FewSOL: A Dataset for Few-Shot Object Learning in Robotic Environments [Supplementary Material]

Jishnu Jaykumar P[1] **Yu-Wei Chao**[2] **Yu Xiang**[1]
[1]The University of Texas at Dallas [2]NVIDIA
{jishnu.p,yu.xiang}@utdallas.edu ychao@nvidia.com

Contents

1 Definitions

A **clean support set** (Clean S) consists of single objects with clean background. A **cluttered support set** (Cluttered S) consists of single objects or objects from a collection of objects in a cluttered scene with different backgrounds and occlusions. An **episodic testing** indicates using a number of episodes in testing where each episode consists of several support and query sets from different classes. A **non-episodic testing** indicates the classification of all the test images in the test set without using episodes. These images can be considered to be a whole query set.

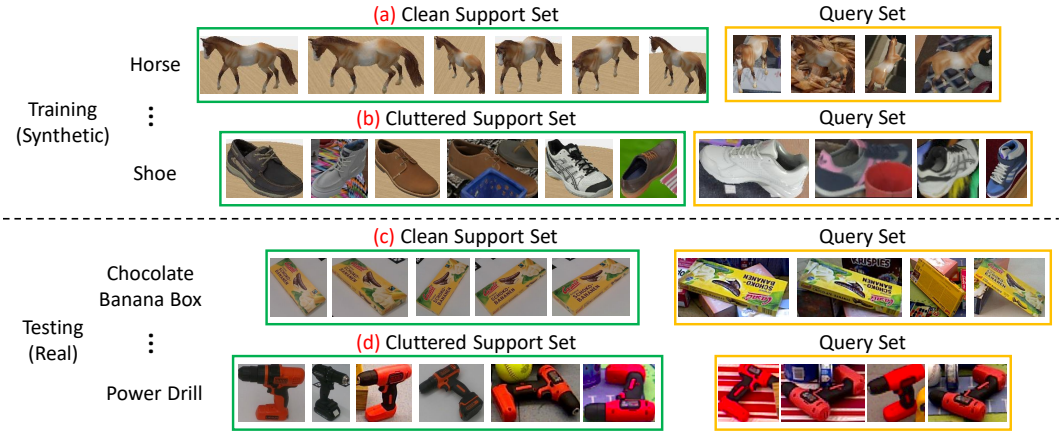


Figure 1: {a,c} and {b,d} are examples of clean and cluttered support sets of training and testing episodes, respectively.

2 Data and Evaluation Metrics

2.1 Data Preparation Pipeline

We follow the same data preparation pipeline devised in META-DATASET [1]. Raw FewSOL dataset is converted to the TFRecord[1] format. The data pipeline utilizes TFRecords to form episodes. The data pipeline randomly shuffles and selects entries from TFRecords to form the train and test episodes which are then used to train and test the respective models mentioned in Section 3.1.

2.2 Dataset Split

Train and Validation Sets. For few-shot classification and joint object segmentation and few-shot classification, training data comprised 125 synthetic object classes. The training data is split into train and validation sets based on a 90:10 ratio of classes. The classes were first sorted in descending order based on the number of support examples. The top 90% classes, i.e., 112 classes form the train set, and the last 10%, i.e., 13 classes form the validation set. This would allow using more variety of support examples for training and validation using fewer support examples which is an apt case for the real world as well. With this setup, it is clear that train and validation classes are disjoint.

Test Sets. For few shot classification, as shown in Table 1, four variants of the test data were generated: (i) **All** comprises of 52 test classes, (ii) **Unseen** comprises of 41 test classes (disjoint from training classes), (iii) **Seen** comprises of 11 test classes (common with training classes) and (iv) **Unseen (Synthetic)** comprises of 13 test classes (common with the validation set portion of the training data). For joint object segmentation and few-shot classification, only {All, Unseen, Seen} variants were used.

[1] https://www.tensorflow.org/tutorials/load_data/tfrecord

Variant	#Classes
All	52
Unseen	41
Seen	11
Unseen (Synthetic)	13

Table 1: Test data variants.

The Seen variant has 11 real object classes that coincide with synthetic object classes, hence a need to keep all 11 synthetic object classes in the train split came up. This is useful for studying the effect on these classes during episodic testing of the Seen variant. This need was catered by transferring any of the 11 synthetic object classes common with the Seen variant test classes from validation split to train split. The void of the transferred classes in the validation split was filled by the bottom classes of the initially sorted train split so that classes with fewer support samples stay in the validation split and the train split gets classes with more support samples which may aid in training the models. Fortunately, with the current setup of $112/13$ after sorting only one class had to be swapped. The last train split class **product_box** was swapped with **sponge** class from validation split. After all these steps, validation classes were product_box, eagle, wood_tower, cream_box, cake_pan, screwdriver, food_can, bottle, paper_roll, honey_dipper, racoon, hard_drive, and rubber_band. Furthermore, the last step of image filtration discussed in Section 2.3 was performed, i.e., all images with either width or height less than 15 pixels were discarded as it may cause the problem shown in Figure 2.

For the real-world setup, all real and synthetic data from the FewSOL dataset were used for training. Hence, the training data comprised 323 (125+198) classes. This forms a mixture of synthetic and real object classes. Since only 52 out of 198 classes had query sets, the query sets for the remaining 146 classes were generated by splitting the available support set images into two halves. One half was considered as the support set while the other formed the query set. A 90:10 split yielded 291 train and 32 validation classes. While these 32 validation classes have real images, there are similar 32 synthetic classes in 291 train classes. This is a good setup as the Sim-to-Real transfer effect could be utilized here. Thus, in this way train and validation classes are disjoint.

2.3 Data preprocessing

Discard Low Resolution Images. As shown in Figure 2, low-resolution images, i.e., images having very small width or height when resized to 84x84x3 or 126x126x3 yields blurred images with no useful feature to learn. Hence, we discard images with width or height less than 15 pixels to avoid including noises while training and testing (except during real-world testing with 198 classes and joint object segmentation and few-shot classification training and testing). The threshold for 15 pixels has been chosen with the heuristic that it is very small and going less than that may not help. Also, increasing this threshold might discard a lot of images that could be useful for training and testing.

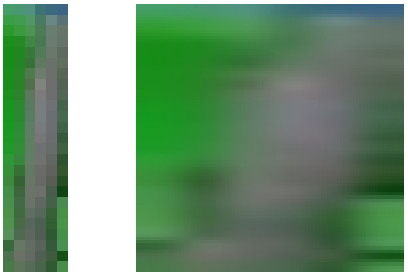


Figure 2: The left (low resolution and blurred) image shows an object after cropping from a cluttered scene while the right image shows the same when resized to 84x84x3 size. This may not help during training as all the images are reshaped to 84x84x3 or 126x126x3 before being fed into any model for few-shot learning.

Support Set Oversampling. This step is required for few shot classification only. As discussed in Section 2.1, the data pipeline randomly shuffles and selects entries from TFRecords to form episodes. This affects FewSOL dataset as it has fewer support samples w.r.t. query (refer Table 4 and 5). Hence, while performing the filtration discussed in Section 3.9, either no or fewer support images might be left. This is a case of set imbalance. Thus with the standard data pipeline of META-DATASET [1], the solution was to oversample the support images to match the cardinality of the query set. This oversampling only happens at the TFRecords level and not on the raw

Backbone	Accuracy	Image Size
four_layer_convnet	25.30%	84
four_layer_convnet	31.17%	126
resnet	33.20%	84
resnet	29.96%	126
resnet34	**34.44%**	84
resnet34	**33.76%**	126
resnet34 (max-stride=16)	**35.14%**	126

Table 2: Ablation study of image size w.r.t. few-shot classification accuracy for selected backbones using k-NN [1] as the learner model. Here, k-NN [1] is trained on ImageNet [6]. k-NN [1] was selected following the suggestion from META-DATASET [1] that some of the best meta-learning models are initialized from the weights of a batch baseline.

images. This gives a fair chance to the support samples to be selected alongside the query samples. Oversampling is done as $N_{ic} = \lceil Q_c/S_c \rceil$, where N_{ic} is the number of times to oversample the i^{th} support image in class C, S_c and Q_c are the number of available support and query images in class C, respectively.

2.4 Evaluation Metrics

For few shot classification similar to META-DATASET [1], we report 95% confidence intervals for classification accuracy with episodic testing consisting of 600 episodes. This is reasonable as variable "N-way K-shot" setup is used to construct test episodes. For joint object segmentation and few-shot classification along with real-world testing, top-1 and top-5 classification accuracy are reported on all query data while utilizing all support data.

3 Experiment Details

3.1 Models

We purposefully restricted our experiments to a selected set of few-shot and meta-learning methods from the growing list of META-DATASET [1], viz., k-NN [1], k-NN-Finetuned [1], Prototypical Networks [2], Matching Networks [3], Model Agnostic Meta-Learning (MAML) [4], Proto-MAML [1], CrossTransformers (CTX) [5] and CTX+SimCLR [5] variant due to the combination explosion of the experiments, hardware constraints and stability of META-DATASET [1] codebase. We think that these methods are a good representative sample to begin with.

3.2 Embedding Networks (Backbones)

Each model described in Section 3.1 needs a backbone to learn useful features. META-DATASET [1] offers a variety of backbone options: four layer convolutional network (four_layer_convnet), resnet18 (resnet), wide_resnet, and resnet34. All models use resnet34 backbone architecture except k-NN-Finetuned [1] which uses resnet due to GPU memory limit. Resnet34 was mainly chosen due to its superior performance during pretraining using ImageNet [6]. Resnet with image_size=84 gave on par performance w.r.t. Resnet34, thus it was the next substitute for k-NN-Finetuned [1]. CTX uses resnet34 with an extra setting of max_stride=16.

3.3 Backbone Pretraining

As in META-DATASET [1], we pretrain the backbones mentioned in Table 2 using ImageNet [6] for 50K updates. These backbones were selected based on the suggestions from META-DATASET [1]. An ablation study on the effect of image size on few shot classification accuracy of selected backbones is shown in Table 2. The accuracy reported is the mean component of the 95% confidence interval. Confidence interval is used mainly due to the variable "N-way K-shot" setup. It is clear from

Model	Backbone	Max Stride	Image Size	Num Training Updates
k-NN [1]	resnet34	-	126	75K
k-NN-Finetuned [1]	resnet	-	84	75K
ProtoNet [2]	resnet34	-	126	75K
MatchingNet [3]	resnet34	-	126	75K
fo-MAML [4]	resnet34	-	126	75K
fo-Proto-MAML [1]	resnet34	-	126	75K
CTX [5]	resnet34	16	126	100K
CTX+SimCLR [5]	resnet34	16	126	400K

Table 3: Details of hyperparameters for training and testing. Num Training Updates denotes the maximum number of updates that training process can perform.

Table 2 that resnet34 performs better. Thus it was selected as the backbone. Since CTX+SimCLR [5] performs better at higher resolution, we chose to go ahead with the image_size=126. We tried using image_size=224 but were constrained due to the GPU memory limit. To maintain uniformity and perform comparison across various methods, resnet34 was used as the backbone with image_size=126. The GPU memory limit in case of k-NN-Finetuned [1] made the option of going ahead with resnet as the backbone with image_size=84 as it performed on par with resnet34 (refer Table 2). The images are resized to (image_size x image_size) before the forward pass similar to META-DATASET [1].

3.4 Hyperparameters

The best hyperparameters reported for each model in META-DATASET [1] have been used for our experiments. The changes to the default hyperparameters related to backbones are discussed in Section 3.3. Following are the changes from our end: (i) Model training involving resnet34 backbone have learning_rate set as 0.001052178216688174, (ii) Number of validation episodes are set to 60 (default 600) taking into consideration the minimum need for constructing a confidence interval and the exceeding time for validation, (iii) EpisodeDescriptionConfig.max_support_size_contrib_per_class is set to 9 instead of default 500, (iv) EpisodeDescriptionConfig.max_ways_upper_bound is set to 52 and 198 for few shot classification and joint object segmentation and few-shot classification respectively.

3.5 Training Details

We have two training setups: (a) using 125 synthetic classes and (b) using all classes of the FewSOL dataset, i.e. 323 classes. The training setup (a) is further divided into two more variants based on the type of support set used for training: (i) using clean support sets and (ii) using cluttered support sets. Figure 1 shows the clean and cluttered support sets for few shot classification training and testing. Each support set setup is further divided into 2 types based on (i) using pretrained backbones and (ii) using backbones initialized randomly. Details of the backbones are mentioned in Section 3.2. We use the same protocol as in META-DATASET [1] for episode composition. We use the variable "N-way K-shot" setup for training. Training setup (a) is used to train models mentioned in Section 3.1.

Training setup (b) uses all synthetic and real data from the FewSOL dataset to train CTX+SimCLR [5]. Only cluttered support sets are used for training this particular model which would be used for real-world testing. Refer Section 4 for more details.

3.6 Selection of the Best Trained Checkpoints

As stated in META-DATASET [1], actual early stopping is not performed, in that training is not stopped early according to validation performance. Instead, checkpoints are recorded every 500^{th} update during training, and validation error is saved at these times. The checkpoint corresponding to the least validation error is chosen as the best one. This procedure is used for selecting the best-trained model checkpoints in all the experiments.

3.7 Testing Details

For the few shot classification experiment, we use the variable "N-way K-shot" setup as in META-DATASET [1] for constructing test episodes. We call this 'episodic testing'. Testing was done on all the variants using the models trained on 4 training setups discussed in Section 3.5. For joint object segmentation and few-shot classification experiment, we used all the query images from {All, Unseen, Seen} variants. Refer Section 2.2. We call this the 'non-episodic' setup. Few shot classification and joint object segmentation and few-shot classification testing share the same trained model checkpoints. They only differ in the type of testing. The former uses episodic whereas the latter uses non-episodic testing.

For real-world testing, we have setup an environment where a Fetch mobile manipulator is facing a desk containing various real-world objects mentioned in Section 4. We have 8 setups each containing 4 different objects. Fetch takes an image containing the objects on the desk then [7] is used for object segmentation. The object masks are then used to crop the detected objects. The cropped object images become the query set whereas the support set of 198 real classes forms the support set of the new real-world test set. This new real-world test set is tested using the model trained using training setup (b) (refer Section 3.5).

3.8 Experimental Hardware

All experiments were conducted on two NVIDIA RTX A5000 24GB GPUs. A Fetch mobile manipulator[2] was used for real-world testing.

3.9 Changes introduced in META-DATASET [1] code

All the images are first converted into tfrecords format. The default tfrecords are a collection of example protocol buffer[3] strings. Each example string consists of two elements, (i) image and (ii) label in bytes format. Meta-Dataset leverages data from 10 different datasets: ILSVRC-2012 (ImageNet [8]), Omniglot [9], Aircraft [10], CUB-200-2011 (Birds [11]), Describable Textures [12], Quick Draw [13], Fungi [14], VGG Flower [15], Traffic Signs [16] and MSCOCO [17]. As we can see, unlike the FewSOL dataset, these data are not divided into support and query sets specifically. Hence, we added a new key to each example string namely **set** which could take one value from {*support, query, ''*}. '' denotes that the particular example can belong to either support or query. This ensures the existing 10 datasets of META-DATASET [1] are compatible with the new protocol buffer structure. The default META-DATASET [1] pipeline shuffles and randomly selects a set of example strings which are then again randomly divided into support and query sets. This is reasonable as there is no bifurcation of support and query sets in the existing META-DATASET [1] datasets. But FewSOL dataset has a clear distinction between support and query and thus a need to select support and query data from the respective category was necessary. For this purpose, a perform_filtration feature is introduced. If set to boolean 'True', the pipeline filters the support and query sets so that the resulting data belongs to the respective categories. If set to boolean 'False', the standard META-DATASET [1] data pipeline is run.

Figure 3: Fetch mobile manipulator facing the objects from Set-1 (Figure 4a) on a table

[2] https://fetchrobotics.com/fetch-mobile-manipulator
[3] https://developers.google.com/protocol-buffers

(a) Set-1; mustard_bottle, water_bottle, jello_box, soup_can

(b) Set-2; soft_scrub_cleanser_bottle, tennis_ball, ball, cracker_box

(c) Set-3; cup, jello_box, meat_can, clock

(d) Set-4; tuna_can, air_duster_can, marker, knife

(e) Set-5; keyboard, game_controller, hand_sanitizer, mouse

(f) Set-6; wood_block, folder, sticky_notes, stapler

(g) Set-7; key, pen, book, headphone

(h) Set-8; mug, charger, cellphone, spoon

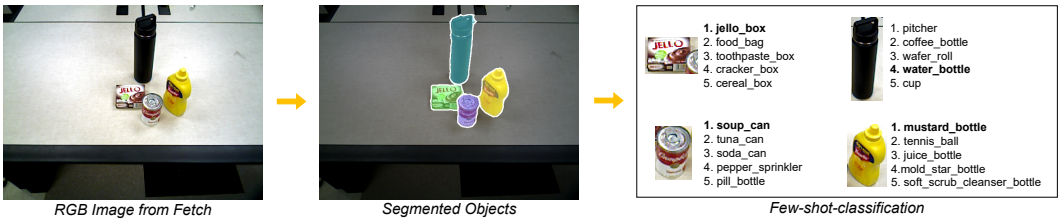
Figure 4: 8 sets, each containing 4 different real world objects

4 Qualitative Results in the Real World

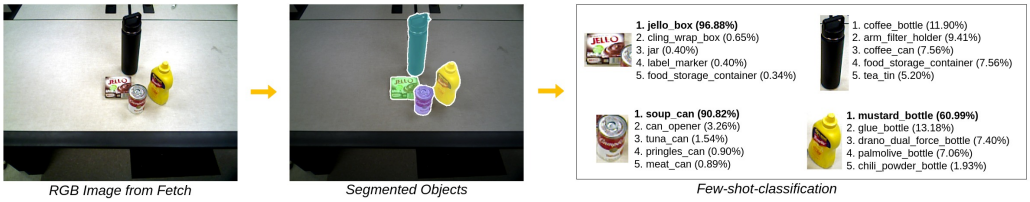
In this experiment, our goal is to build a few-shot classification model that works the best on real-world perception systems. So we train CTX+SimCLR [5] with all real and synthetic data from the FewSOL dataset and then test the trained model in our lab. Cluttered support sets are used for training. This setup is selected due to the superior performance of CTX+SimCLR [5] with cluttered support set training during joint object segmentation and few-shot classification experiment. Figure 3 shows fetch mobile manipulator facing objects from Set-1 (Figure 4a) on a table. RGB-D images are collected from a Fetch mobile manipulator and [7] is used for object segmentation. We tested on 32 objects with 4 objects in an image (refer Figure 4).

4.1 Real World Objects

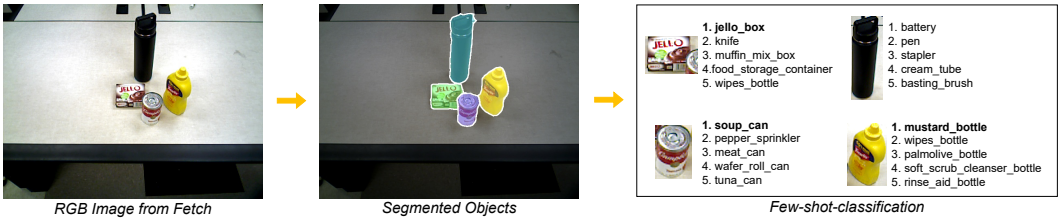
4.2 Joint Object Segmentation and Few-Shot Classification



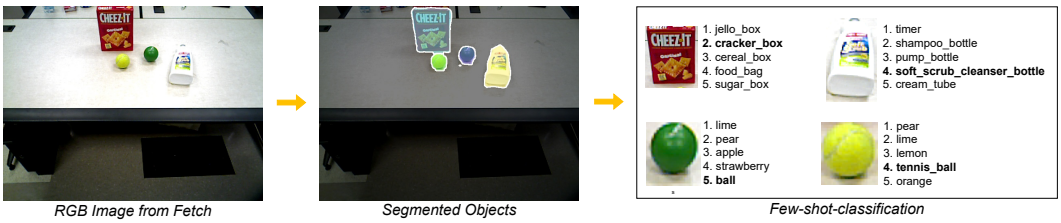
(a) Set-1 prediction using CTX+SimCLR [5]



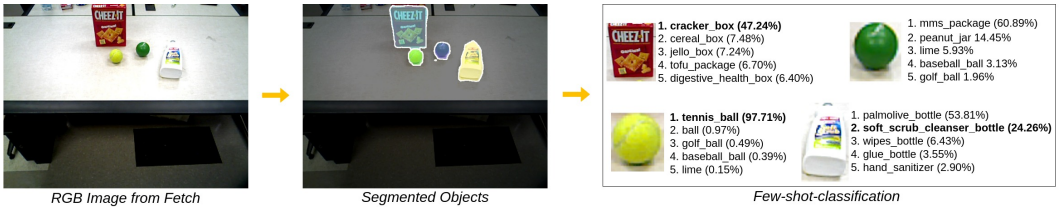
(b) Set-1 prediction using Zero-Shot-CLIP [18]



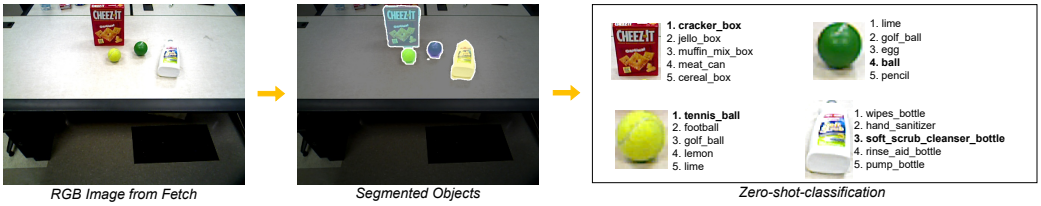
(c) Set-1 prediction using Tip-Adapter [19]



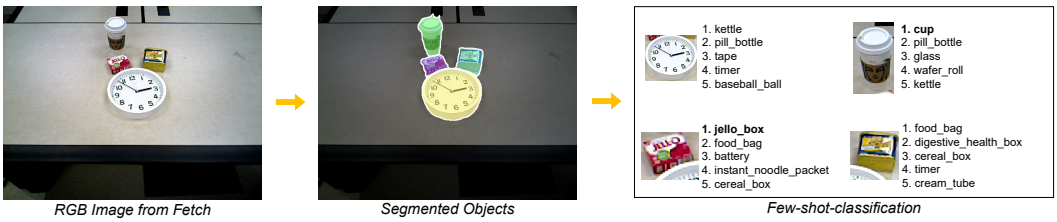
(d) Set-2 prediction using CTX+SimCLR [5]



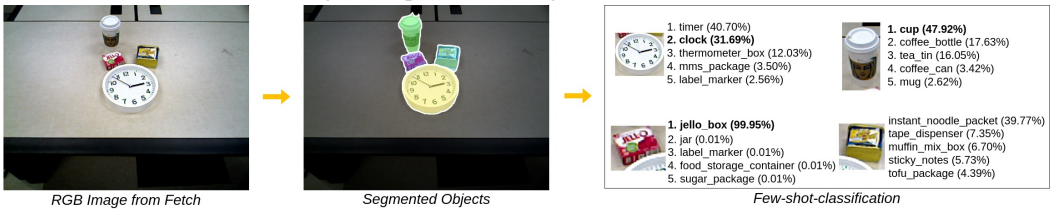
(e) Set-2 prediction using Zero-Shot-CLIP [18]



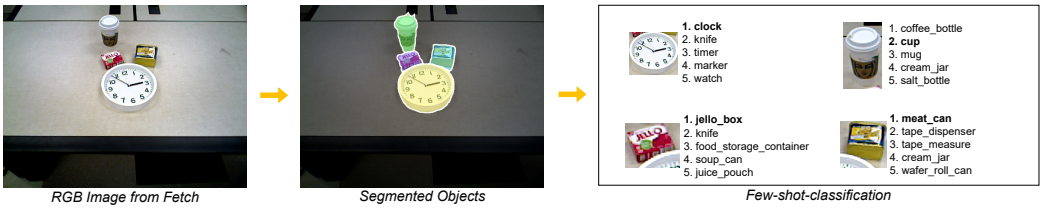
(f) Set-2 prediction using Tip-Adapter [19]



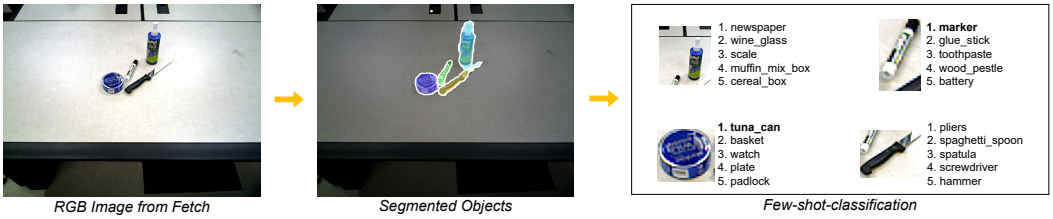
(g) Set-3 prediction using CTX+SimCLR [5]



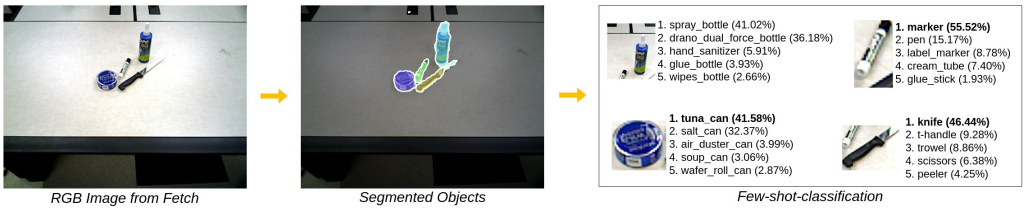
(h) Set-3 prediction using Zero-Shot-CLIP [18]



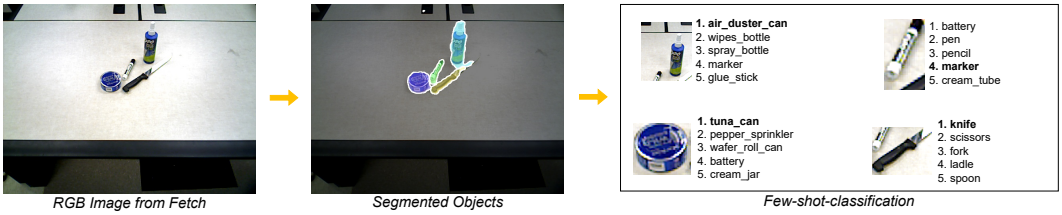
(i) Set-3 prediction using Tip-Adapter [19]



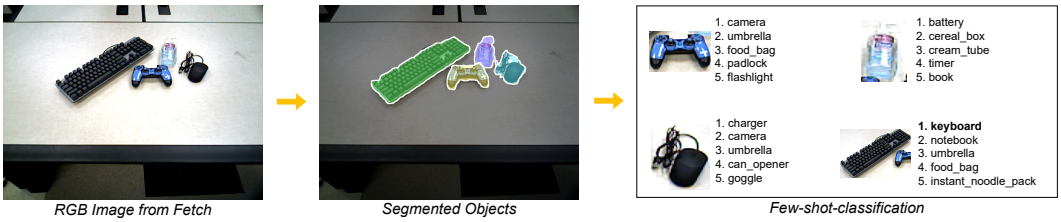
(j) Set-4 prediction using CTX+SimCLR [5]



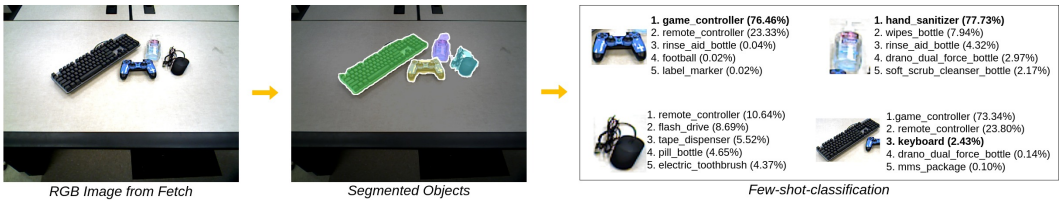
(k) Set-4 prediction using Zero-Shot-CLIP [18]



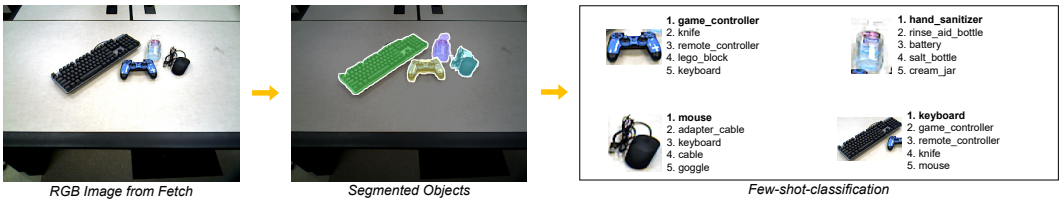
(l) Set-4 prediction using Tip-Adapter [19]



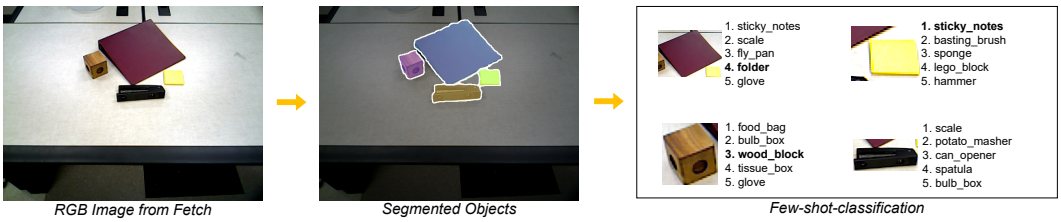
(m) Set-5 prediction using CTX+SimCLR [5]



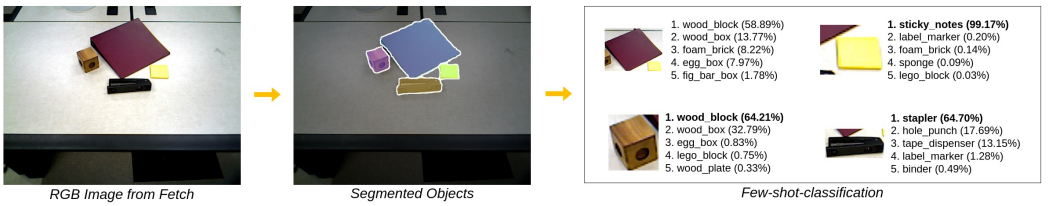
(n) Set-5 prediction using Zero-Shot-CLIP [18]



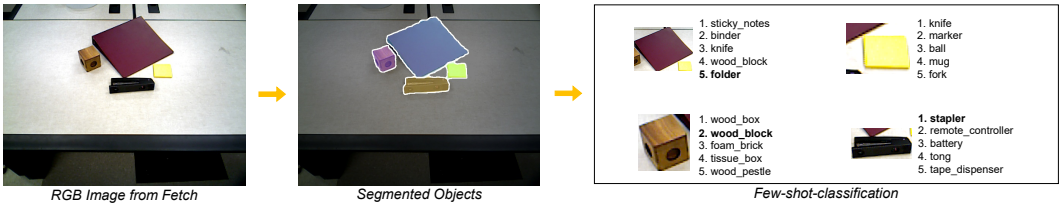
(o) Set-5 prediction using Tip-Adapter [19]



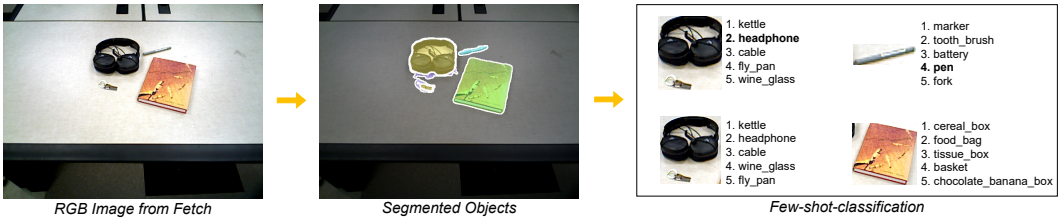
(p) Set-6 prediction using CTX+SimCLR [5]



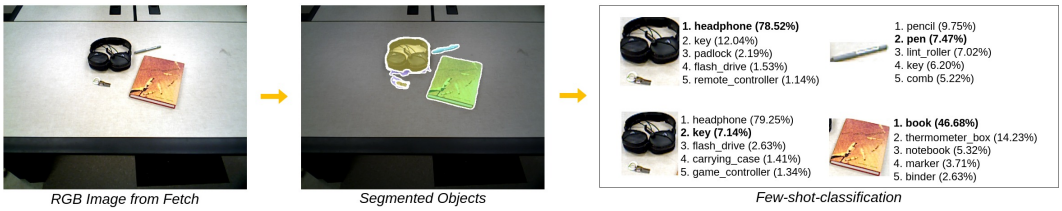
(q) Set-6 prediction using Zero-Shot-CLIP [18]



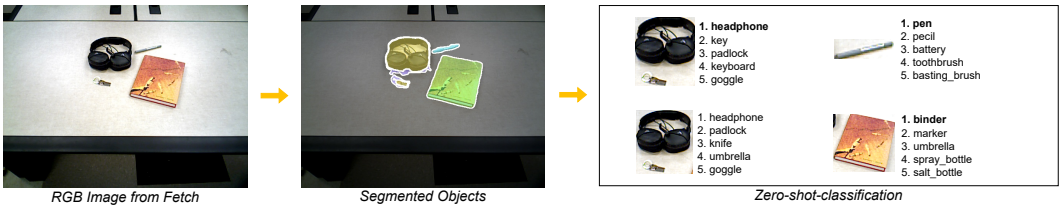
(r) Set-6 prediction using Tip-Adapter [19]



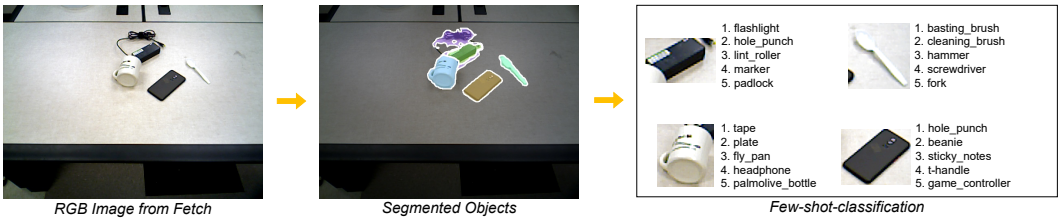
(s) Set-7 prediction using CTX+SimCLR [5]



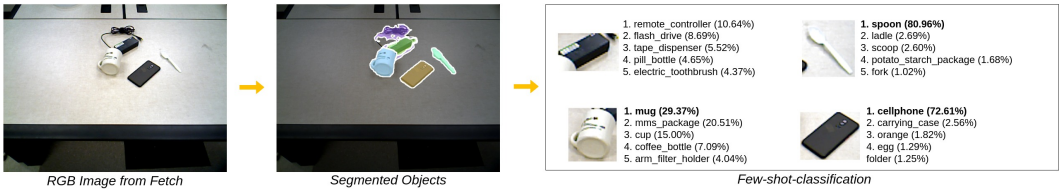
(t) Set-7 prediction using Zero-Shot-CLIP [18]



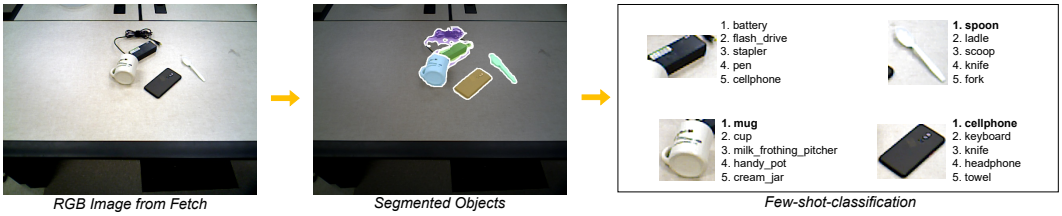
(u) Set-7 prediction using Tip-Adapter [19]



(v) Set-8 prediction using CTX+SimCLR [5]



(w) Set-8 prediction using Zero-Shot-CLIP [18]

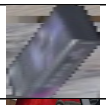
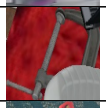
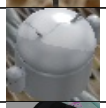
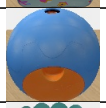
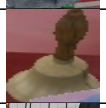
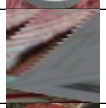


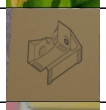
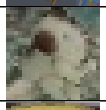
(x) Set-8 prediction using Tip-Adapter [19]

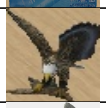
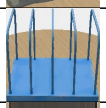
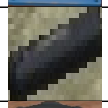
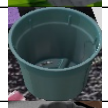
Figure 5: joint object segmentation and few-shot classification on real world objects shown in Figure:4

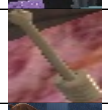
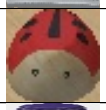
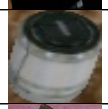
5 Lists of Object Classes

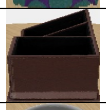
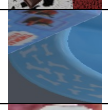
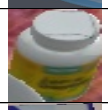
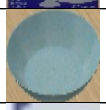
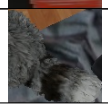
5.1 Synthetic Data

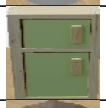
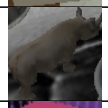
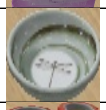
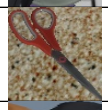
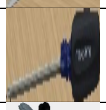
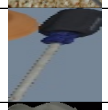
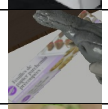
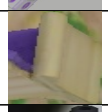
#	Class	#Clean Images	#Cluttered Images	Clean Sample	Cluttered Sample
1	CD	36	15806		
2	C_clamp	9	3745		
3	airplane	9	3704		
4	alarm_clock	9	3985		
5	android	9	3852		
6	backpack	36	13118		
7	ball	9	3684		
8	basket	72	28429		
9	bear	18	6890		
10	bell	9	3549		
11	boot	18	7559		
12	bottle	9	4058		
13	bowl	171	73417		
14	bulb_box	9	4089		
15	butter_dish	9	3847		
16	cable	27	11673		

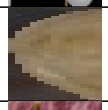
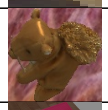
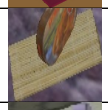
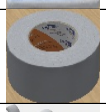
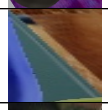
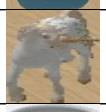
17	cake_pan	9	4194		
18	can_opener	27	12493		
19	candy_bar	9	3796		
20	cardboard_box	9	3887		
21	cereal_box	18	7865		
22	chair	9	4208		
23	coffee_box	18	8173		
24	coffee_can	27	12128		
25	coffee_marker	9	3253		
26	cooker	9	3360		
27	cow	27	11076		
28	cream_box	9	4382		
29	cream_tube	18	7924		
30	cup	18	8231		
31	dinosaur	54	24143		
32	dog	18	7674		
33	doll	9	4037		
34	dragon	9	3780		

35	drink_box	45	18692		
36	eagle	9	3895		
37	elephant	18	7422		
38	file_sorter	9	3947		
39	flash_drive	9	3086		
40	flashlight	9	3990		
41	flower_pot	90	40153		
42	flower_stand	9	3907		
43	fly_pan	9	3847		
44	food_can	9	3943		
45	game_card_box	63	27039		
46	game_controller	27	11991		
47	game_tape	27	11666		
48	garden_swing	9	4385		
49	gift_box	9	4155		
50	hair_dryer	9	3869		
51	hair_straightener	18	7248		
52	hammer	9	3836		

53	hanger	9	3956		
54	hard_drive	9	3924		
55	hat	54	18716		
56	headphone	9	4212		
57	helmet	9	3425		
58	high_heel_shoe	18	7921		
59	honey_dipper	9	3771		
60	horse	9	4174		
61	jug	9	3925		
62	kettle	9	4131		
63	keyboard	36	14118		
64	ladybug	9	3377		
65	lamp	9	3942		
66	laptop	18	6163		
67	lens	9	3994		
68	letter_holder	9	4026		
69	lion	9	4181		
70	lunch_bag	81	34509		

71	lunch_box	9	3910		
72	medicine_box	9	3951		
73	milk_bottle	9	4128		
74	mouse	36	16515		
75	mouse_pad	9	3812		
76	mug	36	16705		
77	panda	9	4030		
78	paper_roll	9	4022		
79	pencil_case	27	12198		
80	pencil_holder	9	3887		
81	pet_dish	36	15117		
82	pill_bottle	90	39322		
83	pill_box	9	3798		
84	plate	81	34261		
85	portable_drive	18	8180		
86	pot	18	8250		
87	product_box	216	95600		
88	racoon	9	3875		

89	ramekin	18	7634		
90	refrigerator	9	4049		
91	rhino	9	4058		
92	rubber_band	9	3500		
93	sanitary_pads	9	3511		
94	saucer	9	3992		
95	scissors	9	3586		
96	screwdriver	9	3949		
97	shark	18	7312		
98	sheep	9	4093		
99	shoe	117	50700		
100	sink	9	4064		
101	slipper	63	28479		
102	smartphone_printer	9	4028		
103	snack_box	126	56524		
104	sofa	9	4060		
105	spatula	9	3917		
106	speaker	18	8103		

107	sponge	9	3805		
108	spoon	9	4079		
109	sprinkler	9	3846		
110	squirrel	9	3861		
111	storage_box	18	6472		
112	sushi_mat	9	3729		
113	table	9	3980		
114	tablet	9	3742		
115	tape	18	8108		
116	teapot	9	4058		
117	toaster	18	7034		
118	towel	180	76769		
119	trash_bin	9	3665		
120	tray	27	11222		
121	unicon	9	4047		
122	utensil_holder	9	3889		
123	whale	9	3413		
124	wood_box	9	3778		

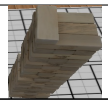
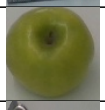
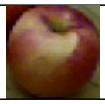
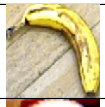
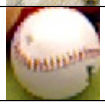
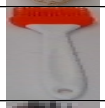
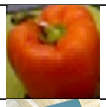
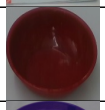
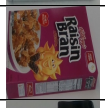
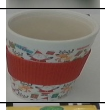
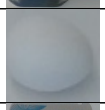
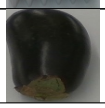
125	wood_tower	9	4034		

Table 4: 125 object classes of the 330 3D models from the Google Scanned Objects [20]

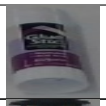
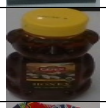
5.2 Real-World Data

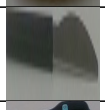
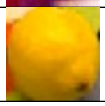
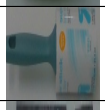
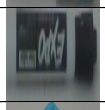
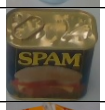
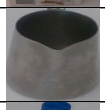
#	Class	#Our Images	#OCID Images	Our Sample	OCID Sample
1	adapter_cable	9	0		-
2	air_duster_can	9	0		-
3	air_purifying_bag	9	0		-
4	aluminum_foil_box	9	0		-
5	apple	27	68		
6	arm_filter_holder	9	0		-
7	avocado	9	0		-
8	baking_tray	27	0		-
9	ball	36	112		
10	ball_of_wool	9	0		-
11	banana	18	54		
12	baseball_ball	18	34		
13	basket	9	0		-
14	basting_brush	9	0		-
15	battery	9	0		-

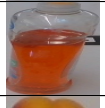
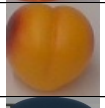
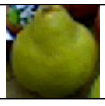
16	beanie	18	0		-
17	bell_pepper	18	12		
18	binder	18	26		
19	book	9	0		-
20	bowl	45	12		
21	bucket	9	0		-
22	bulb_box	18	36		
23	cabbage	9	0		-
24	cable	27	0		-
25	camera	9	0		-
26	can_opener	9	0		-
27	carrot	9	0		-
28	carrying_case	9	0		-
29	cellphone	18	0		-
30	cereal_box	36	252		
31	charger	9	0		-
32	chili_powder_bottle	9	0		-
33	chocolate_banana_box	9	38		

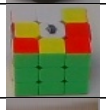
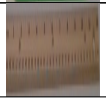
34	clamp	18	14		
35	cleaning_brush	18	0		-
36	cling_wrap_box	9	0		-
37	clock	9	0		-
38	coffee_bottle	9	0		-
39	coffee_can	18	18		
40	comb	18	0		-
41	cracker_box	18	22		
42	cream_jar	9	0		-
43	cream_tube	27	0		-
44	cucumber	9	0		-
45	cup	27	0		-
46	digestive_health_box	9	0		-
47	donut_sticks_box	9	0		-
48	drano_dual_force_bottle	9	0		-
49	egg	9	0		-
50	egg_box	9	0		-
51	eggplant	9	0		-

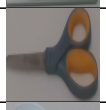
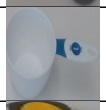
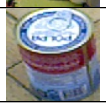
52	electric_toothbrush	9	0		-
53	fig_bar_box	9	0		-
54	flash_drive	9	0		-
55	flashlight	18	18		
56	fly_pan	9	0		-
57	foam_brick	9	18		
58	folder	9	0		-
59	food_bag	27	102		
60	food_storage_container	27	0		-
61	football	9	18		
62	fork	45	0		-
63	game_controller	9	0		-
64	garlic	9	0		-
65	ginger	9	0		-
66	glass	27	0		-
67	glasses_box	9	0		-
68	glove	18	0		-
69	glue_bottle	18	0		-

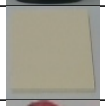
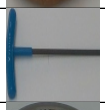
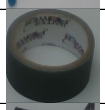
70	glue_stick	27	38		
71	goggle	27	0		-
72	golf_ball	9	18		
73	goo_gone_bottle	9	0		-
74	hair_brush	9	0		-
75	hammer	27	0		-
76	hand_sanitizer	9	0		-
77	handy_pot	27	0		-
78	headphone	27	0		-
79	hole_punch	9	0		-
80	honey_bottle	9	0		-
81	instant_noodle_packet	9	26		
82	jar	9	0		-
83	jello_box	18	52		
84	juice_bottle	18	0		-
85	juice_pouch	9	0		-
86	kettle	9	0		-
87	key	9	0		-

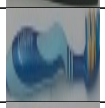
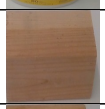
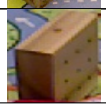
88	keyboard	18	48		
89	kiwi	9	0		-
90	knife	54	0		-
91	label_marker	9	0		-
92	ladle	36	0		-
93	lego_block	18	22		
94	lemon	27	40		
95	lighter	9	0		-
96	lime	18	21		
97	lint_roller	9	0		-
98	marker	27	104		
99	mask	9	0		-
100	meat_can	9	0		-
101	milk_box	18	0		-
102	milk_frothing_pitcher	9	0		-
103	milk_jug	9	0		-
104	mms_package	9	0		-
105	mold_star_bottle	9	0		-

106	mouse	18	0		-
107	muffin_mix_box	9	0		-
108	mug	45	44		
109	mustard_bottle	9	16		
110	newspaper	9	0		-
111	notebook	9	0		-
112	onion	18	0		-
113	orange	18	68		
114	oven_glove	9	0		-
115	padlock	9	0		-
116	palmolive_bottle	9	0		-
117	peach	9	48		
118	peanut_jar	9	0		-
119	pear	18	31		
120	peeler	9	0		-
121	pen	9	0		-
122	pencil	9	0		-
123	pepper_sprinkler	9	0		-

124	pill_bottle	9	0		-
125	pitcher	9	14		
126	pizza_cutter	9	0		-
127	plant_vase	9	0		-
128	plate	45	0		-
129	pliers	18	0		-
130	potato	9	46		
131	potato_masher	9	0		-
132	potato_starch_package	9	0		-
133	power_drill	27	16		
134	pringles_can	9	24		
135	protein_bar	9	0		-
136	pump_bottle	9	0		-
137	remote_controller	18	0		-
138	rinse_aid_bottle	18	0		-
139	root_beer_bottle	9	0		-
140	rubiks_cube	9	16		
141	ruler	9	0		-

142	salt_bottle	18	0		-
143	salt_can	9	0		-
144	scale	9	0		-
145	scissors	27	0		-
146	scoop	9	0		-
147	screwdriver	18	0		-
148	shampoo_bottle	27	92		
149	shaver	9	0		-
150	shoe	9	0		-
151	slipper	18	0		-
152	soda_can	45	76		
153	soft_scrub_cleanser_bottle	9	16		
154	soup_can	36	60		
155	spaghetti_spoon	9	0		-
156	spatula	36	0		-
157	sponge	18	80		
158	spoon	36	0		-
159	spray_bottle	36	0		-

160	stapler	9	28		
161	sticky_notes	9	0		-
162	strawberry	9	0		-
163	sugar_box	9	20		
164	sugar_package	9	0		-
165	sunscreen_bottle	9	0		-
166	sweet_potato	9	0		-
167	t-handle	9	0		-
168	tape	18	0		-
169	tape_dispenser	18	0		-
170	tape_measure	27	0		-
171	tea_tin	9	0		-
172	tennis_ball	9	14		
173	thermometer_box	9	0		-
174	timer	9	22		
175	tissue_box	18	128		
176	tofu_package	18	0		-
177	tomato	18	20		

178	tong	27	0		-
179	toothbrush	9	0		-
180	toothpaste	36	38		
181	toothpaste_box	9	0		-
182	towel	36	60		
183	travel_pillow	9	0		-
184	trowel	9	0		-
185	tuna_can	27	54		
186	umbrella	18	0		-
187	wafer_roll_can	18	0		-
188	watch	18	0		-
189	water_bottle	36	0		-
190	watermelon	9	0		-
191	wine_bottle	9	0		-
192	wine_glass	9	0		-
193	wipes_bottle	9	0		-
194	wood_block	18	24		
195	wood_box	9	22		

196	wood_pestle	9	0		-
197	wood_plate	9	0		-
198	wrench	9	0		-

Table 5: 198 object classes of the 336 objects we captured in our FewSOL dataset. 52 classes have examples from the OCID dataset [21]

References

[1] E. Triantafillou, T. Zhu, V. Dumoulin, P. Lamblin, U. Evci, K. Xu, R. Goroshin, C. Gelada, K. Swersky, P.-A. Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[2] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017.

[3] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[4] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135. PMLR, 2017.

[5] C. Doersch, A. Gupta, and A. Zisserman. Crosstransformers: spatially-aware few-shot transfer. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:21981–21993, 2020.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

[7] Y. Xiang, C. Xie, A. Mousavian, and D. Fox. Learning RGB-D feature embeddings for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, pages 461–470. PMLR, 2021.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[9] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[10] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[12] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3606–3613, 2014.

[13] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, , and N. Fox-Gieg. The quick, draw! – a.i. experiment, quickdraw.withgoogle.com. 2016.

[14] M. Sulc, L. Picek, J. Matas, T. Jeppesen, and J. Heilmann-Clausen. Fungi recognition: A practical use case. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2316–2324, 2020.

[15] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008.

[16] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *The International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013.

[17] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755. Springer, 2014.

[18] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021.

[19] R. Zhang, Z. Wei, R. Fang, P. Gao, K. Li, J. Dai, Y. Qiao, and H. Li. Tip-adapter: Training-free adaption of clip for few-shot classification. *arXiv preprint arXiv:2207.09519*, 2022.

[20] L. Downs, A. Francis, N. Koenig, B. Kinman, R. Hickman, K. Reymann, T. B. McHugh, and V. Vanhoucke. Google scanned objects: A high-quality dataset of 3D scanned household items. *arXiv preprint arXiv:2204.11918*, 2022.

[21] M. Suchi, T. Patten, D. Fischinger, and M. Vincze. Easylabel: A semi-automatic pixel-wise object annotation tool for creating robotic rgb-d datasets. In *International Conference on Robotics and Automation (ICRA)*, pages 6678–6684. IEEE, 2019.