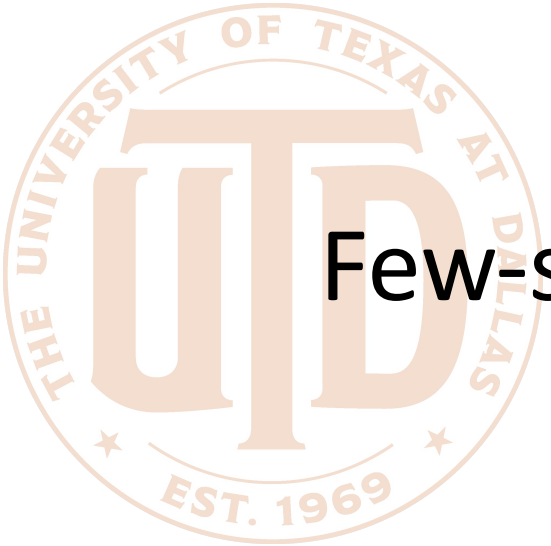


Few-shot object classification in clutter scenes

Jishnu Jaykumar Padalunkal

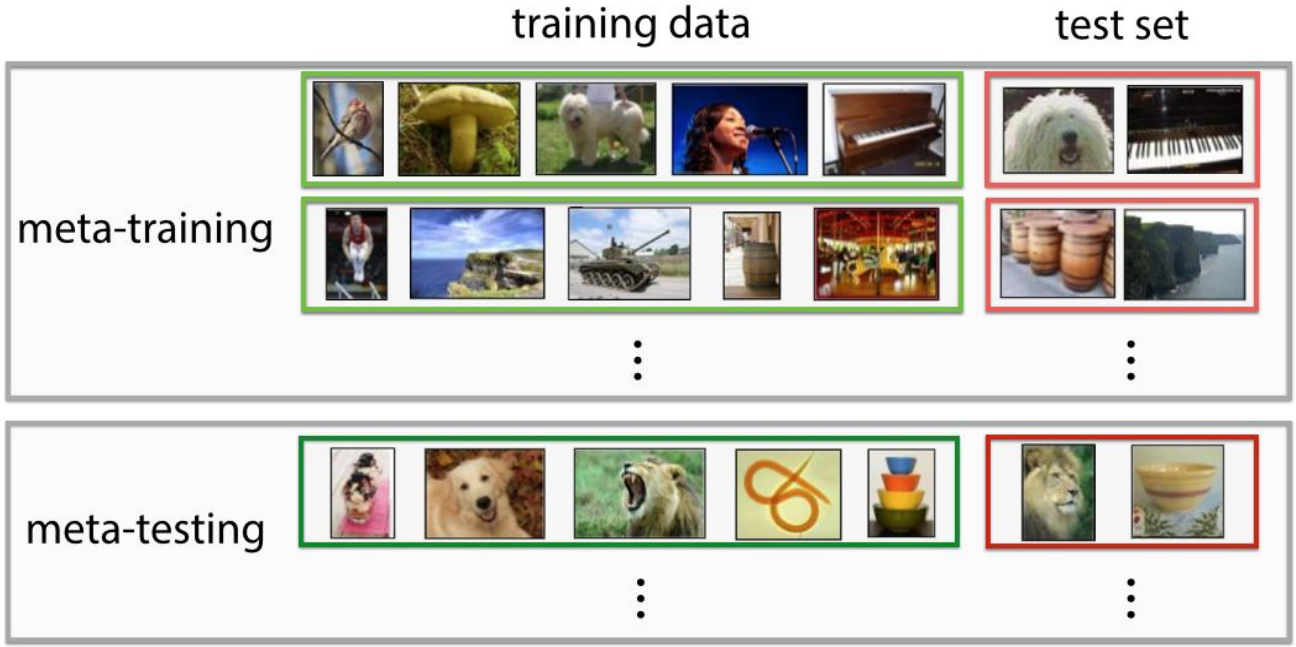
Team-11 | CS 6384 Computer Vision

The University of Texas at Dallas

Few-shot object classification in clutter scenes

- Few-Shot Learning is a sub-area of machine learning. It's about classifying new data when you have only a few training samples with supervised information (neptune.ai).
- Formulated as an N-way-K-shot problem (**Episodes**)
  - N := number of classes
  - K := number of samples per class
    - In a fixed setup, this remains same for all classes
    - In a variable setup, this varies across classes

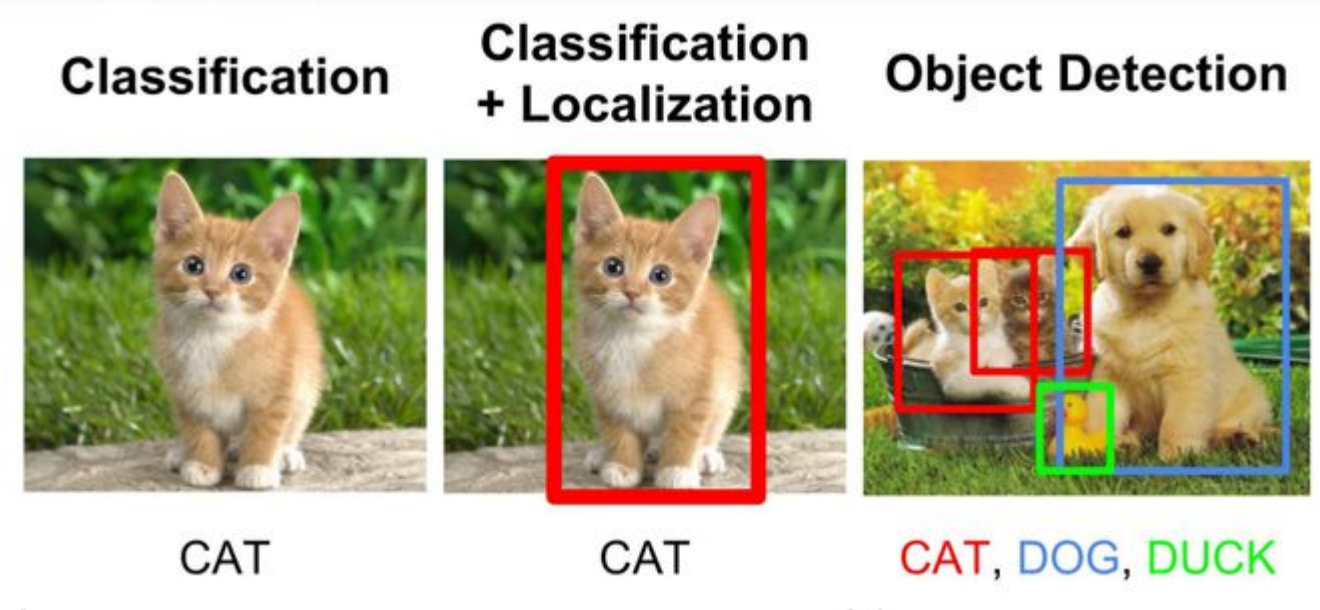
Few-shot object classification in clutter scenes



Remember "**training data**" and "**test set**" as "**Support**" and "**Query**" **set** respectively
Here, it's a 5-way-1-shot setup (fixed episode variant)

Image: https://bair.berkeley.edu/blog/2017/07/18/learning-to-learn

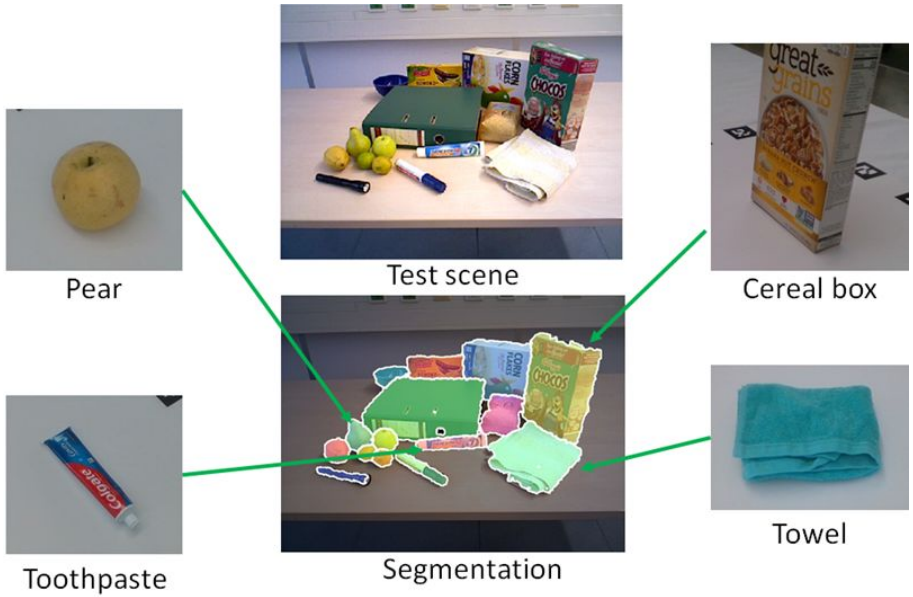
Few-shot **object classification** in clutter scenes



We will be dealing with **classification** only. i.e. Given an image containing a single object, classify it.

Image: https://www.kaggle.com/getting-started/169984

Few-shot object classification in **clutter scenes**



Here, the test scene is an example of clutter scene containing various objects.

Image-courtesy: Dr. Yu Xiang

Motivation

- The benchmarks for few-shot learning
- Omniglot: 50 alphabets, 1,623 classes, 20 images per class (105 x 105 resolution)
  - Mini-ImageNet: 100 classes, 600 images per class (84 x 84)
- The performance almost saturates on these datasets
- In robotics, we would like to employ few-shot learning techniques for object recognition. **There is no good dataset for few-shot object learning in robotic manipulation settings.**
- Also, getting labelled real data is non-trivial as it requires lot of resources.
  - Question is can we learn from simulated examples and perform well on real world data points? Yeah, that's right => Sim2Real
- Thus, we developed a **multi-view** dataset.
  - "**TESLA**" for mul**T**i-view RGB dataset for f**E**w-**S**hot **L**e**A**rning
- This will be used for few shot object classification in clutter scenes.

TESLA Dataset

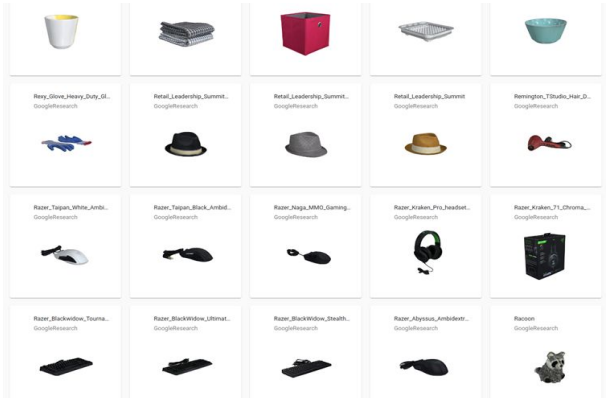
```
- TESLA_root
    - Training (Synthetic, 125 classes)
        - Object-Class
            - Support (Synthetic)
                - S-Img-1
                - S-Img-2
                - ...
            - Query (Google Scenes)
                - Q-Img-1
                - Q-Img-2
                - ...
    - Test (Real, 198 classes, 52 classes have support + query images,
            11 classes are common between train and test which have support + query set)
        - Object-Class ∈ {Object-Class}'
            - Support (Real Object)
                - S'-Img-1
                - S'-Img-2
                - ...
            - Query (OCID)
                - Q'-Img-1
                - Q'-Img-2
                - ...
```

TESLA Dataset creation

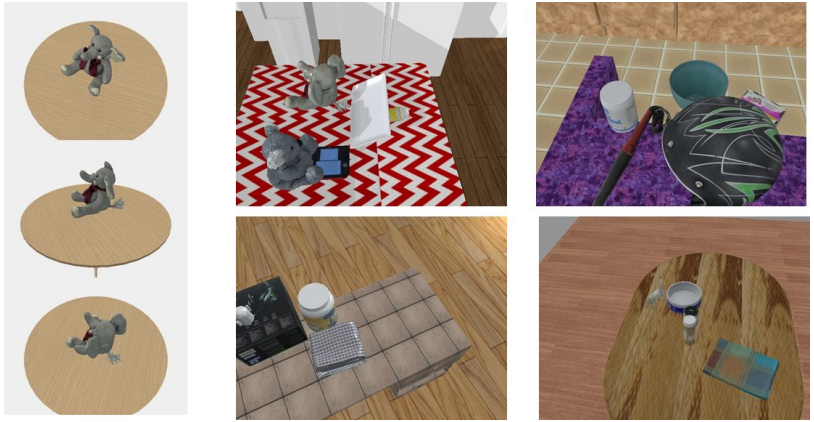
- Training Data [Created using *PyBullet*]
  - Simulated objects
    - 330 3D models of Objects, cropped using object mask
    - Google Dataset (discussed in one of the previous lectures)
  - Support set contains 9 views of each object (**clean**)
  - Query set is formed by cropping the required object from simulated **clutter** scene using object mask.

Cropped support examples

Cropped query examples =>



Google 3D Objects (330)



Simulated single object and clutter scenes

Observation: Query set may have occlusions. Support set is void of it.

TESLA Dataset creation

Observation: Query set may have occlusions. Support set is void of it.

- Test data
  - 336 objects from the real world forms the support set
    - 9 views per class/object (**clean**), cropped using object mask
  - OCID – Object Clutter Indoor Dataset
    - Cropping objects from the real **clutter** scenes using object mask forms the query set

Cropped Support Samples

Cropped Query Samples

OCID Dataset:
https://www.acin.tuwien.ac.at/en/vision-for-robotics/software-tools/object-clutter-indoor-dataset/

TESLA Dataset creation (Test Data: Support)

Source: Dr. Yu Xiang

TESLA Dataset creation (Test Data: Support)

Source: Dr. Yu Xiang

TESLA Dataset on Meta-Dataset Benchmark

- Meta-Dataset is a benchmark with
  - Various datasets
  - Various models for Few-Shot and Meta-Learning
  - The newer version also includes transfer learning artifacts as well
  - Code: https://github.com/google-research/meta-dataset
- TESLA plugged into Meta-Dataset Benchmark (Our experiments)
  - Run on
    - Prototypical Networks
    - Matching Networks
    - MAML
    - Proto-MAML
    - CrossTransformers
    - CrossTransformers + SimCLR

Results
(95% Confidence Interval for Accuracy metric using 600 episodes)

				Test							
				Real + OCID						Synthetic	
Model	Embedding_fn decided based on	Embedding_fn	Num_Episodes (val	Mixture (52 classes)	Mixture-filtered	Unseen (41 classes)	Unseen-filtered	Seen (11 classes)	Seen-filtered	Unseen (13)	Unseen-filtered
Without any pre-trained backbones											
Prototypical	MD-Default	four_layer_convnet	600/600, 75K	0.351786, +/- 0.008372	0.169035, +/- 0.007896	0.374995, +/- 0.008338	0.197848, +/- 0.008278	0.394100, +/- 0.008297	0.226971, +/- 0.009060	0.613413, +/- 0.010703	0.592732, +/- 0.010341
MatchingNet	MD-Default	four_layer_convnet	600/600, 75K	0.539673, +/- 0.011071	0.205882, +/- 0.008823	0.587700, +/- 0.010779	0.220622, +/- 0.007588	0.562442, +/- 0.010648	0.334560, +/- 0.010086	0.803420, +/- 0.007868	0.766079, +/- 0.007914
MatchingNet	Our Setup	resnet, im_h=126	600/600, 75K	0.517028, +/- 0.010846	0.179460, +/- 0.007989	0.559975, +/- 0.010613	0.194610, +/- 0.006971	0.524756, +/- 0.010433	0.293627, +/- 0.008860	0.770191, +/- 0.008516	0.733019, +/- 0.008820
MatchingNet	Our Setup	resnet34, im_h=126	600/600, 75K	0.529191, +/- 0.010521	0.182286, +/- 0.006716	0.558913, +/- 0.010570	0.205379, +/- 0.007458	0.535399, +/- 0.010705	0.325308, +/- 0.009337	0.775070, +/- 0.008376	0.743541, +/- 0.009060
MAML	MD-Default	four_layer_convnet	60/600, 75K	0.497081, +/- 0.011635	0.179157, +/- 0.007556	0.547904, +/- 0.011396	0.202738, +/- 0.008290	0.548973, +/- 0.011200	0.310036, +/- 0.009164	0.748170, +/- 0.010205	0.709779, +/- 0.008960
crosstransformer	MD-Default	resnet34	60/600, 100K (MD-	0.499650, +/- 0.010394	0.184342, +/- 0.007407	0.539116, +/- 0.010151	0.208177, +/- 0.008680	0.579684, +/- 0.009763	0.369341, +/- 0.010254	0.924512, +/- 0.004646	0.898203, +/- 0.005773
crosstransformer-simclr-episodes	MD-Default	resnet34	60/600, 400K(MD-(	0.608311, +/- 0.010610	0.316665, +/- 0.009698	0.637957, +/- 0.010854	0.333432, +/- 0.009913	0.662520, +/- 0.010078	0.516159, +/- 0.011027	0.895779, +/- 0.005742	0.889922, +/- 0.005706
With pre-trained backbones											
As Resnet34 performs better, remaining models will be trained using Resnet34 as pretrained backbone to reduce training permutations (with learning_rate=0.0010521782166881 74 and num_valid_Episodes=60, save_ckpt_every=1000)											
Matching-60	Our Setup	resnet34, im_h=126	60/600, 75K	0.623470, +/- 0.010596	0.285045, +/- 0.009277	0.654113, +/- 0.010557	0.301564, +/- 0.009420	0.704983, +/- 0.009938	0.440081, +/- 0.010261	0.854438, +/- 0.006309	0.840961, +/- 0.006970
Matching-filtered-60	Our Setup	resnet34, im_h=126	60/600, 75K	0.538124, +/- 0.010248	0.263297, +/- 0.009402	0.577684, +/- 0.009264	0.280520, +/- 0.009962	0.618326, +/- 0.009694	0.458149, +/- 0.010653	0.653966, +/- 0.015696	0.851842, +/- 0.006956
Prototypical-60	Our Setup	resnet34, im_h=126	60/600, 75K	0.584256, +/- 0.010171	0.320632, +/- 0.009737	0.613562, +/- 0.010426	0.332194, +/- 0.009586	0.666462, +/- 0.009833	0.496791, +/- 0.011027	0.784394, +/- 0.009504	0.783002, +/- 0.007689
Prototypical-filtered-60	Our Setup	resnet34, im_h=126	60/600, 75K	0.575437, +/- 0.010576	0.344717, +/- 0.010037	0.612460, +/- 0.011133	0.370607, +/- 0.010119	0.658950, +/- 0.010445	0.510729, +/- 0.010442	0.743737, +/- 0.011178	0.813797, +/- 0.006679
MAML-60	Our Setup	resnet34, im_h=126	60/600, 75K	0.560356, +/- 0.010829	0.206363, +/- 0.007598	0.580145, +/- 0.011243	0.212390, +/- 0.008105	0.588134, +/- 0.011163	0.323809, +/- 0.009563	0.791220, +/- 0.009508	0.718783, +/- 0.009990
MAML-filtered-60	Our Setup	resnet34, im_h=126	60/600, 66K	0.459575, +/- 0.011880	0.162691, +/- 0.007760	0.515377, +/- 0.011777	0.160266, +/- 0.007101	0.567824, +/- 0.011271	0.355348, +/- 0.009702	0.725636, +/- 0.010490	0.725636, +/- 0.010490
PROTO_MAML	Our Setup	resnet34, im_h=126	60/600, 75K	0.609769, +/- 0.010054	0.283211, +/- 0.009118	0.631798, +/- 0.010243	0.290773, +/- 0.009308	0.714419, +/- 0.009987	0.469772, +/- 0.010711	0.892056, +/- 0.007020	0.867042, +/- 0.007120
PROTO_MAML-filtered	Our Setup	resnet34, im_h=126	60/600, 75K	0.570884, +/- 0.010422	0.270062, +/- 0.009382	0.602918, +/- 0.010226	0.286915, +/- 0.008756	0.667483, +/- 0.010389	0.443909, +/- 0.011047	0.771586, +/- 0.010995	0.881390, +/- 0.006758
crosstransformer	Our Setup	resnet34, im_h=126	60/600, 100K (MD-	0.562897, +/- 0.009284	0.269304, +/- 0.009098	0.585234, +/- 0.009158	0.273955, +/- 0.009453	0.629967, +/- 0.010086	0.441077, +/- 0.010642	0.945140, +/- 0.003926	0.920558, +/- 0.004791
crosstransformer-filtered	Our Setup	resnet34, im_h=126	60/600, 100K(MD-(	0.566504, +/- 0.010191	0.290602, +/- 0.009877	0.603281, +/- 0.010184	0.299556, +/- 0.009380	0.654695, +/- 0.010354	0.454782, +/- 0.011205	0.906597, +/- 0.006263	0.927241, +/- 0.004542
crosstransformer-simclr-episodes	Our Setup	resnet34, im_h=126	60/600, 400K(MD-(	0.626996, +/- 0.010652	0.385578, +/- 0.011231	0.648641, +/- 0.010854	0.382234, +/- 0.010656	0.731127, +/- 0.009276	0.614708, +/- 0.011141	0.892334, +/- 0.005978	0.900091, +/- 0.004902

Sample output for CrossTransformer(SimCLR) case

Support/Query: Mustard bottle. All but one prediction is wrong which is reasonable as well because cracker box is present behind the mustard bottle

Support/Query: Tennis ball, wrongly predicted as lemon which is reasonable as both have similar visual features

NOTE: Images without/with border are support/query images respectively. Green/Red border indicates that prediction is correct/wrong.

Questions?