

APPENDIX: PERCEPTUAL GROUPING IN VISION-LANGUAGE MODELS

A QUALITATIVE EXAMPLES OF LOCALIZATION

In this section, we showcase more examples of the success and failures of CLIPpy on bottom-up unsupervised segmentation and top-down semantic segmentation.

In Figure 5, our bottom-up unsupervised segmentation on PASCAL VOC follows the same PCA based clustering setup ($K = 8$) to obtain CLIPpy predictions. We highlight how in single object settings with less clutter, the segmentation results are reasonable (column 1-3) while in the other more cluttered cases (column 4-6) our results are poor. Moreover, it fails to distinguish between similar classes such as a dog and a cat in column 3.

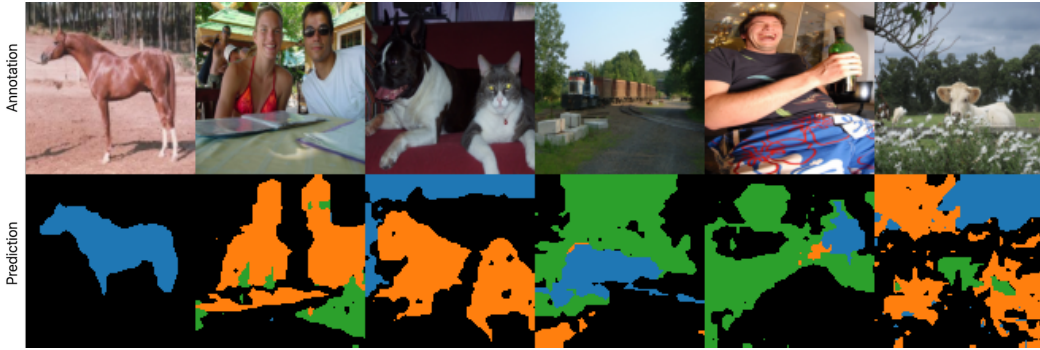


Figure 5: **Qualitative examples of bottom-up unsupervised segmentation with CLIPpy.** We illustrate examples from PASCAL VOC dataset with original image and CLIPpy prediction in the top and bottom rows respectively.

We present additional top-down semantic segmentation examples across all three datasets used for evaluation. Figure 6 contains examples from PASCAL VOC dataset for the same examples from Figure 5. We note how the top-down segmentation is able to correctly separate the dog and cat classes (column 3) and also improve performance in the more cluttered scenes. On the other hand, in column 4, it missed out on a portion of the train that was segmented properly in the bottom-up setting. Segmentations from CLIPpy also contain discontinuities within a single object region in some cases, especially for the background objects (col 2-3).



Figure 6: **Qualitative examples of top-down semantic segmentation with CLIPpy.** We illustrate examples from PASCAL VOC with original image, ground truth annotation, and CLIPpy prediction (for semantic segmentation) in each row from top to bottom respectively.

In Figures 7 and 8, we present more examples of top-down segmentation for COCO and ADE-20K datasets. A notable failure case of CLIPpy for some inputs is reverting to CLIP like behaviour,

predicting the salient object class at all locations. This is visible to some extent in column 2 & 3 in Figure 7 and column 3 in Figure 8. Additionally, CLIPpy results in false positives for cluttered scenes as visible in some examples of these two datasets.



Figure 7: **Qualitative examples of top-down semantic segmentation with CLIPpy.** We illustrate examples from COCO with original image, ground truth annotation, and CLIPpy prediction (for semantic segmentation) in each row from top to bottom.



Figure 8: **Qualitative examples of top-down semantic segmentation with CLIPpy.** We illustrate examples from ADE-20K with original image, ground truth annotation, and CLIPpy prediction (for semantic segmentation) in each row from top to bottom.

To summarize, CLIPpy is able to localize the salient objects well in less cluttered scenes, even when multiple objects belonging to different classes are present. It is also able to coarsely localize some of the salient objects in more cluttered scenes. In terms of drawbacks, CLIPpy often fails to correctly localize some background classes, particularly leading to noisy segmentation with discontinuity across a single object. The representation of the salient class also seems to leak to the surroundings in some cases, and even to the entire image in worst scenarios. It also misses out on some smaller objects in the background in scenes containing objects of multiple classes.

B DETAILS OF HQITP-134M DATASET

High Quality Image Text Pairs (HQITP-134M) consists of ~ 134 million diverse and high quality images paired with descriptive captions and titles. Images range in spatial resolution from 320 to 2048 pixels on the short side. All images are JPEG format and most are RGB. Each example image is associated with a title, and a list of several captions. A small fraction ($\ll 1\%$) of the examples are missing both captions and title. We favor the associated captions, and find that these tokenize to an average length of 20.1 tokens, although the shortest caption is only one token and the longest is over 1000. This dataset was licensed to our research lab by a third party for commercial use.

To preprocess this dataset for training, we first exclude all pairs for which no valid caption exists. We also perform global exact-byte-match image de-duplication across our full training corpus, meaning that valid examples may be dropped due to appearing in other subsets of our overall training dataset. As we draw each example, we create an image-text pair by sampling from the list of available captions. The text is then tokenized, and the image is resized so that the shortest side is 224 pixels, with a further random crop then applied over the longer dimension to produce a 224 x 224 pixel square image. Lastly, we normalize the image using statistics derived from our full training corpus.

C ARCHITECTURE AND TRAINING DETAILS

Our implementations of CLIP and ALIGN employ ViT-B/16 (Dosovitskiy et al., 2020), and EfficientNet-B5 (Tan & Le, 2019) for the image embedding, respectively, to mirror the primary results presented in each respective vision-language model. For the ViT architecture, we experimented with varying patch sizes $P = 8$ and $P = 16$ in order to leverage open-sourced DINO pretrained weights (Caron et al., 2021), but report all of our results with $P = 16$.

We train all models on 224x224 images to provide a fair comparison with Radford et al. (2021). Note however that the published version of ALIGN employed a 640x640 resolution. ViT models may operate on images at arbitrary spatial resolution. At inference time we experimented with spatial resolutions of 224x224 and 448x448, resulting in 196 and 784 tokens, respectively. Results were similar across 224 and 448 resolutions, we report only results at 224 for brevity (except for Figure 4).

During training, we also utilize overlapping patch generation with patch sub-sampling as regularization. In particular, the token sequence length is always maintained at the original value of 224 during training through random sub-sampling (patches selected according to uniform distribution). This also allows obtaining a higher resolution feature map from a fixed resolution image during inference.

We train models with 32 GPUs across 4 machines with PyTorch (Paszke et al., 2019) using the LAMB optimizer (You et al., 2019) with a cosine decayed learning rate with linear warm-up. We employ an initial learning rate of 1e-3, 2000 warm-up steps, and decay the rate with a single period over the entire training regime (32 epochs for CC12M and 10 epochs for HQITP-134M). We additionally employ a weight decay of 1e-2. All training parameters were determined through moderate hyperparameter tuning.

D LIMITATIONS OF CNN-BASED ARCHITECTURES

The primary focus of our presentation is on the Vision Transformer (ViT) architecture (Dosovitskiy et al., 2020). The reason for this focus is that the Transformer architecture is particularly well suited for multimodal learning tasks because one does not need to craft an architecture for each modality, and tune the training set up for each particular architecture. We also recognize that convolutional neural networks (CNN’s) have a long history of providing state-of-the-art CNN results on computer vision related problems. EfficientNet-B5 is a modern state-of-the-art architecture whose meta-architecture and scaling properties were derived from architecture search considerations (Tan & Le, 2019) (but see Bello et al. (2021)), and provides the visual backbone for the ALIGN model (Jia et al., 2021).

We experimented with this model and find that the image featurization from a CNN-derived backbone achieves favorable results with respect to previously published ViT models on localization problems. Table 2 showcases higher mIoU for semantic segmentation than a model with a ViT backbone (CLIP) on ADE20K, COCO and Pascal VOC when trained on the same dataset. Likewise, previous results

published on ALIGN with an EfficientNet-B5 backbone achieved superior results to a ViT model on COCO and ADE20K in terms of semantic segmentation⁵.

Most importantly, in spite of many attempts, we were not able to improve the performance of the EfficientNet-B5 architecture for localization. The best results for a CNN-based architecture we achieved are shown in Table 2, which are notably below previously published results and our best results with a ViT architecture. At best, the addition of maximum pooling at the top layer of a CNN led to marginal gains in terms of mIoU or Jaccard similarity. We suspect that a custom architecture (such as ASPP or FPN) may improve these results further (Chen et al., 2017; Lin et al., 2017), but we consider this out of scope as we are attempting to learn a featurization that does not artificially increase the parameterization in order to solve a specialized task of localization. We suspect that this limitation of CNN architectures may reflect the fact that CNN architecture already have learned a representation that is heavily dependent on the spatial geometry derived from a convolutional kernel. Such an inductive bias may be not provide a suitable mechanism for providing global processing in a segmentation task (Wang et al., 2018).

E PRIOR WORK ON ZERO SHOT SEMANTIC SEGMENTATION

Recent work has made impressive strides on zero-shot semantic segmentation. These works focus on training such models on subsets of segmentation data, whether masks and/or labels and test the performance of the resulting model on other splits of data. The zero-shot performance is assessed by splitting the labeled datasets to ensure that the model is tested in a zero-shot manner on unseen labels. Table 4 summarizes results from several recent papers (Ghiasi et al., 2022; Li et al., 2022; Xian et al., 2019; Bucher et al., 2019).

We note that particularly later forms of models achieve results superior to those presented here (Ghiasi et al., 2022; Li et al., 2022), but we emphasize several important distinctions. First, these models were trained on segmentation masks in order to learn perceptual grouping across visual imagery. Our work instead addresses how a model may learn this information without being explicitly supplied examples teaching such behavior. Second, most of the models were trained using segmentation masks from the COCO dataset. Hence, these models might perform particularly well on this dataset making comparisons on COCO less comparable to our model.

	segment label?	segment mask?	ADE20K	COCO	PASCAL VOC
SPNet ^a	✓	✓			18.3
ZS3Net ^b	✓	✓			38.3
LSeg ^c	✓	✓		27.2	47.4
LSeg+ ^d	✓	✓	18.0	55.1 [†]	59.0
ALIGN w/ proposal ^d		✓	12.9	17.9 [†]	22.4
OpenSeg ^d		✓	21.1	36.1 [†]	70.2
OpenSeg + Narr. ^d		✓	24.8	38.1 [†]	72.2

Table 4: **Performance of prior zero-shot segmentation models trained on segmentation data.** All numbers report the mIoU for semantic segmentation on ADE20K (150 labels), PASCAL-VOC (20 labels) and COCO (50 labels). [†] indicates models that were trained on image segmentation masks from the COCO dataset. Superscript letter denote the result: ^a Xian et al. (2019), ^b Bucher et al. (2019), ^c Li et al. (2022) and ^d Ghiasi et al. (2021).

⁵We note that the previously published ALIGN results were based on a backbone trained with a much larger dataset (1.8B image-text pairs), and it was evaluated at a higher resolution of 640×640 pixels, resulting in a zero-shot image recognition performance of 76.4. In comparison, our baseline and proposed models operate at 224×224 resolution and was trained on a 10× smaller dataset.

F PRIOR WORK ON UNSUPERVISED SEGMENTATION

Unsupervised segmentation attempts to group semantically related concepts using only pixel information. The emergence of multiple self-supervised learning approaches (Caron et al., 2021) successful at various tasks has led to numerous unsupervised segmentation methods building off them (Hamilton et al., 2022; Cho et al., 2021; Gansbeke et al., 2021; Ji et al., 2019). These methods train and operate with no semantic labels, performing bottom-up grouping of image content. A common characteristic is the presence of iterative clustering mechanisms (e.g. K-Means clustering (MacQueen, 1967)). STEGO exhibits (Hamilton et al., 2022) notable performance focusing on better inference strategies, by employing K-Means clustering and Conditional Random Field (CRF) (Krähenbühl & Koltun, 2011) based refinement.

We summarize results from several recent papers in Table 5. While CLIPpy exhibits competitive performance, we also note that it is possible to plug-in certain proposed techniques from methods such as (Hamilton et al., 2022) over the CLIPpy representations to further refine generated segmentations. We also reiterate how CLIPpy does not leverage any architectural components specific for segmentation in contrast to the other approaches such as (Hamilton et al., 2022).

Method	JS
IIC (Ji et al., 2019)	6.4
MDC (Cho et al., 2021)	7.1
PiCIE (Cho et al., 2021)	12.3
STEGO (Hamilton et al., 2022)	21.0
CLIPpy (ours)	18.1

Table 5: **Performance of prior unsupervised segmentation models.** The Jaccard Similarity (JS) on the Cityscapes Dataset 27 class segmentation setup followed in (Hamilton et al., 2022) is reported. We note how CLIPpy exhibits competitive performance even against models containing architectural components specialized for segmentation such as CRFs.

G ADDITIONAL AGGREGATION STRATEGIES

Visual Modality. We explored a range of alternate aggregation strategies that performed subpar to spatial max pooling that is utilized in CLIPpy. Two noteworthy approaches include Text Similarity Pooling (TSP) and Weighted Maximum Pooling (WMP). In TSP, we measure the similarity of each spatial token (corresponding to different positions), obtain a normalized distribution using a softmax operation (with a temperature hyper-parameter for smoothing), and aggregate the visual modality using a weighted averaging operation (where the similarity of each spatial location to the text is the weight). This same idea is used in WMP, but using per-channel per-location embedding values instead of similarity as weights for averaging.

Results for these experiments are presented in Table 7. We note that TSP performs poorly across all variations while WMP works better for lower temperature values. Given the common softmax operation, higher temperature values result in smoother weights for both cases, making them more similar to average pooling. In the case of WMP, lower temperature values make the operation similar to simple spatial maximum pooling, which is reflected in the improved results for lower temperature values.

Language Modality. We also explore how pooling on the text embedding affects overall performance. We replace the default average pooling with a maximum pooling operation to discover drops in performance across all metrics. These results are presented in Table 6. The poor performance of maximum pooling based aggregation for the language modality is consistent with prior works exploring similar aggregation mechanisms (Harwath et al., 2019).

Method	IN	VOC	COCO	ADE-20K
Avg	45.3	50.8	23.8	13.1
Max	42.0	42.6	18.9	10.5

Table 6: **Alternate Pooling Strategies for Text Modality.** Unlike the visual modality, performance drops when replacing the default average pooling (Avg) with maximum pooling (Max).

Method	temp	IN	VOC	COCO	ADE-20K
TSP	0.1	0	2.98	0	0
TSP	1.0	20.15	4.91	1.25	0
TSP	10	24.70	15.83	4.27	2.29
Max		27.05	37.39	17.32	9.69

Method	temp	IN	VOC	COCO	ADE-20K
WMP	0.1	28.36	31.12	13.64	8.12
WMP	1.0	0	0	0	0
WMP	10	0	3.53	0	0
Max		27.05	37.39	17.32	9.69

Table 7: **Alternate Pooling Strategies for Visual Modality.** We report results for TSP and WMP aggregation techniques with models trained for lesser steps on CC12M with no initialization. Max refers to the spatial max operation used in CLIPpy. The temperature value of the softmax operation for TSP / WMP is indicated by *temp*.

H ADDITIONAL ABLATION STUDIES

dataset	image init	T5 init?	ImageNet accuracy	Pascal VOC mIoU	Jaccard
HQITP-134M	DINO	✓	59.0	50.1	54.6
	IN-1K	✓	63.6	21.7	40.2
	random	✓	49.4	37.3	46.3
	DINO		55.2	46.9	53.7
	IN-1K		60.6	20.9	39.1
	random		46.4	37.1	45.8

Table 8: **Additional ablation studies with HQITP-134M.** Parallel results for Table 3 (right) for ablations on weight initialization. We observe similar trends in results across these experiments.

Self-supervised pre-training of the vision head of CLIPpy leads to notable performance gains across tasks as illustrated in 3. We explore how alternate supervised pre-training affects performance. In particular, we pre-train the image backbone using ImageNet-1K in a fully-supervised setting, and use the backbone (without the class-specific linear head) to initialize CLIPpy for the weakly supervised Image-Text training. We present these results in Table 8. Apart from pre-training, all other hyper-parameters are kept constant across experiments. In terms of the visual backbone, in CLIPpy it is pre-trained in a self-supervised setting while in ViT-B/16 it is pre-trained fully-supervised. We note that supervised pre-training leads to considerable performance boosts for ImageNet-1K top-1 accuracy, but results in considerable performance drops in mIoU across all three datasets.

I DETAILS OF ROBUSTNESS ANALYSIS

We calculate the zero-shot prediction of the class in a non-standard manner to exploit the spatial reasoning of CLIPpy. We apply the same zero-shot evaluation to the baseline CLIP model. Specifically, we first calculate the embedding for all labels within each category of *waterbird*, *landbird* and *background*. For each of these categories we calculate the average image embeddings across these labels at each spatial location.

To exploit the spatial knowledge of the model, we focus our analysis on all spatial locations which are *not* labeled as *background*. For all locations which maximally predict a *waterbird*, we calculate the similarity to the embedding for a *waterbird*. Likewise, we do the same for all locations maximized by *landbird*. Finally, our resulting prediction is the class that is closest to its associated embedding.

In order to calculate the target embeddings, we employ the following set of three prompts. The prompts for *waterbirds* and *landbirds* follow Sagawa et al. (2019). The *background* class prompts are derived from the prescribed backgrounds used for synthesizing the dataset. For a *water* background, the backgrounds used are *ocean* or *natural lake*. For a *land* background, the categories used are *bamboo forest* or *broadleaf forest*. See also Welinder et al. (2010).

- *background*: background
- *waterbird*: Black footed Albatross, Laysan Albatross, Sooty Albatross, Crested Auklet, Least Auklet, Parakeet Auklet, Rhinoceros Auklet, Brandt Cormorant, Red faced Cormorant, Pelagic Cormorant, Frigatebird, Northern Fulmar, Gadwall, Eared Grebe, Horned Grebe, Pied billed Grebe, Western Grebe, Pigeon Guillemot, California Gull, Glaucous winged Gull, Heermann Gull, Herring Gull, Ivory Gull, Ring billed Gull, Slaty backed Gull, Western Gull, Long tailed Jaeger, Pomarine Jaeger, Red legged Kittiwake, Pacific Loon, Mallard, Hooded Merganser, Red breasted Merganser, Brown Pelican, White Pelican, Horned Puffin, Artic Tern, Black Tern, Caspian Tern, Common Tern, Elegant Tern, Forsters Tern, Least Tern

- **landbird:** Groove billed Ani, Brewer Blackbird, Red winged Blackbird, Rusty Blackbird, Yellow headed Blackbird, Bobolink, Indigo Bunting, Lazuli Bunting, Painted Bunting, Cardinal, Spotted Catbird, Gray Catbird, Yellow breasted Chat, Eastern Towhee, Chuck will Widow, Bronzed Cowbird, Shiny Cowbird, Brown Creeper, American Crow, Fish Crow, Black billed Cuckoo, Mangrove Cuckoo, Yellow billed Cuckoo, Gray crowned Rosy Finch, Purple Finch, Northern Flicker, Acadian Flycatcher, Great Crested Flycatcher, Least Flycatcher, Olive sided Flycatcher, Scissor tailed Flycatcher, Vermilion Flycatcher, Yellow bellied Flycatcher, American Goldfinch, European Goldfinch, Boat tailed Grackle, Blue Grosbeak, Evening Grosbeak, Pine Grosbeak, Rose breasted Grosbeak, Anna Hummingbird, Ruby throated Hummingbird, Rufous Hummingbird, Green Violetear, Blue Jay, Florida Jay, Green Jay, Dark eyed Junco, Tropical Kingbird, Gray Kingbird, Belted Kingfisher, Green Kingfisher, Pied Kingfisher, Ringed Kingfisher, White breasted Kingfisher, Horned Lark, Western Meadowlark, Mockingbird, Nighthawk, Clark Nutcracker, White breasted Nuthatch, Baltimore Oriole, Hooded Oriole, Orchard Oriole, Scott Oriole, Ovenbird, Western Wood Pewee, Sayornis, American Pipit, Whip poor Will, Common Raven, White necked Raven, American Redstart, Geococcyx, Loggerhead Shrike, Great Grey Shrike, Baird Sparrow, Black throated Sparrow, Brewer Sparrow, Chipping Sparrow, Clay colored Sparrow, House Sparrow, Field Sparrow, Fox Sparrow, Grasshopper Sparrow, Harris Sparrow, Henslow Sparrow, Le Conte Sparrow, Lincoln Sparrow, Nelson Sharp tailed Sparrow, Savannah Sparrow, Seaside Sparrow, Song Sparrow, Tree Sparrow, Vesper Sparrow, White crowned Sparrow, White throated Sparrow, Cape Glossy Starling, Bank Swallow, Barn Swallow, Cliff Swallow, Tree Swallow, Scarlet Tanager, Summer Tanager, Green tailed Towhee, Brown Thrasher, Sage Thrasher, Black capped Vireo, Blue headed Vireo, Philadelphia Vireo, Red eyed Vireo, Warbling Vireo, White eyed Vireo, Yellow throated Vireo, Bay breasted Warbler, Black and white Warbler, Black throated Blue Warbler, Blue winged Warbler, Canada Warbler, Cape May Warbler, Cerulean Warbler, Chestnut sided Warbler, Golden winged Warbler, Hooded Warbler, Kentucky Warbler, Magnolia Warbler, Mourning Warbler, Myrtle Warbler, Nashville Warbler, Orange crowned Warbler, Palm Warbler, Pine Warbler, Prairie Warbler, Prothonotary Warbler, Swainson Warbler, Tennessee Warbler, Wilson Warbler, Worm eating Warbler, Yellow Warbler, Northern Waterthrush, Louisiana Waterthrush, Bohemian Waxwing, Cedar Waxwing, American Three toed Woodpecker, Pileated Woodpecker, Red bellied Woodpecker, Red cockaded Woodpecker, Red headed Woodpecker, Downy Woodpecker, Bewick Wren, Cactus Wren, Carolina Wren, House Wren, Marsh Wren, Rock Wren, Winter Wren, Common Yellowthroat

J DETAILS OF PROMPTS FOR ZERO-SHOT SEGMENTATION

We employed the following prompts for probing our vision-language models for zero-shot semantic segmentation. These prompts were copied from the corresponding label sets of each dataset with some basic considerations, for instance, restoring spaces in compound words. If the prompts is separated by a comma, then the average embedding across all prompts delineated by a comma is associated with each label.

Pascal VOC 2012 (Everingham et al. [2010])

- | | |
|---|--|
| 1. background: background, crops, bush, shrub, tiles, pavement, rug, carpet, box, boxes, speaker, storage, painting, board, panel, poster, clock, cage, drinking glass, park, plaything, toy, fireplace, bag, bag, bed, bench, book, books, building, buildings, cabinet, drawer, ceiling, computer, computer case, cup, cups, door, fence, floor, flower, grass, lawn, turf, ground, soil, dirt, tiles, keyboard, lamp, mountain, hills, mouse, curtain, platform, sign, street, rock, stone, shelf, sidewalk, sky, clouds, snow, track, train track, tree, trees, wall, water, window, wood, woods | 9. cat: cat, cats, kitties, kitty |
| 2. aeroplane: aeroplane, airplane, aeroplanes, airplanes | 10. chair: chair, chairs |
| 3. bicycle: bicycle, bicycles, bike, bikes | 11. cow: cow, cows, calf |
| 4. bird: bird, birds | 12. diningtable: diningtable, dining table, diningtables, dining tables, plate, plates |
| 5. boat: boat, boats | 13. dog: dog, dogs, puppy, puppies |
| 6. bottle: bottle, bottles, water bottle | 14. horse: horse, horses, foal |
| 7. bus: bus, buses | 15. motorbike: motorbike, motorcycle, motorbikes, motorcycles |
| 8. car: car, cars | 16. person: person, child, girl, boy, woman, man, people, children, girls, boys, women, men, lady, guy, ladies, guys, clothes |
| | 17. pottedplant: pottedplant, pottedplants, plant pot, plant pots, planter, planters, potted plant |
| | 18. sheep: sheep |
| | 19. sofa: sofa, sofas |
| | 20. train: train, trains, locomotive, locomotives, freight train |
| | 21. tvmonitor: tvmonitor, monitor, tv, television, television monitor |

COCO 2017 (Lin et al., 2014)

- | | |
|-----------------------------------|----------------------------------|
| 1. airplane: airplane | 41. kite: kite |
| 2. apple: apple | 42. knife: knife |
| 3. backpack: backpack | 43. laptop: laptop |
| 4. banana: banana | 44. microwave: microwave |
| 5. baseball bat: baseball bat | 45. motorcycle: motorcycle |
| 6. baseball glove: baseball glove | 46. mouse: mouse |
| 7. bear: bear | 47. orange: orange |
| 8. bed: bed | 48. oven: oven |
| 9. bench: bench | 49. parking meter: parking meter |
| 10. bicycle: bicycle | 50. person: person |
| 11. bird: bird | 51. pizza: pizza |
| 12. boat: boat | 52. potted plant: potted plant |
| 13. book: book | 53. refrigerator: refrigerator |
| 14. bottle: bottle | 54. remote: remote |
| 15. bowl: bowl | 55. sandwich: sandwich |
| 16. broccoli: broccoli | 56. scissors: scissors |
| 17. bus: bus | 57. sheep: sheep |
| 18. cake: cake | 58. sink: sink |
| 19. car: car | 59. skateboard: skateboard |
| 20. carrot: carrot | 60. skis: skis |
| 21. cat: cat | 61. snowboard: snowboard |
| 22. cell phone: cell phone | 62. spoon: spoon |
| 23. chair: chair | 63. sports ball: sports ball |
| 24. clock: clock | 64. stop sign: stop sign |
| 25. couch: couch | 65. suitcase: suitcase |
| 26. cow: cow | 66. surfboard: surfboard |
| 27. cup: cup | 67. teddy bear: teddy bear |
| 28. dining table: dining table | 68. tennis racket: tennis racket |
| 29. dog: dog | 69. tie: tie |
| 30. donut: donut | 70. toaster: toaster |
| 31. elephant: elephant | 71. toilet: toilet |
| 32. fire hydrant: fire hydrant | 72. toothbrush: toothbrush |
| 33. fork: fork | 73. traffic light: traffic light |
| 34. frisbee: frisbee | 74. train: train |
| 35. giraffe: giraffe | 75. truck: truck |
| 36. hair drier: hair drier | 76. tv: tv |
| 37. handbag: handbag | 77. umbrella: umbrella |
| 38. horse: horse | 78. vase: vase |
| 39. hot dog: hot dog | 79. wine glass: wine glass |
| 40. keyboard: keyboard | 80. zebra: zebra |

ADE-20K (150 frequent labels) (Zhou et al., 2018)

- | | |
|---|---|
| 1. airplane: airplane, aeroplane, plane | 14. basket: basket, handbasket |
| 2. animal: animal, animate, being, beast, brute, creature, fauna | 15. bathtub: bathtub, bathing, tub, bath, tub |
| 3. apparel: apparel, wearing, apparel, dress, clothes | 16. bed: bed |
| 4. arcade: arcade, machine | 17. bench: bench |
| 5. armchair: armchair | 18. bicycle: bicycle, bike, wheel, cycle |
| 6. ashcan: ashcan, trash, can, garbage, can, waste-bin, ash, bin, ash-bin, ashbin, dustbin, trash, barrel, trash, bin | 19. blanket: blanket, cover |
| 7. awning: awning, sunshade, sunblind | 20. blind: blind, screen |
| 8. bag: bag | 21. boat: boat |
| 9. ball: ball | 22. book: book |
| 10. bannister: bannister, banister, balustrade, balusters, handrail | 23. bookcase: bookcase |
| 11. bar: bar | 24. booth: booth, cubicle, stall, kiosk |
| 12. barrel: barrel, cask | 25. bottle: bottle |
| 13. base: base, pedestal, stand | 26. box: box |
| | 27. bridge: bridge, span |
| | 28. buffet: buffet, counter, sideboard |
| | 29. building: building, edifice |
| | 30. bulletin: bulletin, board, notice, board |

31. bus: bus, autobus, coach, charabanc, double-decker, jitney, motorbus, motorcoach, omnibus, passenger, vehicle
32. cabinet: cabinet
33. canopy: canopy
34. car: car, auto, automobile, machine, motorcar
35. case: case, display, case, showcase, vitrine
36. ceiling: ceiling
37. chair: chair
38. chandelier: chandelier, pendant, pendent
39. chest: chest of drawers, chest, bureau, dresser
40. clock: clock
41. coffee: coffee, table, cocktail, table
42. column: column, pillar
43. computer: computer, computing, machine, computing, device, data, processor, electronic, computer, information, processing, system
44. conveyer: conveyer, belt, conveyor, belt, conveyor, transporter
45. counter: counter
46. countertop: countertop
47. cradle: cradle
48. crt: crt, screen
49. curtain: curtain, drape, drapery, mantle, pall
50. cushion: cushion
51. desk: desk
52. dirt: dirt, track
53. dishwasher: dishwasher, dish, washer, dishwasher, washing, machine
54. door: door, double, door
55. earth: earth, ground
56. escalator: escalator, moving, staircase, moving, stairway
57. fan: fan
58. fence: fence, fencing
59. field: field
60. fireplace: fireplace, hearth, open, fireplace
61. flag: flag
62. floor: floor, flooring
63. flower: flower
64. food: food, solid, food
65. fountain: fountain
66. glass: glass, drinking, glass
67. grandstand: grandstand, covered, stand
68. grass: grass
69. hill: hill
70. hood: hood, exhaust, hood
71. house: house
72. hovel: hovel, hut, hutch, shack, shanty
73. kitchen: kitchen, island
74. lake: lake
75. lamp: lamp
76. land: land, ground, soil
77. light: light, light, source
78. microwave: microwave, microwave, oven
79. minibike: minibike, motorbike
80. mirror: mirror
81. monitor: monitor, monitoring, device
82. mountain: mountain, mount
83. ottoman: ottoman, pouf, pouffe, puff, hassock
84. oven: oven
85. painting: painting, picture
86. palm: palm, palm, tree
87. path: path
88. person: person, individual, someone, somebody, mortal, soul
89. pier: pier, wharf, wharfage, dock
90. pillow: pillow
91. plant: plant, flora, plant, life
92. plate: plate
93. plaything: plaything, toy
94. pole: pole
95. pool: pool, table, billiard, table, snooker, table
96. poster: poster, posting, placard, notice, bill, card
97. pot: pot, flowerpot
98. radiator: radiator
99. railing: railing, rail
100. refrigerator: refrigerator, icebox
101. river: river
102. road: road, route
103. rock: rock, stone
104. rug: rug, carpet, carpeting
105. runway: runway
106. sand: sand
107. sconce: sconce
108. screen: screen, door, screen
109. screen: screen, silver, screen, projection, screen
110. sculpture: sculpture
111. sea: sea
112. seat: seat
113. shelf: shelf
114. ship: ship
115. shower: shower
116. sidewalk: sidewalk, pavement
117. signboard: signboard, sign
118. sink: sink
119. sky: sky
120. skyscraper: skyscraper
121. sofa: sofa, couch, lounge
122. stage: stage
123. stairs: stairs, steps
124. stairway: stairway, staircase
125. step: step, stair
126. stool: stool
127. stove: stove, kitchen, stove, range, kitchen, range, cooking, stove
128. streetlight: streetlight, street, lamp
129. swimming: swimming, pool, swimming, bath, natatorium
130. swivel: swivel, chair
131. table: table
132. tank: tank, storage, tank
133. television: television, television, receiver, television, set, tv, tv, set, idiot, box, boob, tube, telly, goggle, box
134. tent: tent, collapsible, shelter
135. toilet: toilet, can, commode, crapper, pot, potty, stool, throne
136. towel: towel
137. tower: tower
138. trade: trade, name, brand, name, brand, marque
139. traffic: traffic, light, traffic, signal, stoplight
140. tray: tray
141. tree: tree
142. truck: truck, motortruck
143. van: van
144. vase: vase
145. wall: wall
146. wardrobe: wardrobe, closet, press

147. washer: washer, automatic, washer, washing, ma-149. waterfall: waterfall, falls
chine
148. water: water
150. windowpane: windowpane, window