24.2	69.8	12.3	60.6
DINO	ViT B/16	59.0	84.5	53.5	81.6
DINO	ViT B/8	61.9	86.8	**56.0**	**85.2**
CLIP	ViT L/14	**66.2**	**86.8**	49.2	81.3

Table 3: **Benchmarking various models on ImageNet-1k with CPP and KMeans**. MAE and DINO models are pre-trained on ImageNet-1k; CLIP model is pre-trained on external data from their official implementation.

A.2 Comparison with more deep clustering methods

In addition to Table 1, we list other *state-of-the-art* deep clustering methods and report the quantitative performance. The primary purpose of showing Table 1 and Table 4 is not to compete with or surpass other deep clustering methods. Instead, we aim to probe the boundaries of image clustering. We clearly list the backbones for reference.

		CIFAR-10		CIFAR-20		CIFAR-100		ImageNet-1k	
Method	Backbone	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
MLC (Ding et al., 2023)	ResNet-18	86.3	76.3	52.2	54.6	49.4	68.3	-	-
SCAN (Van Gansbeke et al., 2020)	ResNet-18	88.3	79.7	50.7	48.6	34.3	55.7	39.9	-
IDFD (Yaling Tao, 2021)	ResNet-18	81.5	71.1	42.5	42.6	-	-	-	-
IMC-SWAV (Ntelemis et al., 2022)	ResNet-18	89.7	81.8	51.9	52.7	45.1	67.5	-	-
RUC+SCAN (Park et al., 2021)	ResNet-18	90.3	-	54.3	-	-	-	-	-
SPICE (Niu & Wang, 2021)	ResNet-34	91.7	85.8	58.4	58.3	-	-	-	-
NMCE (Li et al., 2022)	ResNet-34	88.7	81.9	53.1	52.4	-	-	-	-
TCL (Yunfan et al., 2022)	ResNet-34	88.7	81.9	53.1	52.9	-	-	-	-
C3 (Sadeghi et al., 2022)	ResNet-34	83.8	74.8	45.1	43.4	-	-	-	-
CC (Li et al., 2021)	ResNet-34	79.0	70.5	42.9	43.1	-	-	-	-
ConCURL (Deshmukh et al., 2021)	ResNet-50	84.6	76.2	47.9	46.8	-	-	-	-
Single-Noun Prior (Cohen & Hoshen, 2021)	ViT-B/32	93.4	85.9	48.4	51.5	-	-	-	-
TEMI* (Adaloglou et al., 2023)	ViT L/14	96.9	92.6	61.8	64.5	73.7	79.9	64.0	-
CPP*	ViT L/14	**97.4**	**93.6**	**64.2**	**72.5**	**74.0**	**81.8**	**66.2**	**86.8**

Table 4: **Comparison with more *state-of-the-art* deep clustering models**. Methods marked with (*) use pre-training from OpenAI CLIP, method use (#) use pre-training from OpenCLIP LAION-2B.

B Ablation Study

In this subsection, we conduct ablation studies on 2 components of CPP: pre-train datasets and diversified initialization.

The main results of our proposed methods leverage pre-training from OpenAI CLIP (Radford et al., 2021). To validate the generalizability of CPP, we additionally conduct experiments using ViT-L/14 from OpenCLIP (Ilharco et al., 2021), which is pre-trained on LAION-2B (Schuhmann et al., 2022). We report the results on Table 5.

Backbone	Pretraining	CIFAR-10		CIFAR-20		CIFAR-100		ImageNet-1k	
		CPP	KMeans	CPP	KMeans	CPP	KMeans	CPP	KMeans
ViT-L/14*	LAION-2B	96.5	94.0	70.2	58.7	76.7	62.3	66.8	50.2
ViT-L/14#	OpenAI	97.4	83.5	64.2	47.3	74.0	52.3	66.2	49.2

Table 5: **Ablation study on pre-training**. We report the clustering accuracy and observe that CPP consistently outperforms KMeans on two different pretrained models. (*) leverages open source pretraining from OpenCLIP, (#) leverages pretraining from official CLIP.

We then conduct an ablation study on the contribution of diversified initialization (described in Section 3.2). We diversify the representation via optimizing the objective in equation (4). This procedure improves the clustering performance on various datasets with results reported in Table 6.

Initialization	CIFAR-10		CIFAR-20		CIFAR-100		ImageNet-1k	
	ACC	NMI	ACC	NMI	ACC	NMI	ACC	NMI
Random	87.6	90.6	57.4	69.9	67.9	78.3	63.7	84.7
Diversified	**97.4**	**93.6**	**64.2**	**72.5**	**74.0**	**81.8**	**66.2**	**86.8**

Table 6: **Ablation study on the contribution of diversified initialization.** We randomly initialize pre-feature layer, cluster head and feature head (*Top*) and compare the performance with the diversified initialization (*Bottom*) in equation (4).

C Training Details

This section provides the training details - network architecture, datasets, optimization and hyperparameters.

Datasets. CIFAR contains $50,000$ training and $10,000$ test images, which are divided evenly into 10, 20 or 100 ground-truth classes, which we refer to as CIFAR-10, -20 or -100; note that the classes of CIFAR-20 are given by merging those of CIFAR-100. ImageNet incorporates around 1.2 million training images and $100,000$ test images, spread across $1,000$ classes, called ImageNet-1k. We process data in a manner identical to that used in CLIP (Radford et al., 2021), which involves resizing and center cropping images to dimensions of 224×224.

Network Architecture. We use a ViT L/14 model (Dosovitskiy et al., 2020) pre-trained via CLIP (Radford et al., 2021), with checkpoint from OpenAI[1]. As shown in Figure 1, we freeze the backbone during training and add a pre-feature layer composed of Linear-BatchNorm-ReLU-Linear-ReLU after the backbone. For feature head and cluster head, we use a Linear layer with mapping from the hidden dimension to feature dimension d respectively. Unified architecture is applied on all the experiments across different datasets except adjusted hidden dimension and feature dimension d. Finally, to learn a ideal representation that spans a union of orthogonal subspaces as in §3.1, note that the feature dimension d should be larger than or equal to the expected number of clusters. We leave more details in the Appendix C.

Details of Added Layers. For all datasets, we utilize a simple architecture composed of three parts: pre-feature layer, feature head and cluster head. Pre-feature layer has a structure with Linear-BatchNorm-ReLU-Linear-ReLU, with detailed setting in Table 7a. For feature head and cluster head, we use a linear layer respectively as is described in Table 7b.

Dimensions. Dimension d and d_{hidden} for each dataset are provided in Table 8a, note that d should be larger than or equal to the expected number of clusters to satisfy the orthogonal subspace assumption.

Optimizers. We specify two independent optimizers to simultaneously optimize the MLC objective with detailed parameters in Table 8b.

[1] https://github.com/openai/CLIP

(a) Pre-feature layer

Linear: $\mathbb{R}^{768} \to \mathbb{R}^{d_{hidden}}$
BatchNorm1d(d_{hidden})
ReLU
Linear: $\mathbb{R}^{d_{hidden}} \to \mathbb{R}^{d_{hidden}}$
ReLU

(b) Feature head and cluster head

Feature head	Linear: $\mathbb{R}^{d_{hidden}} \to \mathbb{R}^{d}$
Cluster head	Linear: $\mathbb{R}^{d_{hidden}} \to \mathbb{R}^{d}$

Table 7: Network Architecture

Sinkhorn Distance. The doubly stochastic membership matrix $\mathbf{\Pi}$ is computed by a sinkhorn distance projection on $\mathcal{C}^\top \mathcal{C}$, where the parameters γ regulate the sparsity of the membership matrix as is described in Section 3.1. Details with this parameter are recorded in Table 8c.

Optimization. As describe in §3.2, we first warmup our network by 1-2 epochs by training $R(\mathcal{Z}; \varepsilon)$ alone, then simultaneously optimize both feature head and cluster head using (MLC). For both the feature head and cluster head, we train with SGD optimizer, learning rate set to 0.0001, momentum set to 0.9 and weight decay set to 0.0001 and 0.005 respectively.

Initialization and Training Epochs. Details in initialization (simply optimize $R(\mathcal{Z}; \varepsilon)$) epochs and total training epochs are recorded in Table 8d.

(a) Model Parameters. We adjust the dimension of learned features for the different expected numbers of clusters.

Datasets/Parameters	d	d_{hidden}
CIFAR-10	128	4096
CIFAR-20	128	4096
CIFAR-100	128	4096
ImageNet-1k	1024	2048
MS-COCO	128	4096
LAION-Aesthetics	1024	2048

(b) Optimizers. We optimize the objective function using SGD optimizer with unified parameters as below:

Optimizers	Type	lr	wd	momentum
Feature	SGD	0.0001	0.0001	0.9
Cluster	SGD	0.0001	0.005	0.9

(c) Sinkhorn Distance Parameters while Training

Datasets	γ	Iter
CIFAR-10	0.175	5
CIFAR-20	0.13	5
CIFAR-100	0.1	5
ImageNet-1k	0.12	5
COCO	0.12	5
LAION	0.09	5

(d) Initialization epoch, total training epoch, batch size. Batch size doesn't affect too much on the performance. All experiments can be conducted on a single A100.

Datasets	$epoch_{init}$	$epoch_{total}$	bs
CIFAR-10	1	5	1024
CIFAR-20	1	15	1024
CIFAR-100	1	50	1500
ImageNet-1k	2	20	1024
COCO	1	20	1200
LAION	2	20	1024

Table 8: Core hyperparameters selected in experiments.

D SUBSPACE CLUSTERING PARAMETERS

We conduct subspace clustering methods on CLIP features and report the highest accuracy after searching for optimal parameters.

EnSC. Both EnSC and SSC-OMP[2] estimate a membership matrix via solving some convex optimizations that depend only on CLIP features, and then run spectral clustering on the resulting membership. For EnSC, we use the efficient active-set solvers from (You et al., 2016a) to solve the convex optimization. EnSC has two parameters γ, τ. Roughly speaking, $\tau \in [0, 1]$ balances between an ℓ_1 and an ℓ_2 penalty on the membership, with larger τ giving sparser affinity; $\gamma > 0$ is the weight of the data fidelity error, aside from the regularizing term.

SSC-OMP. (k_{max}, ϵ) We use the OMP solver for SSC (You et al., 2016b). k_{max} is the maximum number of non-zero entries of each row of the membership, while ϵ controls the allowed data fidelity error.

Spectral Clustering. γ denotes the parameter for sink horn distance projection, which is the same as the one mentioned in previous sections. For a given batch of CLIP's feature $\mathcal{C}' \in \mathbb{R}^{d \times n}$, we first normalize each feature vector and then do inner production plus sink horn distance projection, i.e. $\mathbf{\Pi}_{CLIP} = \text{proj}_{\Omega,\gamma}(\mathcal{C}'^{\top}\mathcal{C}')$. We then do spectral clustering on this membership matrix $\mathbf{\Pi}_{CLIP}$.

Table 9: Parameter search with the following parameters for EnSC, SSC-OMP and spectral clustering. We report the highest performance on Table 2.

Datasets	Parameters
EnSC	$\gamma \in [1, 5, 10, 50, 100], \tau \in [0.9, 0.95, 1.0]$
SSC-OMP	$k_{max} \in [3, 5, 10], \epsilon \in [1e-4, 1e-5, 1e-6, 1e-7]$
Spectral Clustering	$\gamma \in [0.2, 0.18, 0.16, 0.1, 0.09, 0.08, 0.07, 0.06]$

E EVALUATION ON IMBALANCED DATASETS

We evaluate CPP on imbalanced CIFAR-10 and imbalanced CIFAR-100, where images of odd classes (i.e. 1, 3, ...) are reduced to half of the original. Additionally, some large, uncurated datasets, like LAION-Aesthetic, exhibit natural imbalance. To demonstrate CPP's proficiency in identifying minority groups, we also present visualizations of clusters with fewer members in Figure 7.

F MORE RESULTS ON OPTIMAL NUMBER OF CLUSTERS

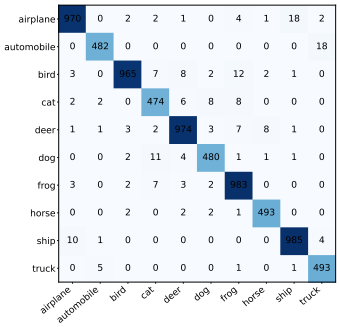
We additionally measure the coding length for ImageNet as is shown in Figure 8. For all datasets, we compute the coding length with $\epsilon^2 = 0.1$, which is consistent with the one in MLC objective function.

G IMAGE-TO-IMAGE SEARCH

Pipeline. Figure 9 demonstrates the pipeline of image-to-image search. In practice, the image repository is composed of 1.2M images from ImageNet's training split while the target image is randomly picked from ImageNet's validation split. We search the images in the repository via measuring the Euclidean distance and plot the 64 most similar images.

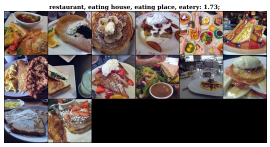
Results. Here, we provide 10 more image-to-image search results in Figure 10. We observe from these results that CPP learned better representation that facilitates image-to-image search.

[2]The implementations are provided by the authors at https://github.com/ChongYou/subspace-clustering.



(a) Imb. CIFAR-10

(b) LAION-Aesthetic Cluster (i)



(c) LAION-Aesthetic Cluster (ii)

Figure 7: Performance of CPP on imbalanced datasets. CPP achieved 97.3% and 71.3% clustering accuracy on Imb. CIFAR-10 and Imb. CIFAR-100 respectively; The confusion matrix demonstrates the prediction results on Imb. CIFAR-10 validation set (*Left*). LAION-Aesthetic is also a natural imbalanced dataset, where two clusters with few members are visualized, each composed of 0.73% and 0.47% images respectively from the dataset (clustering on 30k random samples from LAION-Aesthetic) (*Middle*, *Right*)

.

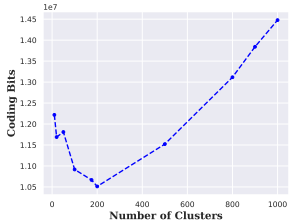


Figure 8: ImageNet (200)

H More Results on Clustering and Labelling with Text

Text Candidates Selection. Ideally, an open vocabulary source with tons of highly reliable labels best suits our needs. However, text candidates of this quality and scale are usually hard to be obtained. In practice, In practice, we leverage powerful LLMs, such as GPT-4, to generate text candidates as an economical substitute. More specifically, we employ prompts like "generate 2000 names of real-world objects/creatures, generate 100 words of art styles, give me 100 words describning the content of paintings..." during the generation process. To guarantee the diversity and reliability, we furtherly mix them up with 1000 ImageNet class labels to construct the final 3000+ text candidates. It is noteworthy to mention that we did not utilize any label information from MS-COCO, LAION-Aesthetic or WikiArt.

Pipeline. We introduce a cluster-labeling algorithm after we obtain a well-trained CPP. First, we do spectral clustering upon the membership matrix given by CPP and get clusters of images. Then, for images in each cluster, we conduct weighted voting for the common labels. The voting algorithm is described in Algorithm 2.

Results. In this section, we visualize more cluster-captioning results for datasets including CIFAR-100, ImageNet-1k, COCO and LAION-Aesthetics. We also follow the optimal number of clusters measured in Section 4.3 for each dataset.

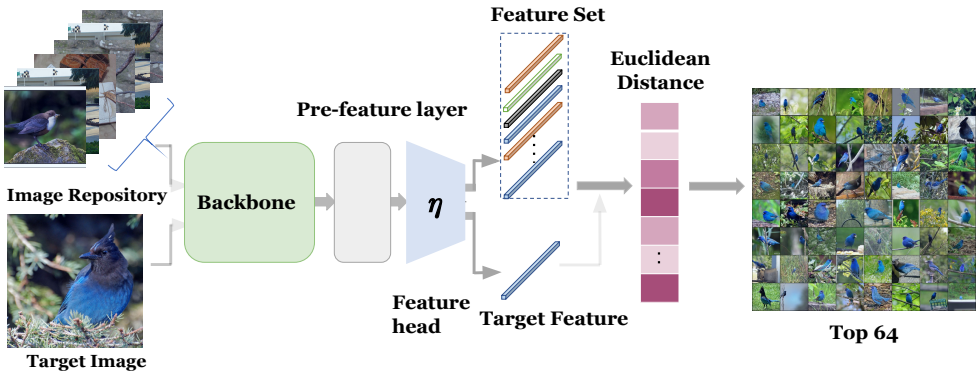


Figure 9: **Image-to-Image Search Pipeline.**

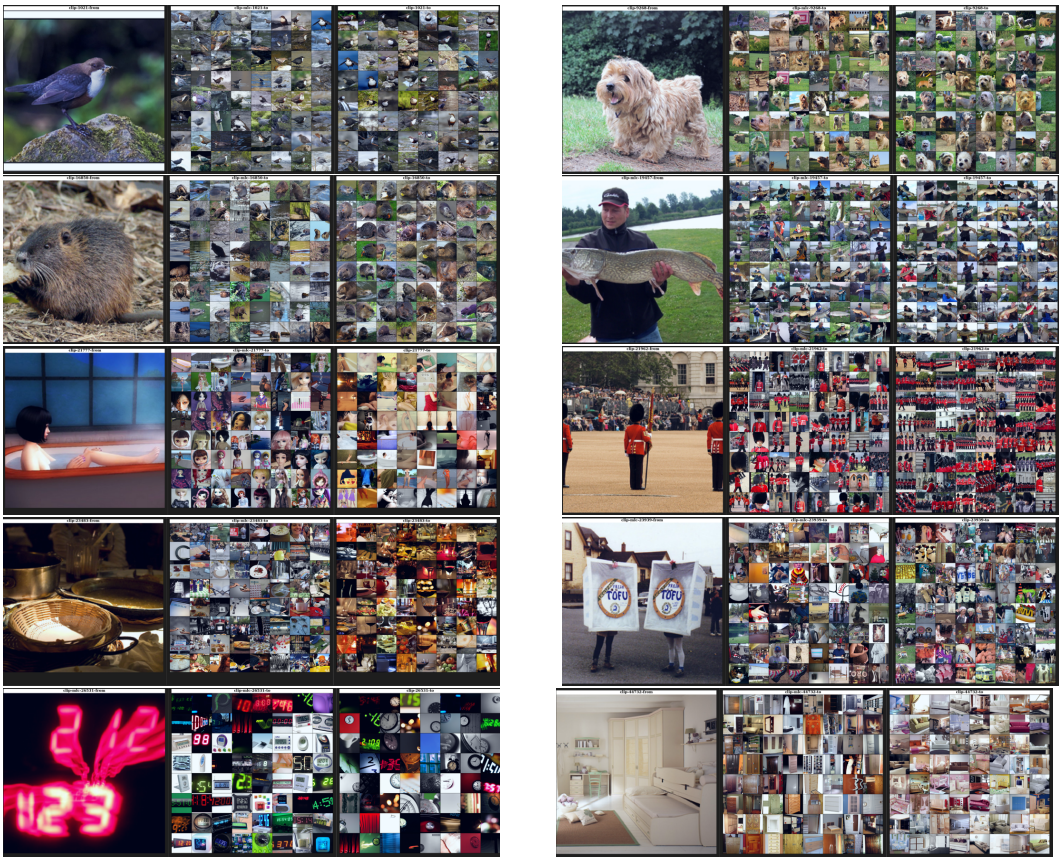


Figure 10: **Searching for similar images in ImageNet's training split**. *Left*: Target image from validation split in ImageNet.*Middle*: searched images via CPP's representation; *Right*: searched images via CLIP's representation.

Algorithm 2: Captioning one cluster

Input: Images from one learned cluster $\mathcal{X} \in \mathbb{R}^{N\times3\times224\times224}$, M text candidates, 0 initialized voting result vector $V \in \mathbb{R}^M$

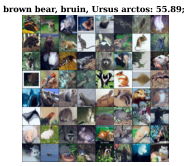
$$\mathcal{Z}_{img} \leftarrow \text{CLIP: encode images}(\mathcal{X})$$
$$\mathcal{Z}_{txt} \leftarrow \text{CLIP: encode texts (text candidates)}$$

For $i \leftarrow 1, \ldots, N$:

$$\text{Scores4labels} \leftarrow \text{Cosine Similarity for}(\mathcal{Z}^i_{img}, \mathcal{Z}_{txt}) \tag{8}$$
$$\text{Valid Score} \leftarrow \text{Score4labels[top5]} \tag{9}$$
$$V \leftarrow V + \text{Valid Score} \tag{10}$$

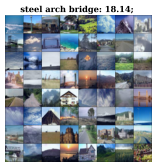
Output: Caption for this cluster: text candidates[$\operatorname{argmax} V$]

(a) tricycle.

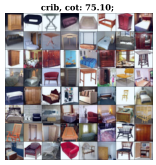


(b) brown bear.

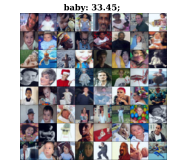
(c) goblet



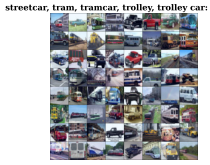
(d) steel arch bridge



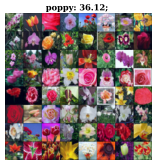
(e) crib



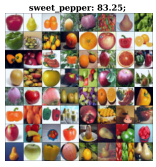
(f) baby



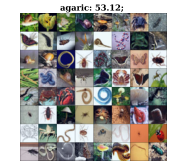
(g) streetcar.



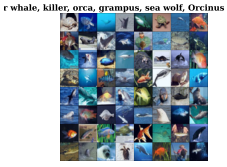
(h) poppy



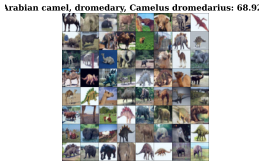
(i) sweet pepper



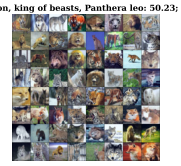
(j) agaric



(k) whale.

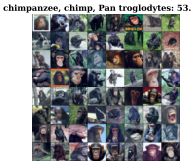


(l) Arabian camel.

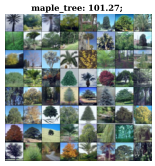


(m) lion.

(n) armored vehicle.

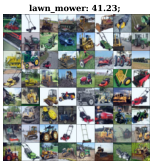


(o) chimp.

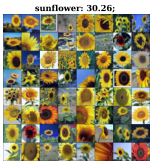


(p) maple tree

(q) dial telephone.



(r) lawn mower

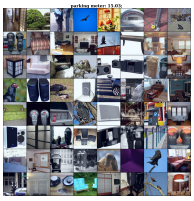


(s) sunflower

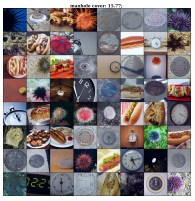
(t) bee

Figure 11: Clustering CIFAR-100 into 20 clusters and relabeling them using our pipeline.

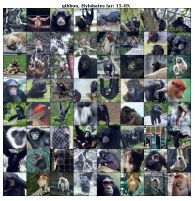
(a) head cabbage



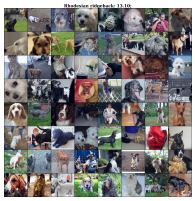
(b) parking meter



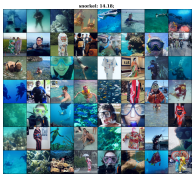
(c) manhole cover



(d) gibbon

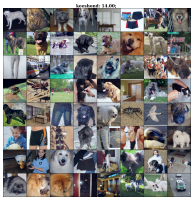


(e) Rhodesian ridgeback

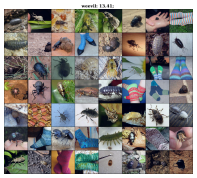


(f) snorkel

(g) lorikeet

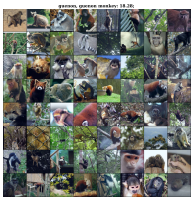


(h) keeshond

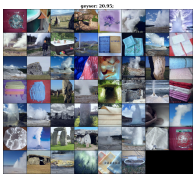


(i) weevil

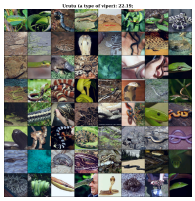
(j) buckeye



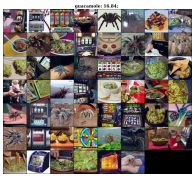
(k) guenon



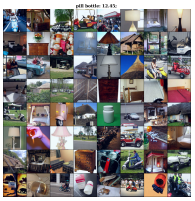
(l) geyser



(m) Urutu

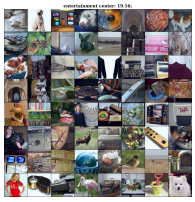


(n) guacamole

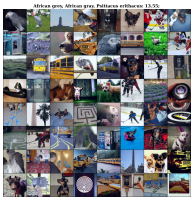


(o) pill bottle

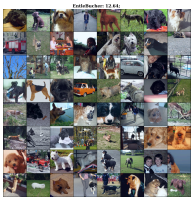
(p) French horn



(q) entertainment center



(r) African grey

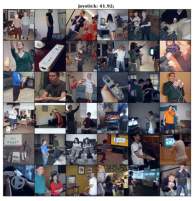


(s) EntleBucher

(t) rose hip

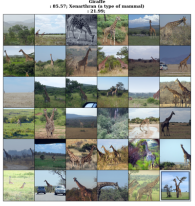
Figure 12: Clustering ImageNet (15k random samples from train split) into 200 clusters and relabeling them using our pipeline. (Randomly selected 20 clusters) Non-square figures represent that images within that cluster are not enough to fulfill the 8×8 grid.

(a) ballplayer



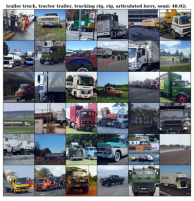
(b) joystick

(c) stove

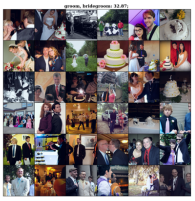


(d) Giraffe

(e) motor scooter

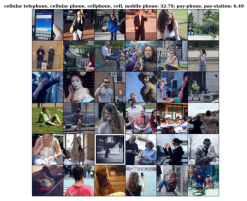


(f) trailer truck



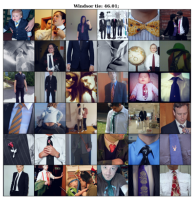
(g) groom

(h) church

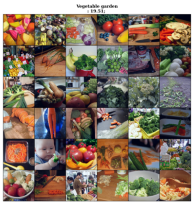


(i) cellular phone

(j) Police car



(k) Windsor tie

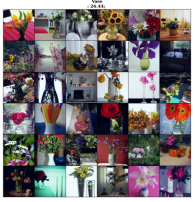


(l) Vegetable garden

(m) desktop computer

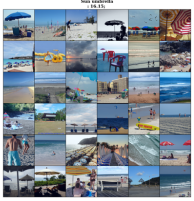
(n) surfboard

(o) Tennis racket



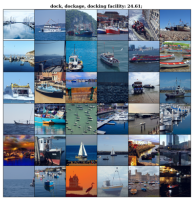
(p) Vase

(q) Fire extinguisher



(r) Sun umbrella

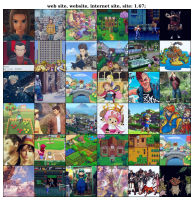
(s) soccer ball



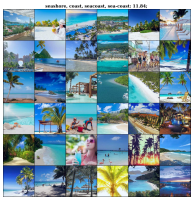
(t) dock

Figure 13: Clustering COCO (30k random samples) into 150 clusters and labeling them using our pipeline. (Randomly selected 20 clusters)

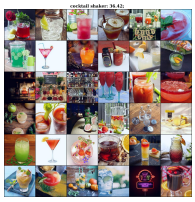
(a) Dollhouse kit



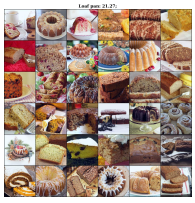
(b) website



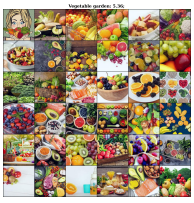
(c) seashore



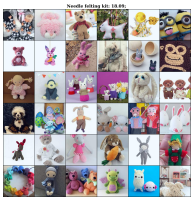
(d) cocktail



(e) Loaf pan

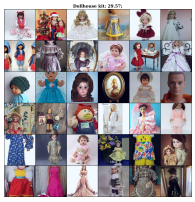


(f) Vegetable garden

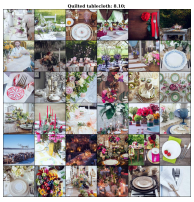


(g) needle fitting kit

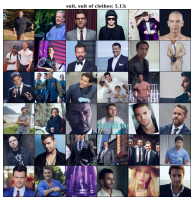
(h) fountain



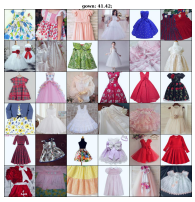
(i) Dollhouse kit



(j) Quiltered tablecloth



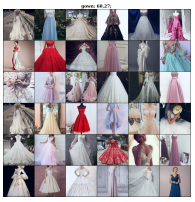
(k) suit



(l) gown

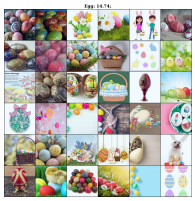
(m) accordion

(n) suit



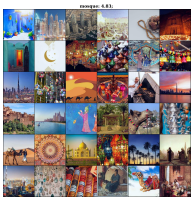
(o) gown

(p) Quiltered tablecloth

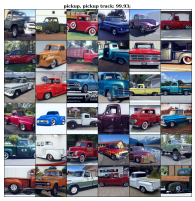


(q) Egg

(r) Model T



(s) mosque



(t) pickup truck

Figure 14: Clustering LAION-Aesthetic into 300 clusters and labeling them using our pipeline. (Randomly selected 20 clusters)