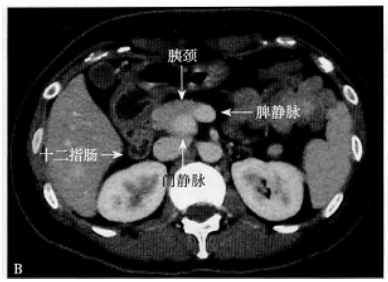
Lecture 24: Recent Progress in Deep Learning: Few-shot Learning

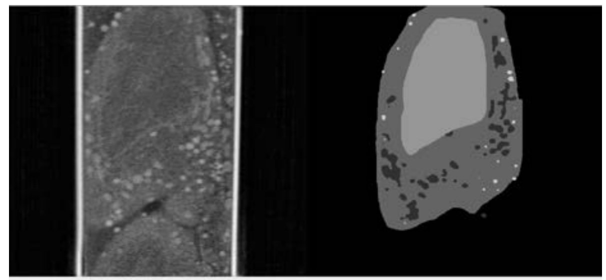
Xuming He
SIST, ShanghaiTech
Fall, 2020

Real-world scenarios

- Data annotation is costly
  - Many specific domain and cross modality tasks



Medical image understanding
(image credit: 廖飞. 胰腺影像学. 2015.)

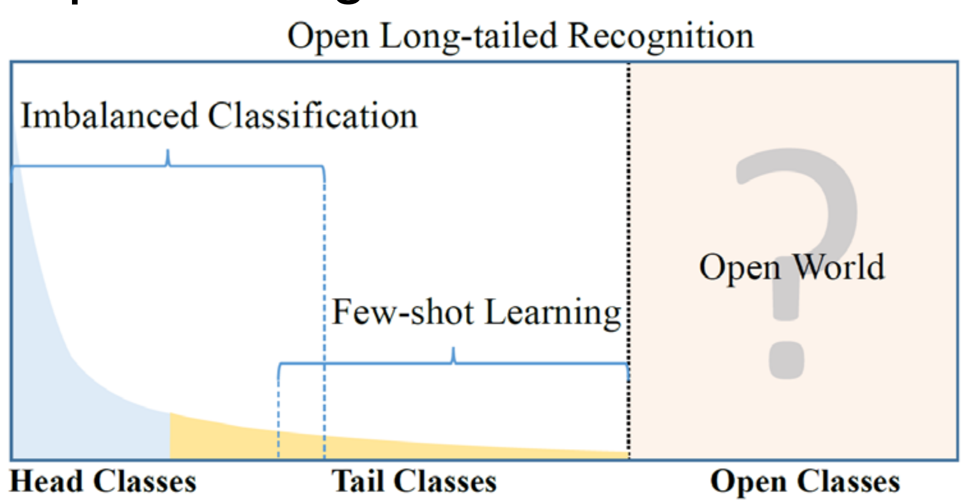


Biological image analysis
(Zhang and He, 2019)

A house cat laying on a couch beside a remote.

Vision & Language (MSCOCO)

- Visual concept learning in wild



(Liu et al CVPR 2019)

Challenges

- Limitation in naïve transfer learning
  - Insufficient instance variations of novel classes
  - Fine-tuning usually fails given a few examples per class

Image Credit: Ravi & Larochelle et al 2017

- Human (child) performance is much better
  - How do we achieve such data efficiency?
  - What representations are used?
  - What are the underlying learning algorithms?

Few-shot learning problem

- Learning from (very) limited annotated data
- Typical setting:
  - Classification using a few training examples per visual category
  - Formally, given a small dataset $D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L}$
    - N categories $y_i \in \mathcal{Y}, |\mathcal{Y}| = N,$
    - K shot: each class has K examples, or $L = N \times K$
  - The goal is to learn a model *F* parametrized by θ to minimize

$$E_{D_{test}}\left[\text{loss}(y_i, F_\theta(x_i))\right]$$

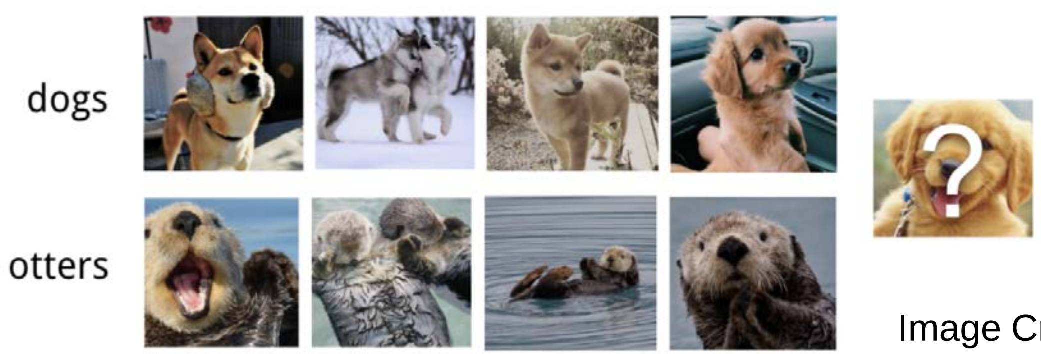


Image Credit: Weng, Lil-log, 2018

Few-shot learning problem

- For a single isolated task, this is difficult
  - But *if* we have access to many *similar* few-shot learning tasks, we can exploit such prior knowledge.
- Main idea is to consider task-level learning
  - Learn a representation shared by all those tasks
  - Learn an efficient classifier learning algorithm that can be applied to all the tasks

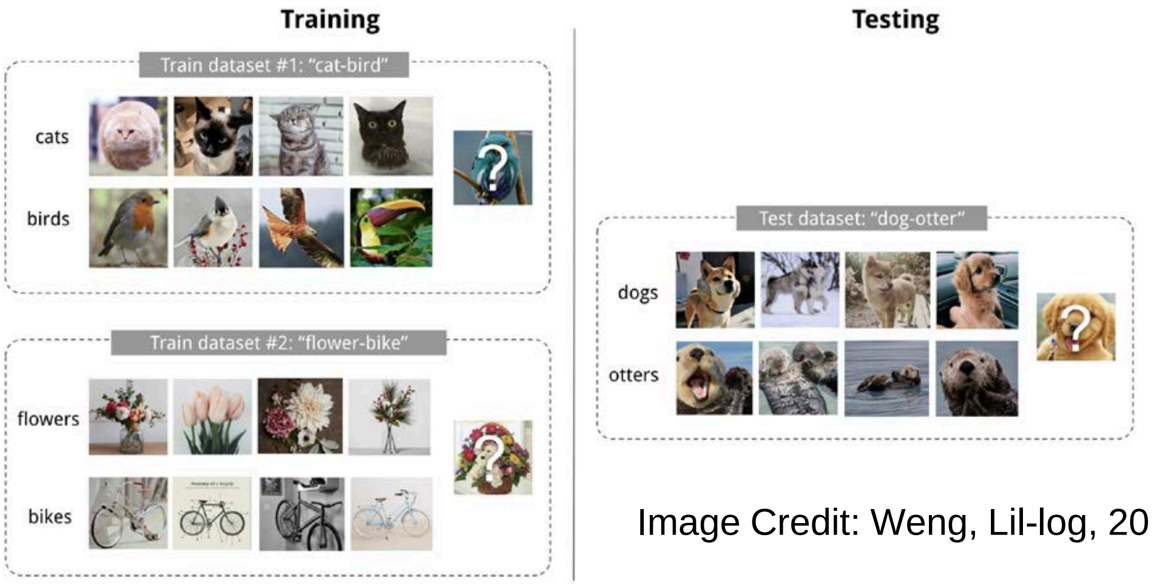


Image Credit: Weng, Lil-log, 2018

Main intuitions in few-shot learning

- Prior knowledge in different vision tasks
  - Similarity between visual categories
    - Feature representations, etc.
  - Similarity between visual recognition tasks
    - Learning a classifier, etc.

- Focusing on generic aspects of similar tasks
  - Generic visual representations
    - Not category-specific
  - Transferrable learning strategies
    - Very data-efficient

Meta-learning framework

- Problem formulation
  - Each few-shot classification problem as a task

    Each Task: $T \in \mathcal{T}$ $\quad T \sim P(T)$

  - Each task (or an *episode*) consists of

    $T = (D_{train}, D_{test}, \mathcal{Y}_T)$

  - Task-train (*support*) set

    $D_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{L} \quad \forall y_i \in \mathcal{Y}_T$

  - Task-test set (query) D_{test}
  - For each task, we adopt an learning algorithm A_ϕ
    - to learn its own classifier F_θ via $F_\theta = A_\phi(D_{train})$
    - to perform well on the task-test set D_{test}

Meta-learning formulation

- Key assumptions:
  - The learning algorithm A_ϕ is shared across tasks
  - We can sample many tasks to learn a good A_ϕ

- A meta-learning strategy
  - Input: meta-training set $\mathcal{D}_{meta-train} = \{(D^{(n)}_{train}, D^{(n)}_{test})\}_{n=1}^{N}$
  - Output: algorithm parameter ϕ^*
  - Objective: good performance on meta-test set

$$\mathcal{D}_{meta-test} = \{(D'^{(n)}_{train}, D'^{(n)}_{test})\}_{n=1}^{N'}$$

  - Minimizing the empirical loss on the meta-training set

$$\min_\phi E_{\mathcal{D}_{meta-train}} \left[\mathrm{loss}(F_\theta^{(n)}, D^{(n)}_{test}) \right]$$

    - Each meta-train task $F_\theta^{(n)} = A_\phi(D^{(n)}_{train})$

Meta-learning formulation

- Analogy to standard supervised learning

Supervised-Learning	Train	Test	One data point
Meta-learning	Meta-training	Meta-testing	One task

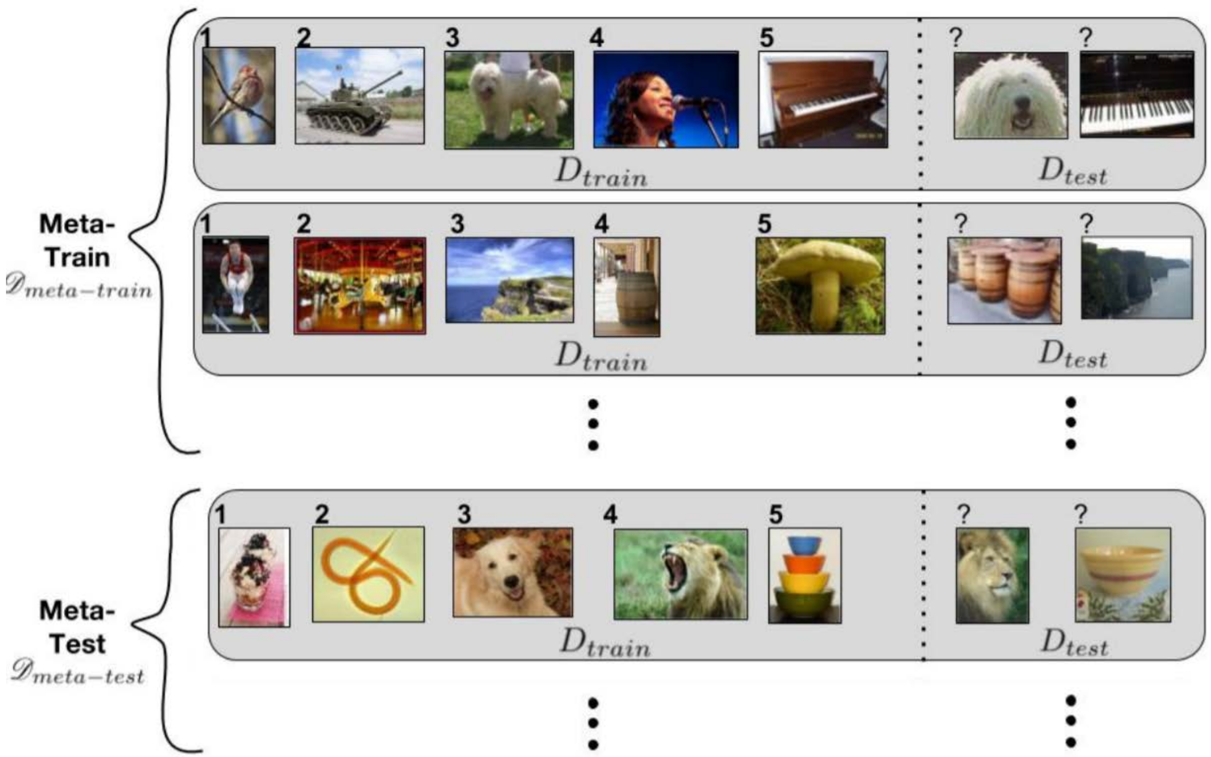
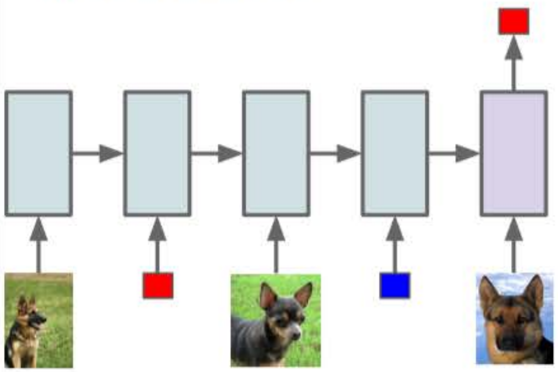


Image Credit: Ravi & Larochelle et al 2017

Overview of existing methods

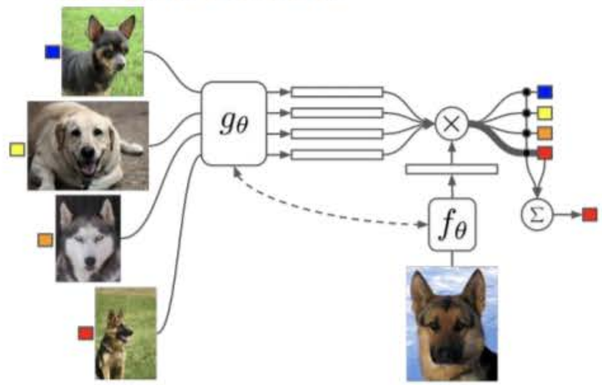
- Depending on the meta-learners used in few-shot tasks

Model Based



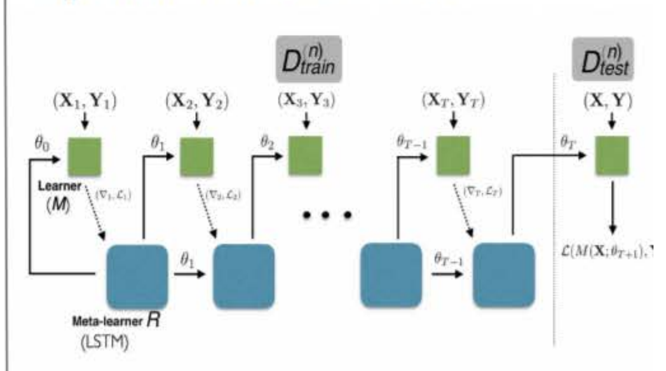
- Santoro et al. '16
- Duan et al. '17
- Wang et al. '17
- Munkhdalai & Yu '17
- Mishra et al. '17

Metric Based



- Koch '15
- Vinyals et al. '16
- Snell et al. '17
- Shyam et al. '17
- Sung et al. '17

Optimization Based



- Schmidhuber '87, '92
- Bengio et al. '90, '92
- Hochreiter et al. '01
- Li & Malik '16
- Andrychowicz et al. '16
- Ravi & Larochelle '17
- Finn et al. '17

Slide Credit: Vinyals, NIPS 2017

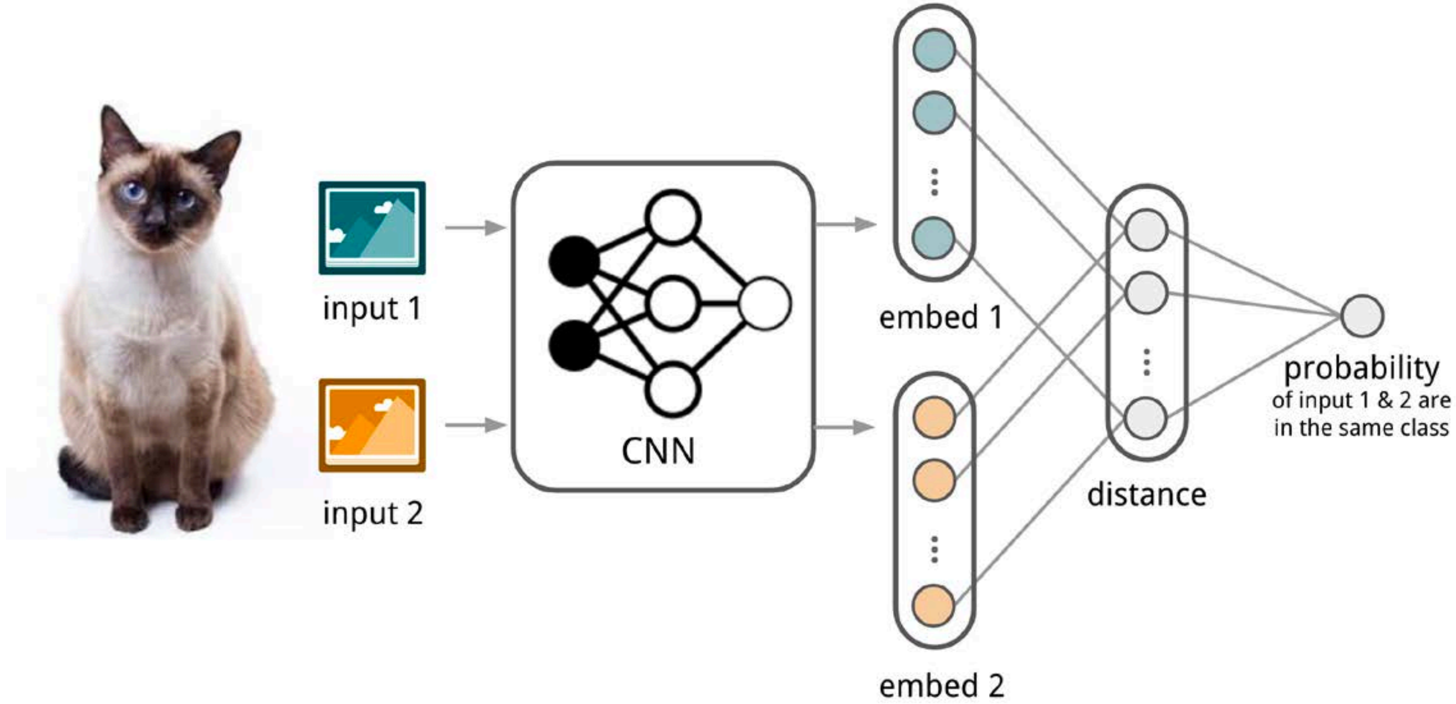
Metric-based methods

- Basic idea: Learn a generic distance metric

$$P_\theta(y|\mathbf{x}, D_{train}) = \sum_{(\mathbf{x}_i, y_i)\in D_{train}} k_\theta(\mathbf{x}, \mathbf{x}_i) y_i$$

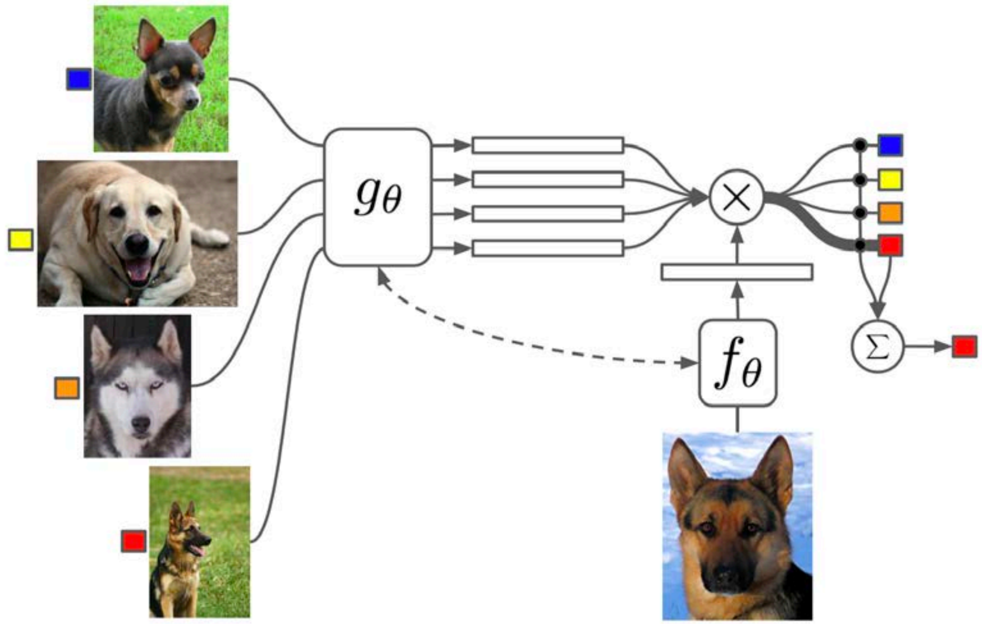
- Typical methods
  - Siamese network (Koch, Zemel & Salakhutdinov, 2015)
  - Matching network (Vinyals et al, 2016)
  - Relation network (Sung et al. 2018)
  - Prototypical network (Snell, Swersky & Zemel, 2017)

Siamese Neural Network



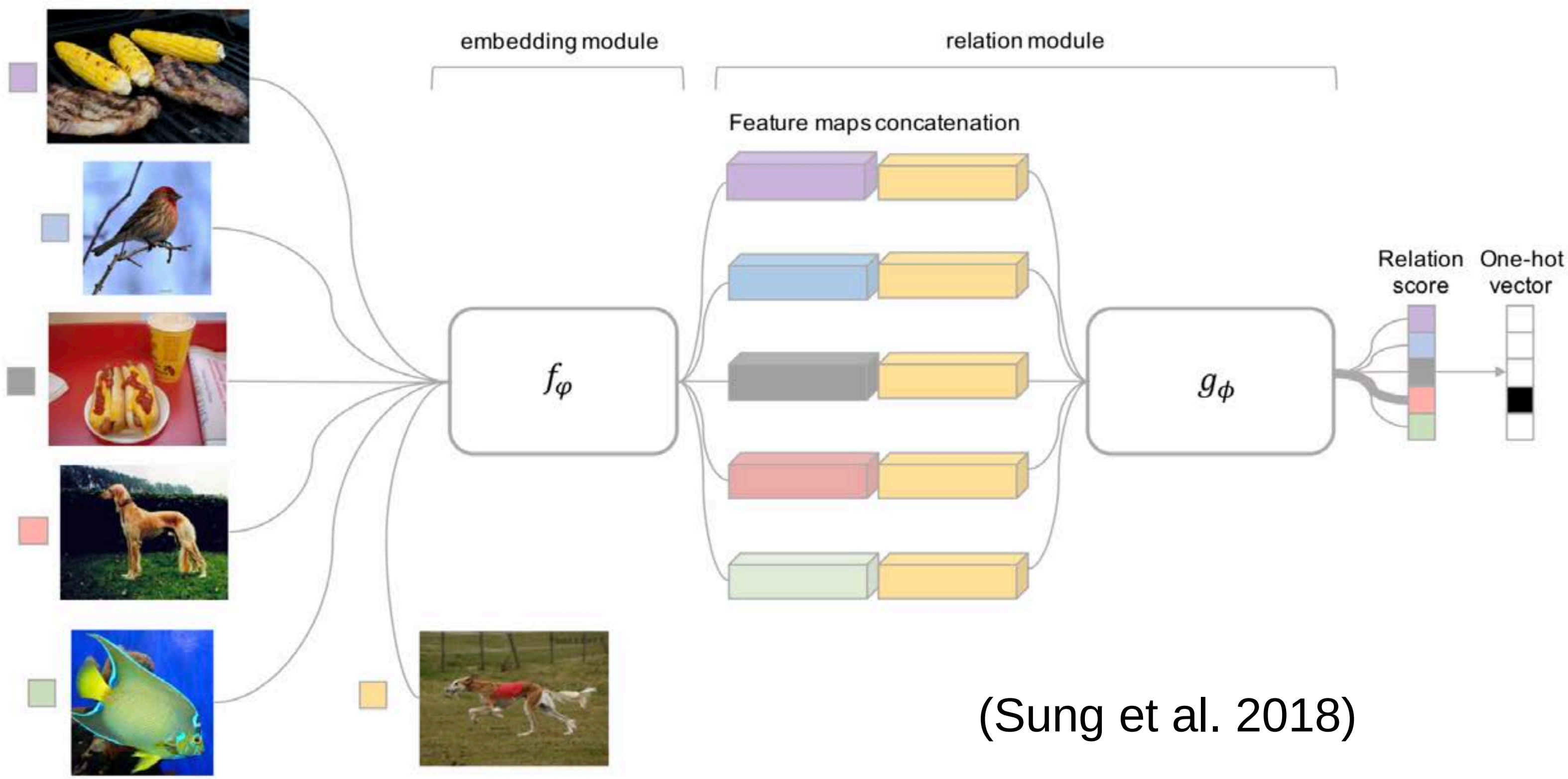
- The learned embedding can be generalized to unknown categories (Koch, Zemel & Salakhutdinov, 2015)

Matching Networks



- Full Contextual Embedding (Vinyals et al, 2016)
  - Encoding input in the context of the entire support set
  - The learned embedding can be adjusted based on the relationship with other support samples.

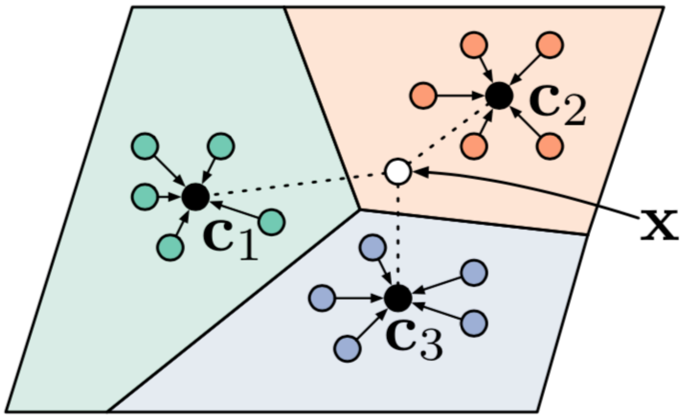
Relation Network



(Sung et al. 2018)

- Similar to Siamese network
- More complex metric learning $r_{ij} = g_\phi([\mathbf{x}_i, \mathbf{x}_j])$

Prototypical Networks



(a) Few-shot

$$\mathbf{v}_c = \frac{1}{|S_c|} \sum_{(\mathbf{x}_i, y_i) \in S_c} f_\theta(\mathbf{x}_i)$$

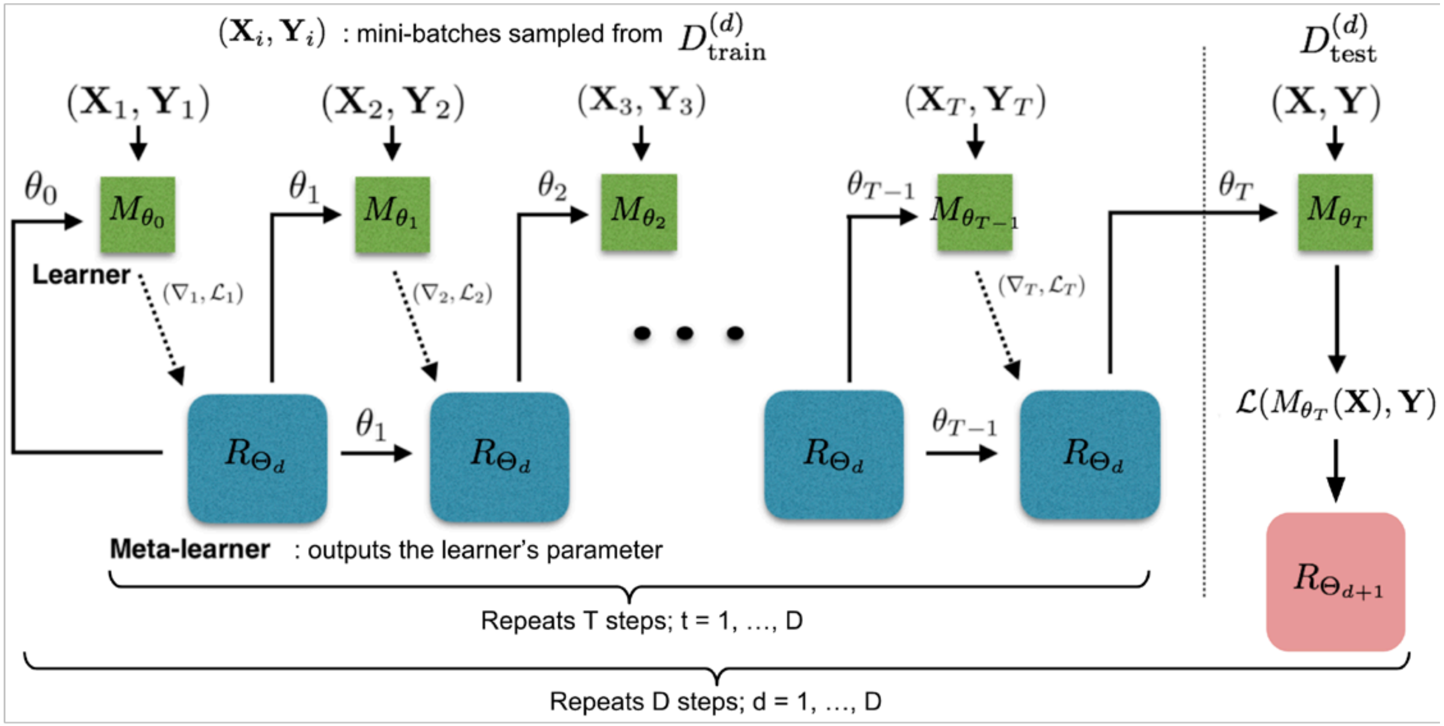
- Prototype vectors (Snell, Swersky & Zemel, 2017)

$$P(y = c|\mathbf{x}) = \mathrm{softmax}(-d_\varphi(f_\theta(\mathbf{x}), \mathbf{v}_c))$$

Optimization-based methods

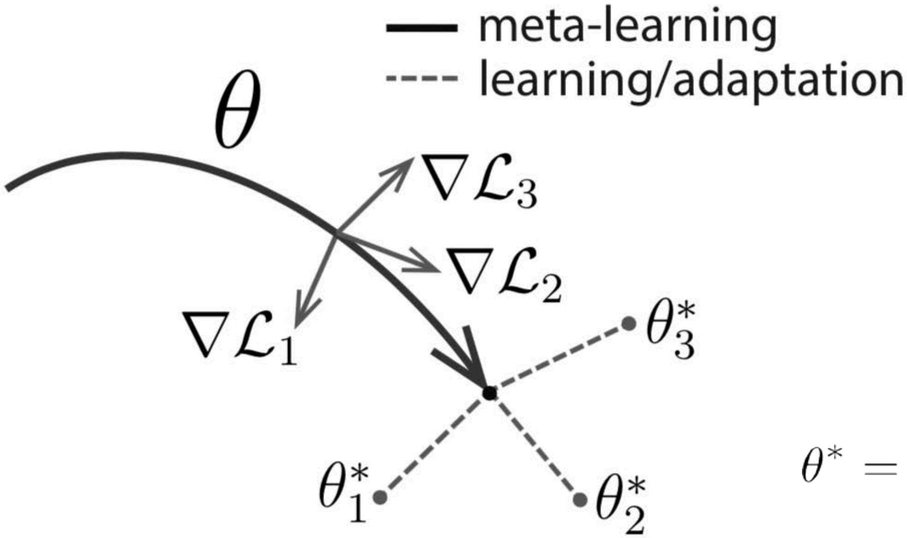
- Basic idea: Adjust the optimization in model learning so that the model can effectively learn from a few examples
- Typical methods
  - LSTM meta-learner (Ravi & Larochelle, 2017)
  - MAML (Finn, et al. 2017)
  - Reptile (Nichol, Achiam & Schulman, 2018)

LSTM meta-learner



- The optimization algorithm is explicitly modeled based on an LSTM meta-learner (Ravi & Larochelle, 2017)

MAML



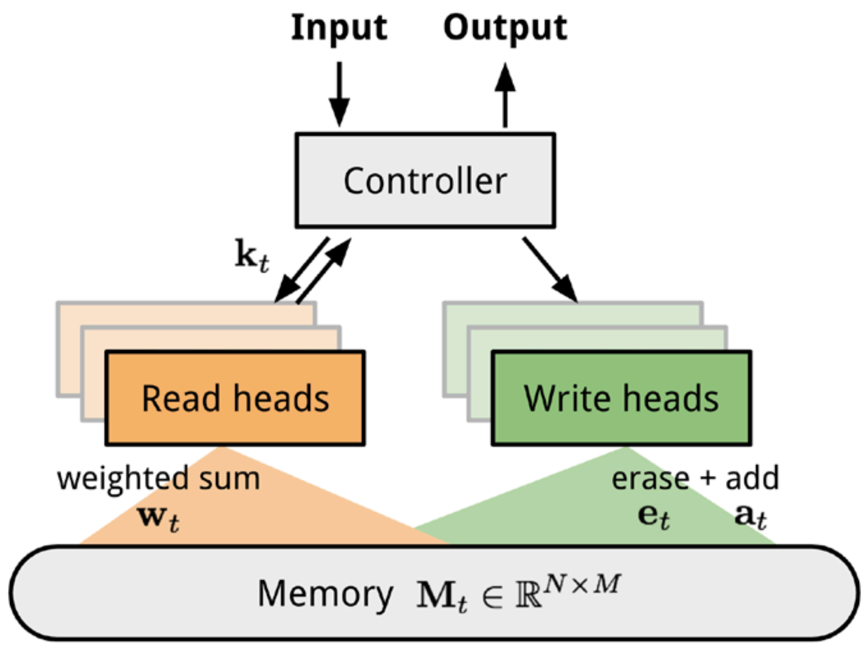
$$\theta^* = \arg\min_{\theta} \sum_{\tau_i \sim p(\tau)} \mathcal{L}^{(1)}_{\tau_i}\left(f_{\theta - \alpha \nabla_\theta \mathcal{L}^{(0)}_{\tau_i}(f_\theta)}\right)$$

- Model-Agnostic Meta-Learning (Finn, et al. 2017) aims to generate a fast gradient based learner

Model-based methods

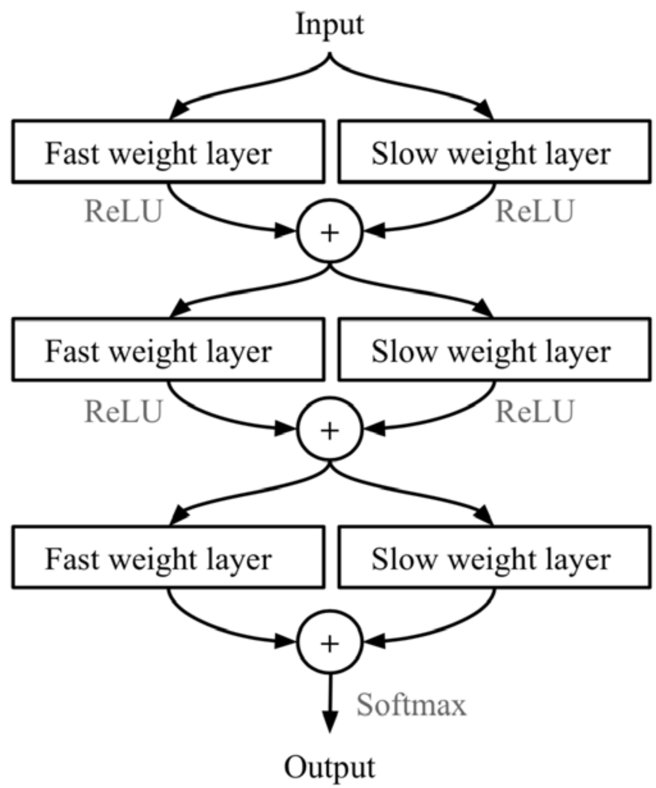
- Basic idea: Using a black-box neural network designed specifically for fast learning
- Typical methods
  - Memory-augmented network (Santoro et al., 2016)
  - Meta networks (Munkhdalai & Yu, 2017)
  - SNAIL (Mishra et al., 2018)

Memory-augmented network



- With an explicit storage buffer, it is easier for the network to rapidly incorporate new information.
- (Santoro et al., 2016) train it in a way that the memory can encode and capture information of new tasks fast and is easily and stably accessible.

Meta networks



- The MetaNet relies on “fast weights” to achieve rapid generalization across tasks (Munkhdalai & Yu, 2017)

Main limitations

- A global representation of inputs
  - Sensitive to nuisance parameters: background clutter, occlusions, etc.
- Mixed representation and predictor learning
  - Complex architecture, difficult to interpret
  - Sometimes slow convergence
- Focusing on classification tasks
  - Non-trivial to apply to other vision tasks: localization, segmentation, etc.

Our proposed solutions

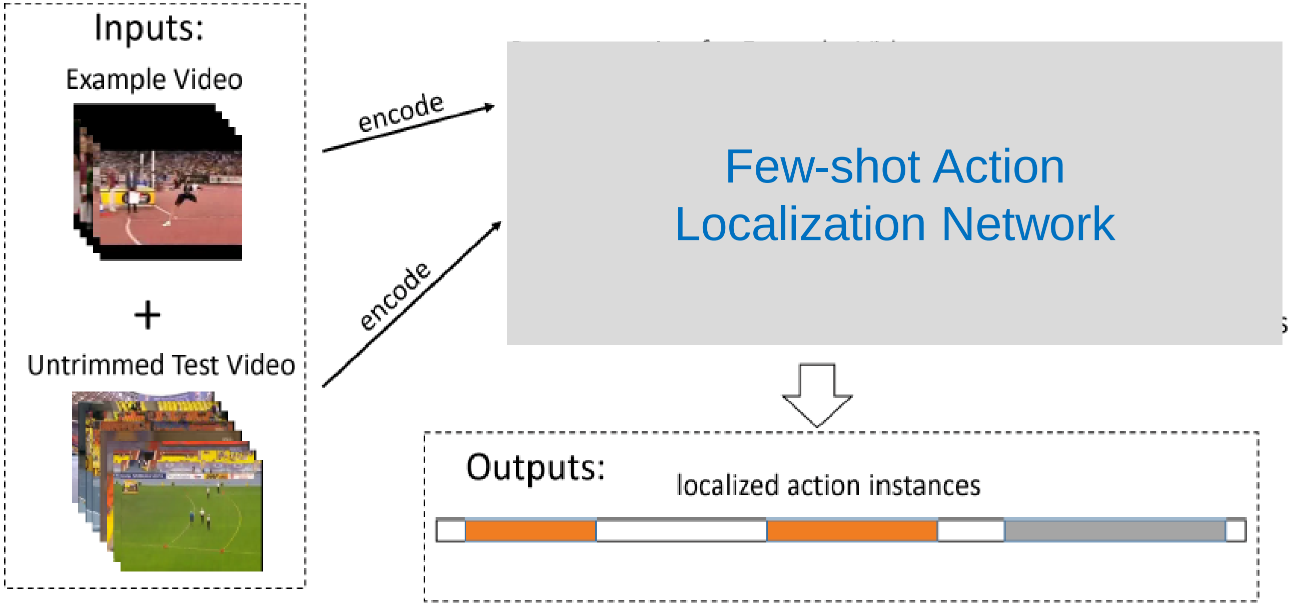
- Structure-aware data representation
  - Spatial/temporal representations for semantic objects/actions
- Decoupling representation and classifier learning
  - Improving representation learning
- Generalizing to other visual tasks
  - Instance localization and detection with few-shot learning

Temporal action localization

- Our goal: Jointly classify action instances and localize them in an untrimmed video
  - Important for detailed video understanding
  - Broad range of applications in video surveillance/analytics

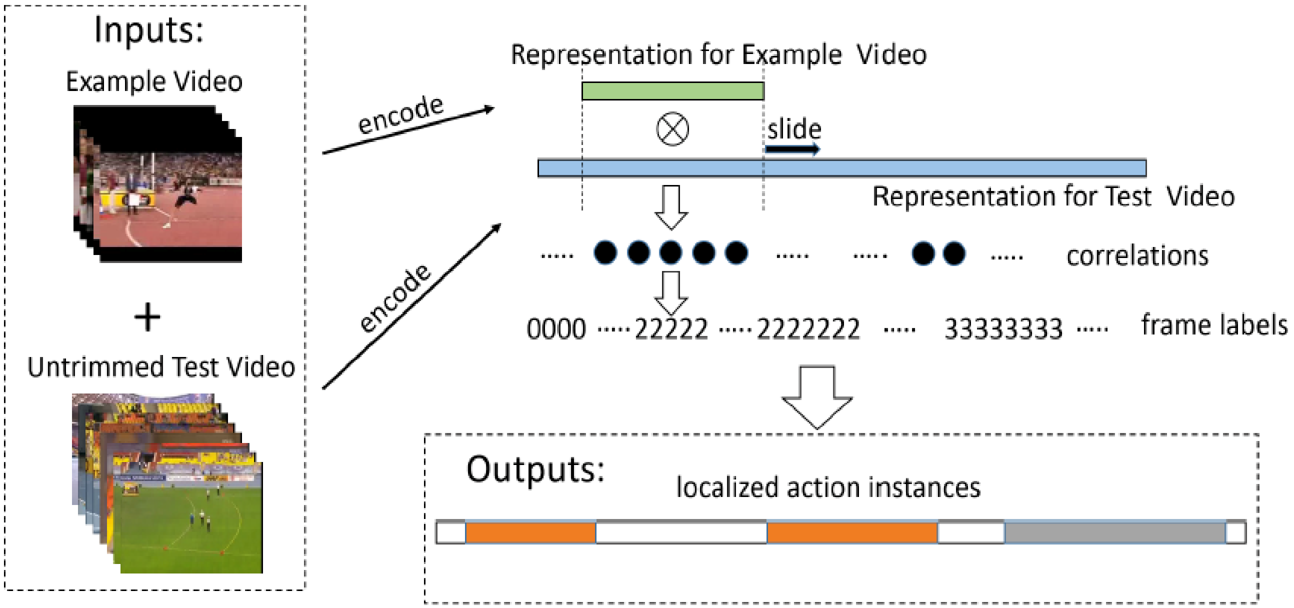
Our problem setting

- We conceptualize an example-based action localization strategy
  - Few-shot learning of action classes and
  - Being sensitive to action boundaries

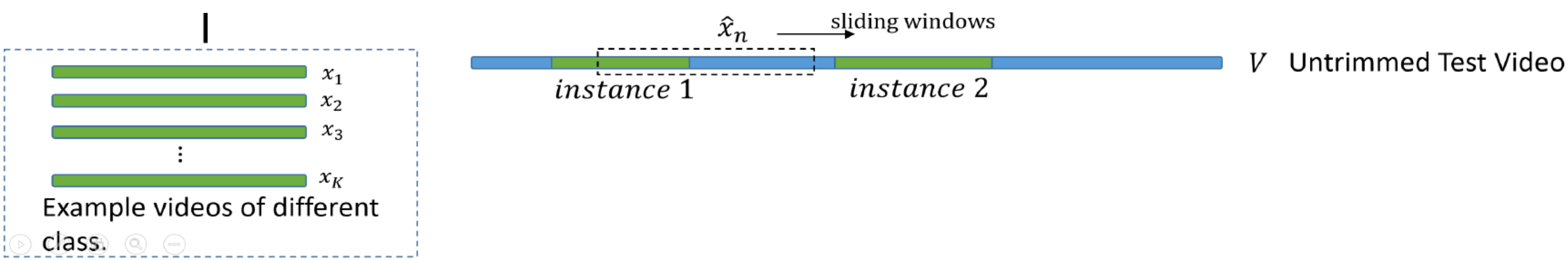


Main ideas

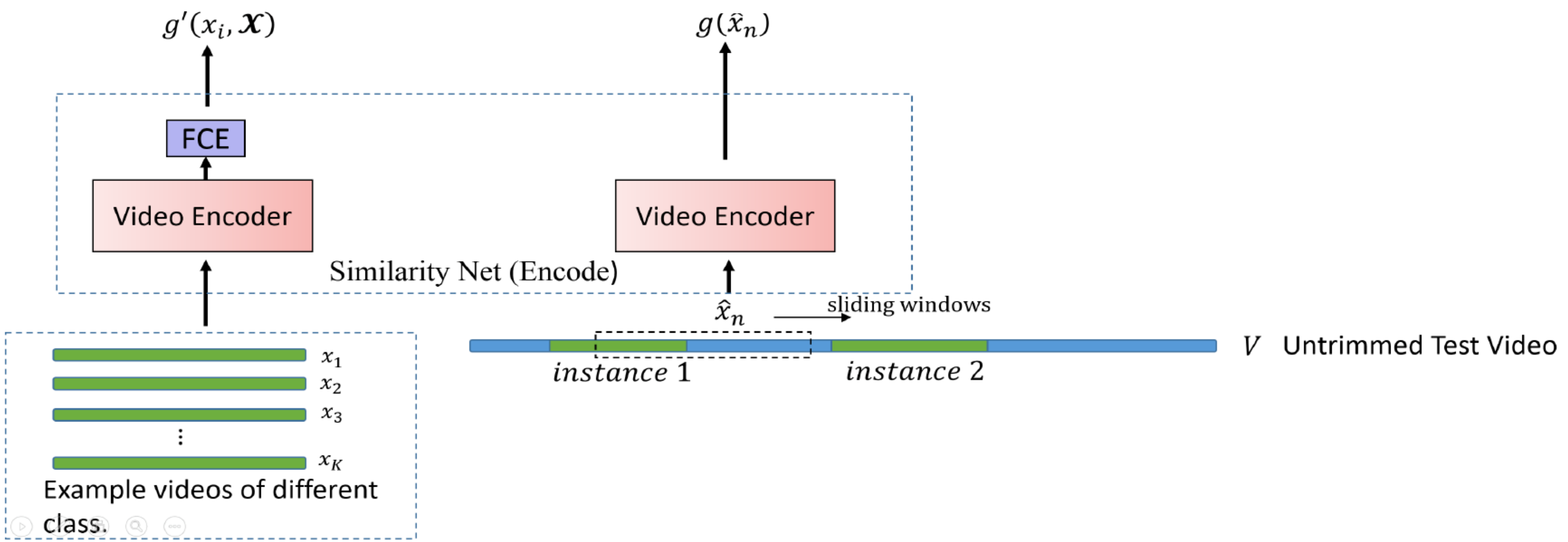
- Meta-learning problem formulation
  - Learning how to transfer the labels of a few action examples to a test video
    - Encode action instance into a structured representation
    - Learn to match (partial) action instances
    - Exploit the matching correlation scores



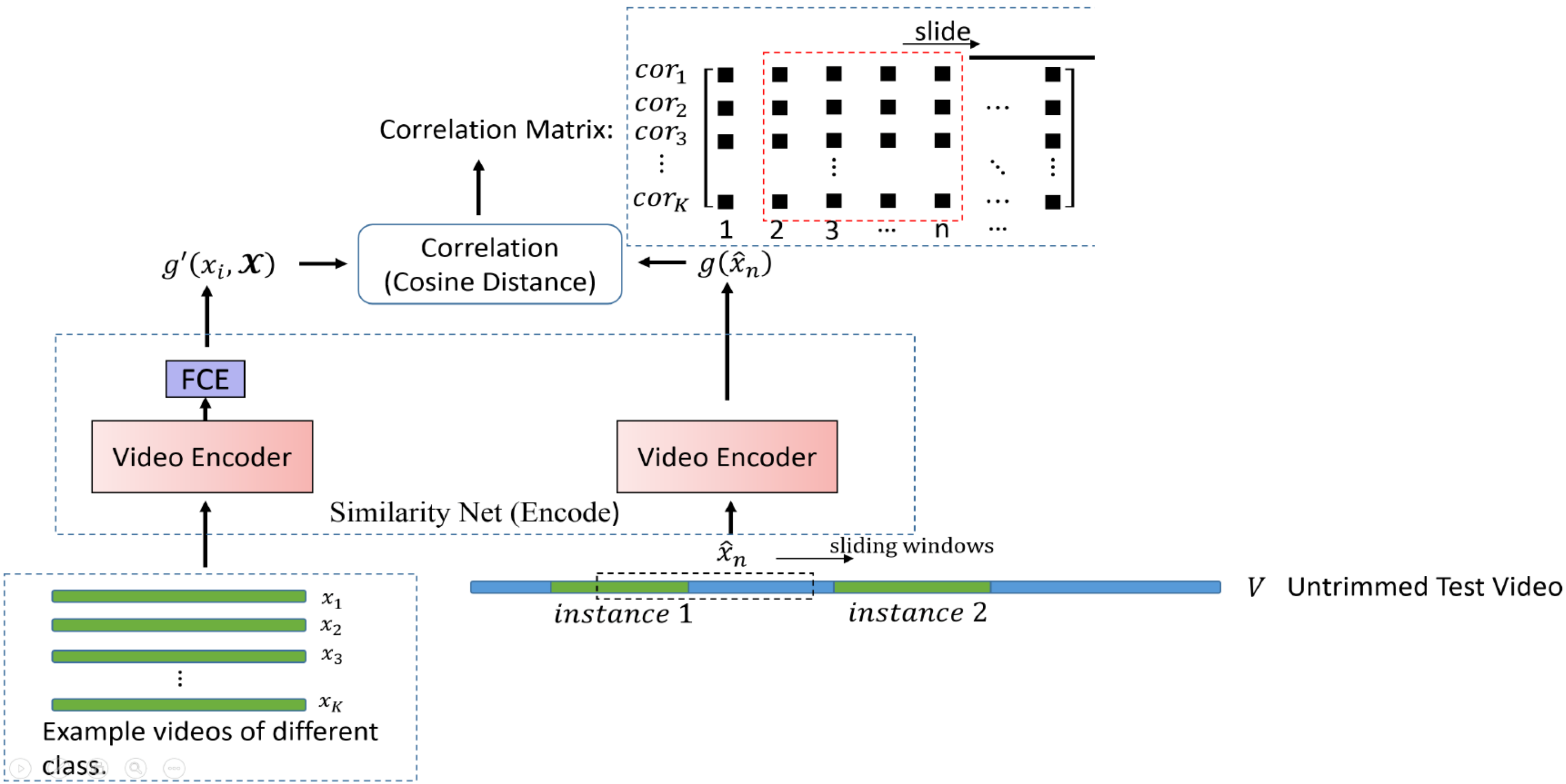
Overview of our method



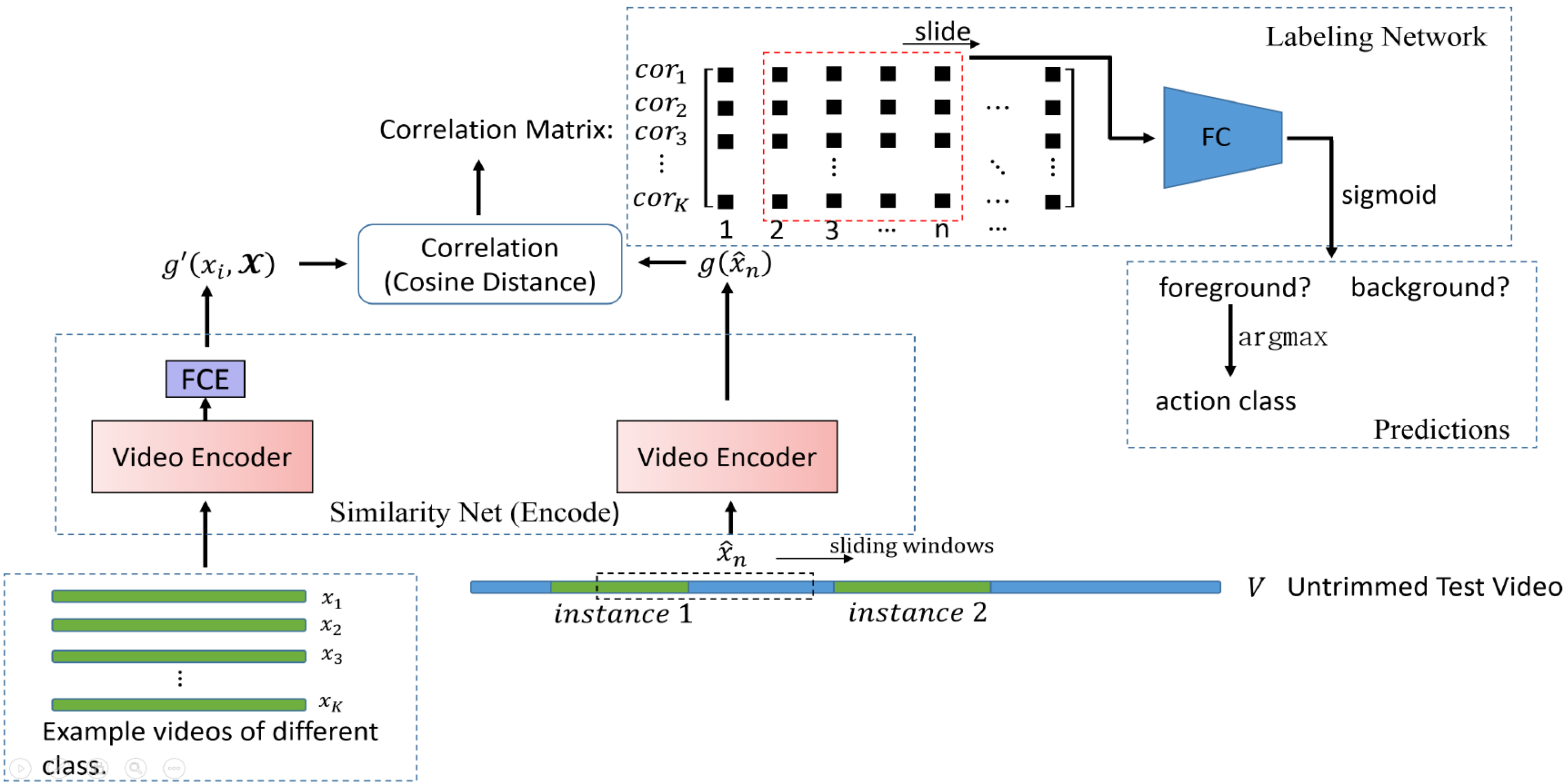
Overview of our method



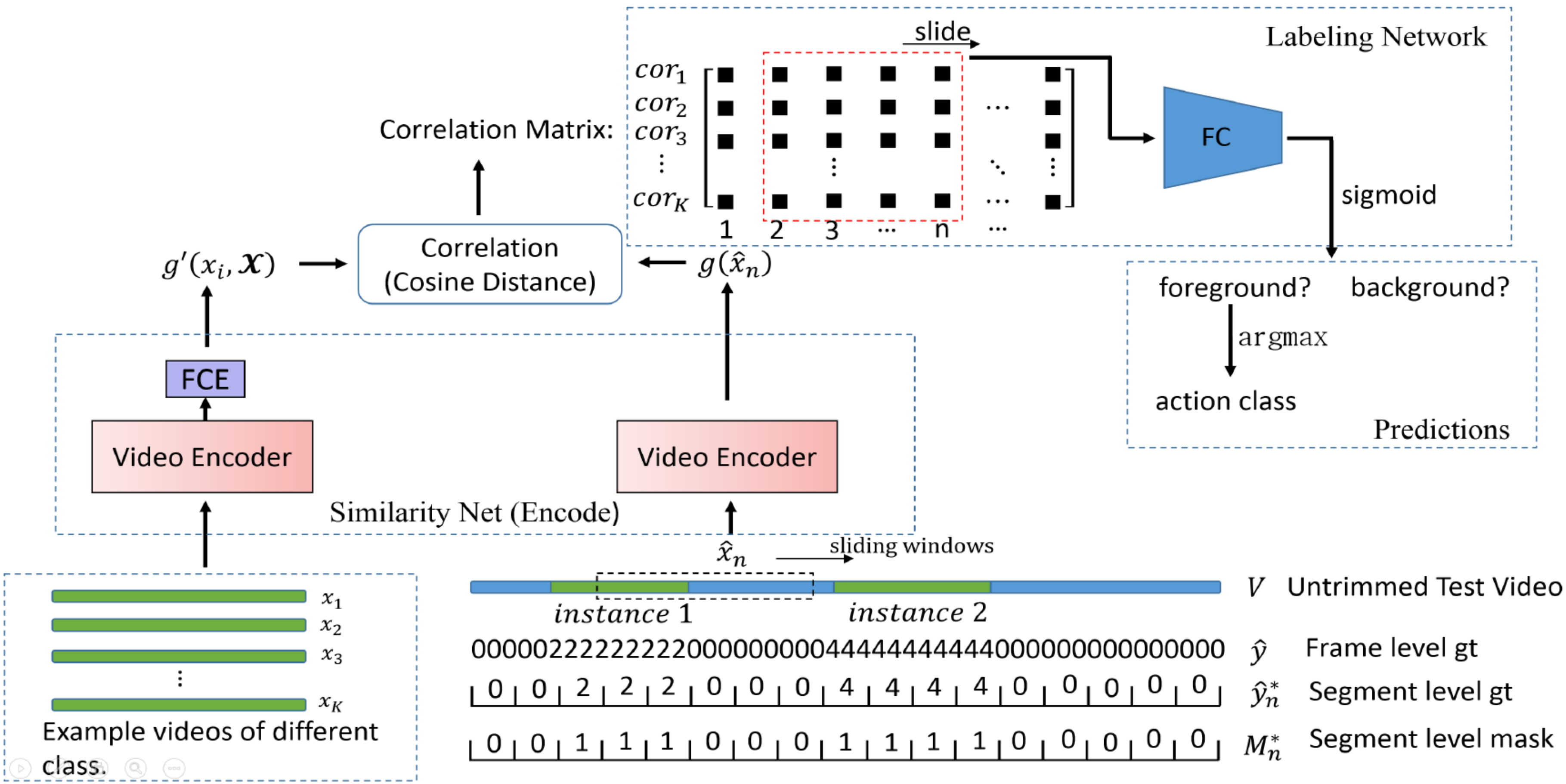
Overview of our method



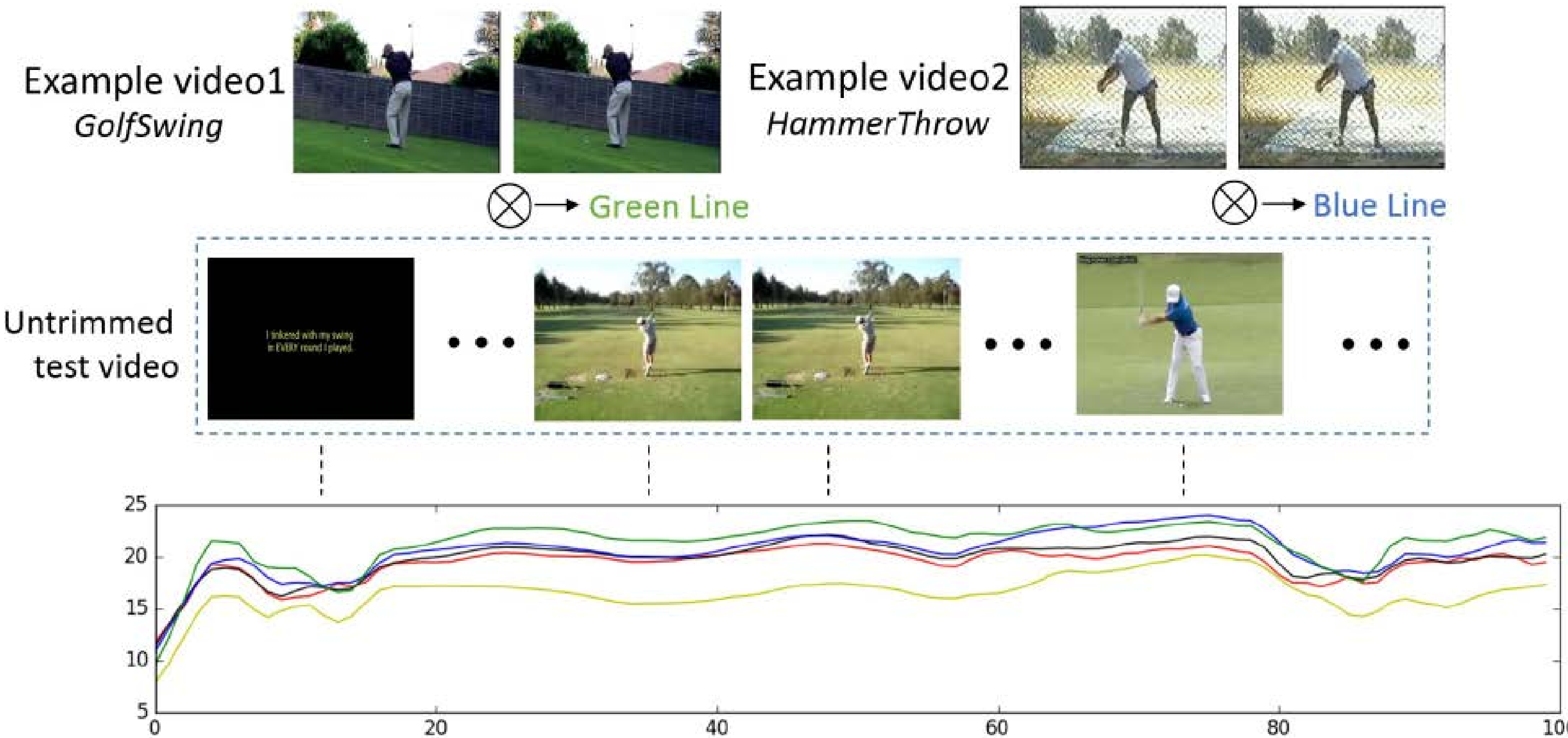
Overview of our method



Overview of our method



Matching examples



Matching score trajectories

Meta-learning strategy

- Meta-training phase
  - Meta-training set $\mathcal{T}_{meta-train} = \{\mathcal{X}, \hat{\mathcal{X}}, \mathcal{L}(\mathcal{X}, \hat{\mathcal{X}}, \theta)\}$
  - Task-train (support set) $\mathcal{X} = \{x_i, y_i\}$
  - Task-test (query) $\hat{\mathcal{X}} = \{\hat{x}_j, \hat{y}_j\}$
  - Loss function $\mathcal{L}$
- Our loss function
  - Localization loss: foreground vs background (cross entropy)
  - Classification loss: action class (log loss)

$$L = \mathbb{E}_{\mathcal{T} \sim \mathcal{T}_{meta-train}}[L_{loc} + L_{cls}]$$

  - Ranking loss: replacing localization loss to encourage partial alignment

Experimental evaluation

- Few-shot performance summary
  - ~80 classes for meta-training and ~20 for meta-test

Fully supervised	mAP	Few-shot	mAP
Heilbron *et al.* [5]	13.5	Ours@1	13.6
Yeung *et al.* [49]	17.1	Ours@5	14.0
Yuan *et al.* [50]	17.8	Ours@15	14.7
S-CNN [35]	19.0	CDC@1	6.4
S-CNN + SST [4]	23.0	CDC@5	6.5
CDC [34]	23.3	CDC@15	6.8

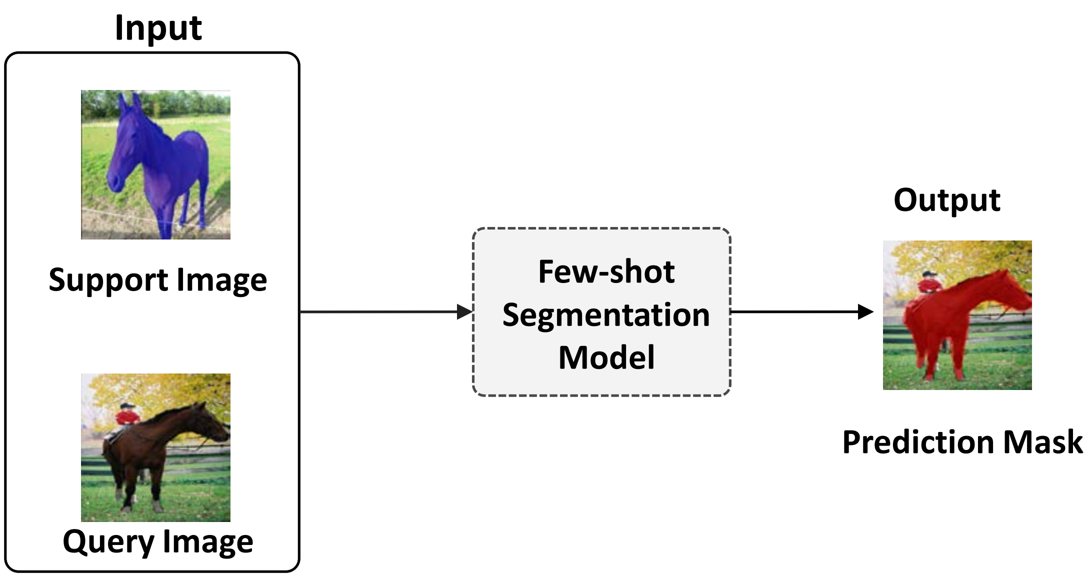
Thumos14

	mAP@0.5	Average mAP
TCN [8]	37.4	23.5
R-C3D [48]	-	26.8
Wang *et al.* [26]	42.2	14.8
Lin *et al.* [27]	48.9	32.2
Xiong *et al.* [47]	41.1	24.8
CDC [34]	43.8	22.7
Ours@1	22.3	9.8
Ours@5	23.1	10.0
CDC@1	8.2	2.4
CDC@5	8.6	2.5

ActivityNet

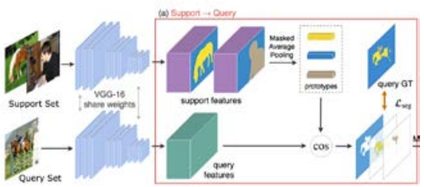
Problem Definition

- Few-shot semantic segmentation aims to segment **new** semantic objects in an image with only **a few annotated examples**.



Related Works

Prototype-Based Algorithm:



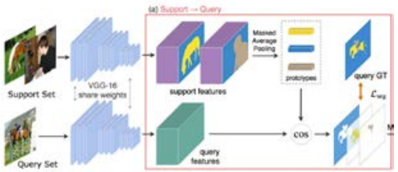
Wang et al. *ICCV19*
Dong et al. *BMVC18*

- Hard to model object's scale and appearance variations
- Easy to saturate with multi-shots

立志成才报国裕民

Related Works

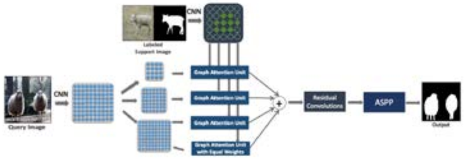
Prototype-Based Algorithm:



Wang et al. *ICCV19*
Dong et al. *BMVC18*

- Hard to model object's scale and appearance variations
- Easy to saturate with multi-shots

Parametric-Based Algorithm:



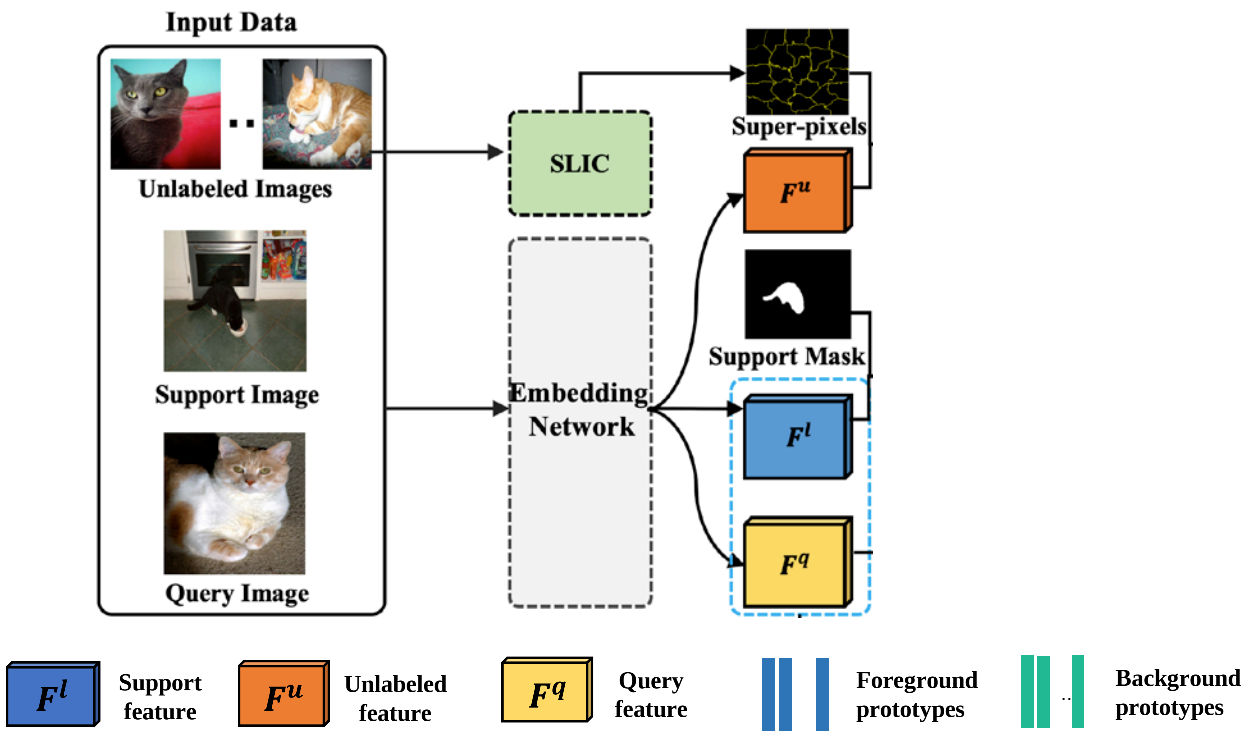
Zhang et al. *ICCV19*
Zhang et al. *CVPR19*

- Hard to adapt to multi-way few shot segmentation
- High model complexity

Challenges

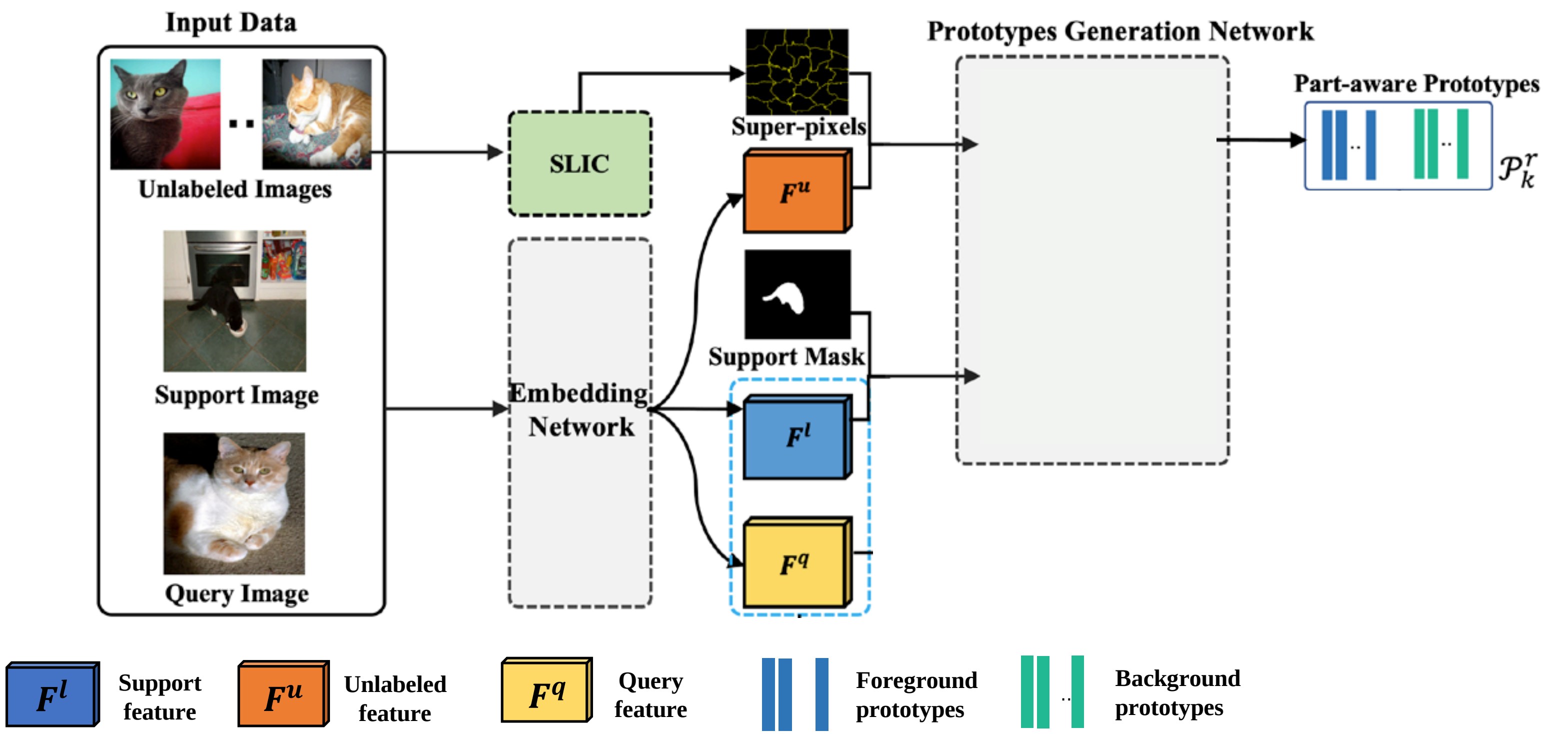
Challenges	Our solutions
Global prototype representation lacks detailed information of novel objects.	Part-based prototype representation reserves more detailed information.
Large appearance & scale variation between support and query images.	Utilize the unlabeled images to handle variations.
Less effective to learn a good visual representation for segmentation.	Add semantic branch to learn a better visual representation.

Method



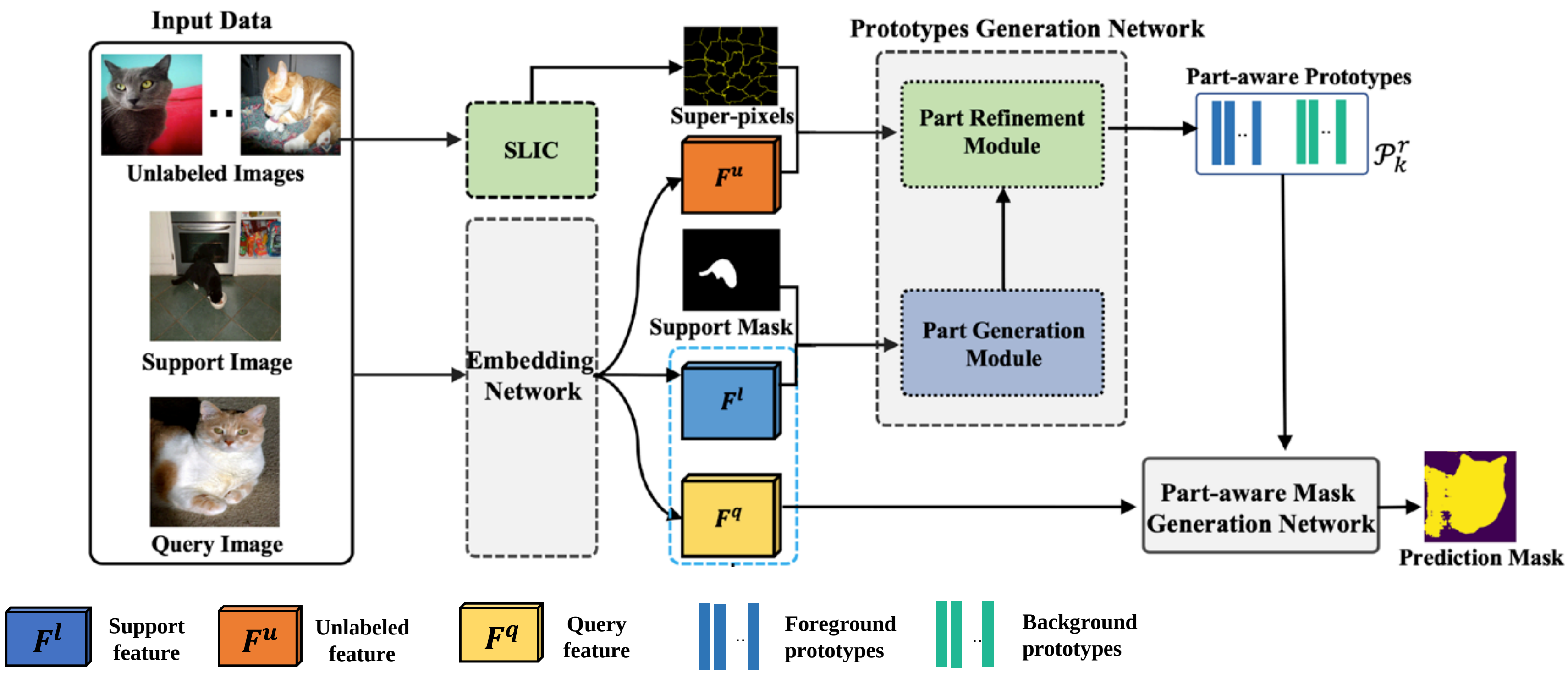
立志成才报国裕民

Method



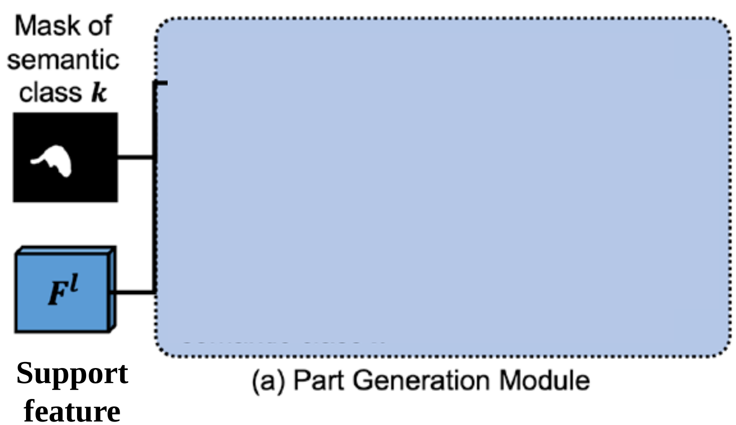
Input Data
Unlabeled Images
Support Image
Query Image
SLIC
Embedding Network
Super-pixels
F^u
Support Mask
F^l
F^q
Prototypes Generation Network
Part-aware Prototypes
$\mathcal{P}_k^r$
F^l
Support feature
F^u
Unlabeled feature
F^q
Query feature
Foreground prototypes
Background prototypes

Method


Input Data
Unlabeled Images
Support Image
Query Image
SLIC
Embedding Network
Super-pixels
F^u
Support Mask
F^l
F^q
Prototypes Generation Network
Part Refinement Module
Part Generation Module
Part-aware Prototypes
$\mathcal{P}_k^r$
Part-aware Mask Generation Network
Prediction Mask
F^l Support feature
F^u Unlabeled feature
F^q Query feature
Foreground prototypes
Background prototypes

Method

Part Generation Module



(a) Part Generation Module

Part Generation Module aims to generate **the initial part-aware prototypes** on support images.

- Build a set of part-aware prototypes to capture **fine-grained part-level variation.**
- Further augment each initial prototype with a **global context** of the semantic class.

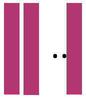
Method

Part Refinement Module

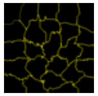
(b) Part Refinement Module

Part Refinement Module **further improve** part-aware prototypes representation with unlabeled images features.

Unlabeled image feature



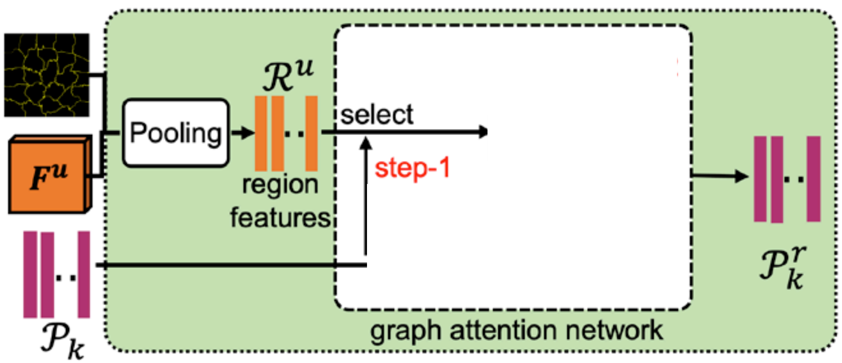
$\mathcal{P}_k$: part-aware prototypes



Unlabeled image superpixel

Method

Part Refinement Module

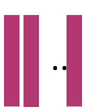


(b) Part Refinement Module

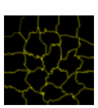
Part Refinement Module **further improve** part-aware prototypes representation with unlabeled images features.

Step-1: Relevant feature generation.

Unlabeled image feature



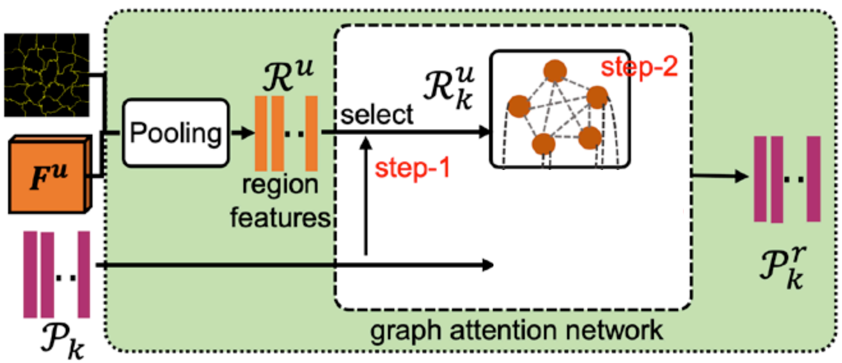
$\mathcal{P}_k$: part-aware prototypes



Unlabeled image superpixel

Method

Part Refinement Module



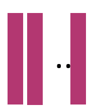
(b) Part Refinement Module

Part Refinement Module **further improve** part-aware prototypes representation with unlabeled images features.

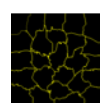
Step-1: Relevant feature generation.

Step-2: Unlabeled feature augmentation.

Unlabeled image feature



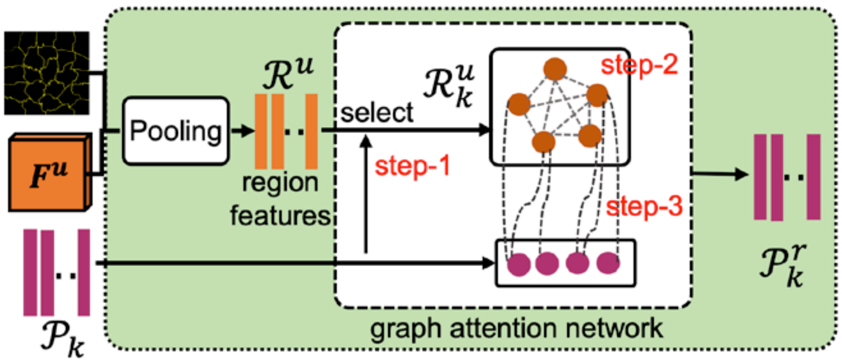
$\mathcal{P}_k$: part-aware prototypes



Unlabeled image superpixel

Method

Part Refinement Module



(b) Part Refinement Module

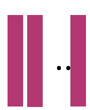
Part Refinement Module **further improve** part-aware prototypes representation with unlabeled images features.

Step-1: Relevant feature generation.

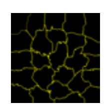
Step-2: Unlabeled feature augmentation.

Step-3: Part-aware prototype refinement.

Unlabeled image feature

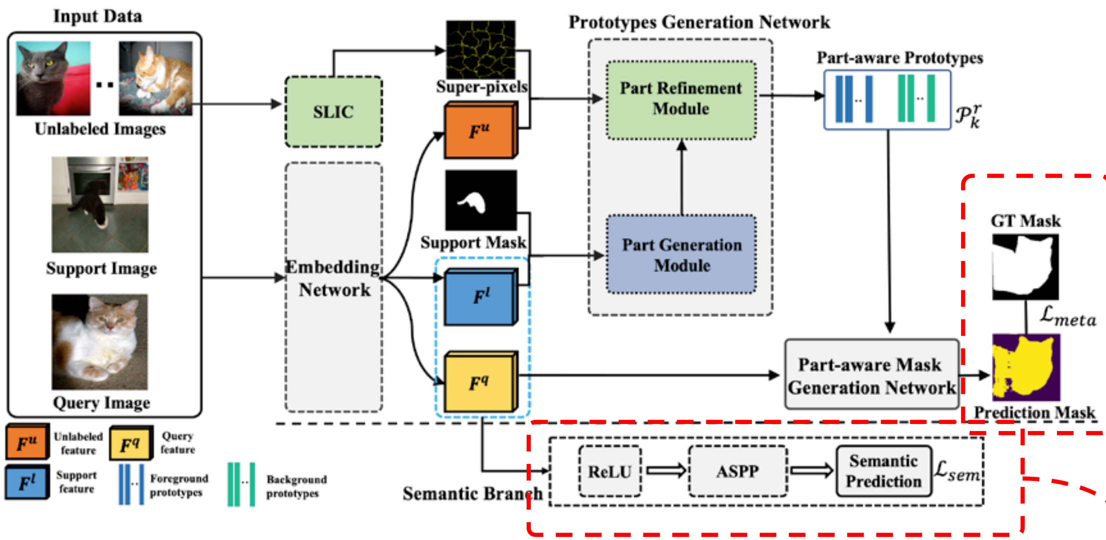


$\mathcal{P}_k$: part-aware prototypes



Unlabeled image superpixel

Model Learning



① $$\boldsymbol{L_{meta}} = \boldsymbol{L_{ce}}(\hat{Y}^q, Y^q) + \boldsymbol{L_{ce}}(\hat{Y}^l, Y^l)$$

- Y^q, Y^l are ground-truth mask.

② $$\boldsymbol{L_{sem}} = \boldsymbol{L_{ce}}(\hat{Y}^q_{sem}, Y^q_{sem}) + \boldsymbol{L_{ce}}(\hat{Y}^l_{sem}, Y^l_{sem})$$

- Y^q_{sem}, Y^l_{sem} are ground-truth mask over global semantic classes.

$$\boldsymbol{L_{full}} = \boldsymbol{L_{meta}} + \boldsymbol{\beta L_{sem}}$$

Results

COCO-20^i Performance (1-way)

Methods	Split	Backbone	1-shot fold-1	1-shot fold-2	1-shot fold-3	1-shot fold-4	1-shot mean	5-Shot fold-1	5-Shot fold-2	5-Shot fold-3	5-Shot fold-4	5-Shot mean
PANet [34]	A	VGG16	28.70	21.20	19.10	14.80	20.90	39.43	28.30	28.20	22.70	29.70
PANet* [34]	A	RN50	31.50	22.58	21.50	16.20	22.95	45.85	29.15	30.59	29.59	33.80
Our(**w/o** $\mathcal{S}^u$)	A	RN50	34.53	25.44	24.33	18.57	25.71	48.30	30.90	35.65	30.20	36.24
Our	A	RN50	**36.48**	**26.53**	**25.99**	**19.65**	**27.16**	**48.88**	**31.36**	**36.02**	**30.64**	**36.73**
FWB [21]	B	RN101	16.98	17.78	20.96	**28.85**	21.19	19.13	21.46	23.39	30.08	23.05
Our	B	RN50	**28.09**	**30.84**	**29.49**	27.70	**29.03**	**38.97**	**40.81**	**37.07**	**37.28**	**38.53**

Split A +4.21 +2.93

Split B +7.84 +15.48

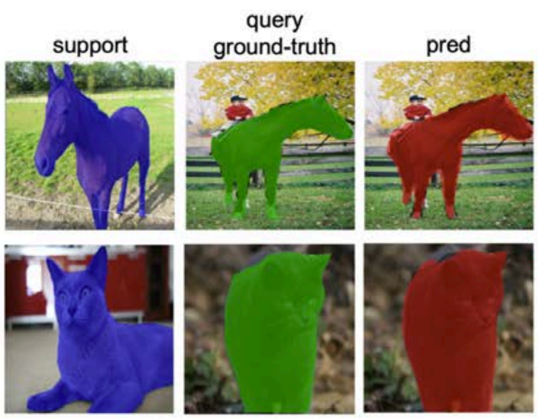
COCO-20^i Performance (2-way & 5-way)

Methods	Backbone	2-way, 1-shot fold-1	fold-2	fold-3	fold-4	mean	5-way, 1-shot fold-1	fold-2	fold-3	fold-4	mean
PANet [34]	VGG16	29.88	21.13	20.46	15.37	21.71	24.94	19.85	19.28	14.11	19.55
PANet* [34]	RN50	31.86	21.47	21.31	16.43	22.76	27.20	21.50	19.66	15.35	20.93
PPNet(**w/o** $\mathcal{S}^u$)	RN50	33.87	23.98	22.75	17.59	24.55	29.12	22.29	21.10	16.37	22.22
PPNet	RN50	**34.20**	**24.21**	**23.39**	**19.06**	**25.22**	**30.84**	**23.03**	**21.32**	**17.93**	**23.28**

+2.46 +2.35

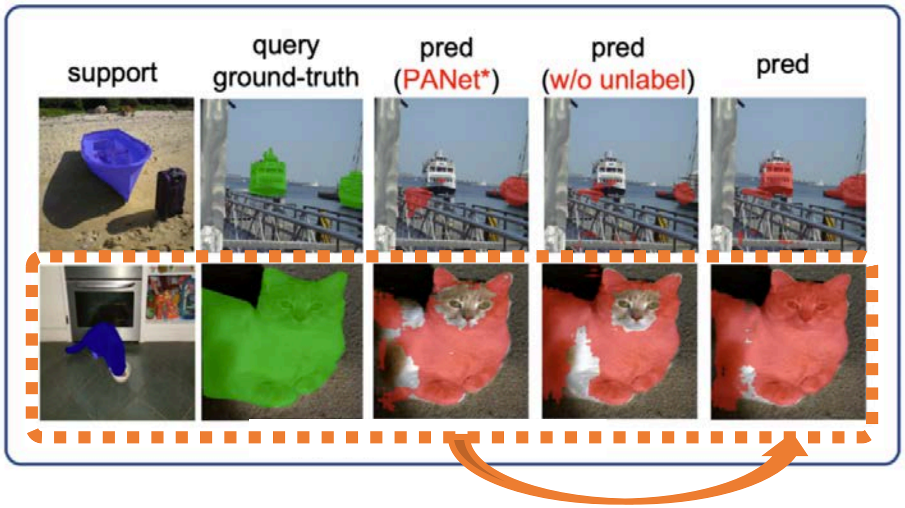
Visualization

Part visualization on Pascal 5^i (1-way)

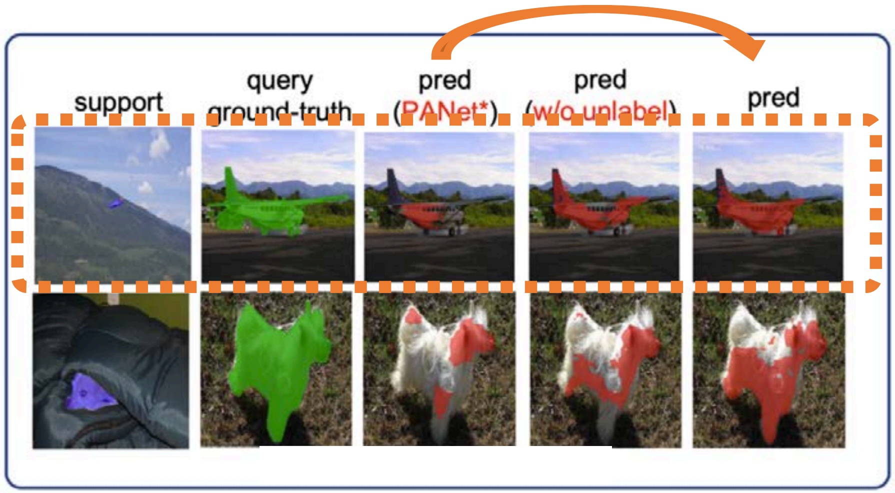


Visualization

Qualitative Visualization by utilizing unlabeled data on Pascal 5^i visualization (1-way)



Appearance Variations



Scale Variations