Contributions

KR led the project by performing most of experiments, evaluations and visualizations. BM helped build the codebase, debugged many issues, worked on the evaluations, and discussed many of the pooling methods explored. SR ran early experiments on localization datasets to explore potential applications in transfer learning and detection. YY discussed all aspects of the project idea and scope, built and trained the initial CLIP implementation, and contributed many ideas to the pooling and initialization. AT advised on the project, proposed many pretraining and pooling experiments, analyzed results, contributed to writing and reviewed code. JS organized the project, set the research direction, discussed all aspects of the project idea and scope, and wrote much of the paper.

A. Additional Quantitative Results

In this section, we present additional quantitative evaluations for better understanding our experimental outcomes. We first extend our robustness analysis, followed by additional ablations on aggregation and pretraining strategies.

A.1. Robustness Analysis

We explore the open-source CLIP model from [85], evaluating its performance on the waterbirds dataset for signs of spurious correlations learned by the model. These results presented in Tab. 11 illustrate how classification performance for the same set of foreground objects drops drastically against different conflicting backgrounds, even in spite of the task involving fine-grained categories of birds.

CLIP [85]	water	land	Δ
waterbird	88.3	70.1	**-18.2**
landbird	90.5	99.1	**-8.6**

Table 11: Waterbirds evaluation for OpenAI CLIP model. We demonstrate how even a CLIP model trained on significantly more data (400M image-text pairs) contains stronger sensitivty to spurious correlations than CLIPpy trained with an order of magnitude less data (12M). The first two columns report top-1 classification accuracy (%) and the last column reports difference of diagonal and off-diagonal terms (delta). Higher spurious correlations increase the absolute value of delta.

A.2. Alternate Aggregation Strategies

Having explored two standard visual aggregation techniques as baselines, we ask how aggregation on the textual encoder affects CLIPpy performance. In detail, we first explore maximum pooling for the language modality. Thereafter, we draw further attention to the visual modality, exploring more complex pooling strategies.

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

Language Modality. In Tab. 12, we explore how pooling on the text embedding affects overall performance. We replace default average pooling with a maximum pooling operation to discover drops in performance across all metrics. This poor performance of maximum pooling based aggregation for language is consistent with prior works [42]. We hypothesize the reasoning as the nature of our task: while we attempt a localization across the visual modality, on the language end we reason with the entire text prompt as a single unit.

Visual Modality. We explored a range of alternate aggregation strategies that performed subpar to spatial max pooling 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) and obtain a normalized distribution using a softmax operation (including a temperature for smoothing). We aggregate the visual modality using a weighted averaging operation where the similarity of each spatial location to the text is the weight. WMP follows the same idea but employs per-channel per-location embedding values instead of similarity as weights for the aggregation operation. Tab. 14 shows results for TSP, WMP and max pooling. TSP performs poorly across all variations while WMP works better at lower temperatures. Given the common softmax operation, higher temperature values result in smoother weights for both cases, making WMP and TSP more similar to average pooling. In the case of WMP, lower temperatures make the operation similar to simple spatial maximum pooling, which is reflected in the improved results for lower temperature values.

A.3. Pretraining Ablations

Self-supervised pretraining of the vision head of CLIPpy leads to notable performance gains. We also explore how alternate supervised pretraining affects performance. In particular, we pretrain the image backbone using ImageNet-1K

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. Note that colors correspond to clusters and *not* semantic labels.

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

in a fully-supervised setting, and use the penultimate features to initialize CLIPpy. Apart from pretraining, all hyperparameters are unchanged. We present these results in Tab. 13. We observe when training with HQITP-134M that supervised pretraining leads to considerable performance gains for ImageNet-1K top-1 accuracy, but a considerable drop in segmentation performance across all three datasets. However, we note that supervised ImageNet pre-training provides unfair advantage in the case of ImageNet accuracy, in particular given the ability of overfitting those visual concepts. So we focus more on the segmentation results. Interestingly, entirely eliminating visual pre-training, while degrading segmentation performance, outperforms the ImageNet supervised pre-training initialized model. This reaffirms our hypothesis of better generalization of self-supervised features for segmentation tasks. We also note that results on HQITP-134M indicate trends similar to those with CC-12M.

B. Qualitative examples of localization

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

Fig. 5 shows examples of bottom-up unsupervised segmentation on PASCAL VOC for CLIPpy. The left three examples highlight the strength of the method for images with less clutter in which there is a single object of interest. The right three examples shows examples highlights the failures in the presence of scene clutter.

Fig. 6 present additional top-down semantic segmentation examples across all three datasets used for evaluation. The examples from PASCAL VOC dataset are the same examples from Fig. 5. For PASCAL VOC, note that the contrast between the top-down and bottom-up segmentations. 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 may contain discontinuities within a single object region in some cases, especially for the background objects (columns 2-3).

COCO and ADE-20K provide more challenging datasets and highlight several potential failure modes. A notable failure mode for CLIPpy is to reverting to the baseline CLIP behaviour by predicting the salient object class at all locations. This is visible to some extent in column 2 & 3 in the COCO examples and column 3 in the ADE-20K examples. Additionally, CLIPpy results in false positives for cluttered scenes as visible in some examples of these two datasets.

In summary, CLIPpy is able to localize the salient objects well in less cluttered scenes, even when multiple objects belonging to different classes are present. CLIPpy is also able to coarsely localize some of the salient objects in more cluttered scenes. In terms of limitations, CLIPpy fails to

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 14: Alternate Pooling Strategies for Visual Modality. We report results for TSP and WMP with models trained for lesser steps on CC-12M with no initialization. Max refers to the spatial max operation used in CLIPpy. The temperature of the softmax operation for TSP and WMP is indicated by *temp*.

correctly localize some background classes, fails to correctly recognize object boundaries, and may miss smaller objects in cluttered scenes.

C. Details of HQITP-134M dataset

High Quality Image Text Pairs (HQITP-134M) consists of $\sim$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 (« 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.

D. Architecture and Training Details

Our implementations of CLIP and ALIGN employ ViT-B/16 [27], and EfficientNet-B5[100] 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 [12], but report all of our results with $P = 16$.

We train all models on 224×224 images to provide a fair comparison with [85]. Note however that the published version of ALIGN employed a 640×640 resolution. ViT models may operate on images at arbitrary spatial resolution. At inference time we experimented with spatial resolutions of 224×224 and 448×448, 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 Fig. 4).

We train models with 32 GPUs across 4 machines with PyTorch [80] using the LAMB optimizer [116] 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; 10 epochs for HQITP-134M). We employ a weight decay of 1e-2. All training parameters were determined through moderate hyperparameter tuning.

Patch Sub-Sampling: 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.

E. Limitations of CNN-based architectures

The primary focus of our presentation is on the Vision Transformer (ViT) architecture [27]. 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 [100] (but see [5]), and provides the visual backbone for the ALIGN model [50].

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 6 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 segmentation [5].

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 6, 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 [16, 63], 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 [107].

F. 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 15 summarizes results from several recent papers [36, 57, 111, 8]

We note that particularly later forms of models achieve results superior to those presented here [36, 57], 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.

[5]We 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.

G. Prior work on unsupervised segmentation

The task of unsupervised segmentation groups semantically related concepts using only pixel information. Recent advances in self-supervised learning (e.g. [12]) has led to numerous opportunities across tasks [61, 94], including for unsupervised segmentation or weakly-supervised localization tasks [41, 18, 31, 49, 93]. 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 [69]). Notably, STEGO [41] employs k-Means and Conditional Random Fields (CRF) [56] to refine and improve the inference procedure. We hope to apply CRFs (e.g. [47]) to CLIPpy in future. While CLIPpy exhibits competitive performance, we note that techniques proposed by other methods (e.g. [41]) may be leveraged (and combined with CLIPpy) to refine our segmentation performance. A key distinction of our work is the focus on learning a new representation space aligned to language that exhibits strong perceptual grouping.

H. 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.

We find that these results and the corresponding robustness vary substantially due to the selection of prompts for each of the three categories. This matches observations in [85]. In Sec. 4.4 we focus on the results of the model trained on CC-12M using the prompts listed below which contain a minimal amount of prompt engineering. In particular, the prompts for `waterbirds` and `landbirds` follow [91]. See also [108].

- `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

	segment label?	segment mask?	ADE20K	COCO	PASCAL VOC
SPNet [111]	✓	✓			18.3
ZS3Net [8]	✓	✓			38.3
LSeg [57]	✓	✓		27.2	47.4
LSeg+ [35]	✓	✓	18.0	55.1 †	59.0
ALIGN w/ proposal [35]		✓	12.9	17.9 †	22.4
OpenSeg [35]		✓	21.1	36.1 †	70.2
OpenSeg + Narr. [35]		✓	24.8	38.1 †	72.2

Table 15: 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.

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

I. 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. For prompts separated by a comma, the average embedding across all prompts delineated by commas is associated with each label.

Pascal VOC 2012 [30]

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
2. `aeroplane`: aeroplane, airplane, aeroplanes, airplanes
3. `bicycle`: bicycle, bicycles, bike, bikes
4. `bird`: bird, birds
5. `boat`: boat, boats
6. `bottle`: bottle, bottles, water bottle
7. `bus`: bus, buses
8. `car`: car, cars
9. `cat`: cat, cats, kitties, kitty
10. `chair`: chair, chairs
11. `cow`: cow, cows, calf
12. `diningtable`: diningtable, dining table, diningtables, dining tables, plate, plates
13. `dog`: dog, dogs, puppy, puppies
14. `horse`: horse, horses, foal
15. `motorbike`: motorbike, motorcycle, motorbikes, motorcycles
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, televison, television monitor

COCO 2017 [64]

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

ADE-20K (150 frequent labels) [125]

1. `airplane`: airplane, aeroplane, plane
2. `animal`: animal, animate, being, beast, brute, creature, fauna
3. `apparel`: apparel, wearing, apparel, dress, clothes
4. `arcade`: arcade, machine
5. `armchair`: armchair
6. `ashcan`: ashcan, trash, can, garbage, can, wastebin, ash, bin, ash-bin, ashbin, dustbin, trash, barrel, trash, bin
7. `awning`: awning, sunshade, sunblind
8. `bag`: bag
9. `ball`: ball
10. `bannister`: bannister, banister, balustrade, balusters, handrail
11. `bar`: bar
12. `barrel`: barrel, cask
13. `base`: base, pedestal, stand
14. `basket`: basket, handbasket
15. `bathtub`: bathtub, bathing, tub, bath, tub
16. `bed`: bed
17. `bench`: bench
18. `bicycle`: bicycle, bike, wheel, cycle
19. `blanket`: blanket, cover
20. `blind`: blind, screen
21. `boat`: boat
22. `book`: book
23. `bookcase`: bookcase

24. `booth`: booth, cubicle, stall, kiosk
25. `bottle`: bottle
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, conveyer, 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, dishwashing, 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, machine
148. `water`: water
149. `waterfall`: waterfall, falls
150. `windowpane`: windowpane, window

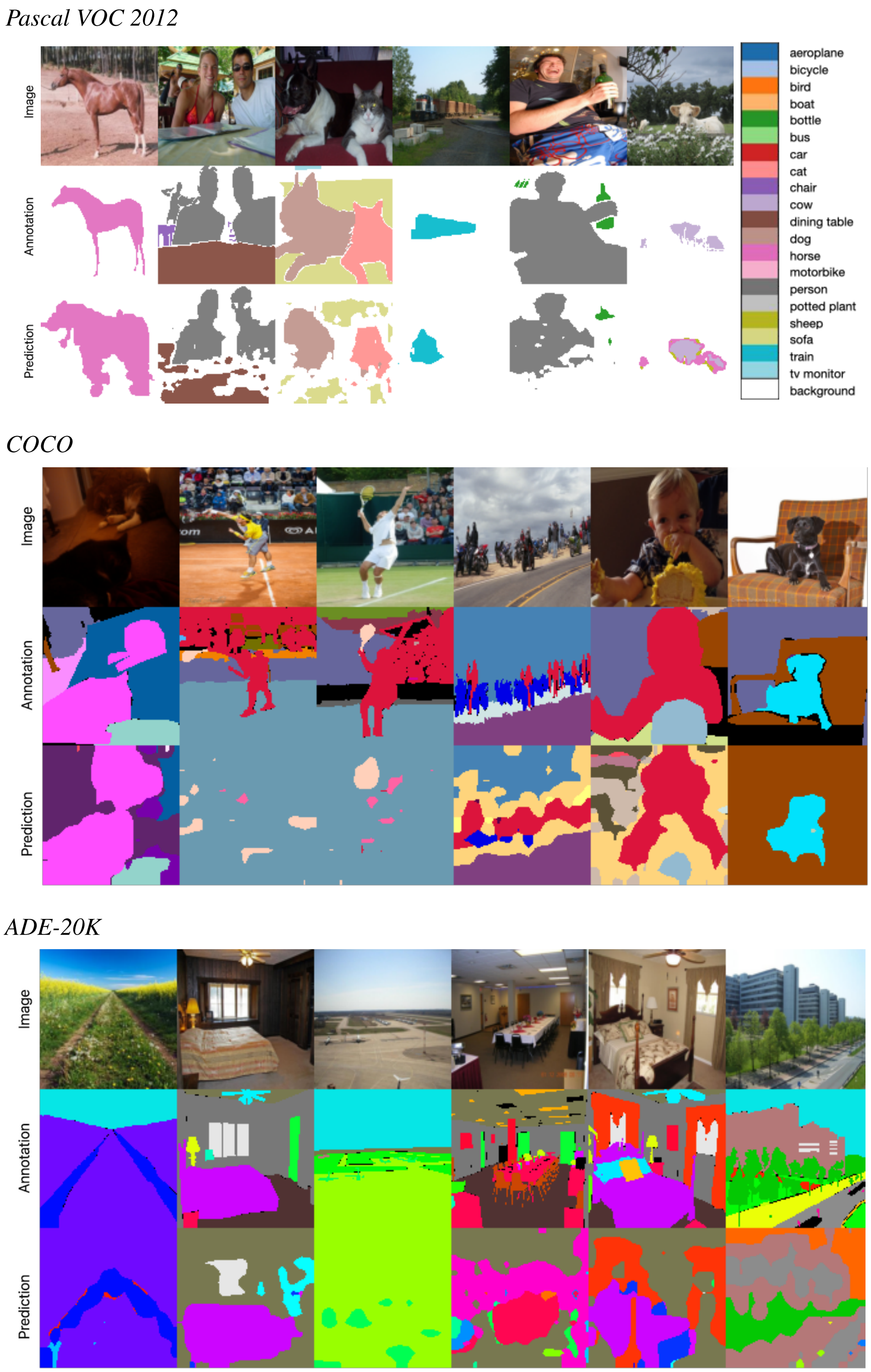

Figure 6: Qualitative examples of top-down semantic segmentation with CLIPpy from PASCAL VOC, COCO and ADE-20K. For Pascal VOC, we supply a color legend for the 20 label classes. Note that the Pascal VOC examples correspond to the same examples from Fig. 5.