Deep Learning for Computer Vision

Fall 2021

http://vllab.ee.ntu.edu.tw/dlcv.html (Public website)

https://cool.ntu.edu.tw/courses/8854 (NTU COOL)

Yu-Chiang Frank Wang 王鈺強, Professor

Dept. Electrical Engineering, National Taiwan University

2021/12/7

Some Updates…

- Syllabus

11	12/07	Meta-Learning for Visual Analysis (I)		
12	12/14	Meta-Learning for Visual Analysis (II); Self-Supervised Learning for Visual Analysis		HW #3 due; HW #4-1 out Final Project Announcement
13	12/21	Vision and Language		HW #4-2 (bonus & optional?)
14	12/28	Beyond 2D Vision (3D and Depth)		
15	01/04	Guest Lectures (TBD)		HW #4 due
16	01/11	Final Week (no class)		
17	01/18	Presentation for Final Projects		

- Final Challenge/Project
  - At least one company is sponsoring the final challenge, still confirming another
  - Considering the size of the class, 4 students per group is preferable
    - No fewer then 3 and no more than 5
    - Inter/intra group evaluation will be conducted
    - Start looking for your teammates!

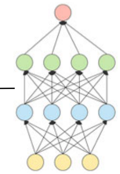
What to Cover Today…

- Meta-Learning
  - Definition
  - Parametric & Non-Parametric based Approaches
- Meta-Learning for Few-Shot Learning
  - Few-Shot Classification
    - Metric Learning vs. Data Hallucination
  - Few-Shot Image Segmentation
  - Few-Shot Object Detection (probably next lecture)
- Meta-Learning for Domain Generalization (probably next lecture)
  - From Domain Adaptation to Domain Generalization
- Challenges in Few-Shot Learning Tasks (next lecture)

Selected slide credits: C. Finn, S. Levine, & H.-Y. Lee

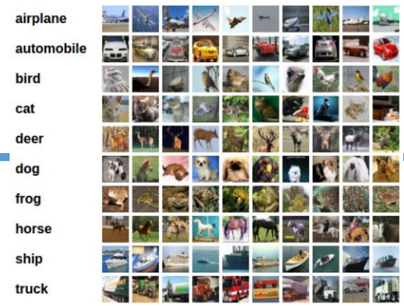
Meta Learning 元學習

- Meta Learning ⊆ Supervised Learning
- For Supervised Learning,
  - Given training data **D** = {**X**, **Y**}, learn function/model ***f*** so that ***f***(x_i) = y_i

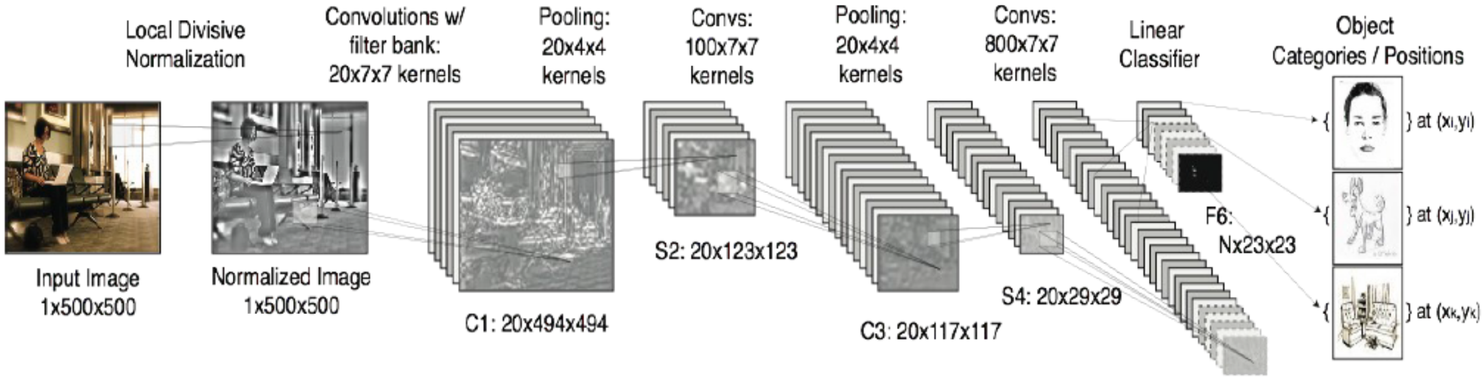


$f($  $)$ = “Cat”

Training data **X**

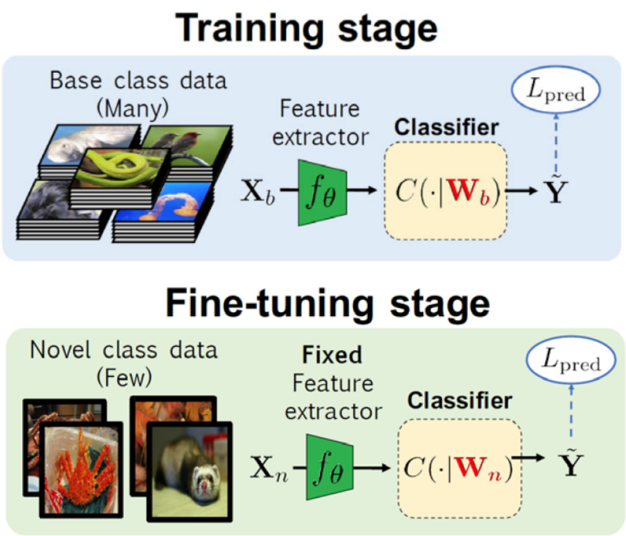


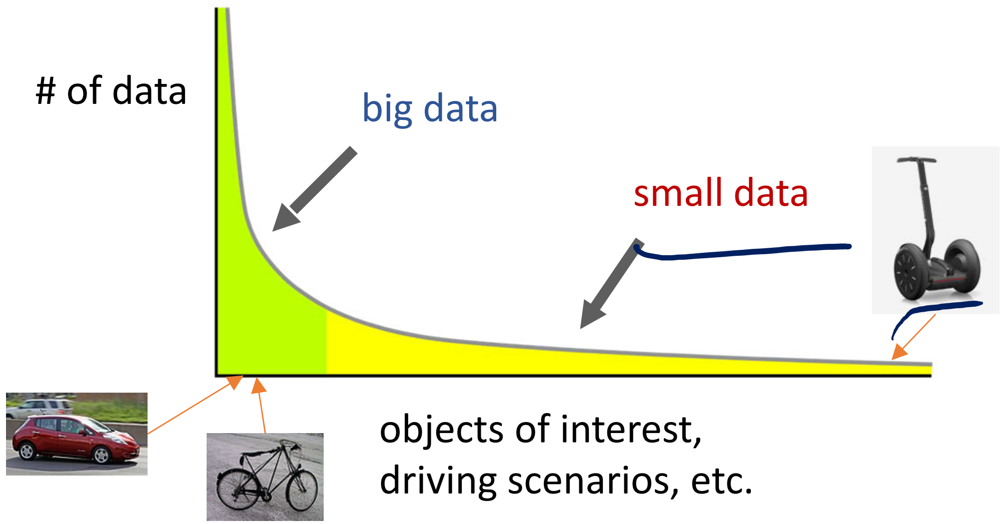
Ground truth labels **Y**



What If Only Limited Amount of Data Available?

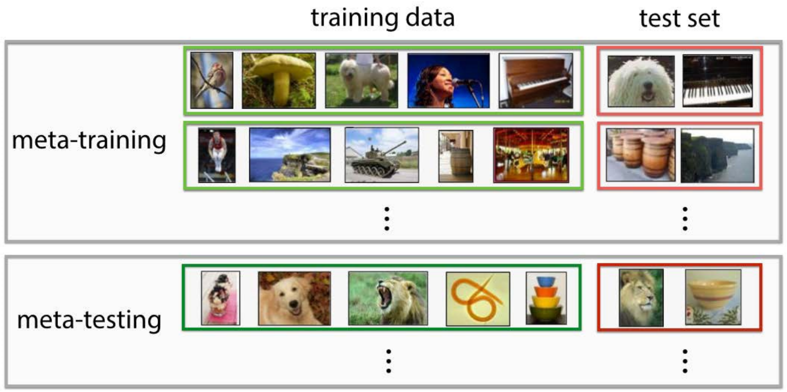
- **Naive transfer?**
  - **Model finetuning**:
    e.g., Train a learning model (e.g., CNN) on large-size data (base classes), following by finetuning on small-size data (novel classes).
    - That is, freeze feature backbone (learned from base classes) and learn classifier weights for novel classes.
  - Possibly **poor generalization** ☹





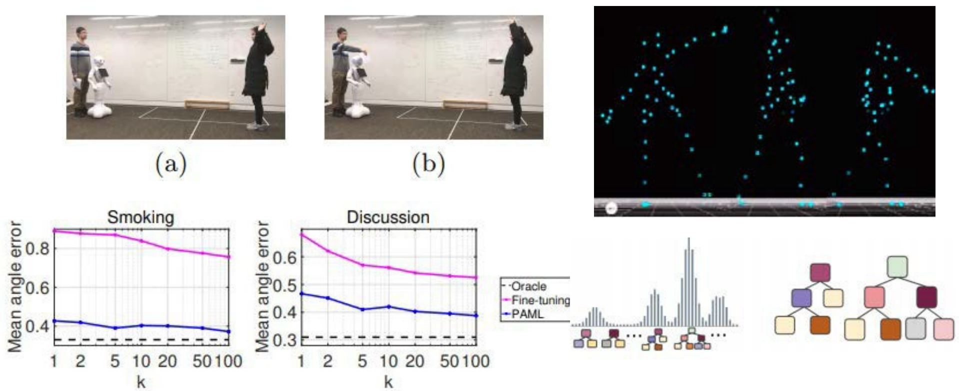
Selected Applications of Few-Shot Learning in Computer Vision

- ✓ Few-Shot Image Classification



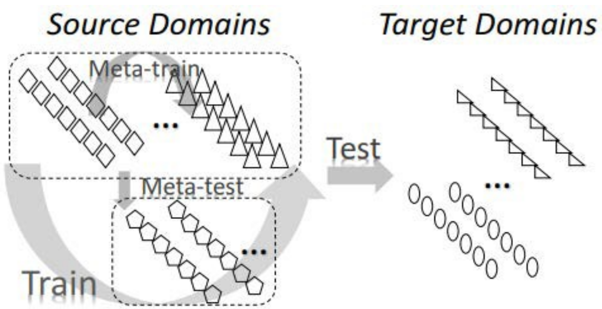
Vinyals et al., NIPS 2016

- Human Pose/Motion Prediction



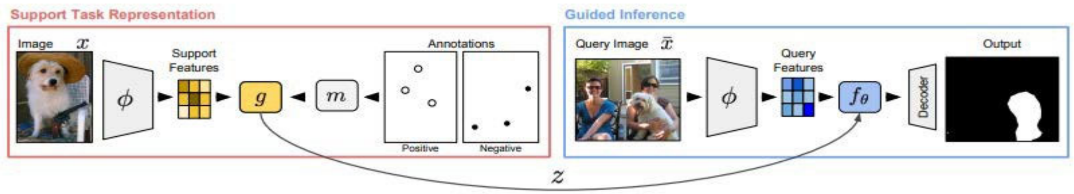
Gui et al., ECCV 2018

- Domain Transfer/Generalization



Li et al., AAAI 2018

- ✓ Few-Shot Image Segmentation



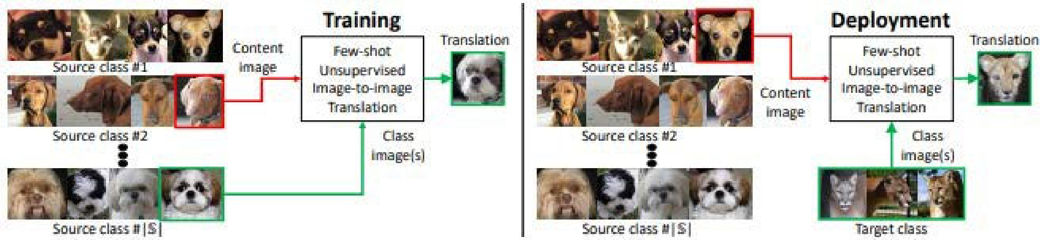
Wang et al., ICCV 2019

Selected Applications of
Few-Shot Learning in Computer Vision

- Few-Shot Image Generation

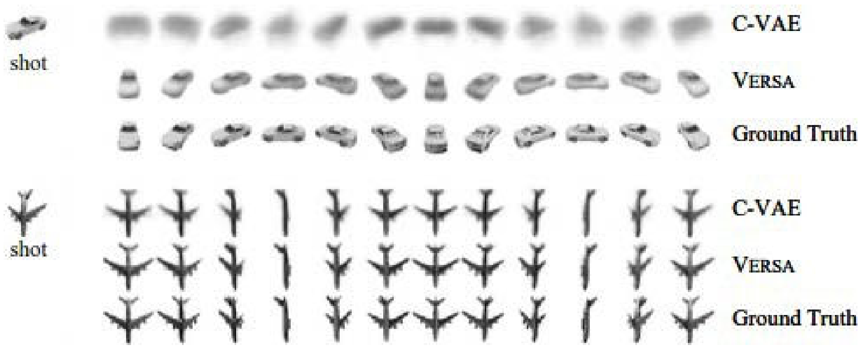
Reed et al., ICLR 2018

- Few-Shot Image-to-Image Translation



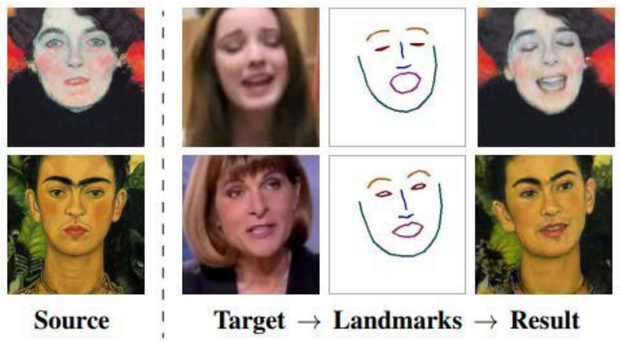
Liu et al., ICCV 2019

- Generation of Novel Viewpoints



Gordon et al., NIPS Workshop 2018

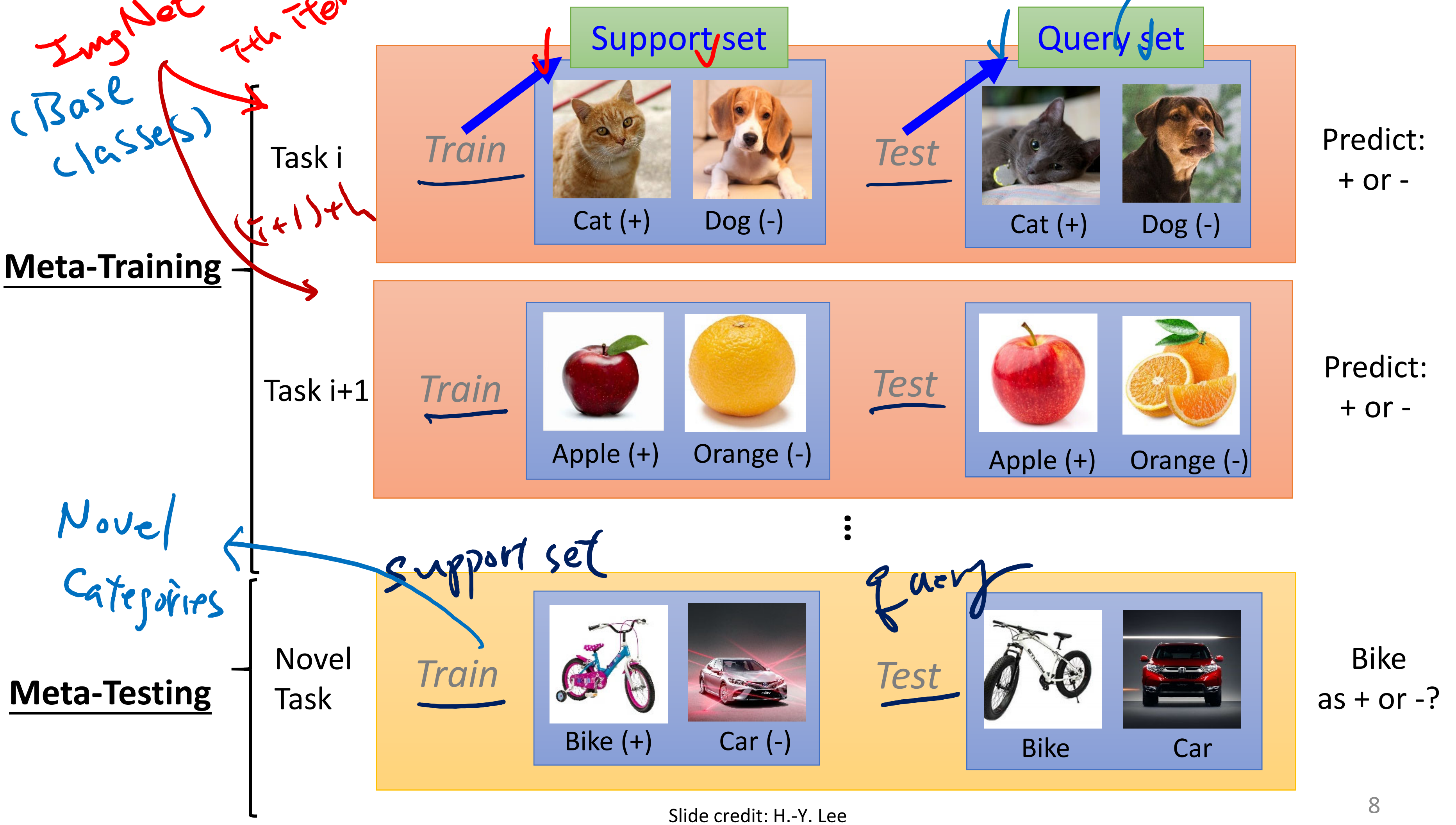
- Generating Talking Heads from Images



Zakharov et al., ICCV 2019

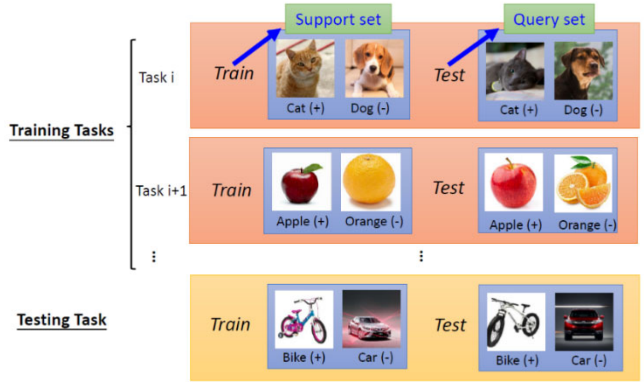
Meta Learning = Learning to Learn

- A powerful solution for learning from few-shot data
- Let's consider the following "2-way 1-shot" learning scheme:



Slide credit: H.-Y. Lee

Meta Learning (cont'd)



- Two Ways to View Meta Learning
  - ***Probabilistic*** **View**
    - Extract prior info from a set of (meta training) tasks, allowing efficient learning of a new task
    - Learning a new task uses this prior and (small) training set to infer most likely posterior model parameters
    - Easy to understand meta learning algorithms
  - ***Mechanistic*** **View**
    - A learning model (e.g., DNN) reads in a meta-training dataset, which consists of many datasets, each for a different task
    - Then, the model observes new data points (for a novel task) and make predictions accordingly
    - Easy to implement meta learning algorithms

Some ML Backgrounds (if time permits…)

- (Standard) Supervised Learning

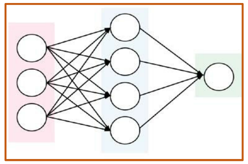
"cat"

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

input (e.g., image) — label

$$\arg\max_{\phi} \log p(\phi|\mathcal{D})$$

model parameters — training data



$$= \arg\max_{\phi} \log p(\mathcal{D}|\phi) + \log p(\phi)$$

data likelihood — regularizer (e.g., weight decay)

$$= \arg\max_{\phi} \sum_i \log p(y_i|x_i, \phi) + \log p(\phi)$$

- We know the biggest problem is that…
  - Can't always collect a large amount of labeled data **D** in advance.

- Now, for the *Meta Learning* scheme…

supervised learning:

$$\arg\max_{\phi} \log p(\phi|\mathcal{D})$$

➡ can we incorporate *additional* data?

➡ $\arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}})$

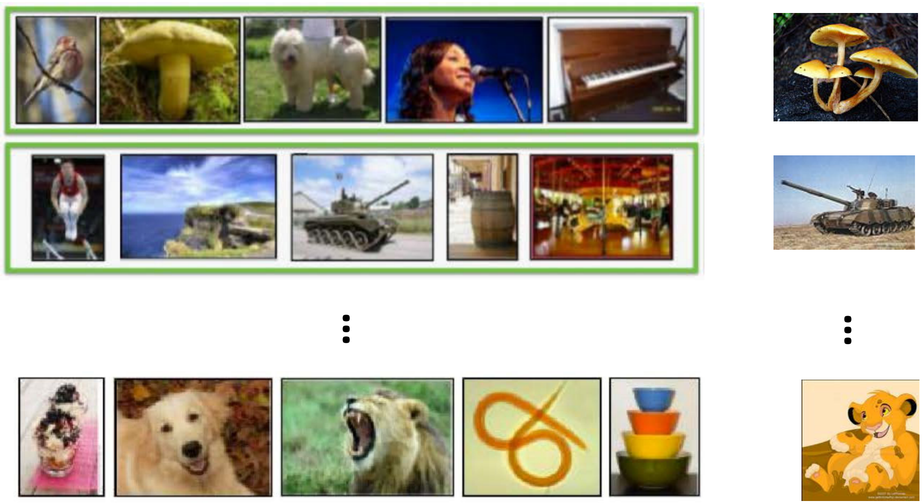
Few-shot data domain of interest

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

$\mathcal{D}_{\text{meta-train}}$ $\mathcal{D}_1$ $\mathcal{D}_2$ ⋮ $\mathcal{D}$



What Meta Learning Solves:

Object label: "cat"

Object ID: "person"

$$\arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}})$$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

➡ what if we don't want to keep $\mathcal{D}_{\text{meta-train}}$ around forever?

➡ learn *meta-parameters* θ: $p(\theta|\mathcal{D}_{\text{meta-train}})$

whatever we need to know about $\mathcal{D}_{\text{meta-train}}$ to solve new tasks

➡ $$\log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) = \log \int_{\Theta} p(\phi|\mathcal{D}, \theta) p(\theta|\mathcal{D}_{\text{meta-train}}) d\theta$$

$$\approx \log p(\phi|\mathcal{D}, \theta^{\star}) + \log p(\theta^{\star}|\mathcal{D}_{\text{meta-train}})$$

Novel Base

What Meta Learning Solves:

Object label: "cat"

Object ID: "person"

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

$$\arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}})$$

- $$\log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) = \log \int_{\Theta} p(\phi|\mathcal{D}, \theta) p(\theta|\mathcal{D}_{\text{meta-train}}) d\theta \approx \log p(\phi|\mathcal{D}, \theta^\star) + \log p(\theta^\star|\mathcal{D}_{\text{meta-train}})$$

- $$\arg\max_{\phi} \log p(\phi|\mathcal{D}, \mathcal{D}_{\text{meta-train}}) \approx \arg\max_{\phi} \log p(\phi|\mathcal{D}, \theta^\star)$$

- What meta learning cares is the learning of Φ from D (and implicitly from $D_{\text{meta-train}}$)

- What makes meta learning challenging is the learning of optimal Θ* from $D_{\text{meta-train}}$:

$$\theta^\star = \arg\max_{\theta} \log p(\theta|\mathcal{D}_{\text{meta-train}})$$

A Quick Example

Base class

Person ID: "Brad Pitt"

➡ **Meta training:** $\theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$

➡ **Meta testing:** $\phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}, \theta^\star)$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

$$\mathcal{D}_i = \{(x_1^i, y_1^i), \ldots, (x_k^i, y_k^i)\}$$

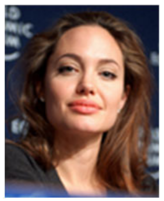
meta-training

(x_1^i, y_1^i) (x_2^i, y_2^i) (x_3^i, y_3^i) → $\theta^\star$

x^{ts} → y^{ts} ← [1, 0, 0]? [0, 1, 0]? [0, 0, 1]?

3 way 1 shot

$\mathcal{D}_i$

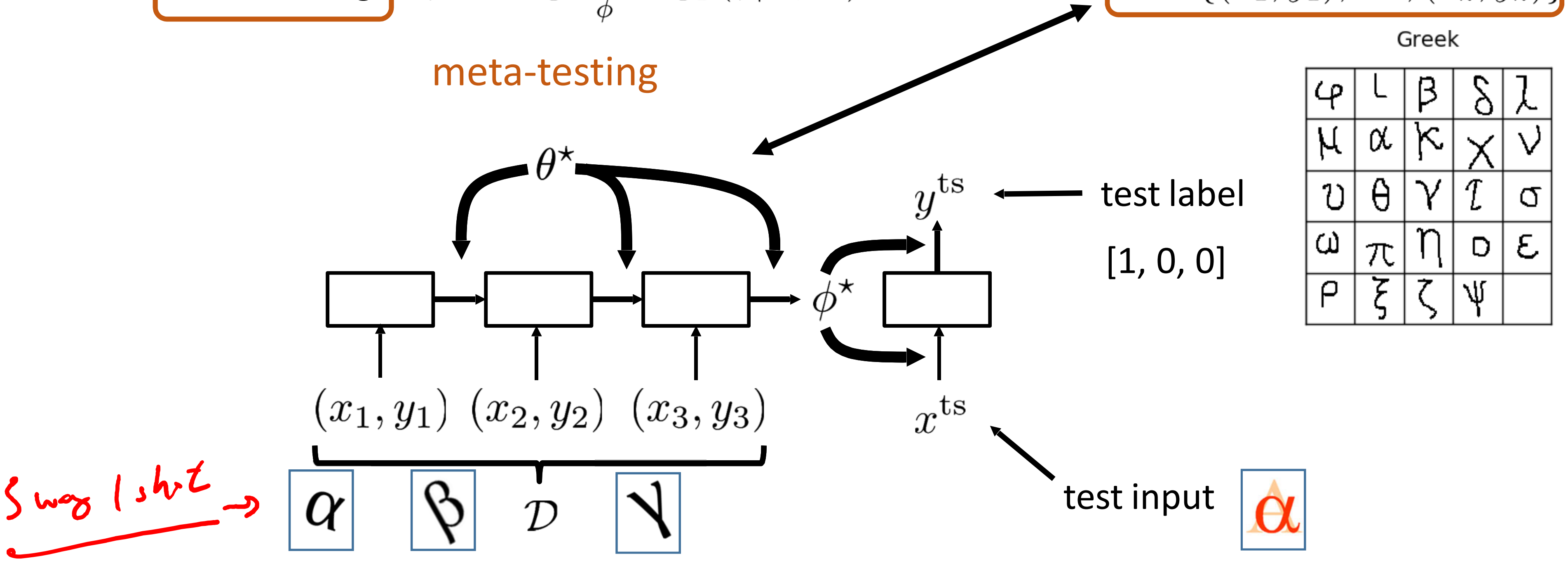


A Quick Example (cont'd)

- Meta training: $\theta^\star = \arg\max_\theta \log p(\theta|\mathcal{D}_{\text{meta-train}})$
- Meta testing: $\phi^\star = \arg\max_\phi \log p(\phi|\mathcal{D}, \theta^\star)$

$$\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$$

$$\mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$$

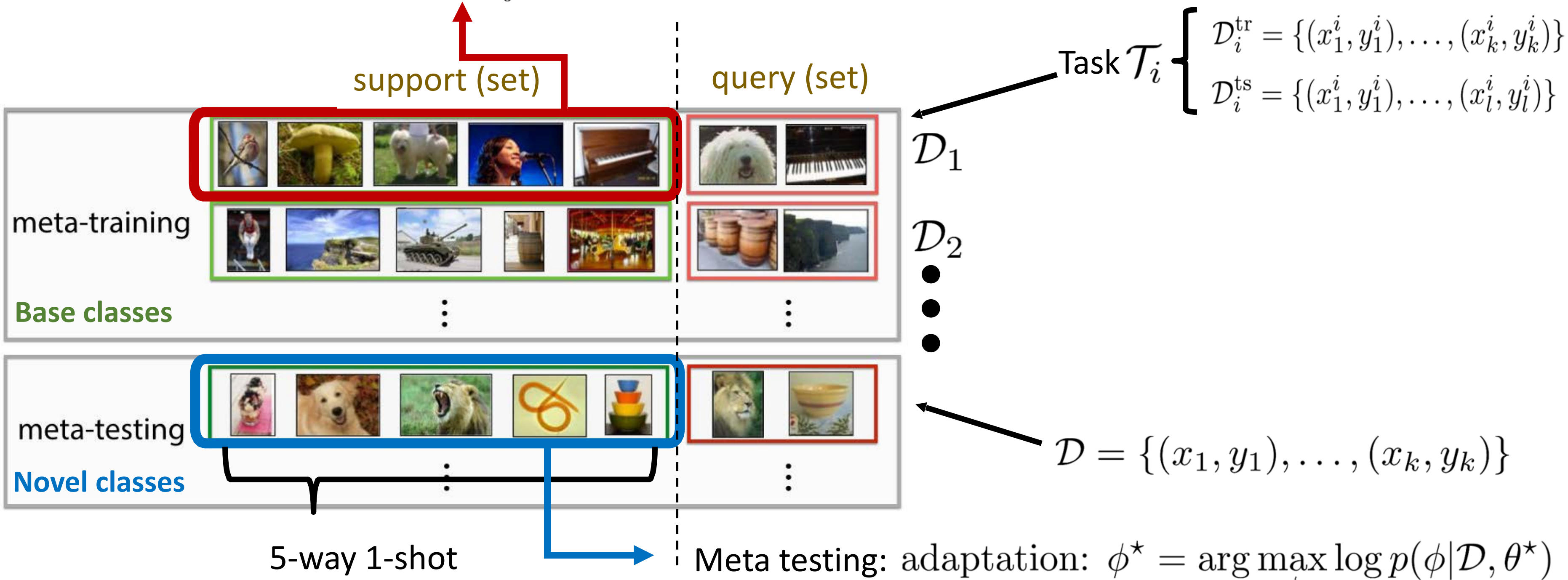


✓ **Key Idea:**

The condition/mechanism of meta-training and meta-testing must match.
In other words, meta learning is to learn the mechanism, ***not*** to fit the data/labels.

Meta-Learning Terminology

$$\text{meta-learning: } \theta^\star = \arg\max_{\theta} \log p(\theta | \mathcal{D}_{\text{meta-train}})$$

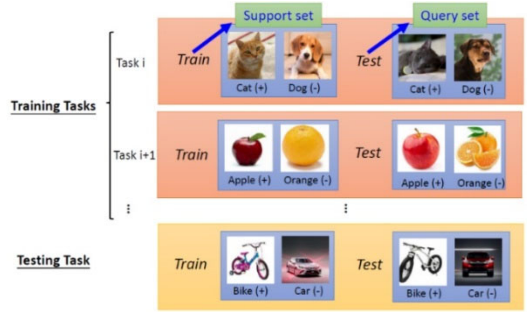


Meta testing: $\text{adaptation: } \phi^\star = \arg\max_{\phi} \log p(\phi | \mathcal{D}, \theta^\star)$

✓ **Remarks**

- Meta learning: learn a N-way K-shot learning mechanism, ***not*** fitting data/labels
- The conditions (i., N-way K-shot) of meta-training and meta-testing must match.
- Additional remarks on N & K for affecting the learning performance?

A Closely Related Yet Different Task: Multi-Task Learning



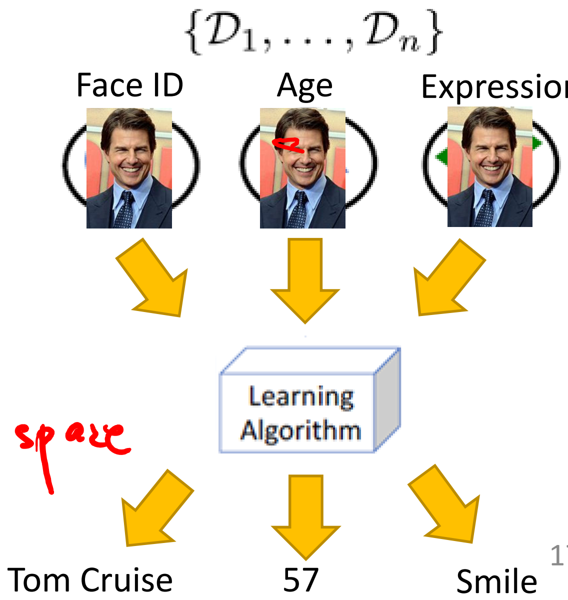
- Meta Learning
  - Meta training: $\theta^\star = \arg\max_\theta \log p(\theta | \mathcal{D}_{\text{meta-train}})$ $\quad \mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_1, \ldots, \mathcal{D}_n\}$
  - Meta testing: $\phi^\star = \arg\max_\phi \log p(\phi | \mathcal{D}, \theta^\star)$ $\quad \mathcal{D} = \{(x_1, y_1), \ldots, (x_k, y_k)\}$

- Multi-Task Learning
  - Learn model with parameter Θ* that simultaneously solves multiple tasks

$$\theta^\star = \arg\max_\theta \sum_{i=1}^{n} \log p(\theta | \mathcal{D}_i)$$

  - Can be viewed as a special case where

$$\phi_i = \theta \ \text{(i.e., } f_\theta(\mathcal{D}_i) = \theta)$$



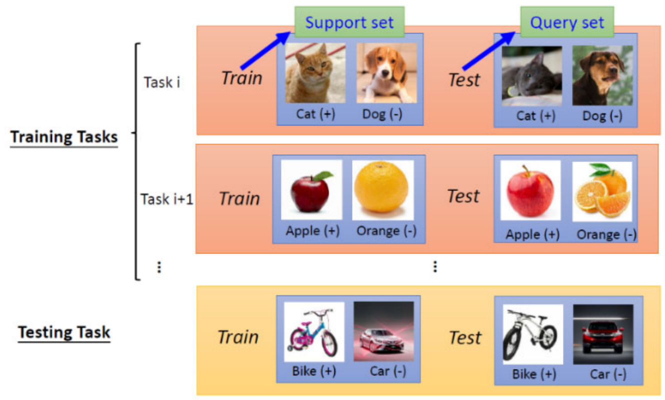
- What about Transfer Learning?

DA: $D_S = \{ X_S, Y_S \}$, $D_T = \{ X_T \}$ } → shared label space

What to Cover Today…

- Meta-Learning
  - Definition
  - Parametric & Non-Parametric based Approaches
- Meta-Learning for Few-Shot Learning
  - Few-Shot Classification
    - Metric Learning vs. Data Hallucination
  - Few-Shot Image Segmentation

Approaches



- Two Ways to View Meta Learning
  - ***Probabilistic* View (e.g., optimization-based)**
    - Extract prior info from a set of (meta training) tasks, allowing efficient learning of a new task (i.e., meta-testing)
    - Learning a new task uses this prior and (small) training set to infer most likely posterior model parameters

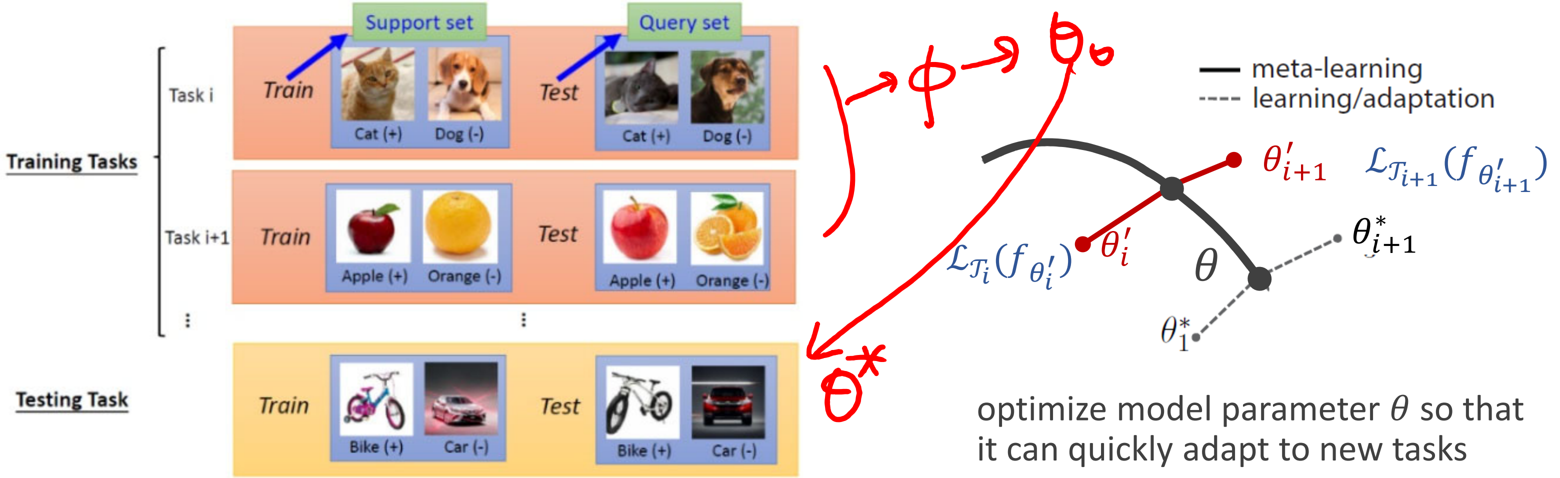
    → Easy to understand meta learning algorithms

  - ***Mechanistic* View (e.g., metric-learning based)**
    - Meta training: A learning model (e.g., DNN) reads in a meta-dataset which consists of many datasets, each for a different task
    - Meta-testing: the model observes new data points (for a novel task) and make prediction accordingly

    → Easy to implement meta learning algorithms

Approach #1: Optimization-Based Approach

- **M**odel-**A**gnostic **M**eta-**L**earning (MAML)*
  - Key idea:
    - Train over many tasks (with a small amount of data & few gradient steps), so that the learned model parameter would generalize to novel tasks
    - Learning to initialize/fine-tune
  - Meta-Learner $\Phi \rightarrow \Theta_0$:
    - Learn a parameter initialization Θ_0 of model that transfers/generalizes to novel tasks well.
    - That is, learn model Θ_0 which can be fine-tuned by novel tasks efficiently/effectively.



*Finn, Abbeel, Levine, ICML 2017

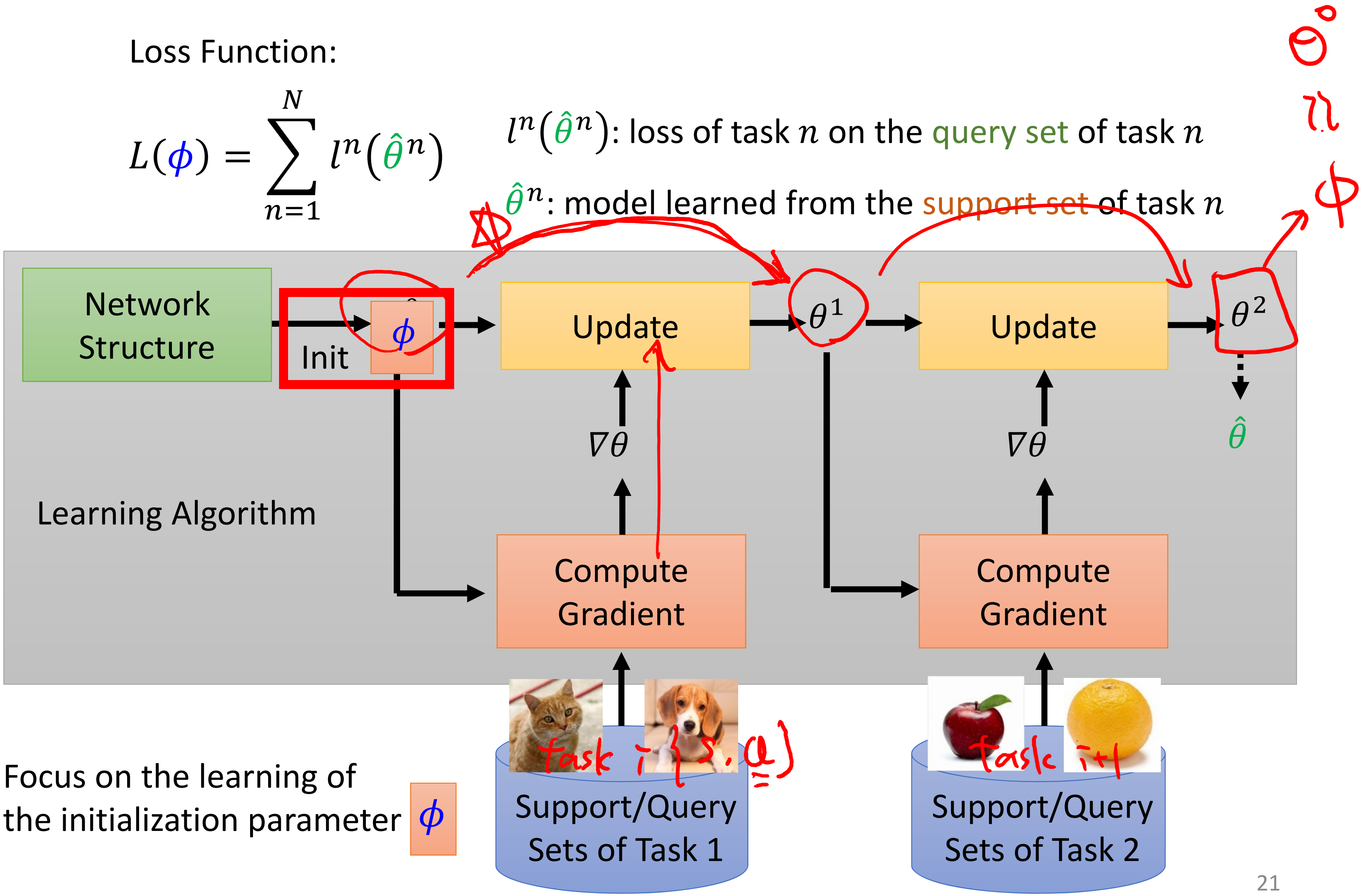
MAML

Loss Function:

$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

$l^n(\hat{\theta}^n)$: loss of task n on the query set of task n

$\hat{\theta}^n$: model learned from the support set of task n



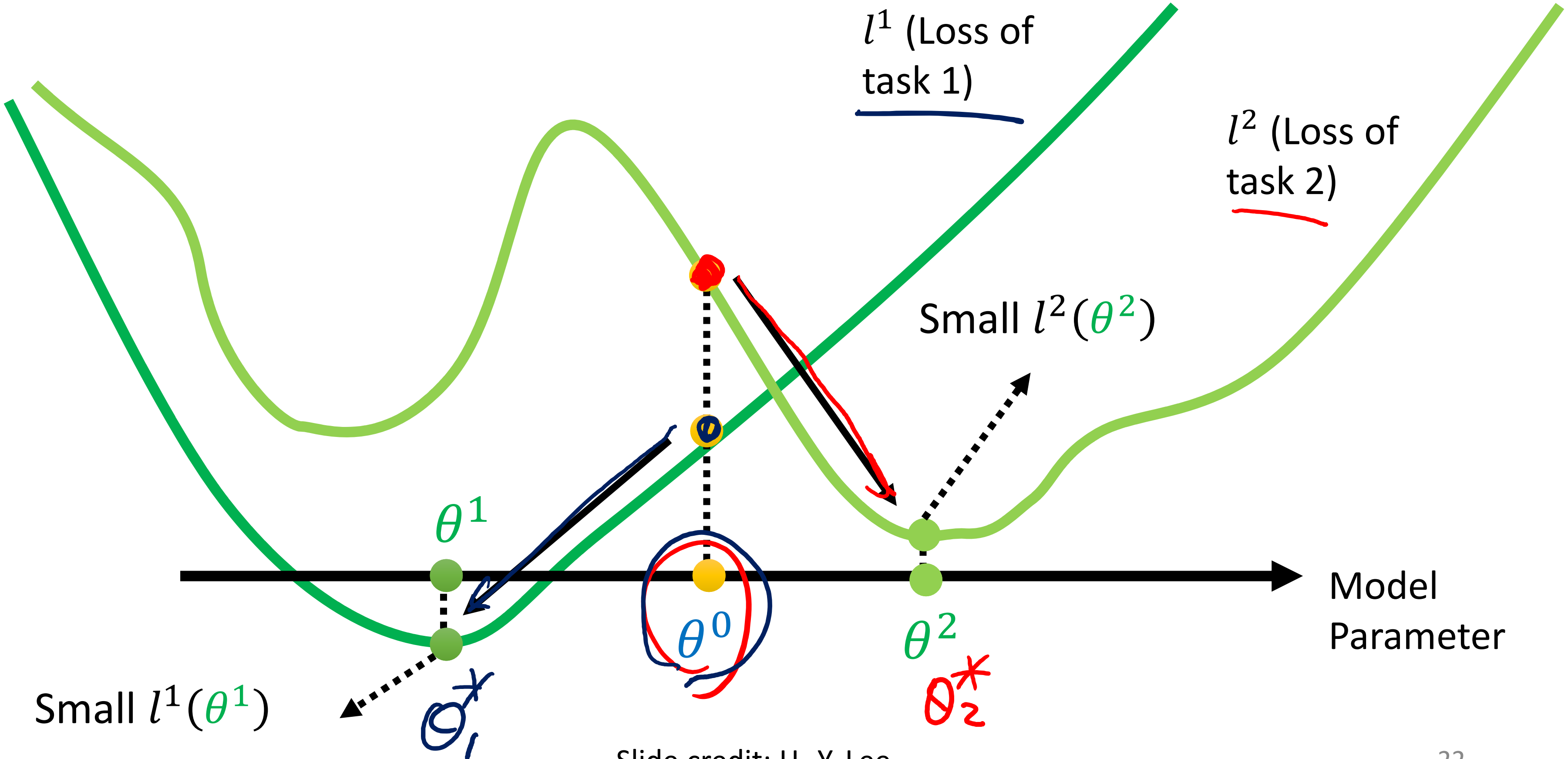
Focus on the learning of the initialization parameter ϕ

- **Illustration of MAML**

MAML doesn't care how model θ^0 performs on each task.

It only cares how model θ^n performs for task *n* when starting from a properly learned θ^0. In other words, a good θ^0 matters!

$$L(\theta^0) = \sum_{n=1}^{N} l^n(\theta^n)$$

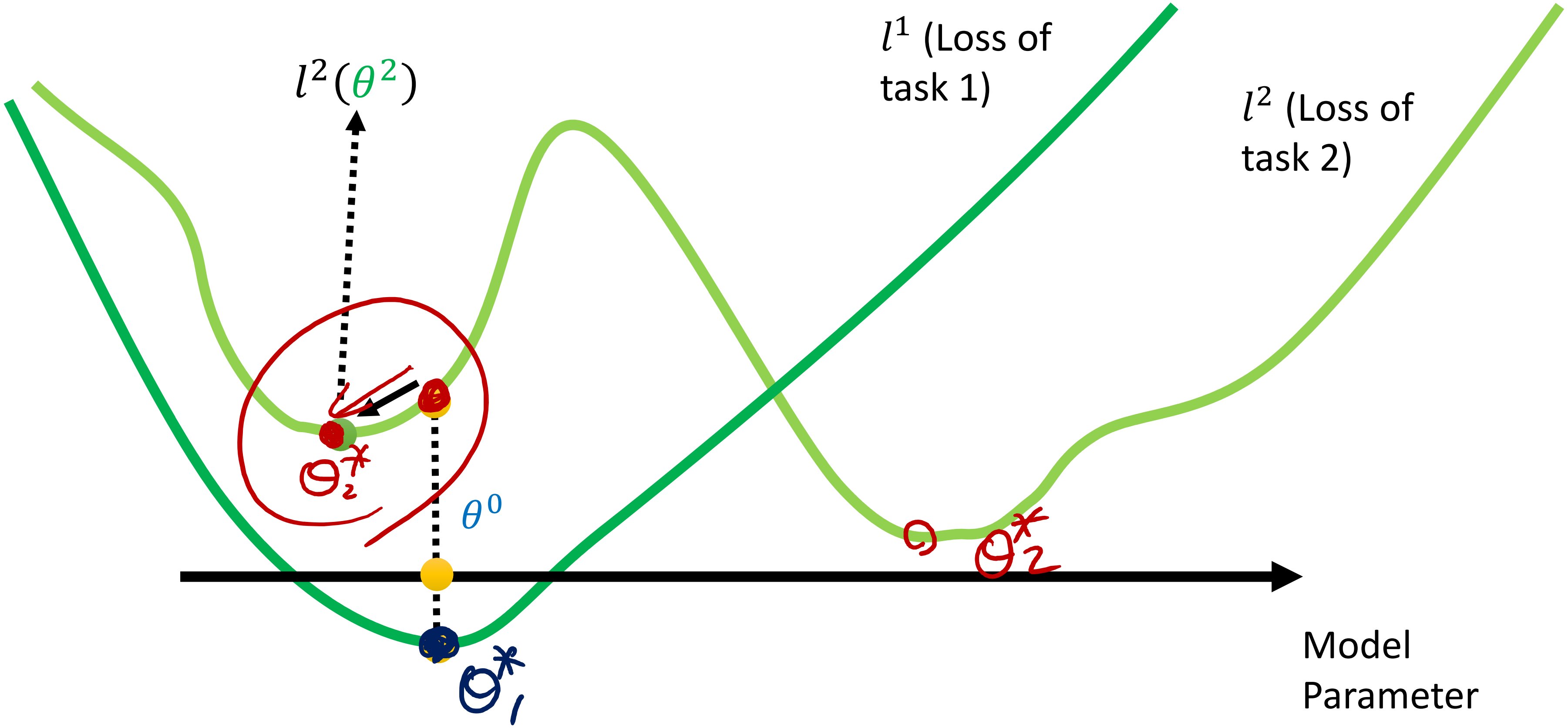


- Comparison:
  Model Pre-Training or
  Multi-Task Learning

$$L(\theta^0) = \sum_{n=1}^{N} l^n(\theta^0)$$

Determine the best θ^0 for all existing tasks

However, no guarantee that θ^0 is preferable for learning good θ^n for task *n*.
Again, a good θ^0 really matters!



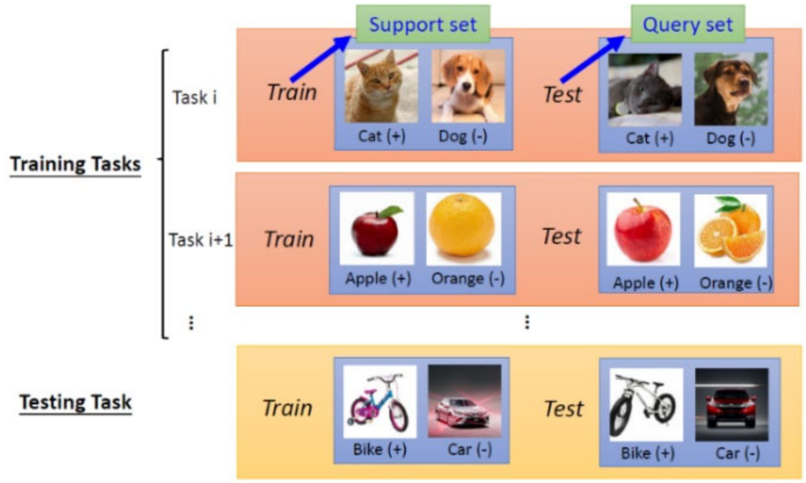
Slide credit: H.-Y. Lee

MAML

- Remarks
  - Train a good initialized parameter set Φ (i.e., θ^0) for quick adaptation/generalization
  - Meta-training:

$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

$$\phi \leftarrow \phi - \eta \nabla_\phi L(\phi)$$

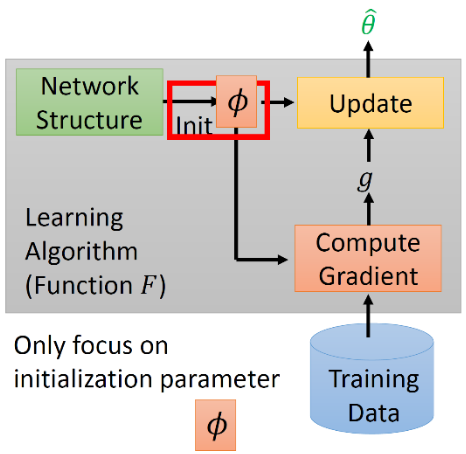


  - Meta-testing (for adaptation):

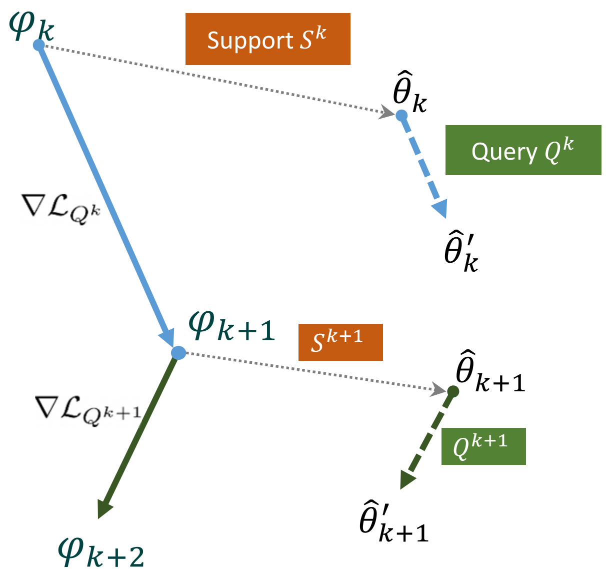
$$\hat{\theta} = \phi - \varepsilon \nabla_\phi l(\phi)$$

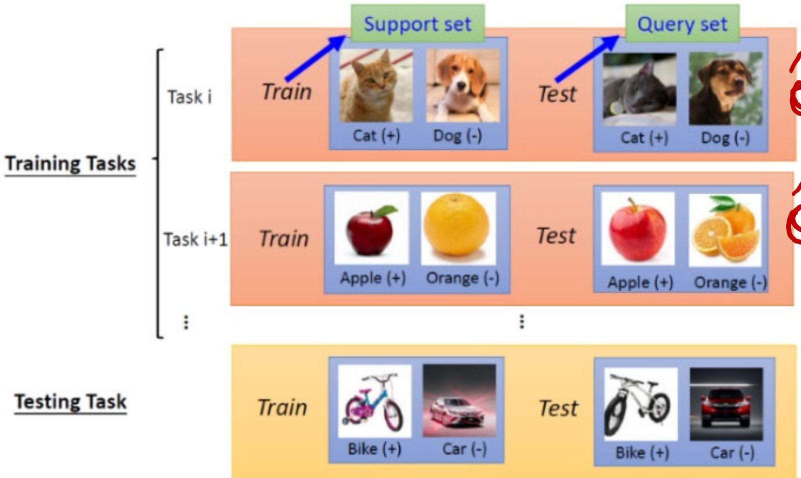
novel task

Note that one or multiple updates can be performed during meta-testing.



Meta-Training in MAML





φ: initial model parameters

$\hat{\theta}$: model parameters updated via the support set

$$\varphi \leftarrow \varphi - \eta \cdot \nabla_\varphi L(\varphi) \tag{1}$$

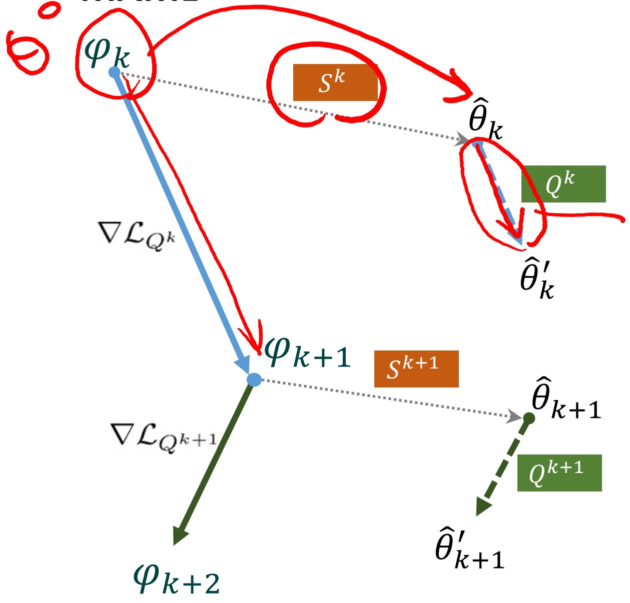
$$L(\varphi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n) \tag{2}$$

$$\hat{\theta} = \varphi - \varepsilon \cdot \nabla_\varphi l(\varphi) \tag{3}$$

$$\nabla_\varphi L(\varphi) = \sum_{n=1}^{N} \nabla_\varphi l^n(\hat{\theta}^n) \tag{4}$$

$$\nabla_\varphi l\,(\hat{\theta}) = \begin{bmatrix} \frac{\partial l(\hat{\theta})}{\partial \varphi_1} \\ \frac{\partial l(\hat{\theta})}{\partial \varphi_2} \\ \vdots \\ \frac{\partial l(\hat{\theta})}{\partial \varphi_i} \end{bmatrix} \tag{5}$$

MAML



$$\hat{\theta} = \varphi - \varepsilon \cdot \nabla_{\varphi} l(\varphi) \qquad (3)$$

$$\nabla_{\varphi} L(\varphi) = \sum_{n=1}^{N} \nabla_{\varphi} l^n(\hat{\theta}^n) \qquad (4)$$

$$\nabla_{\varphi} l\left(\hat{\theta}\right) = \begin{bmatrix} \frac{\partial l(\hat{\theta})}{\partial \varphi_1} \\ \frac{\partial l(\hat{\theta})}{\partial \varphi_2} \\ \cdots \\ \frac{\partial l(\hat{\theta})}{\partial \varphi_i} \end{bmatrix} \qquad (5)$$

$$\frac{\partial l(\hat{\theta})}{\partial \varphi_i} = \sum \frac{\partial l(\hat{\theta})}{\partial \hat{\theta}_j} \frac{\partial \hat{\theta}_j}{\partial \varphi_i}$$

First-order approximation:

$$\hat{\theta}_j = \varphi_j - \varepsilon \cdot \frac{\partial l(\varphi)}{\partial \varphi_j}$$

If $i \neq j$, then:

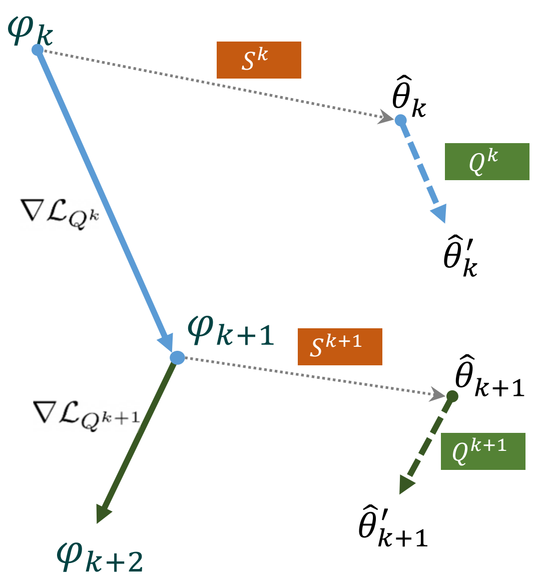
$$\frac{\partial \hat{\theta}_j}{\partial \varphi_i} = -\varepsilon \cdot \frac{\partial l(\varphi)}{\partial \varphi_j \partial \varphi_i} \approx 0$$

If $i = j$, then:

$$\frac{\partial \hat{\theta}_j}{\partial \varphi_i} = 1 - \varepsilon \cdot \frac{\partial l(\varphi)}{\partial \varphi_j \partial \varphi_i} \approx 1$$

φ: initial model parameters
$\hat{\theta}$: model parameters updated via the support set

MAML



$$\nabla_{\varphi} l\left(\hat{\theta}\right) = \begin{bmatrix} \frac{\partial l(\hat{\theta})}{\partial \varphi_1} \\ \frac{\partial l(\hat{\theta})}{\partial \varphi_2} \\ \cdots \\ \frac{\partial l(\hat{\theta})}{\partial \varphi_i} \end{bmatrix} = \begin{bmatrix} \frac{\partial l(\hat{\theta})}{\partial \hat{\theta}_1} \\ \frac{\partial l(\hat{\theta})}{\partial \hat{\theta}_2} \\ \cdots \\ \frac{\partial l(\hat{\theta})}{\partial \hat{\theta}_i} \end{bmatrix} = \nabla_{\hat{\theta}} l\left(\hat{\theta}\right)$$

$$\nabla_{\varphi} L(\varphi) = \sum_{n=1}^{N} \nabla_{\varphi} l^n\left(\hat{\theta}^n\right) = \sum_{n=1}^{N} \nabla_{\hat{\theta}} l^n\left(\hat{\theta}^n\right)$$

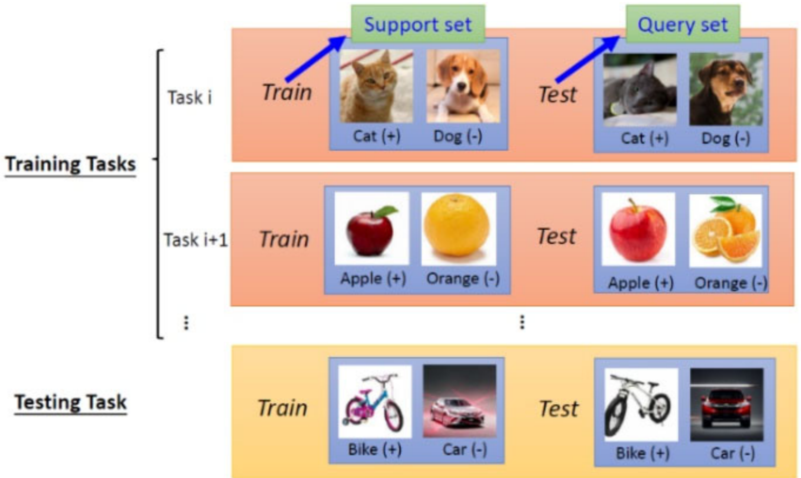
$$\varphi \leftarrow \varphi - \eta \cdot \nabla_{\varphi} L(\varphi) = \varphi - \eta \cdot \nabla_{\hat{\theta}} L(\hat{\theta})$$

Recap: MAML

- Remarks
  - Train a good initialized parameter set Φ (i.e., θ^0) for quick adaptation/generalization
  - Meta-training:

$$L(\phi) = \sum_{n=1}^{N} l^n(\hat{\theta}^n)$$

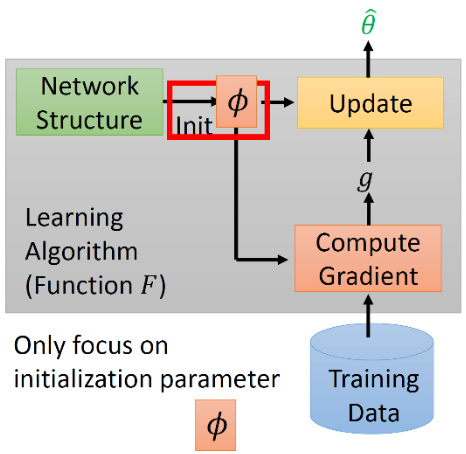
$$\phi \leftarrow \phi - \eta \nabla_\phi L(\phi)$$



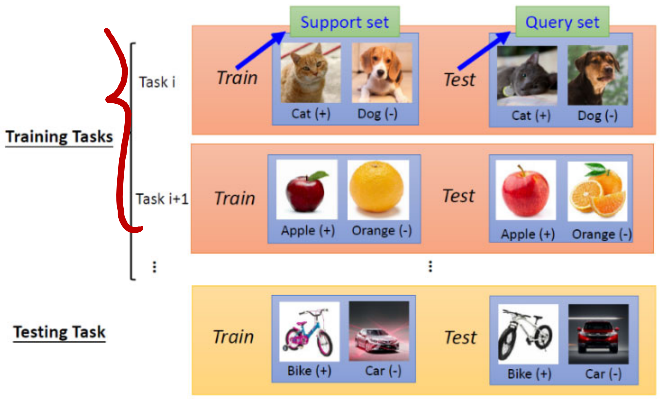
  - Meta-testing (for adaptation):

$$\hat{\theta} = \phi - \varepsilon \nabla_\phi l(\phi)$$

Note that one or multiple updates
can be performed during meta-testing.



Approaches



- Two Ways to View Meta Learning
  - *Probabilistic* View (e.g., optimization-based)
    - Extract prior info from a set of (meta training) tasks, allowing efficient learning of a new task (i.e., meta-testing)
    - Learning a new task uses this prior and (small) training set to infer most likely posterior model parameters

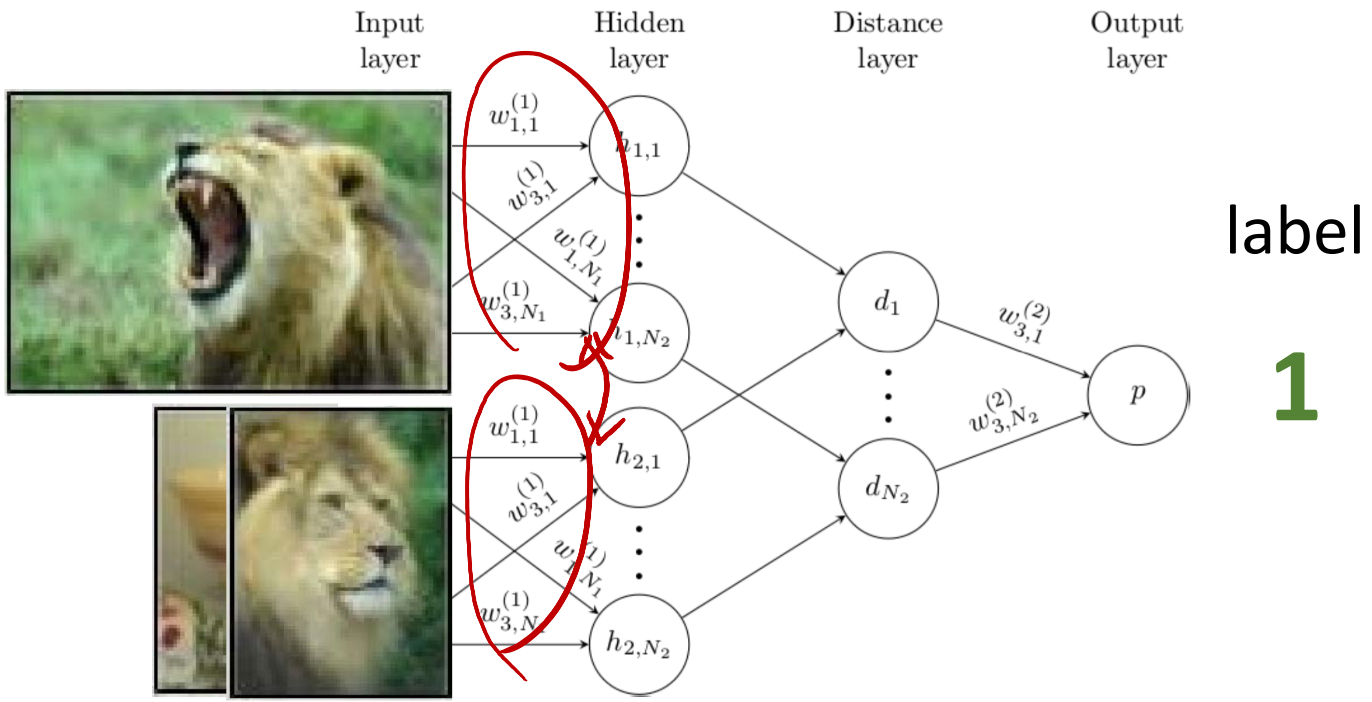
    → Easy to understand meta learning algorithms

  - ***Mechanistic* View (e.g., metric-learning based)**
    - Meta training: A learning model (e.g., DNN) reads in a meta-dataset which consists of many datasets, each for a different task
    - Meta-testing: the model observes new data points (for a novel task) and make prediction accordingly

    → Easy to implement meta learning algorithms

Approach #2: Non-Parametric Approach

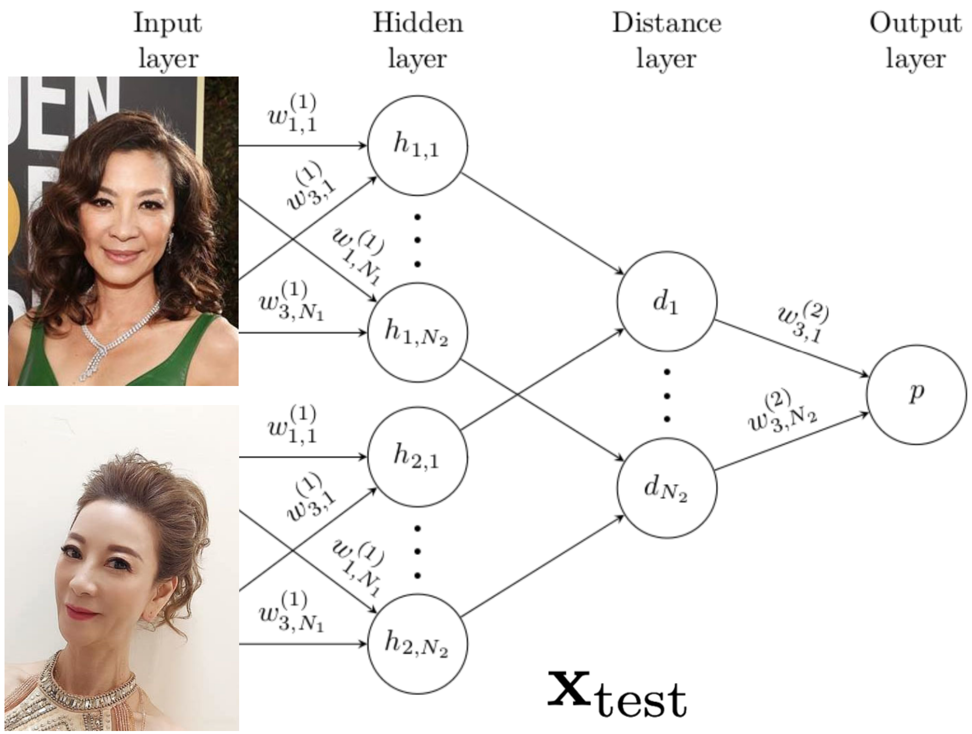
- Can models learn to compare?
- E.g., Siamese Network
  - Learn a network to determine whether a pair of images are of the same category.



Koch et al., Siamese Neural Networks for One-Shot Image Recognition, ICML WS 2015

Learn to Compare (cont’d)

- Siamese Network (cont’d)
  - Meta-training/testing: learn to match (i.e., 2-way image matching)
    - Question: output label of the following example is **1** or **0**?
      (i.e., **same ID** or **not**)

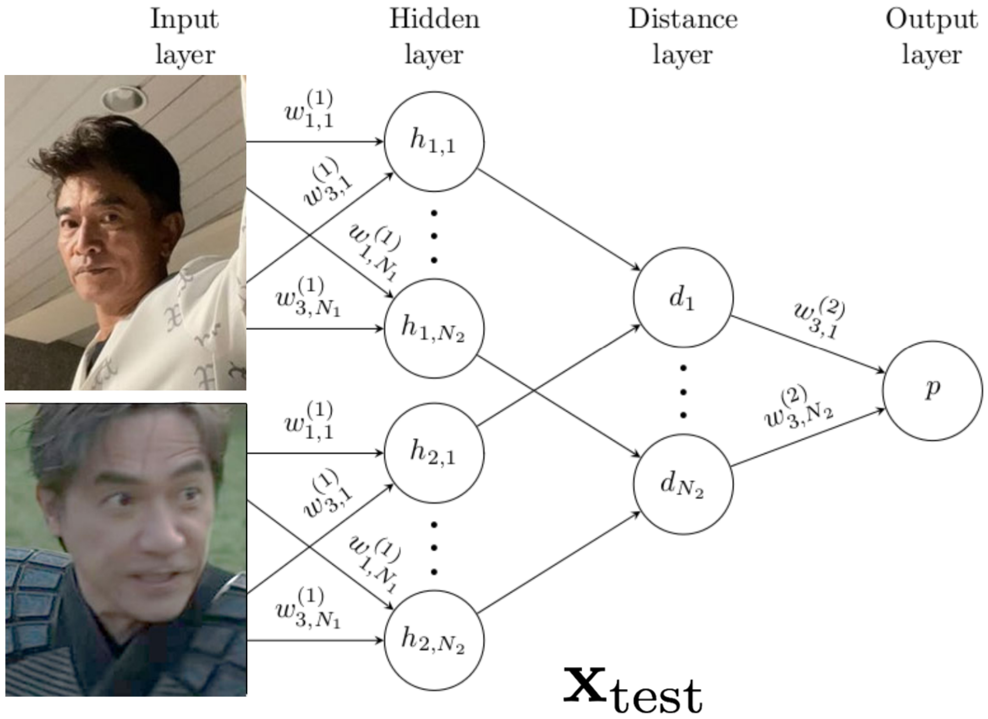


label

?

Learn to Compare (cont'd)

- Siamese Network (cont'd)
  - Meta-training/testing: learn to match (i.e., 2-way image matching)
    - Question: output label of the following example is 1 or 0?
      (i.e., same ID or not)

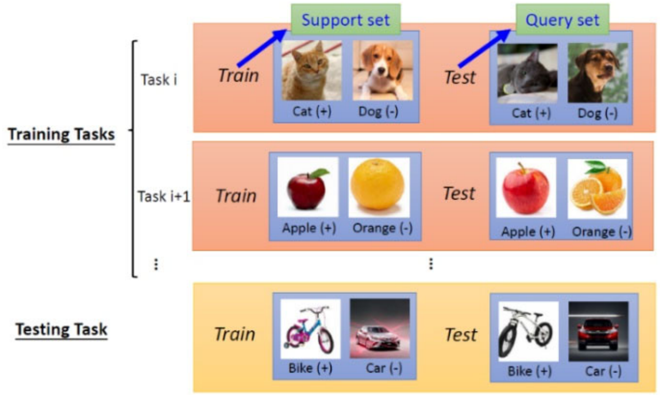


label

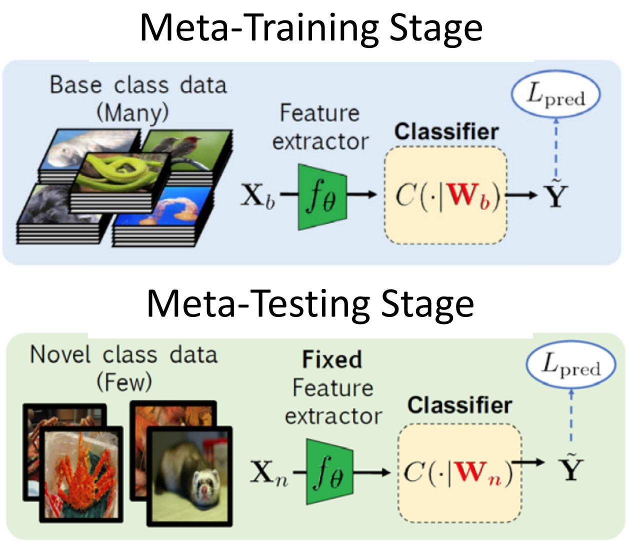
?

- What have we learned from these examples?
- And, can we perform multi-way classification (beyond matching)?

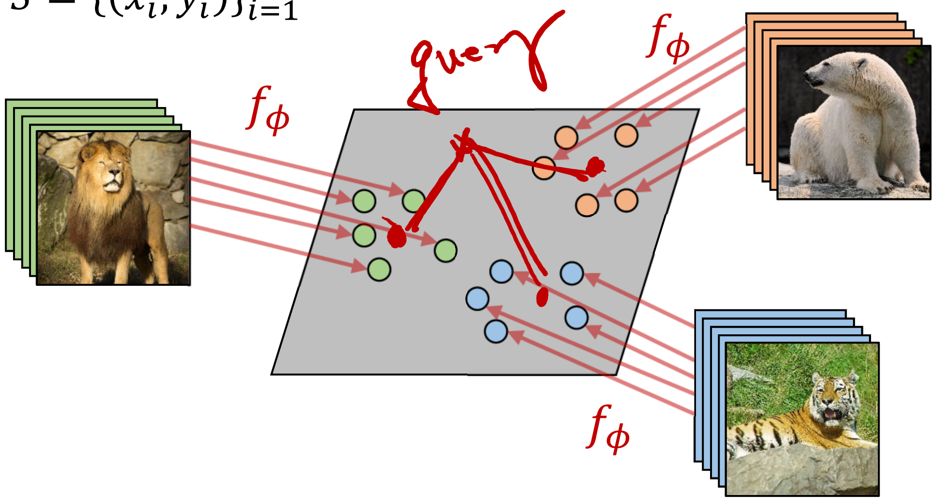
Learn to Compare…
with the Representative Ones!



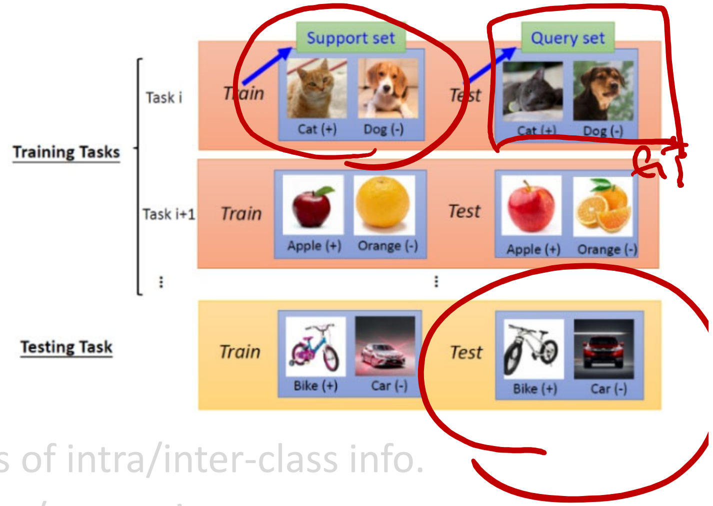
- Prototypical Networks (NIPS’17)
  - Learn a model which properly describes data in terms of intra/inter-class info.
  - It learns a **prototype** for each class, with **data similarity/separation** guarantees.



support set
$S = \{(x_i, y_i)\}_{i=1}^{k}$

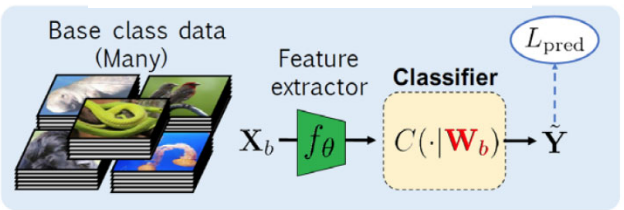


Learn to Compare…
with the Representative Ones!

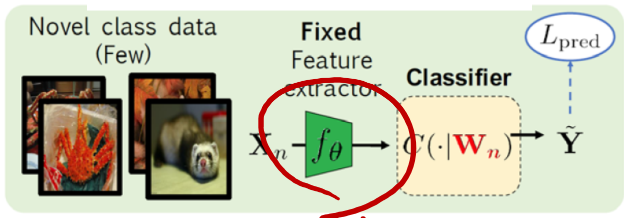


- Prototypical Networks
  - Learn a model which properly describes data in terms of intra/inter-class info.
  - It learns a prototype for each class, with data similarity/separation guarantees.
    For DL version, the learned feature space is derived by a non-linear mapping f_θ and the representatives (i.e., prototypes) of each class is the **mean feature vector $\mathbf{c}_k$**.

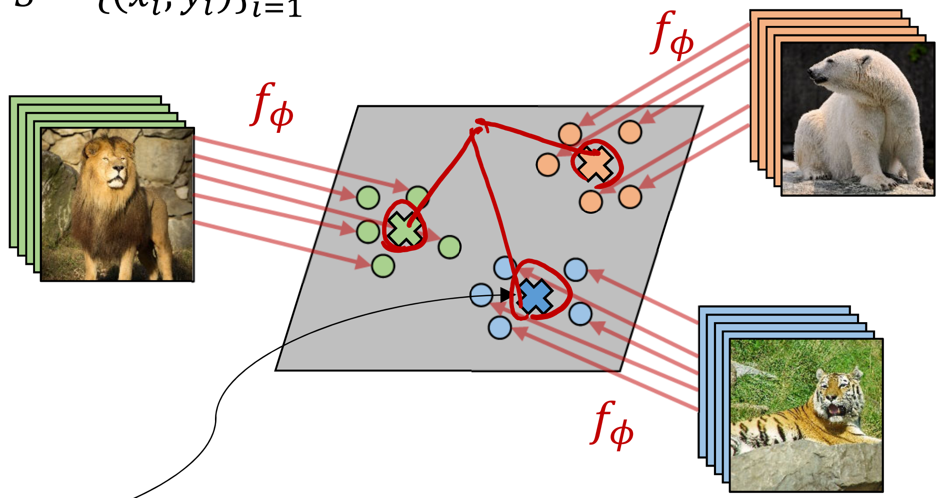
Meta-Training Stage



Meta-Testing Stage



support set
$S = \{(x_i, y_i)\}_{i=1}^{k}$



$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\theta(\mathbf{x}_i)$$, where $S_k \subset S$ indicates features of class k from support set S

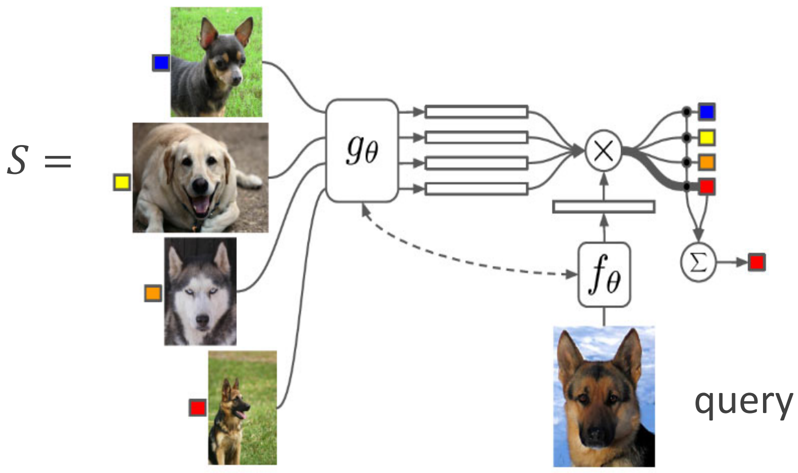
Learn to Compare

- **Matching Networks**
  - Inspired by the **attention** mechanism,
    access an augmented memory containing useful info to solve the task of interest
  - The authors proposed a weighted nearest-neighbor classifier,
    with attention over a learned embedding from the support set $S = \{(x_i, y_i)\}_{i=1}^{k}$,
    so that the label of the query $\hat{x}$ can be predicted.

$c(.,.)$: cosine similarity

$$\hat{y} = \sum_{i=1}^{k} a(\hat{x}, x_i) y_i \quad \text{with} \quad a(\hat{x}, x_i) = e^{c(f(\hat{x}), g(x_i))} / \sum_{j=1}^{k} e^{c(f(\hat{x}), g(x_j))}$$

support set $\{(x_i, y_i)\}_{i=1}^{k}$



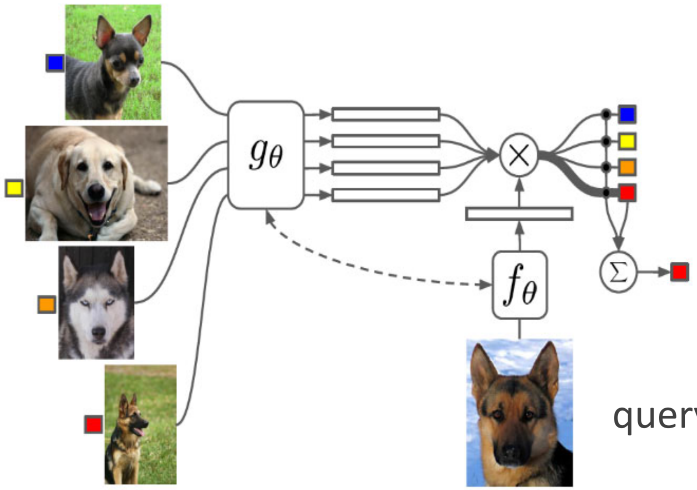
query example $\hat{x}$

Vinyals et al., "Matching Networks for One Shot Learning," NIPS, 2016

Learn to Compare

- **Matching Networks** (cont'd)
  - If we have $g = f$,
    the model turns into a Siamese network like architecture
  - Also similar to prototypical network for **one**-shot learning

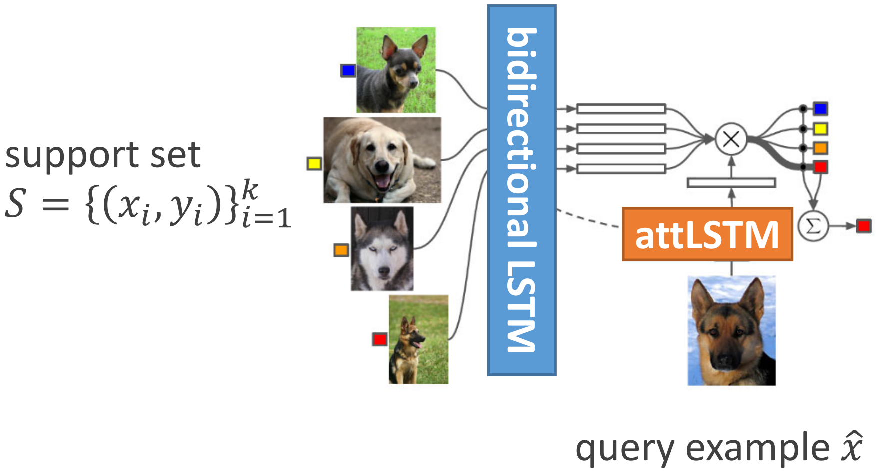
support set $S = \{(x_i, y_i)\}_{i=1}^{k}$



query example $\hat{x}$

Vinyals et al., "Matching Networks for One Shot Learning," NIPS, 2016

- **Matching Networks** (cont'd)
  - Full context embedding (FCE):
  - Each element in S should not be embedded independently of other elements
    - $g(x_i)$ → $g(S)$ as a **bidirectional LSTM** by considering the whole S as a **sequence**
  - Also, S should be able to modify the way we embed $\hat{x}$
    - $f(\hat{x})$ → $f(\hat{x}, S)$ as an **LSTM** with **read-attention** over $g(S)$: $\mathrm{attLSTM}(f'(\hat{x}), g(S), K)$, where $f'(\hat{x})$ is the (fixed) CNN feature, and K is the number of unrolling steps
  - Experiment results on *mini*ImageNet

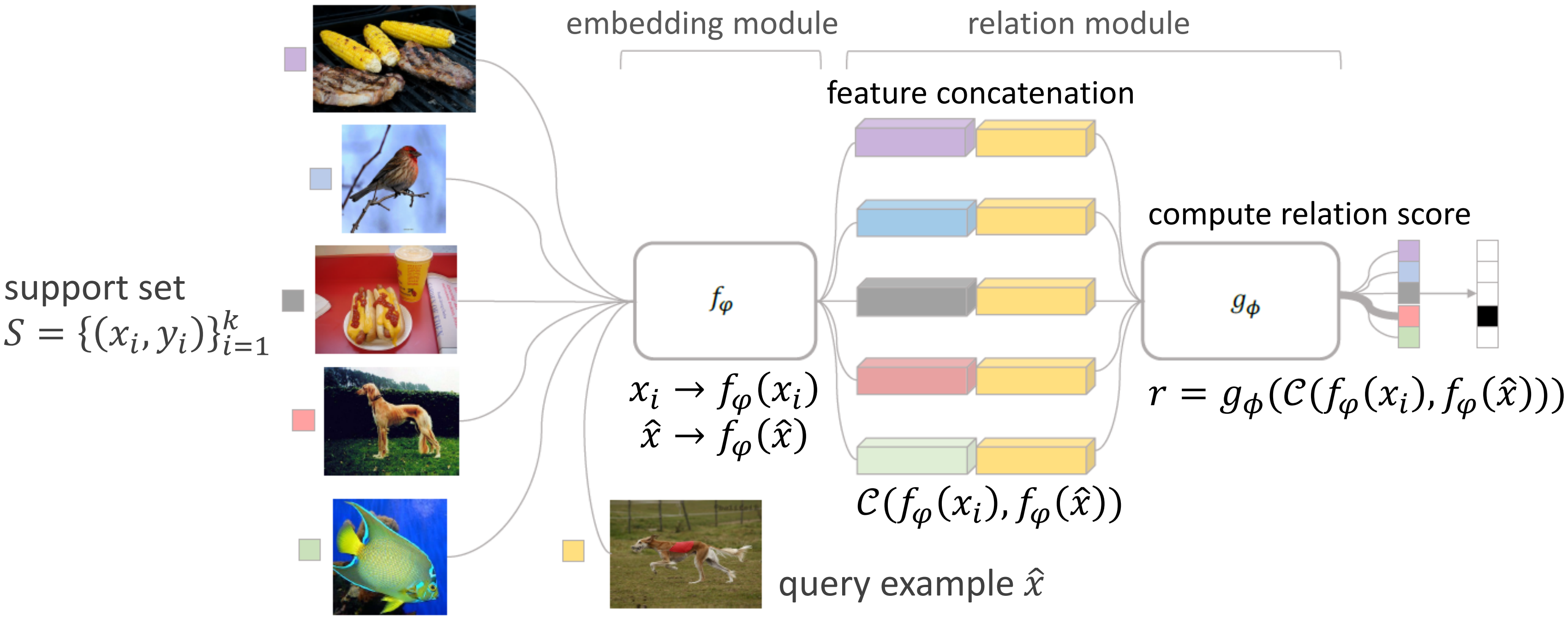


Model	Matching Fn	Fine Tune	5-way Acc 1-shot	5-way Acc 5-shot
PIXELS	Cosine	N	23.0%	26.6%
BASELINE CLASSIFIER	Cosine	N	36.6%	46.0%
BASELINE CLASSIFIER	Cosine	Y	36.2%	52.2%
BASELINE CLASSIFIER	Softmax	Y	38.4%	51.2%
MATCHING NETS (OURS)	Cosine	N	41.2%	56.2%
MATCHING NETS (OURS)	Cosine	Y	42.4%	58.0%
MATCHING NETS (OURS)	Cosine (FCE)	N	44.2%	57.0%
MATCHING NETS (OURS)	Cosine (FCE)	Y	**46.6%**	**60.0%**

Vinyals et al., "Matching Networks for One Shot Learning," NIPS, 2016

Learn to Compare…with Self-Learned Metrics!

- **Relation Network**
  - Metric-learning approaches typically focus on learning an embedding function with a **fixed metric** (e.g., Euclidean distance, cosine similarity, …)
  - The authors proposed to train a **Relation Network** (RN) to explicitly learn a transferrable **deep distance metric** comparing the relation between images

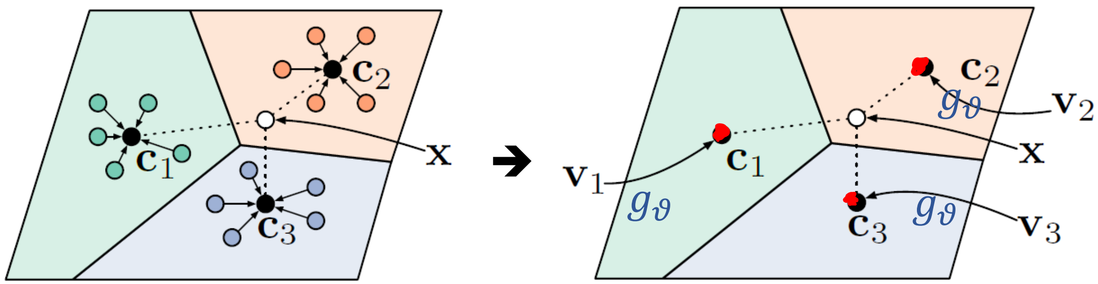


Sung et al., "Learning to Compare: Relation Network for Few-Shot Learning," CVPR, 2018

Relation Networks (cont'd)

- Some works can be extended to **zero-shot learning**:
  - Instead of few-shot images, the support set contains a **semantic embedding vector** ($\mathbf{v}_k$) for each of the training classes.
  - Thus, we can use a second **heterogeneous** embedding function to embed the semantic embedding vectors.
  - Extension of **Prototypical Network**:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_\phi(\mathbf{x}_i) \;\rightarrow\; \mathbf{c}_k = g_\vartheta(\mathbf{v}_k)$$



  - Relation Networks: $r = g_\phi(\mathcal{C}(f_\varphi(x_i), f_\varphi(\hat{x}))) \;\rightarrow\; r = g_\phi(\mathcal{C}(f_{\varphi_2}(\mathbf{v}_k), f_{\varphi_1}(\hat{x})))$

Some Takeaways
for Existing Meta-Learning Approaches

Parametric-based

+ handles **varying** & **large K** well
+ **structure lends well** to **out-of-distribution tasks**

- **second-order optimization**

Non-parametric based

Siamese
Prototypical
Matching
⋮

+ **simple**
+ entirely **feedforward**
+ **computationally fast** & **easy to optimize**

- **harder to generalize** to **varying K**
- hard to scale to **very large K**
- so far, **limited to classification**

Generally, well-tuned versions of each perform **comparably** on existing FSL benchmarks.

What to Cover Today…

- Meta-Learning
  - Definition
  - Parametric & Non-Parametric based Approaches
- Meta-Learning for Few-Shot Learning
  - Few-Shot Classification
    - Metric Learning vs. Data Hallucination
  - Few-Shot Image Segmentation

Learn to Augment…Data Hallucination for FSL

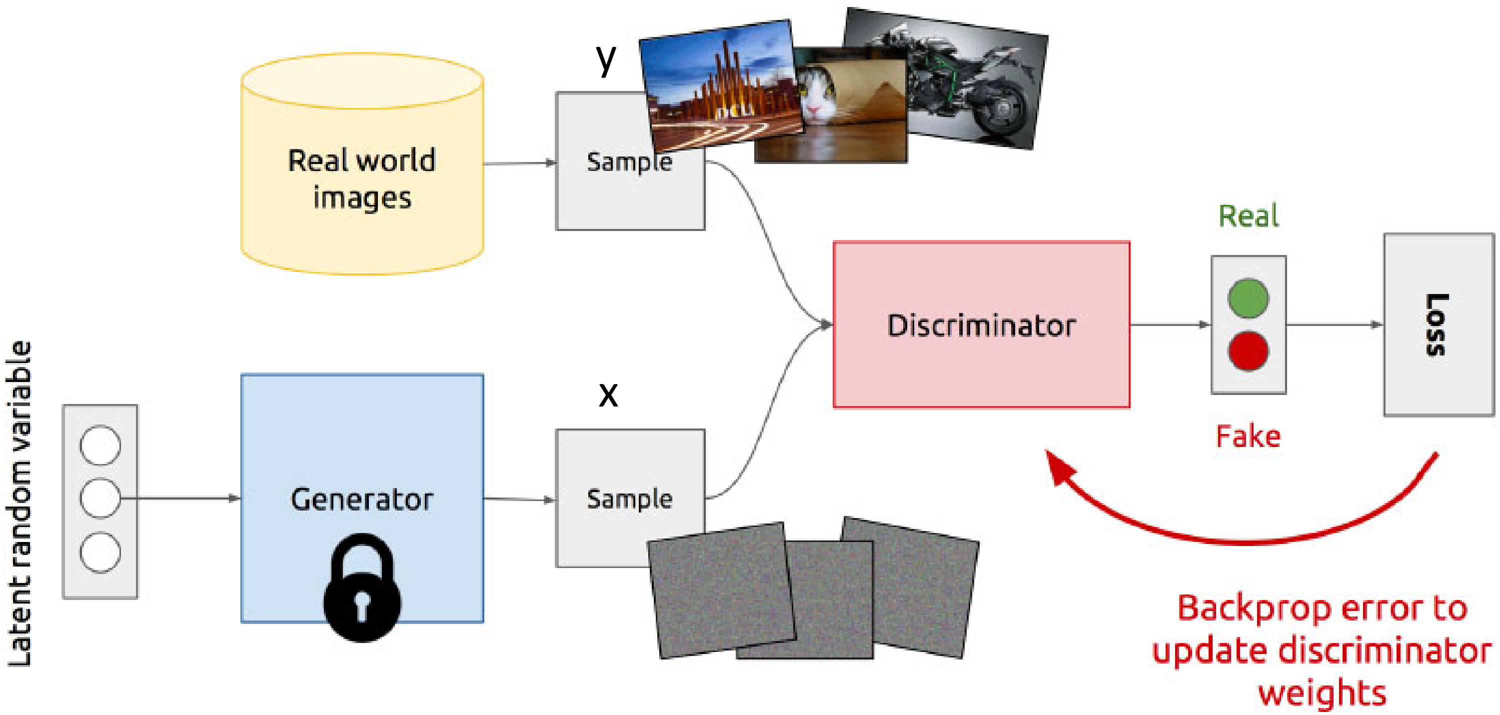
- Data Hallucination
  - Many modes of intra-class variation (e.g., camera pose, translation, lighting changes, and even articulation) are shared across categories.
  - As humans, our knowledge of such intra-class variations allow us to visualize what a novel object might look like in other poses or surroundings.

  - We can thus *hallucinate* additional examples for novel classes by transferring variation modes from the base classes.
  - Typical data augmentation techniques only use a limited amount of **a priori known invariances** (e.g., translations, rotations, flips, addition of Gaussian noise, etc.).

Wang et al., "Low-Shot Learning from Imaginary Data," CVPR, 2018

A Super Brief Review for
Generative Adversarial Networks (GAN)

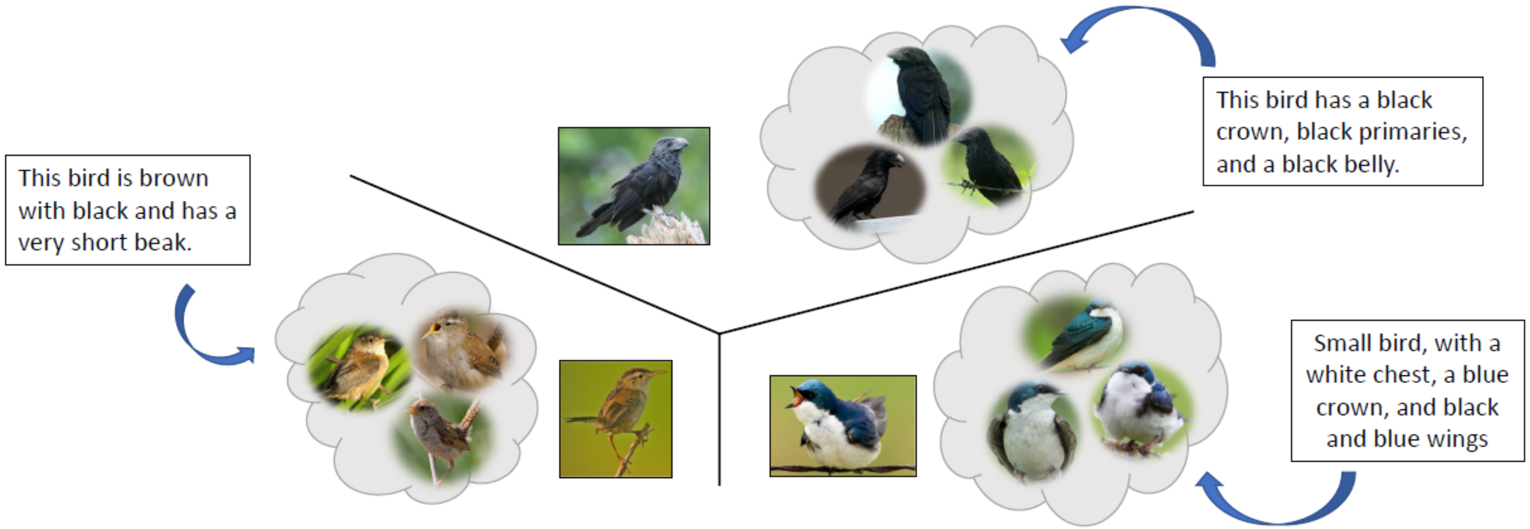
- Design of GAN
  - Loss: $\mathcal{L}_{GAN}(G, D) = \mathbb{E}[\log(1 - D\ (G(x)))] + \mathbb{E}[\log D(y)]$



Goodfellow et al., Generative Adversarial Nets, NIPS, 2014

Learn to Augment…Data Hallucination for FSL

- Cross-Modal Hallucination
  - The lack of data in one modality (e.g., image) can be compensated by abundant data in the other modality (e.g., text) through properly learned **alignments** between two modalities.
  - Here, fine-grained images with detailed textual descriptions are used to build a **text-conditional GAN** for image generation
  - Generated images should be not only realistic but also **class-discriminative**.



Pahde et al., "Cross-modal Hallucination for Few-shot Fine-grained Recognition," CVPR WS 2018

- Cross-Modal Hallucination (cont'd)
  - **Discriminative** text-conditional GAN (tcGAN)
  - First, train a tcGAN on samples from $\mathcal{C}_{\text{base}}$ with regular objective function:

$$\mathcal{L}_{tcGAN}(G,D) = \mathbb{E}_{I,T}[logD(I,T)] + \mathbb{E}_{z,T}[log(1 - D(G(z,T),T))]$$

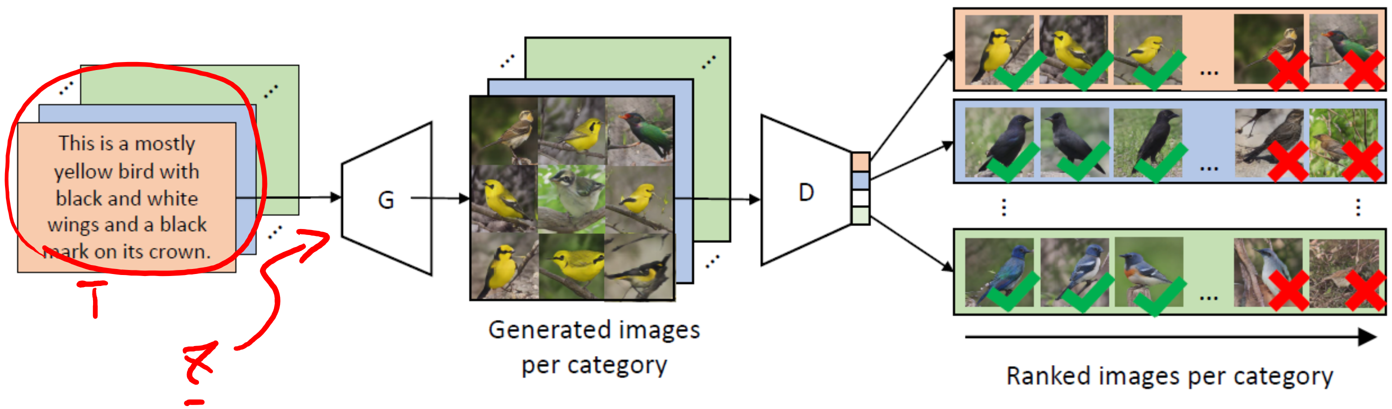
T: text embedding
I: image embedding

  - Next, augment $\mathcal{L}_{tcGAN}$ by adding a **class-discriminative loss** (similar to ACGAN) and fine-tune the tcGAN on the few-shot samples from $\mathcal{C}_{\text{novel}}$ with the compound losses:

$$\mathcal{L}(D) = \mathcal{L}_{tcGAN}(G,D) + \mathbb{E}[P(c|I)]$$
$$\mathcal{L}(G) = \mathcal{L}_{tcGAN}(G,D) - \mathbb{E}[P(c|G(z,T))]$$

c: class label

  - $D^* = \text{argmax}_D \mathcal{L}(D)$ and $G^* = \text{argmin}_G \mathcal{L}(G)$

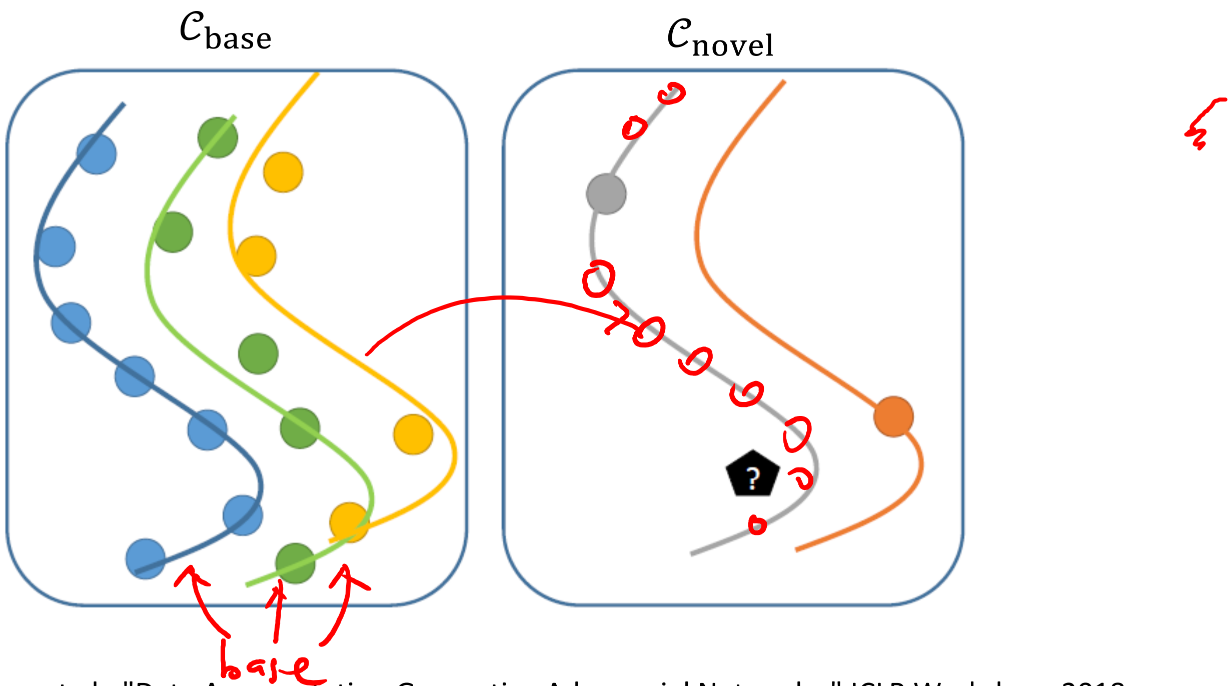


Select top-scored generated images computed by D^*

Pahde et al., "Cross-modal Hallucination for Few-shot Fine-grained Recognition," CVPR WS 2018

Learn to Augment…Data Hallucination for FSL

- Data Hallucination GAN
  - Previous hallucination approaches leveraged datasets with **expensive annotations**
  - Moreover, the modes of intra-class variations typically come from fixed pre-specified rules (e.g., pre-specified **instance-level** textual descriptions)
  - Can we learn a model of a **larger invariance space**, through training a conditional GAN in the **source domain** ($\mathcal{C}_{\text{base}}$), and apply it to the **target domain** ($\mathcal{C}_{\text{novel}}$)?



Antoniou et al., "Data Augmentation Generative Adversarial Networks," ICLR Workshop, 2018

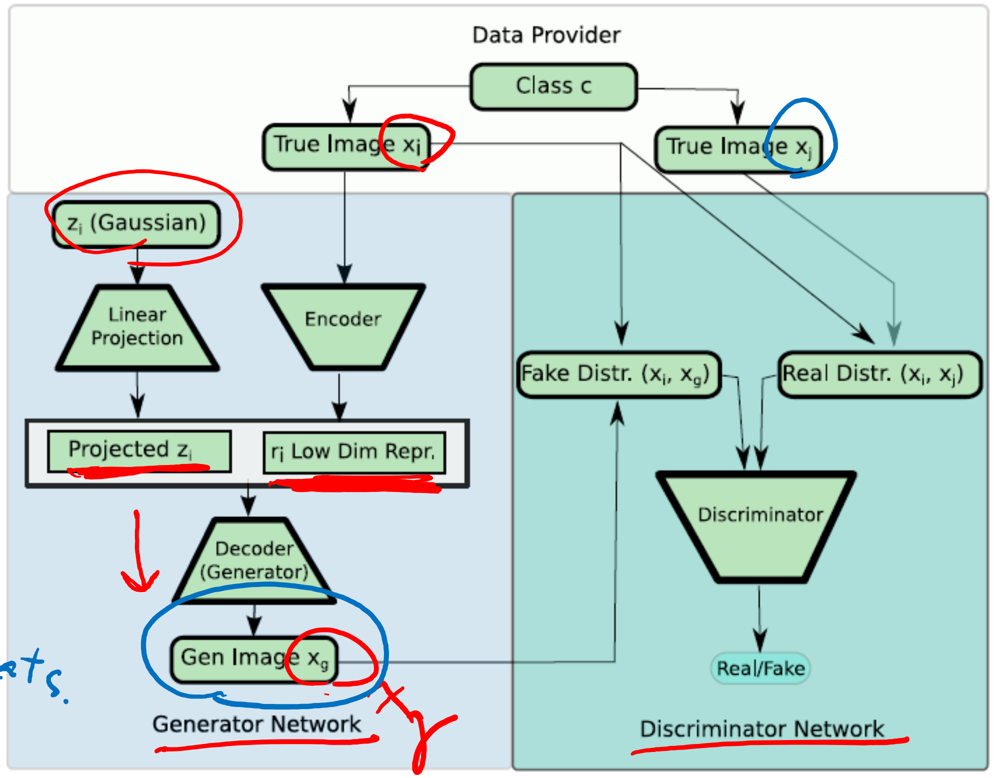
- Data Augmentation GAN

(Left) **Generator**

$\mathbf{r}_i = Enc(\mathbf{x}_i)$

$\mathbf{z}_i \sim N(\mathbf{0}, \mathbf{I})$

$\mathbf{x}_g = Dec(\mathbf{z}_i, \mathbf{r}_i)$



(Right) **Discriminator**

$D(\mathbf{x}_i, \mathbf{x}_j)$ ➔ Real **pair**

$D(\mathbf{x}_i, \mathbf{x}_g)$ ➔ Fake **pair**

Why not just discriminate between $\mathbf{x}_j$ and $\mathbf{x}_g$?

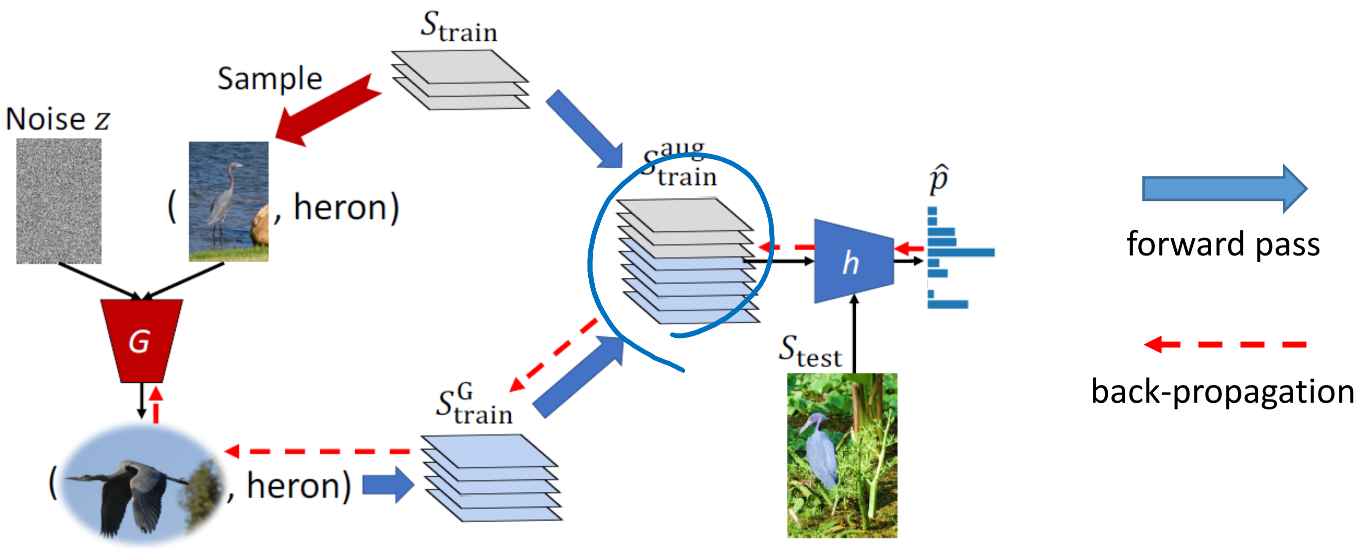
➔ To prevent…

➔ That is, to improve…

Antoniou et al., "Data Augmentation Generative Adversarial Networks," ICLR Workshop, 2018

Learn to Augment…Data Hallucination for FSL

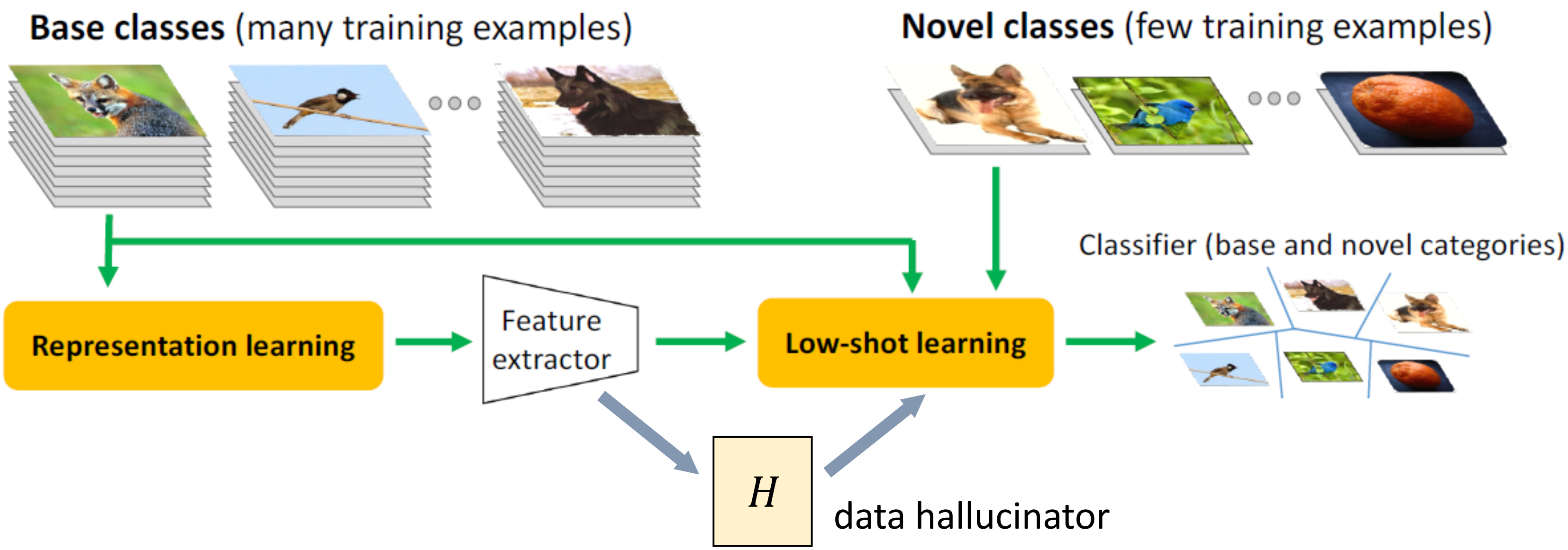
- Jointly Trained Hallucinator
  - The hallucinated examples should be **useful** for classification tasks, rather than just being **diverse** or **realistic** (that may fail to improve FSL performances).
  - The authors proposed to train a **conditional-GAN-based** data hallucinator ($G(x, z)$) **jointly** with the meta-learning module (h) in an **end-to-end** manner.



Wang et al., Low-shot learning from imaginary data, CVPR 2018

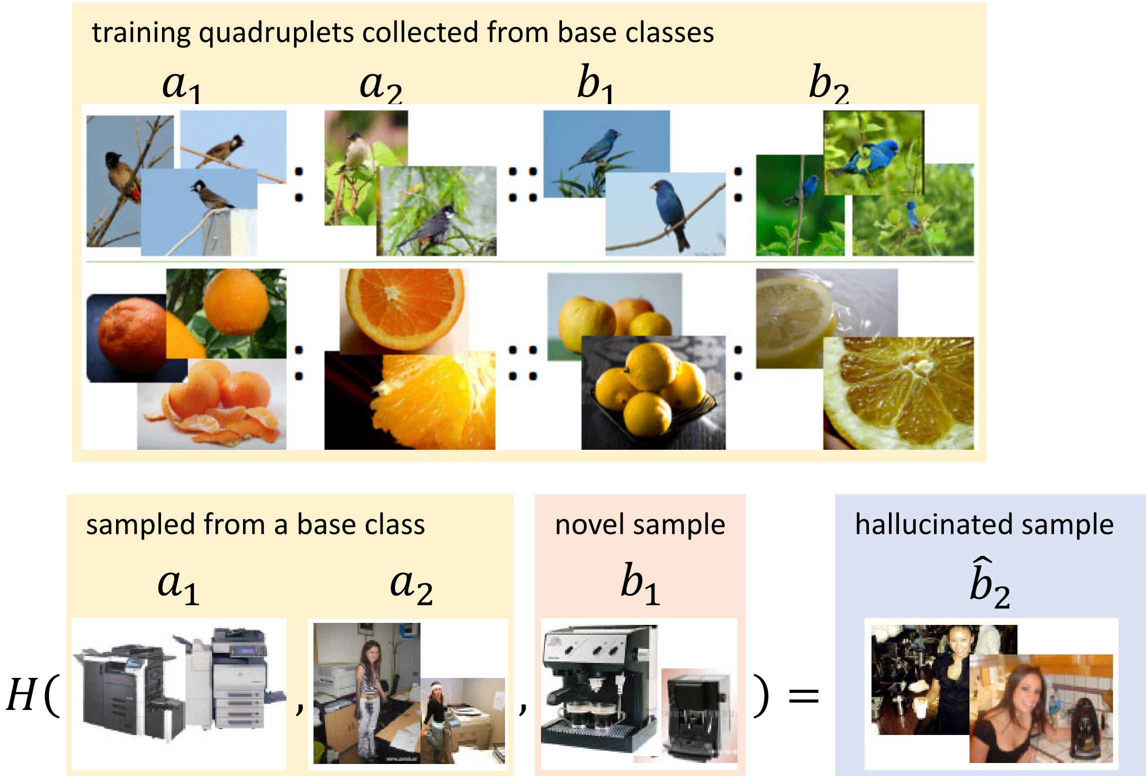
Learn to Augment…Data Hallucination for FSL

- Hallucination by Analogy
  - Modern recognition models are trained on large labeled datasets like ImageNet
  - To deal with the above challenges faced by **recognition systems in the wild**, the authors proposed a FSL benchmark in two phases:



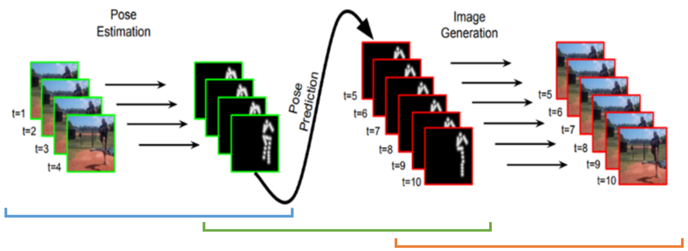
Hariharan et al., Low-shot Visual Recognition by Shrinking and Hallucinating Features, ICCV, 2017

- Hallucination by Analogy (cont'd)
  - Analogy-based Data Hallucinator
    - Train H using **analogy quadruplets** (a_1, a_2, b_1, b_2), where (a_1, a_2) belong to some class, (b_1, b_2) belong to another class, and $a_1 : a_2 :: b_1 : b_2$ holds.



Hariharan et al., Low-shot Visual Recognition by Shrinking and Hallucinating Features, ICCV, 2017

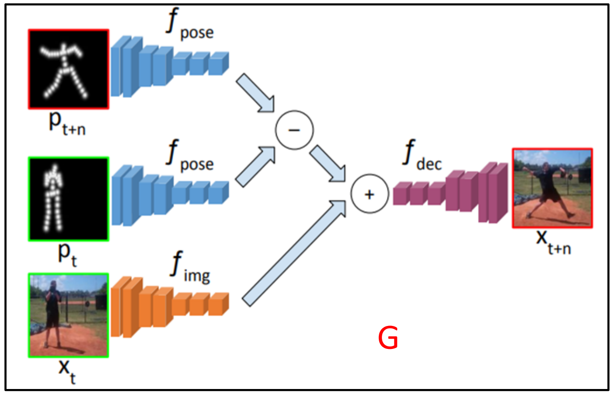
Recap:
Data Analogy in Video Prediction

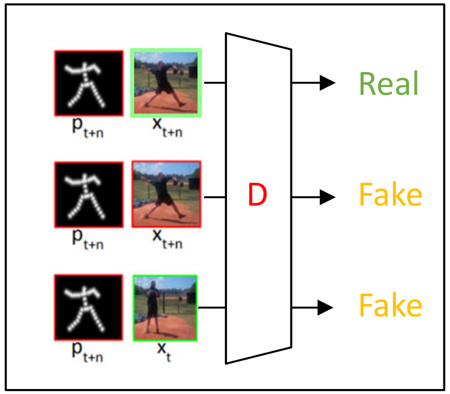


- Learning to generate long-term future via hierarchical prediction
  (Villegas et al., ICML'17)

Step 3:

Image Generation





Visual-Structure Analogy

Objective Function:

Adversarial Training -> alternately minimize L & LDisc

Update Image Generation Network (G)

$$\mathcal{L} = \mathcal{L}_{\text{img}} + \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{Gen}}$$

$$\mathcal{L}_{\text{img}} = \|\mathbf{x}_{t+n} - \hat{\mathbf{x}}_{t+n}\|_2^2$$

$$\mathcal{L}_{\text{feat}} = \|C_1(\mathbf{x}_{t+n}) - C_1(\hat{\mathbf{x}}_{t+n})\|_2^2 + \|C_2(\mathbf{x}_{t+n}) - C_2(\hat{\mathbf{x}}_{t+n})\|_2^2$$

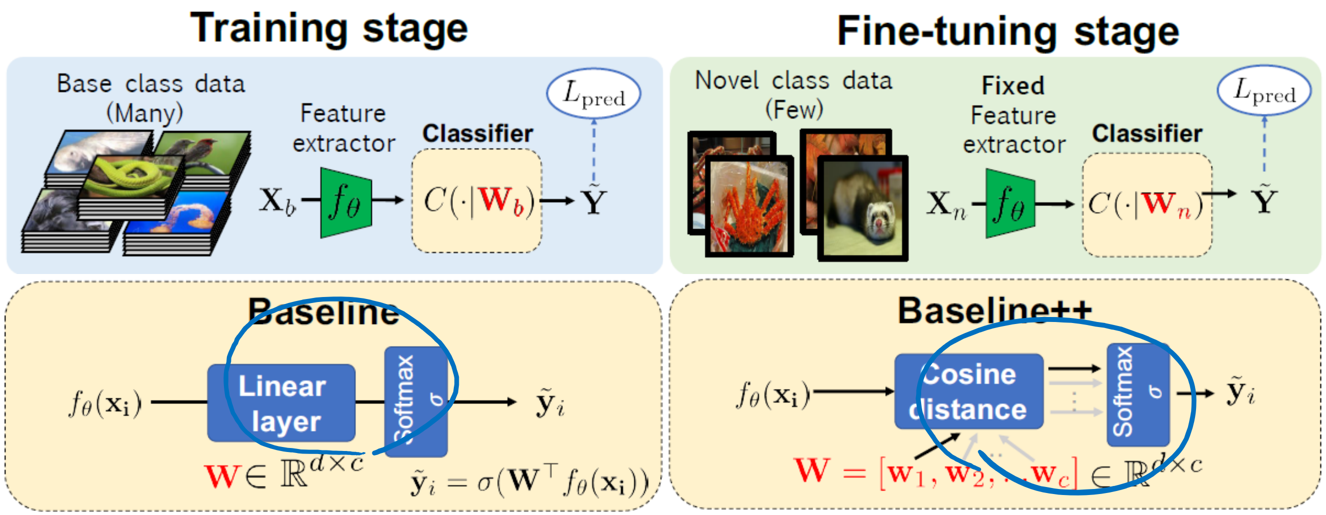
$$\mathcal{L}_{\text{Gen}} = -\log D([\mathbf{p}_{t+n}, \hat{\mathbf{x}}_{t+n}])$$

Update Discriminator (D)

$$\mathcal{L}_{\text{Disc}} = -\log D([\mathbf{p}_{t+n}, \mathbf{x}_{t+n}]) - 0.5\log(1 - D([\mathbf{p}_{t+n}, \hat{\mathbf{x}}_{t+n}])) - 0.5\log(1 - D([\mathbf{p}_{t+n}, \mathbf{x}_t])),$$

A Closer Look at FSL (1/3)

- Idea
  - **Deeper backbones** significantly reduce the gap across existing FSL methods. (with decreased domain shifts between base and novel classes)
  - A slightly modified baseline method (**baseline++**) surprisingly achieves competitive performance.
  - Simple baselines (**baseline** and **baseline++**: trained on base and fine-tuned on novel) outperform representative FSL methods when the domain shift grows larger.

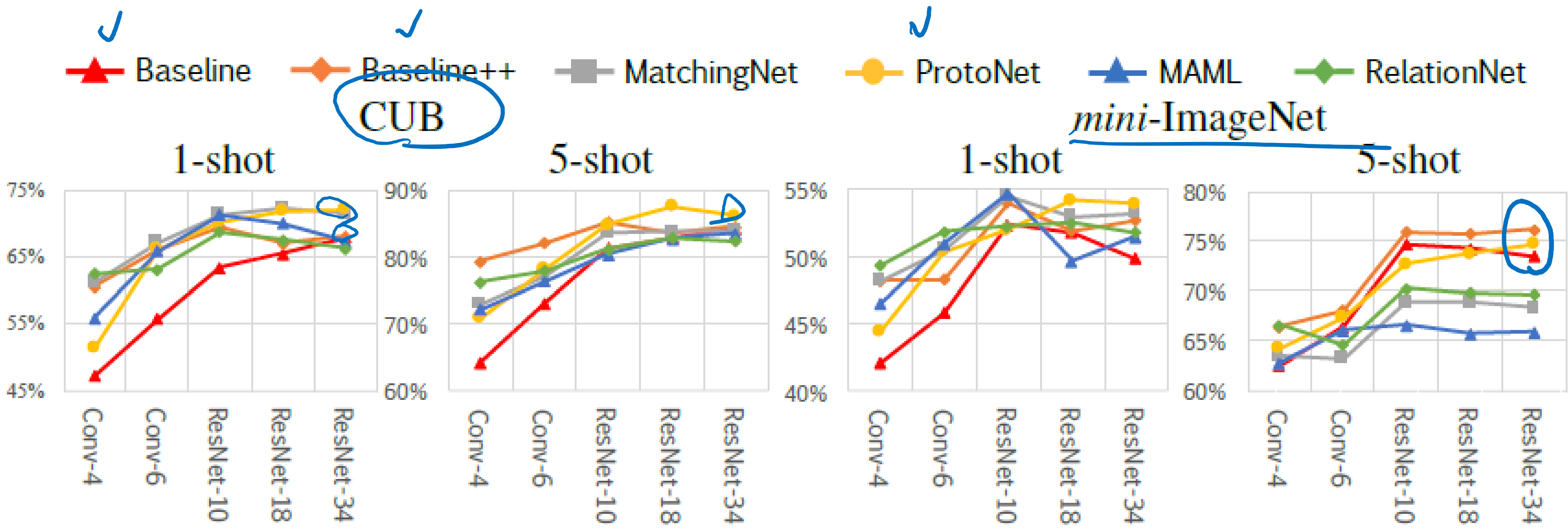


use **cosine distances** between the input feature and the weight vector for each class to reduce intra-class variations

Chen et al., A Closer Look at Few-shot Classification, *ICLR*, 2019

A Closer Look at FSL (2/3)

- Performance with deeper backbones
  - For CUB, gaps among different methods diminish as the backbone gets deeper.
  - For mini-ImageNet, some meta-learning methods are even beaten by baselines with a deeper backbone.

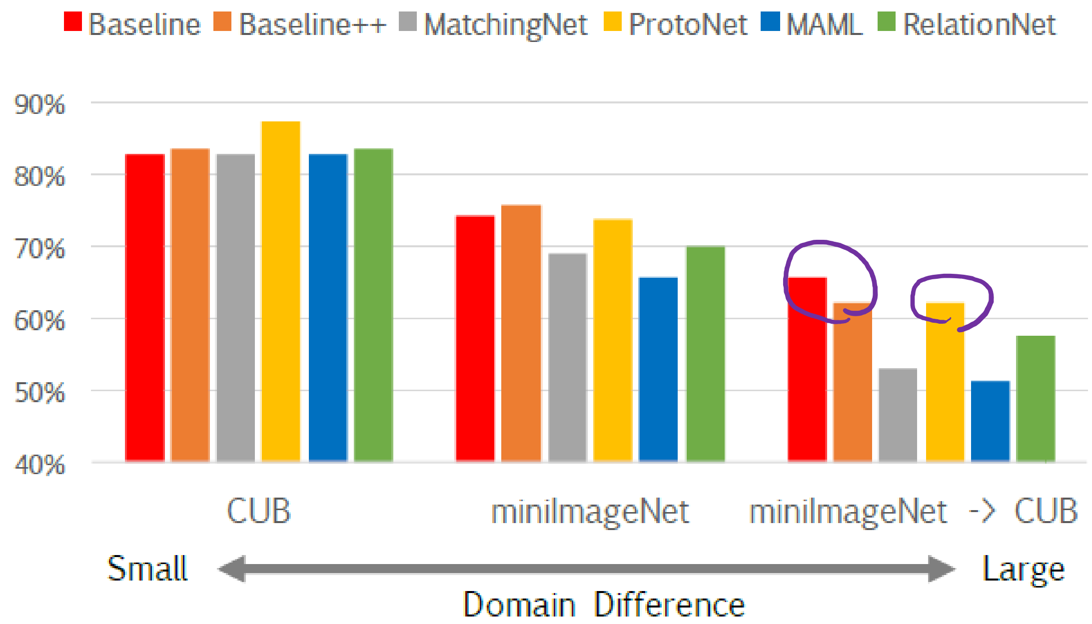


Chen et al., A Closer Look at Few-shot Classification, *ICLR*, 2019

A Closer Look at FSL (3/3)

meta-train : Base class mINet
meta-test : novel CUB

- Performance with domain shifts (using ResNet-18)
  - Existing FSL methods fail to address large domain shifts (e.g., mini-ImageNet → CUB) and are inferior to the baseline methods.
  - This highlights the importance of learning to adapt to domain differences in FSL.



Chen et al., A Closer Look at Few-shot Classification, *ICLR*, 2019

What to Cover Today…

- Meta-Learning
  - Definition
  - Parametric & Non-Parametric based Approaches
- Meta-Learning for Few-Shot Learning
  - Few-Shot Classification
    - Metric Learning vs. Data Hallucination
  - Few-Shot Image Segmentation

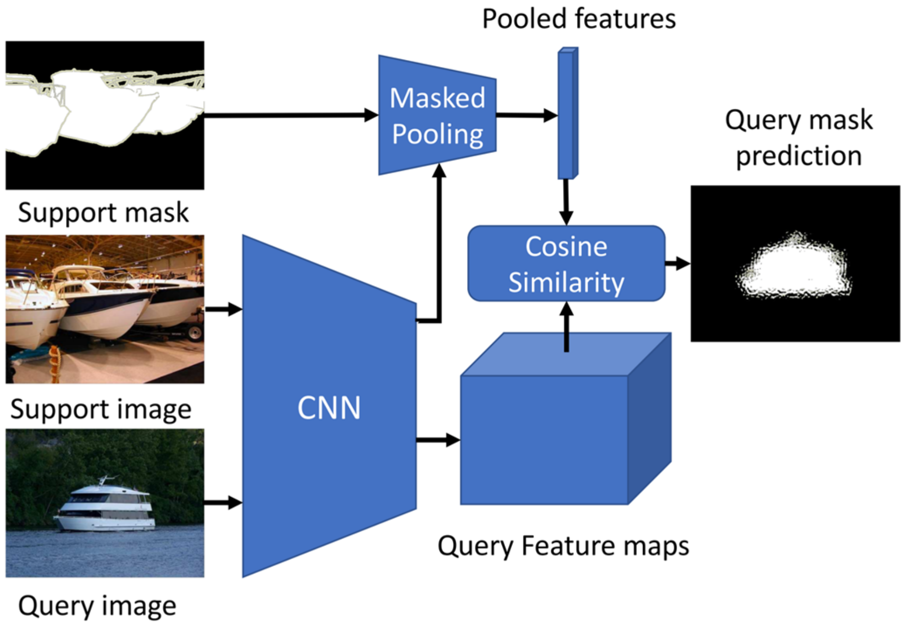
Selected slide credits: C. Finn, S. Levine, & H.-Y. Lee

Semantic Segmentation

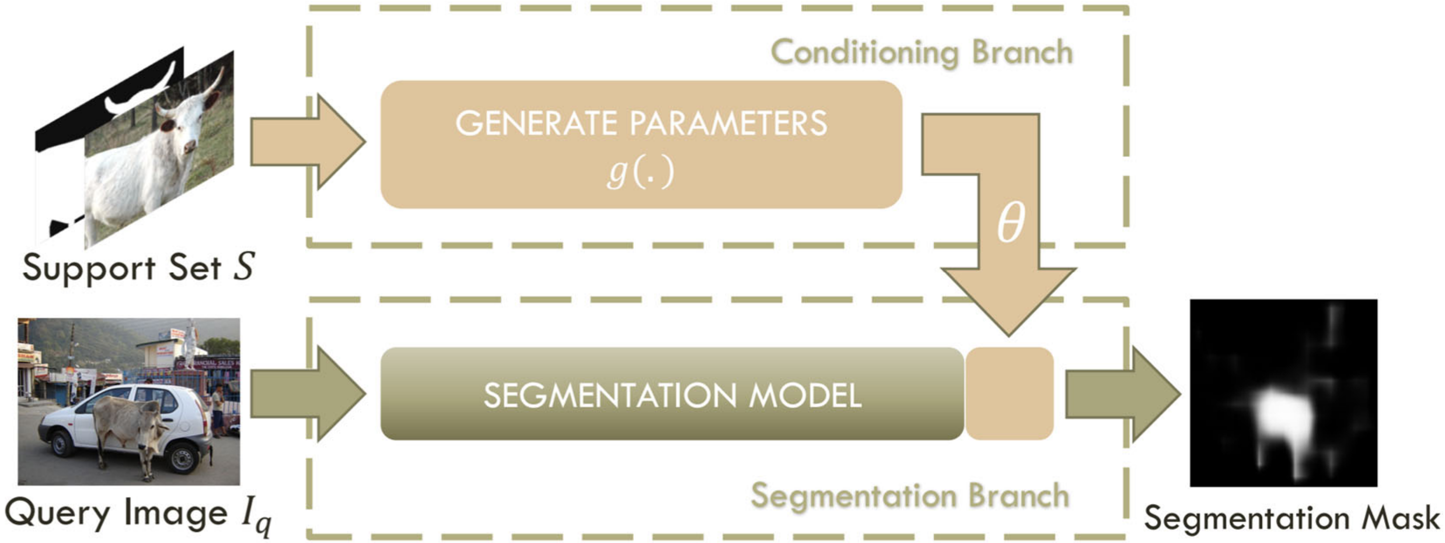
- Goal
  - Assign a class label to each pixel in the input image
  - Don’t differentiate instances, only care about pixels

Few-Shot Segmentation

- A large number of image categories are with pixel-wise ground truth labels, while a small number of them are with limited.
- A **shared backbone** produces feature maps for both support and query images.
- **Prototypes** for each class is obtained by **masked pooling** from support feature maps.
- Query feature maps are then compared with the pooled prototypes **pixel-by-pixel.**
- Typically, **cosine similarity** is adopted for pixel-wise feature comparison.



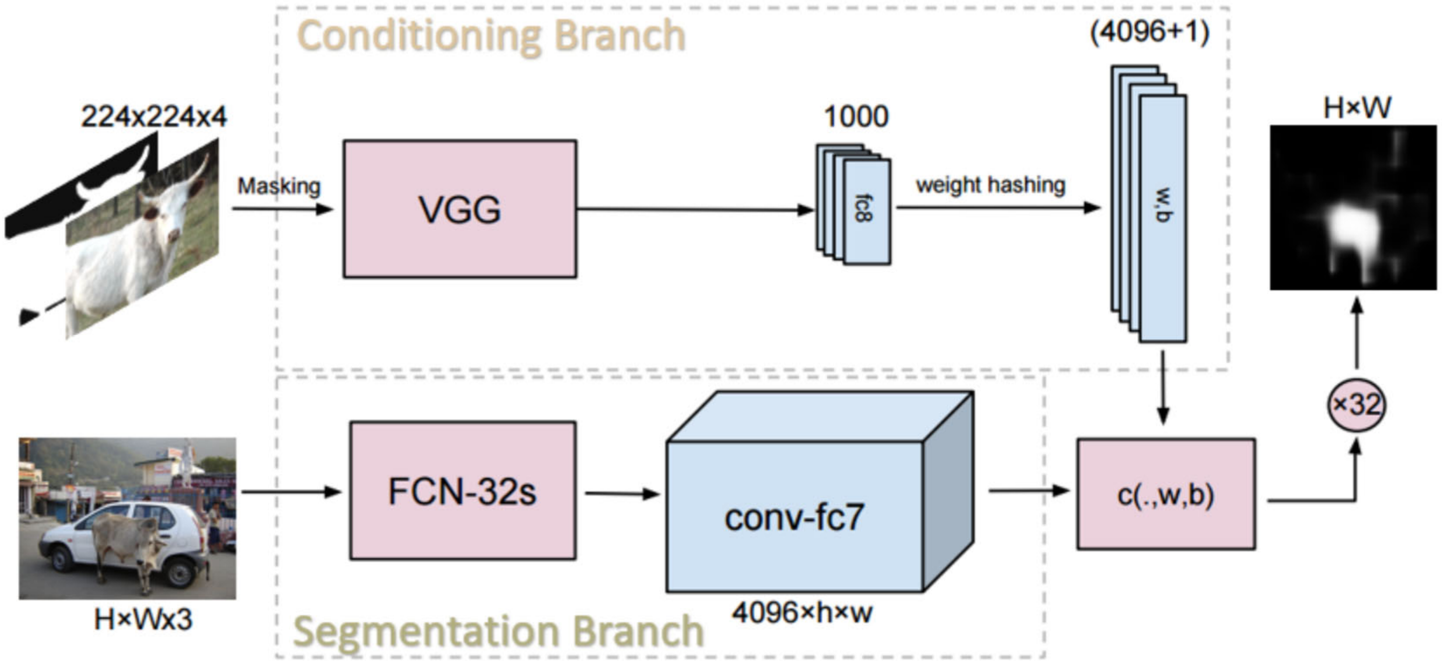
OSLSM [BMVC 2017]



- S is an annotated image from a new semantic class
- Input S to a function g that outputs a set of parameters θ
- θ is used to parameterize part of the segmentation model which produces a segmentation mask given I_q

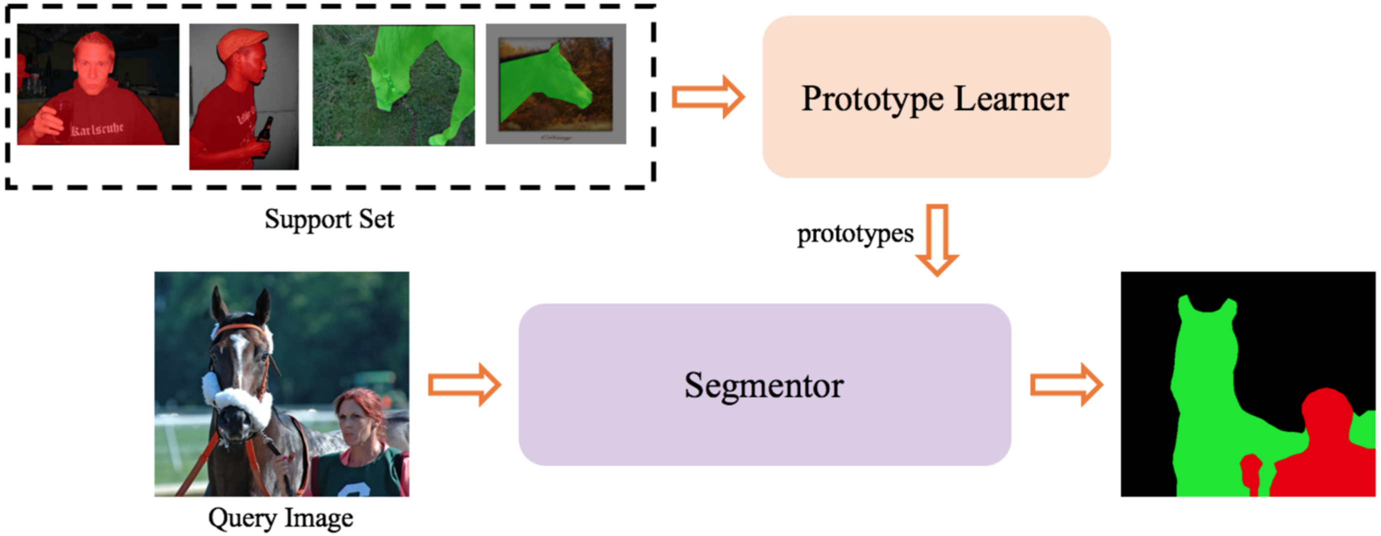
Shaban, Amirreza, et al. "One-shot learning for semantic segmentation." BMVC 2017

OSLSM [BMVC 2017]



Shaban, Amirreza, et al. "One-shot learning for semantic segmentation." BMVC 2017

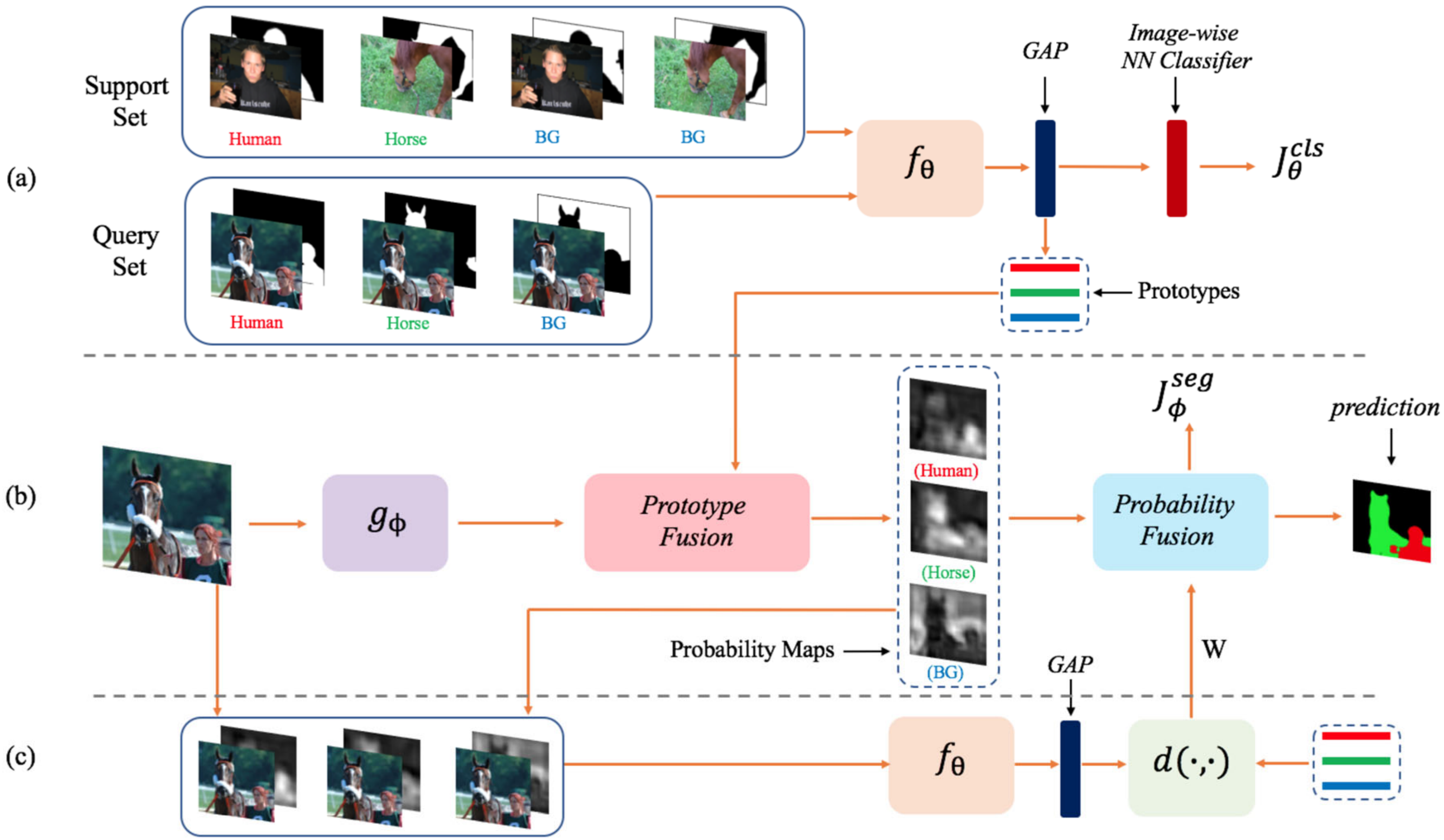
Prototype Learning [BMVC 2018]



- A prototype is learned for each foreground class and the background class.
- Prototypes are used to predict rough segmentation maps for each class.
- The final prediction is optimized using probability fusion.

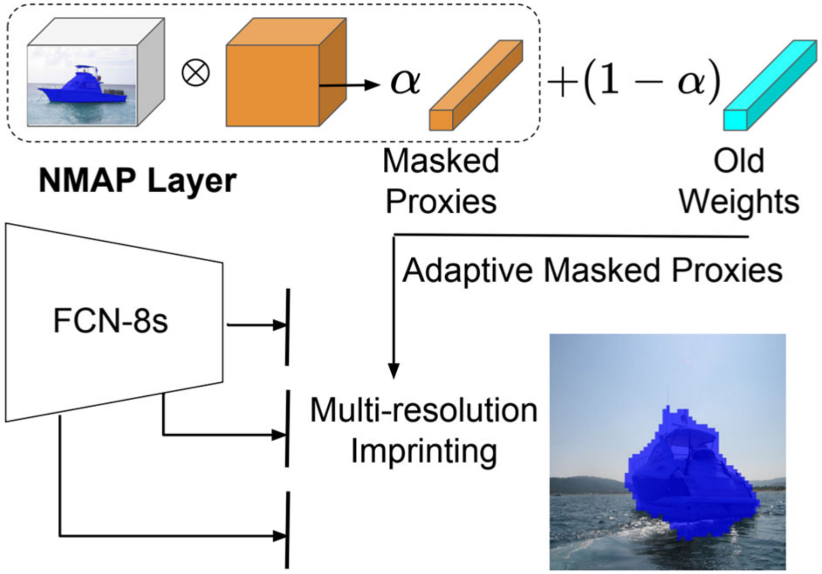
Dong, Nanqing, and Eric Xing. "Few-Shot Semantic Segmentation with Prototype Learning." BMVC. 2018.

PL [BMVC 2018]



Dong, Nanqing, and Eric Xing. "Few-Shot Semantic Segmentation with Prototype Learning." BMVC. 2018.

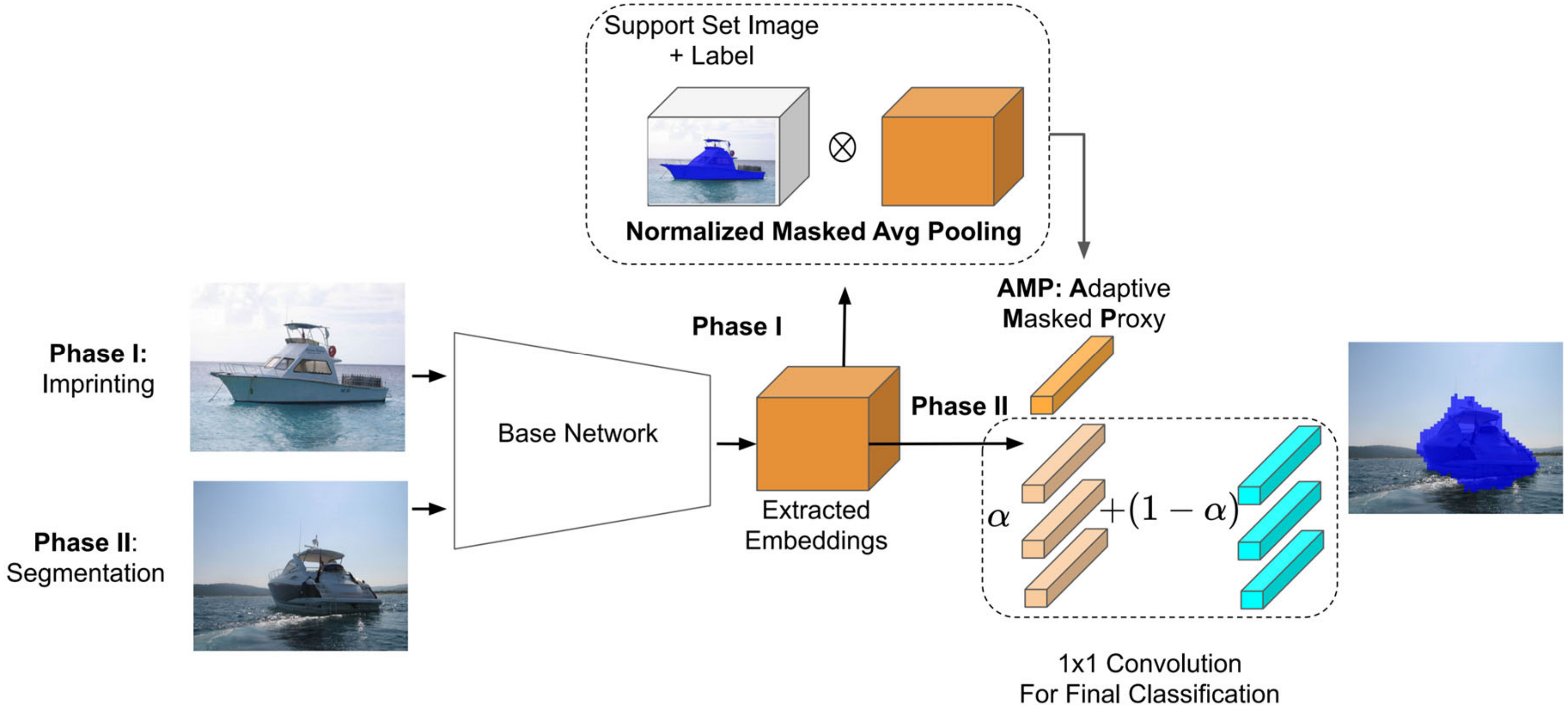
AMP [ICCV 2019]



- Adaptive masked proxies (i.e., prototypes') are extracted for ach semantic class.
- Proxies update themselves in a continuous stream of data (e.g., video).
- Proxies from different resolution levels are used in multi-resolution imprinting

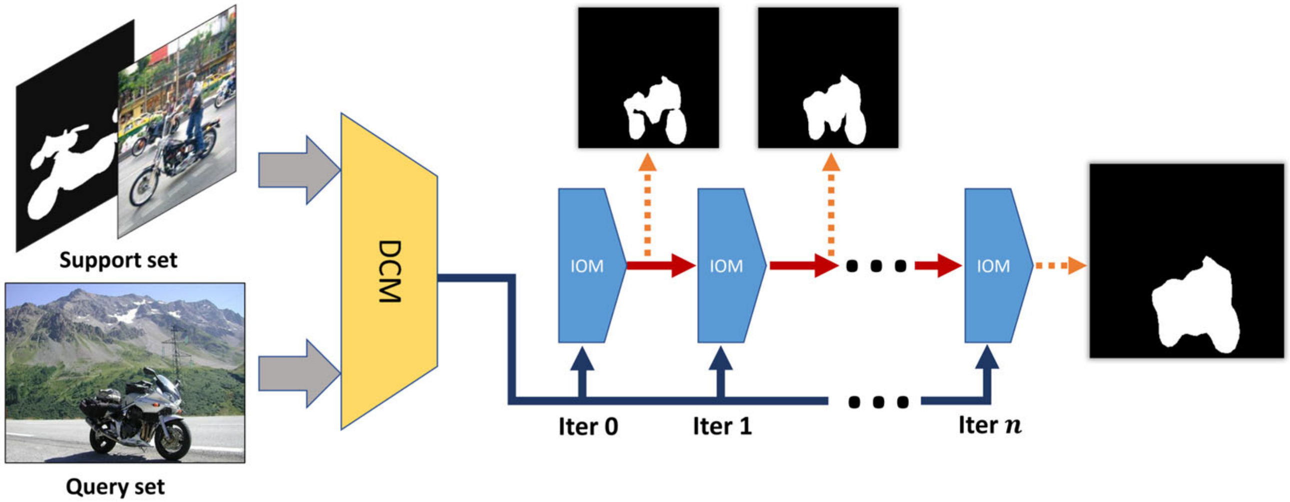
Jagersand et al., AMP: Adaptive masked proxies for few-shot segmentation, ICCV 2019.

AMP [ICCV 2019]



Jagersand et al., AMP: Adaptive masked proxies for few-shot segmentation, ICCV 2019.

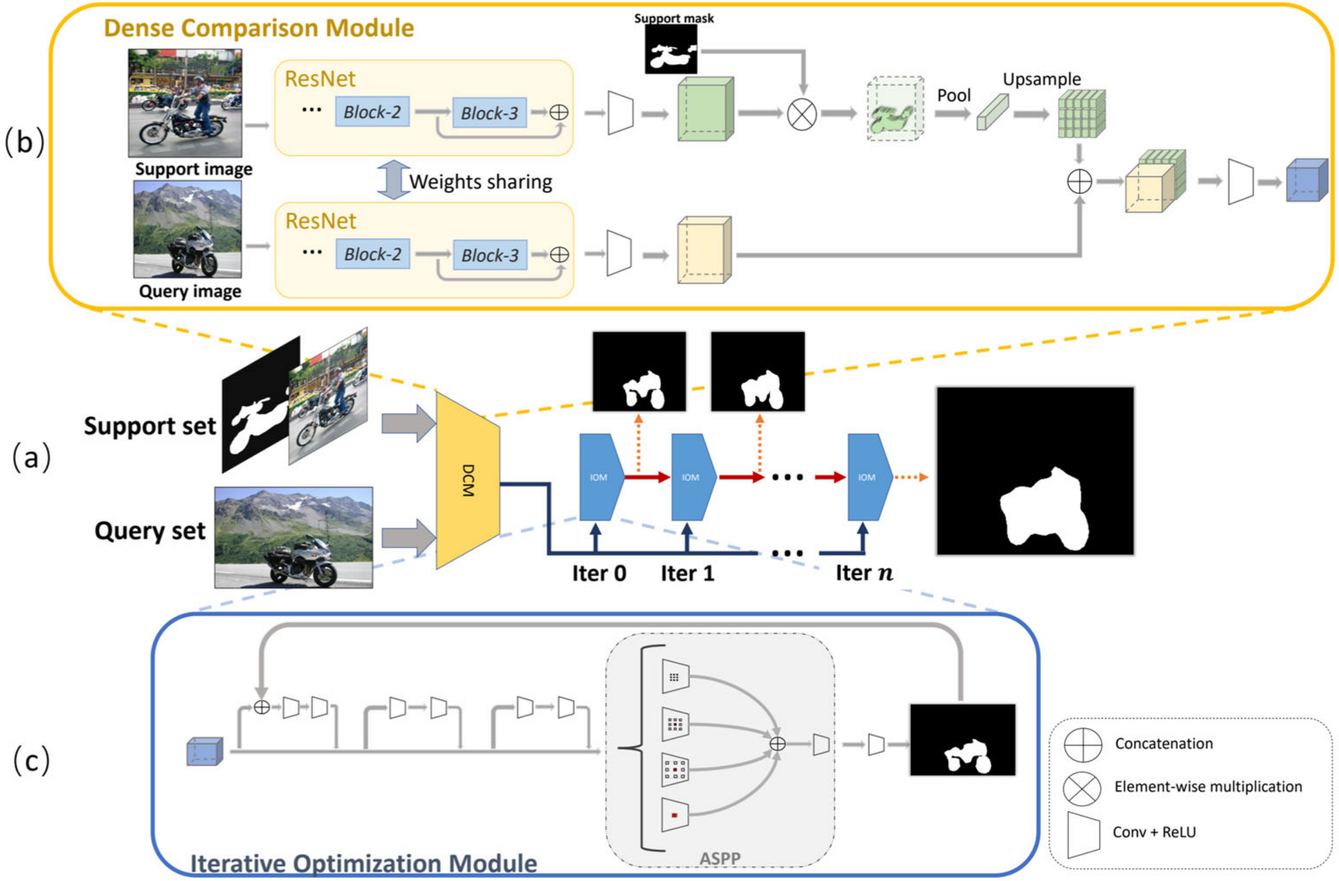
CANet [CVPR 2019]



- Dense comparison module (DCM) concatenates prototypes to each spatial location in query feature map
- Rough segmented maps are produced after comparing with mask-pooled feature prototypes
- The final result is optimized in an iterative manner

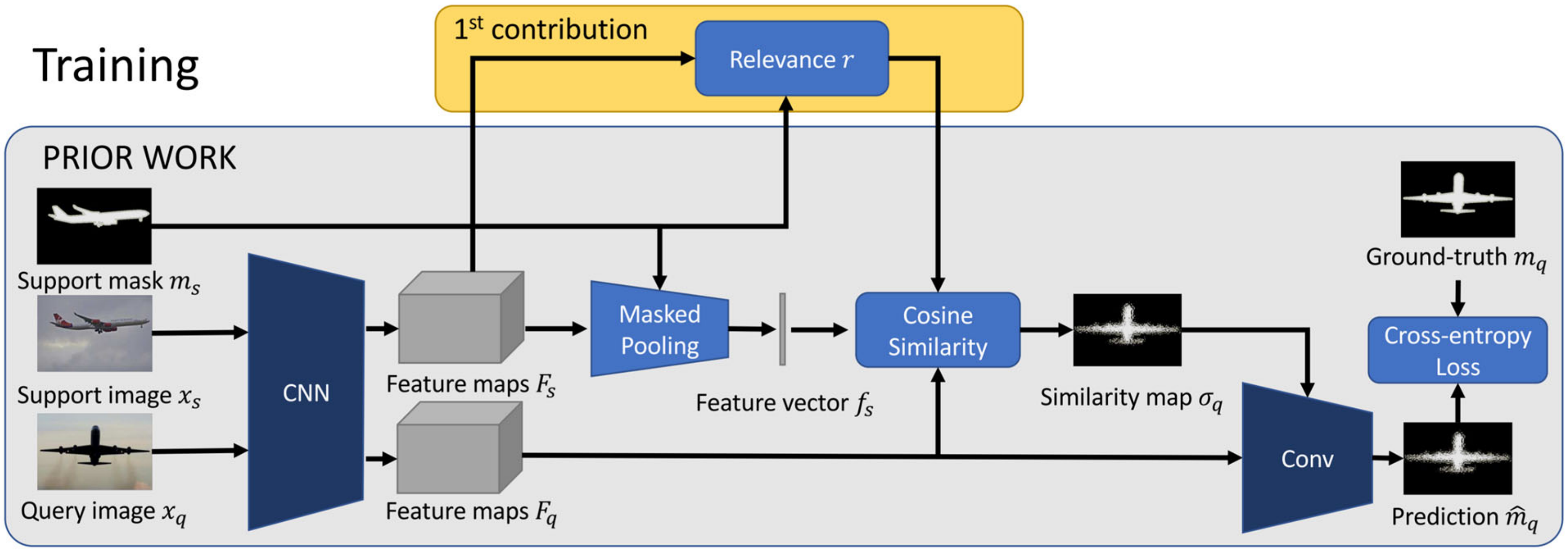
Chi, et al. CANet: Class-agnostic segmentation networks with iterative refinement and attentive FSL, CVPR 2019.

CANet [CVPR 2019]



Chi, et al. CANet: Class-agnostic segmentation networks with iterative refinement and attentive FSL, CVPR 2019.

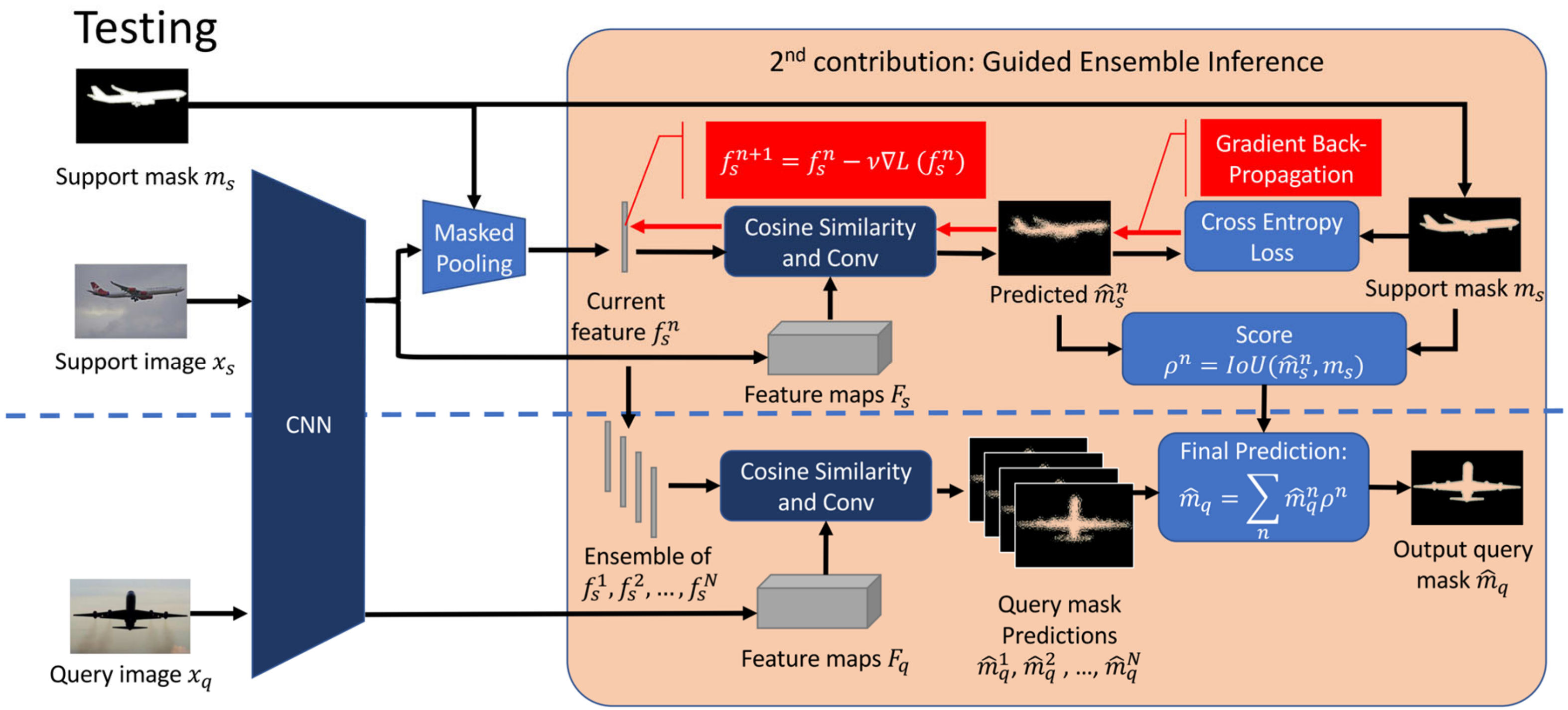
FWB [ICCV 2019]



- Standard FSL methods (e.g., shared backbone, masked pooling…) are used during training.
- A 'relevance' factor is added and taken into account during cosine similarity computation.

Todorovic et al., Feature weighting and boosting for few-shot segmentation, ICCV 2019.

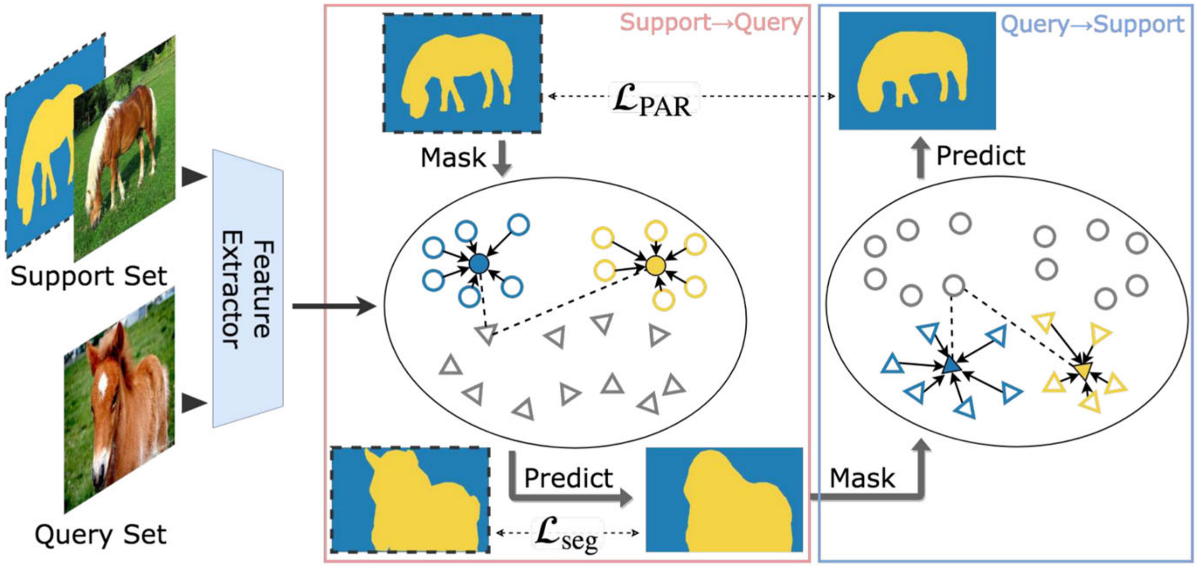
FWB [ICCV 2019]



- During inference, ensemble is utilized to select the best set of parameters
- Prototypes are used to predict the support masks reversely, which can be compared to the ground truth.

Todorovic et al., Feature weighting and boosting for few-shot segmentation, ICCV 2019.

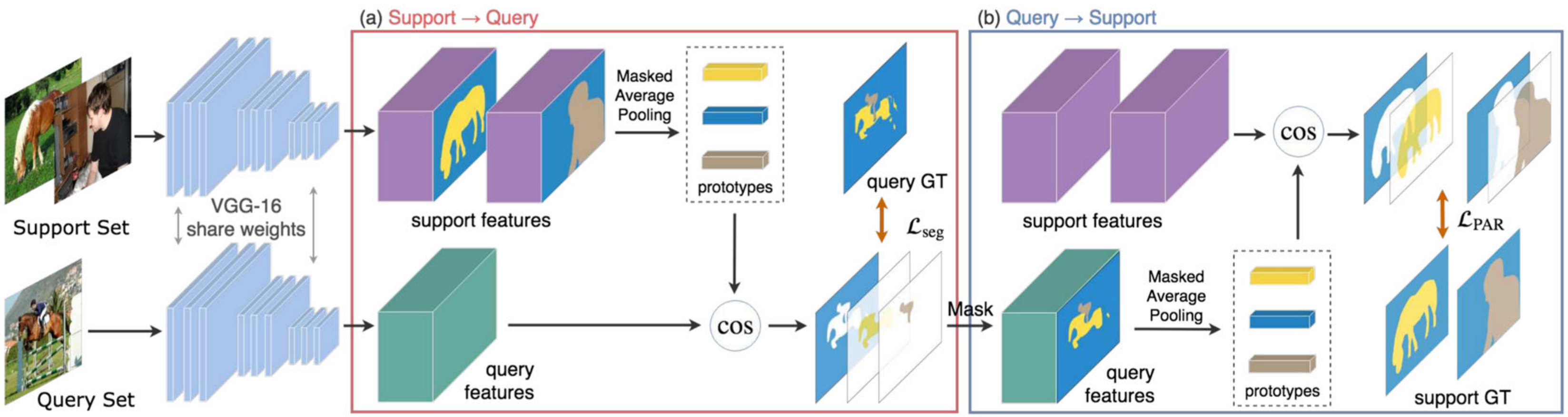
PANet [ICCV 2019]



- Extracted prototypes are first used to predict query masks, as standard FSL methods do.
- Predicted query masks are used to generate new prototypes and reversely predict support masks
- Similar concept to that of the 'cycle consistency' (support→query; query→support)

Kaixin, et al., Panet: Few-shot image semantic segmentation with prototype alignment, ICCV 2019.

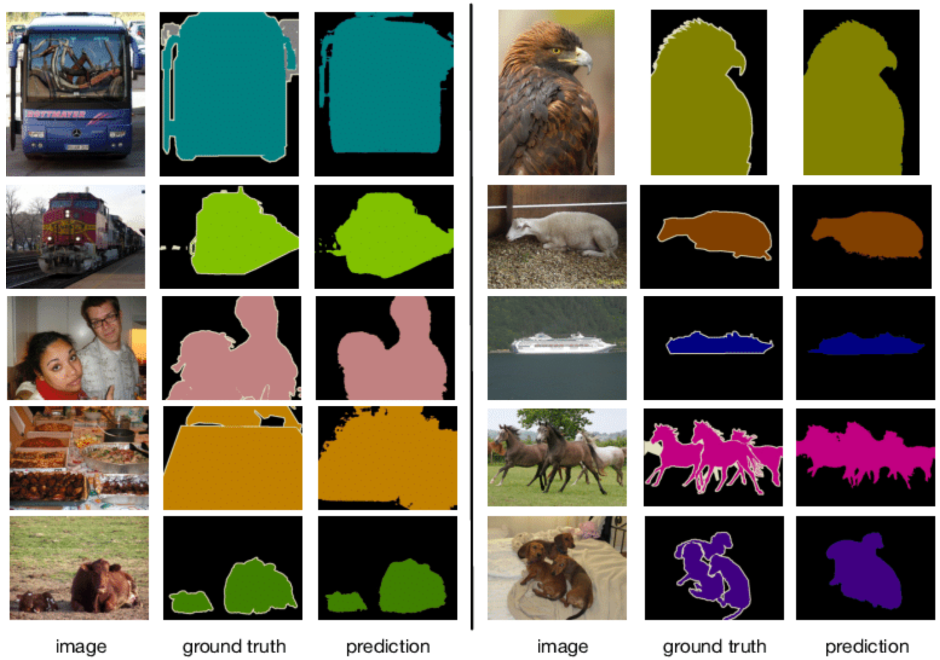
PANet [ICCV 2019]



Kaixin, et al., Panet: Few-shot image semantic segmentation with prototype alignment, ICCV 2019.

Dataset & Evaluation Metric

- **Datasets**
  - **PASCAL VOC 2012** (main)
    - 20 classes
    - Split: (15 *base* + 5 *novel*)
  - coco (secondary)
- **Evaluation Metrics**
  - **Binary-mIoU** (difficult)
  - FB-mIoU (easy)
    - Foreground/Background IoU



Performance Comparisons

Method		Split-0	Split-1	Split-2	Split-3	Mean
Reduced-DFCN8s		39.2	48.0	39.3	34.2	40.2
OSLSM	BMVC 2017	33.6	55.3	40.9	33.5	40.8
co-FCN	ICLRW 2018	36.7	50.6	44.9	32.4	41.2
AMP	ICCV 2019	41.9	50.2	46.7	34.7	43.4
SG-One		40.2	58.4	48.4	38.4	46.4
PANet	ICCV 2019	42.3	58.0	51.1	41.2	48.1
PRNet		51.6	61.3	53.1	47.6	53.4
Co-att		49.5	65.5	50.0	49.2	53.5
CANet	CVPR 2019	52.5	65.9	51.3	51.9	55.4
PGNet	ICCV 2019	56.0	66.9	50.6	50.4	56.0
FWB	ICCV 2019	51.3	64.5	56.7	52.2	**56.2**

What We've Covered Today…

- Meta-Learning
  - Definition
  - Parametric & Non-Parametric based Approaches
- Meta-Learning for Few-Shot Learning
  - Few-Shot Classification
    - Metric Learning vs. Data Hallucination
  - Few-Shot Image Segmentation
  - Few-Shot Object Detection (next lecture)
- Meta-Learning for Domain Generalization (next lecture)
  - From Domain Adaptation to Domain Generalization
- Challenges in Few-Shot Learning Tasks (next lecture)

Selected slide credits: C. Finn, S. Levine, & H.-Y. Lee