Meta Learning: Learn to learn

Hung-yi Lee

What does “meta” mean? meta-X = X about X

Source of image: https://medium.com/intuitionmachine/the-brute-force-method-of-deep-learning-innovation-58b497323ae5 (Denny Britz’s graphic)

這門課的作業在做甚麼？

感謝 沈昇勳 同學提供圖檔

Industry

Using 1000 GPUs to try
1000 sets of hyperparameters

Academia

“Telepathize” (通靈) a set of
good hyperparameters

Can machine automatically determine the hyperparameters?

Machine Learning 101

Machine Learning

= Looking for a function

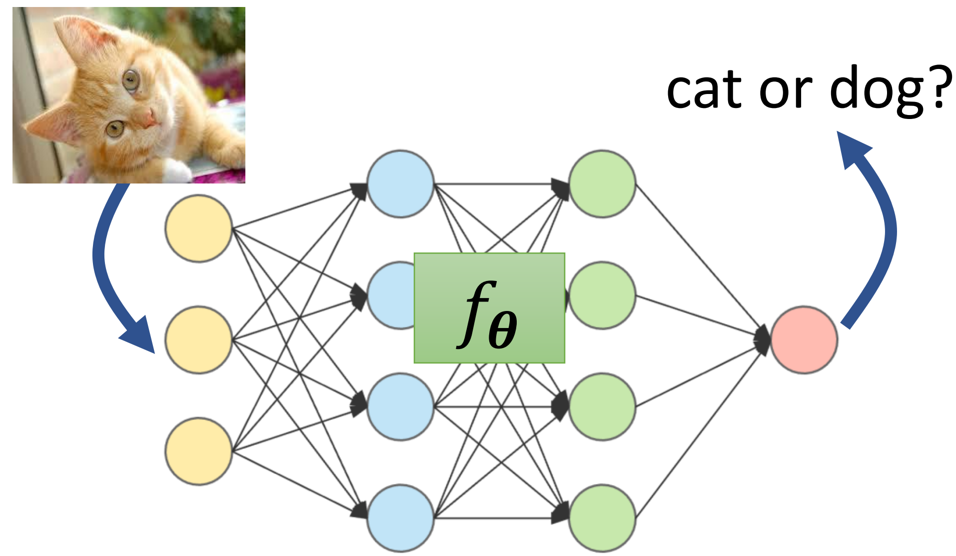
Dog-Cat Classification

$f($  $)$ = "cat"

Step 1: Function with unknown

Step 2: Define loss function

Step 3: Optimization



Weights and biases of neurons are unknown parameters (*learnable*).

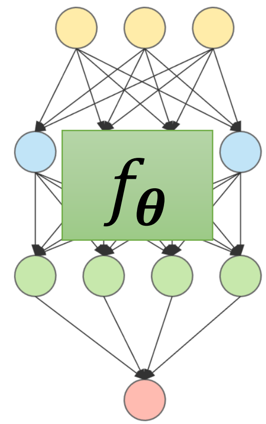
Using $\boldsymbol{\theta}$ to represent the unknown parameters.

Machine Learning

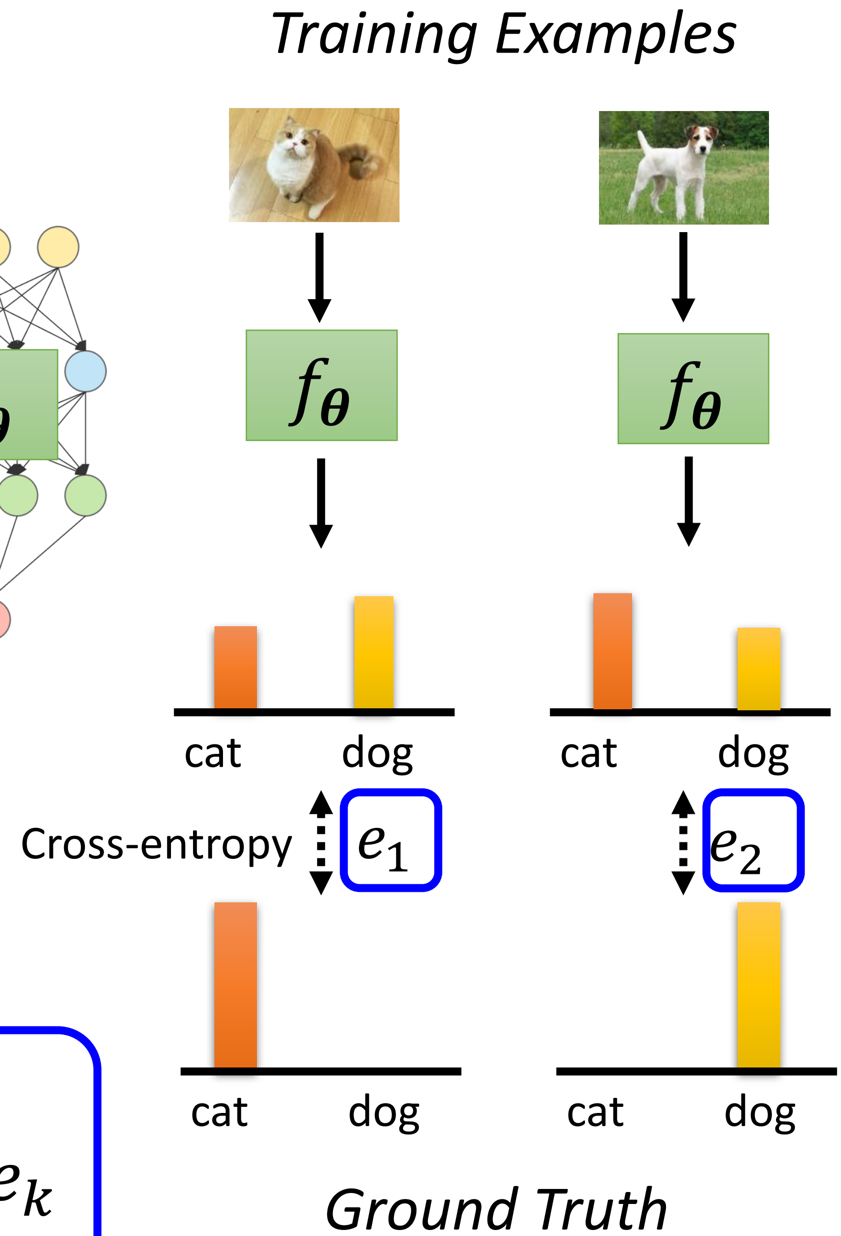
Step 1: Function with unknown

Step 2: Define loss function

Step 3: Optimization



$L(\boldsymbol{\theta})$



$$\boxed{L(\boldsymbol{\theta})} = \boxed{\sum_{k=1}^{K} e_k}$$

Machine Learning 101

Step 1: Function with unknown

Step 2: Define loss function

Step 3: Optimization

loss: $L(\boldsymbol{\theta}) = \sum_{k=1}^{K} e_k$ sum over training examples

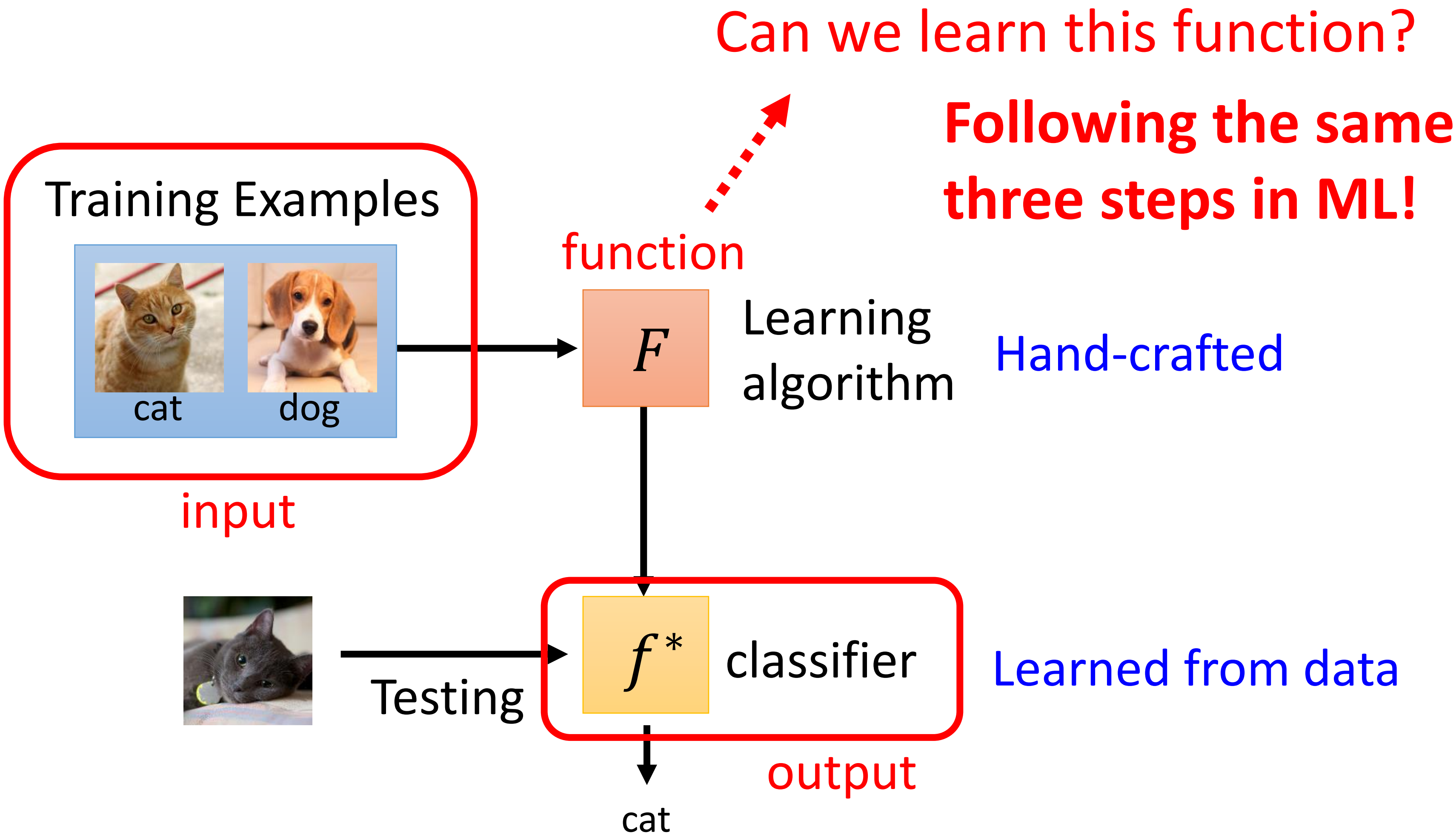
$\boldsymbol{\theta}^* = arg \min_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$

done by gradient descent

$f_{\boldsymbol{\theta}^*}$ is the function learned by learning algorithm from data

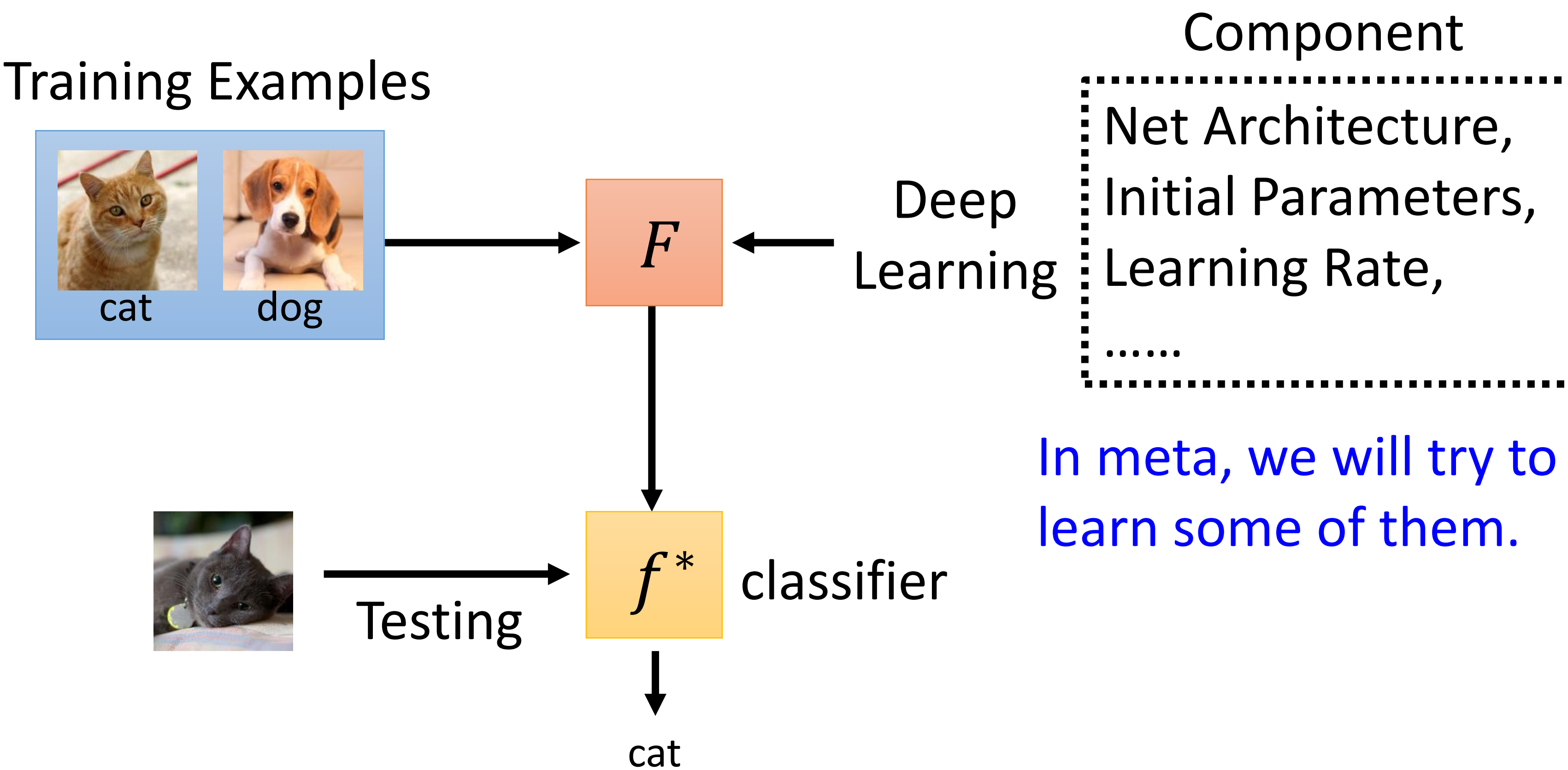
Introduction of Meta Learning

What is Meta Learning?



Meta Learning – Step 1

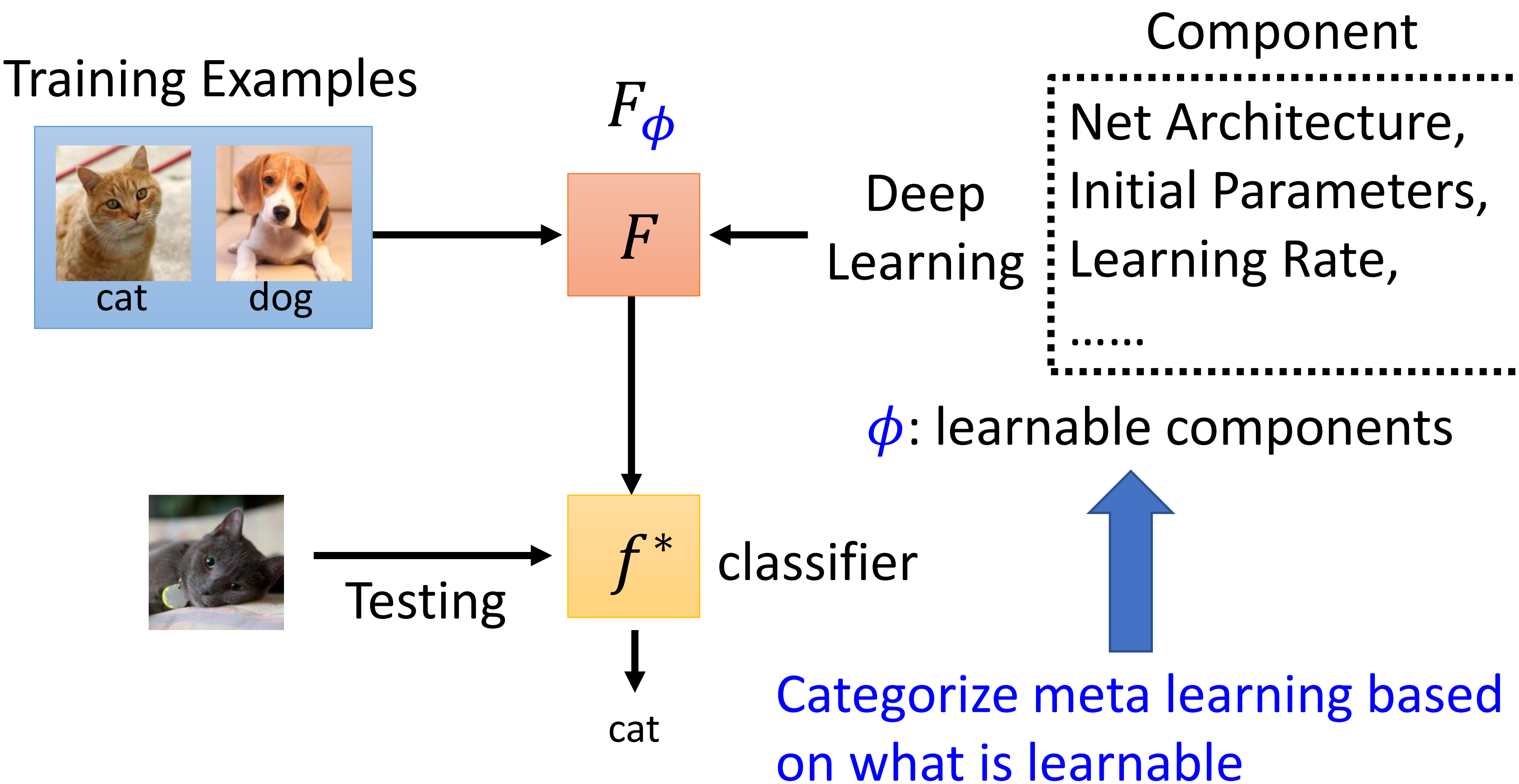
- What is ***learnable*** in a learning algorithm?



In meta, we will try to learn some of them.

Meta Learning – Step 1

- What is ***learnable*** in a learning algorithm?

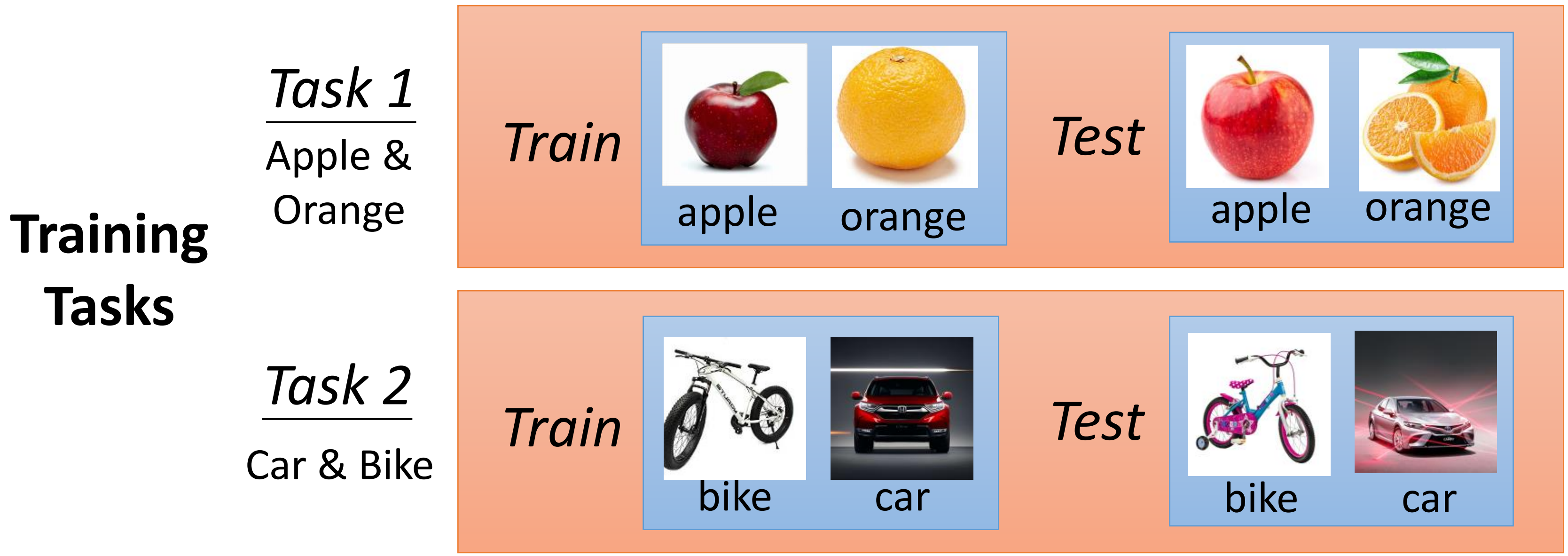


Meta Learning – Step 2

- Define ***loss function*** for ***learning algorithm*** F_{ϕ}

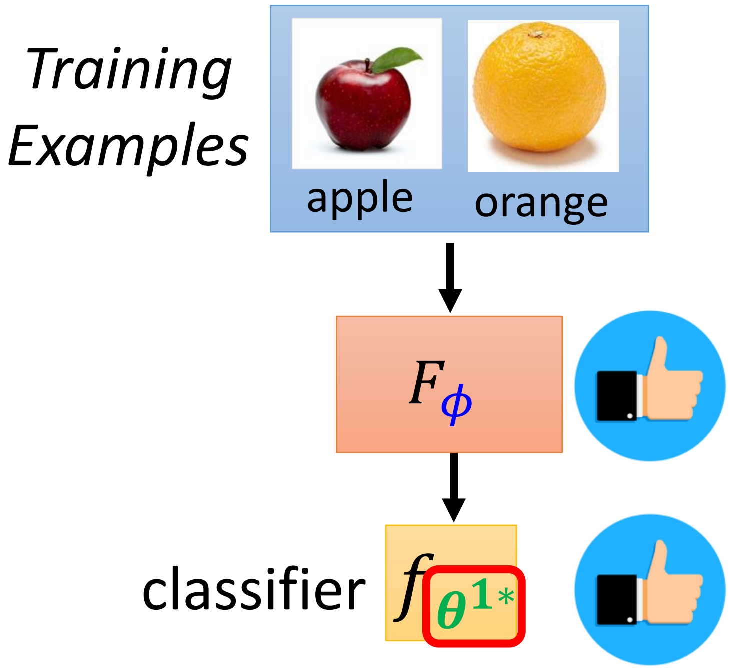
$$L(\phi)$$

$L(\phi)\downarrow$ $L(\phi)\uparrow$



Meta Learning – Step 2

Task 1



How to define $L(\phi)$

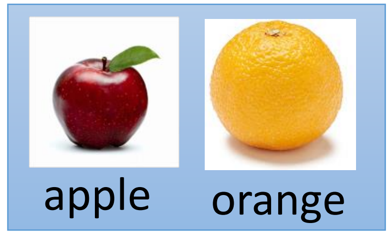
$L(\phi)$ ↓

$\boxed{\theta^{1*}}$: parameters of the classifier learned by F_{ϕ} using the training examples of task 1

Meta Learning – Step 2

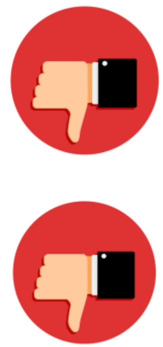
Task 1

Training Examples



F_{ϕ}

classifier $f_{\theta^{1*}}$

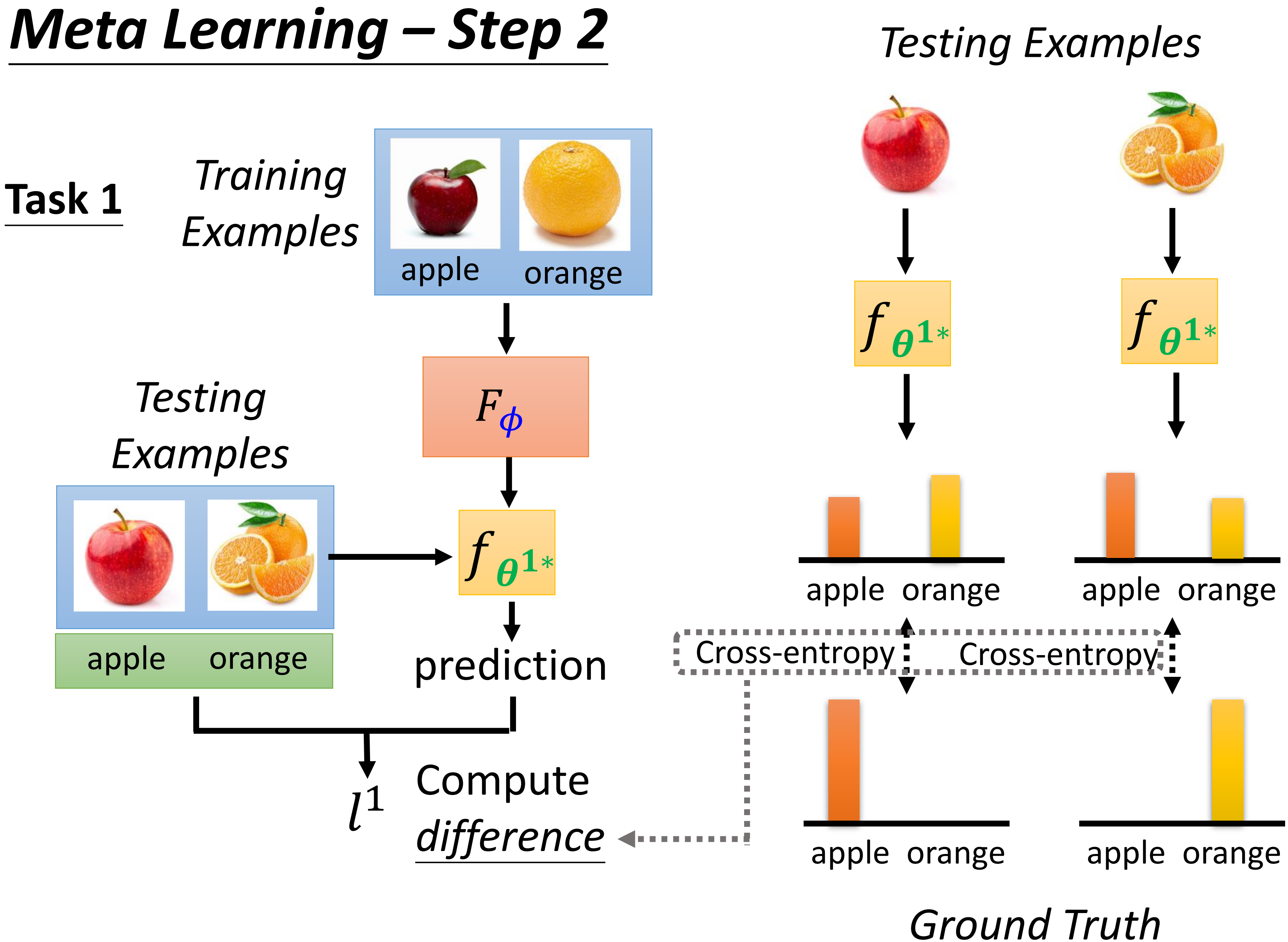


How can we know a classifier is good or bad?

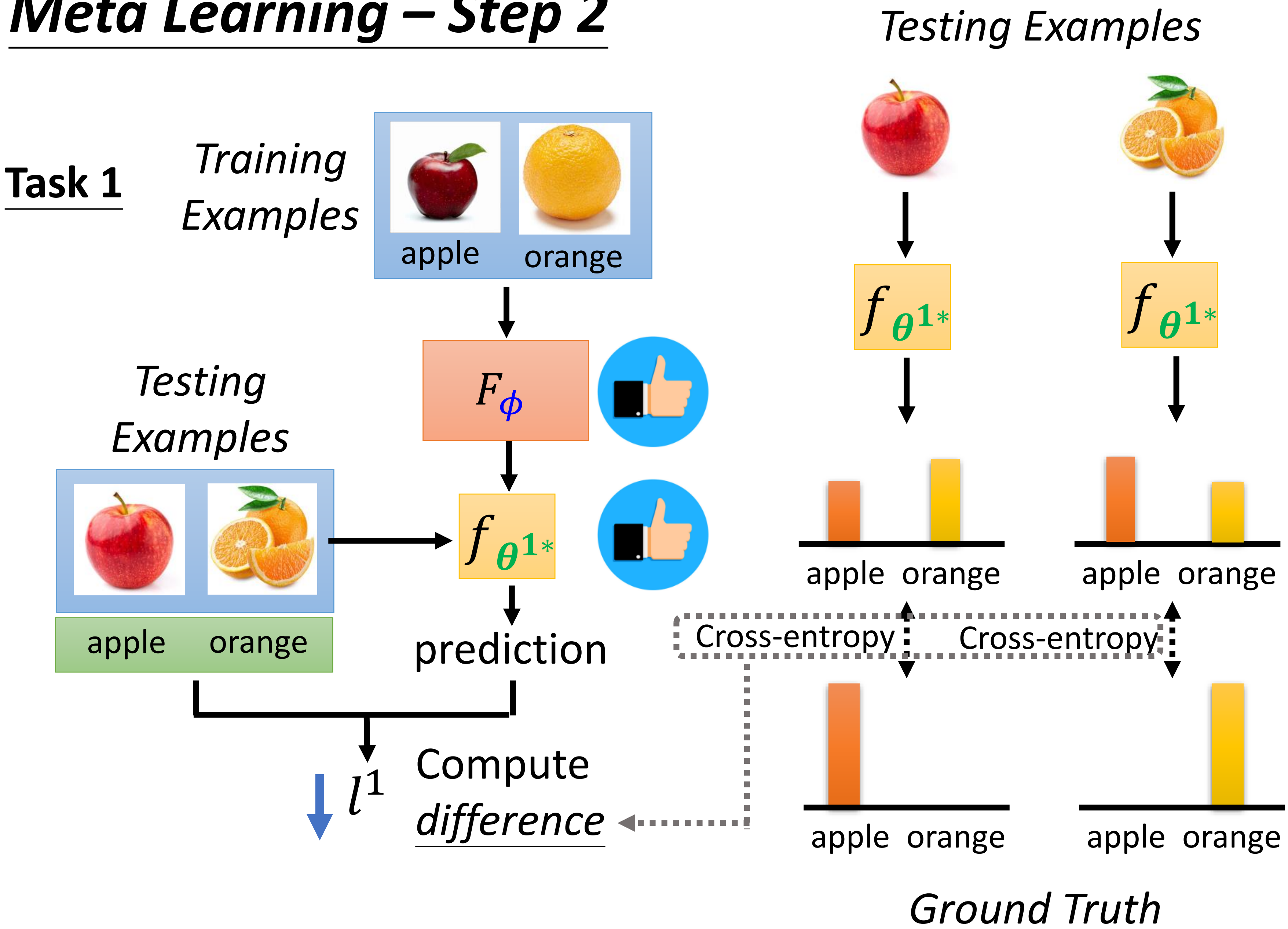
Evaluate the classifier on testing set

How to define $L(\phi)$

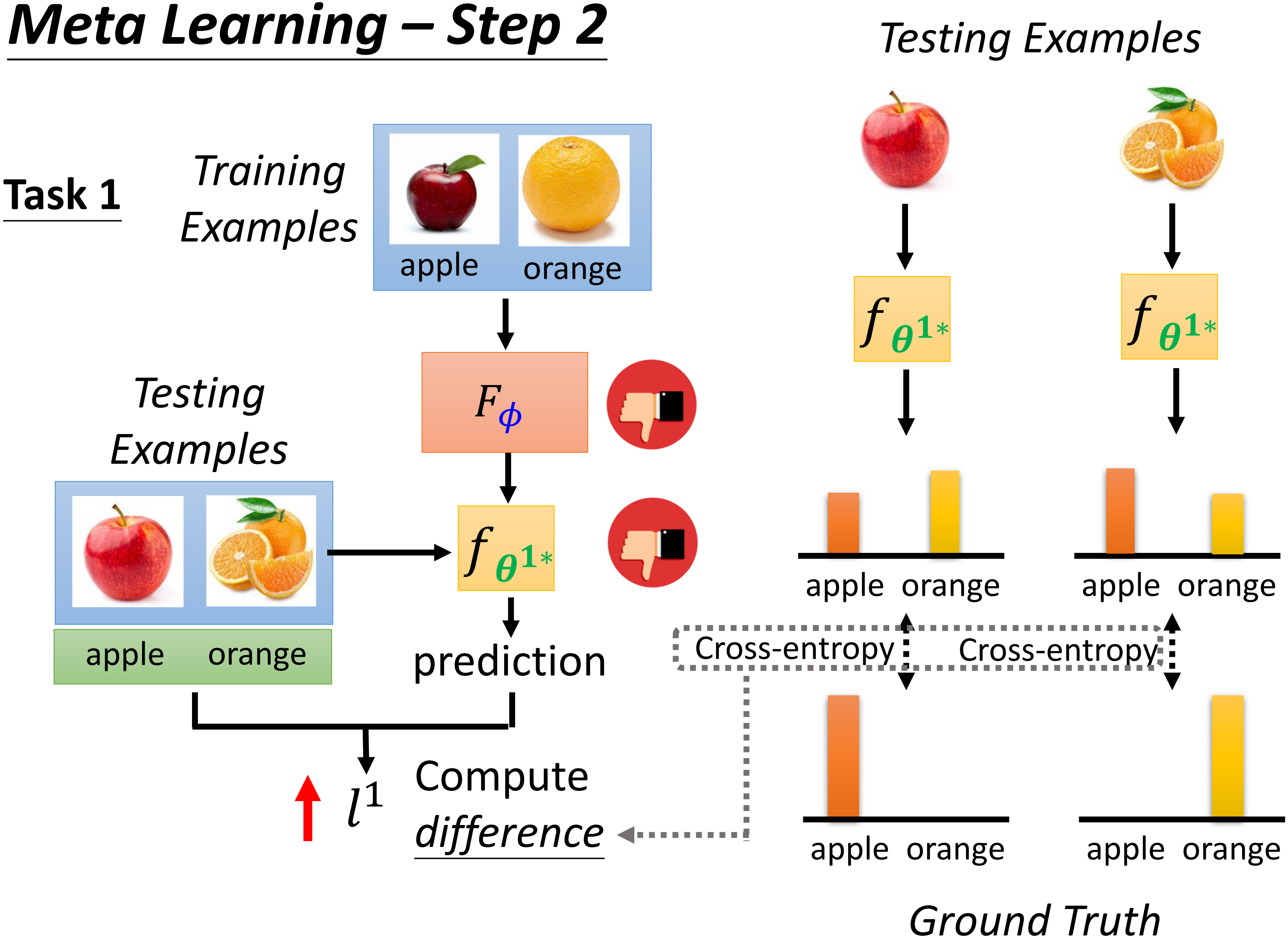
$L(\phi)$ ↑


Meta Learning – Step 2
Task 1
Training Examples
apple
orange
F_ϕ
Testing Examples
apple
orange
$f_{\theta^{1*}}$
prediction
l^1
Compute difference
Testing Examples
$f_{\theta^{1*}}$
$f_{\theta^{1*}}$
apple orange
apple orange
Cross-entropy
Cross-entropy
apple orange
apple orange
Ground Truth

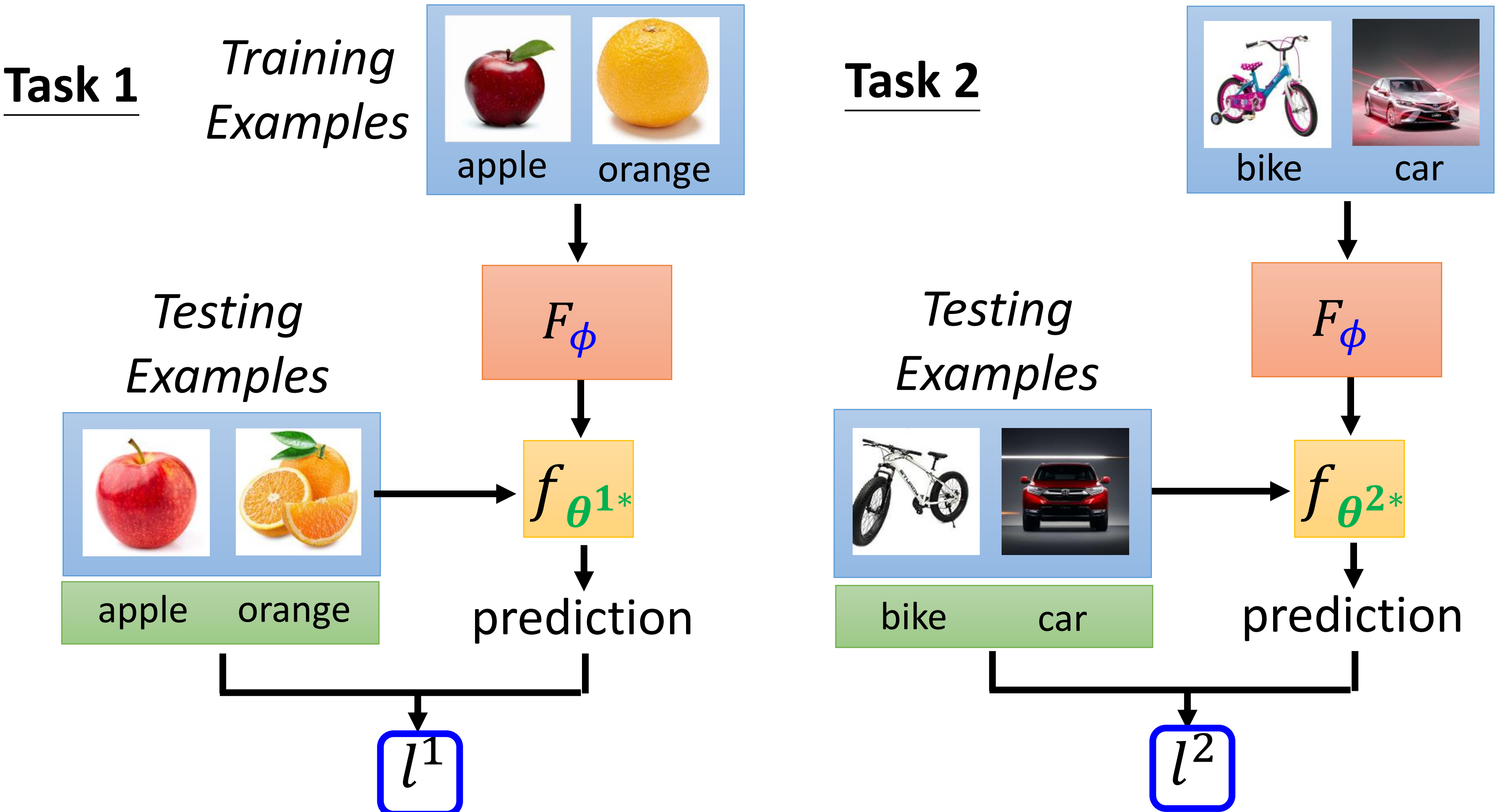
Meta Learning – Step 2



Meta Learning – Step 2

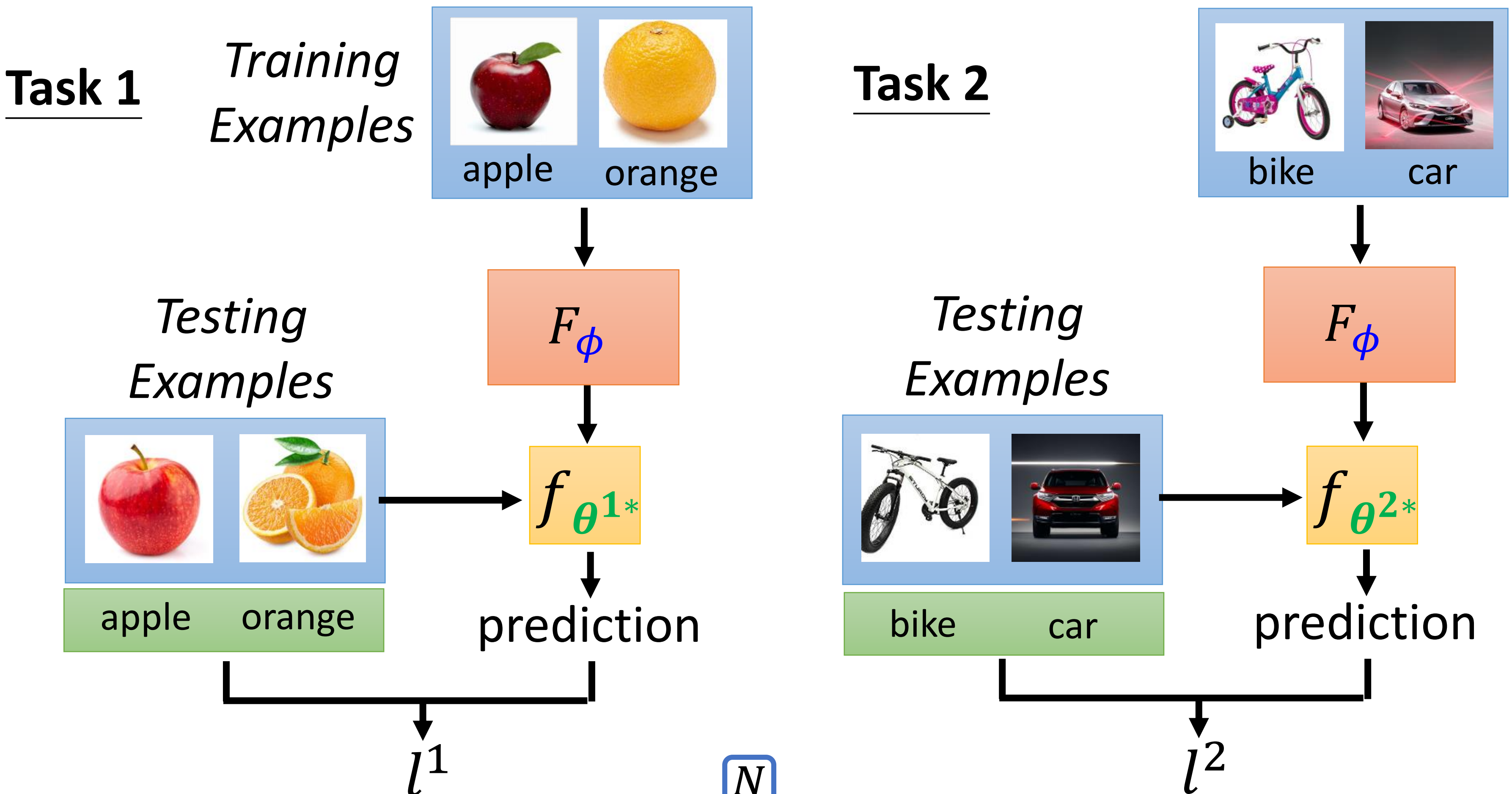


Meta Learning – Step 2



Total loss: $L(\phi) = \boxed{l^1 + l^2}$ (sum over all the training tasks)

Meta Learning – Step 2



Total loss: $L(\phi) = \sum_{n=1}^{\boxed{N}} l^n$ (N is the number of the training tasks)

Meta Learning – Step 2

Testing Examples

$f_{\theta^{1*}}$ $f_{\theta^{1*}}$

Task 1

In typical ML, you compute the loss based on training examples

In meta, you compute the loss based on testing examples

Hold on! You use testing examples during training???

apple orange

prediction

l^1

Compute *difference*

apple orange apple orange

Ground Truth

Meta Learning – Step 2

Testing Examples

$f_{\theta^{1*}}$ $f_{\theta^{1*}}$

Task 1

In typical ML, you compute the loss based on training examples

In meta, you compute the loss based on testing examples

of training tasks.

apple	orange

prediction

l^1

Compute *difference*

Meta Learning – Step 3

- Loss function for learning algorithm $L(\phi) = \sum_{n=1}^{N} l^n$
- Find ϕ that can minimize $L(\phi)$ $\phi^* = arg \min_{\phi} L(\phi)$
- Using the optimization approach you know

If you know how to compute $\partial L(\phi) / \partial \phi$

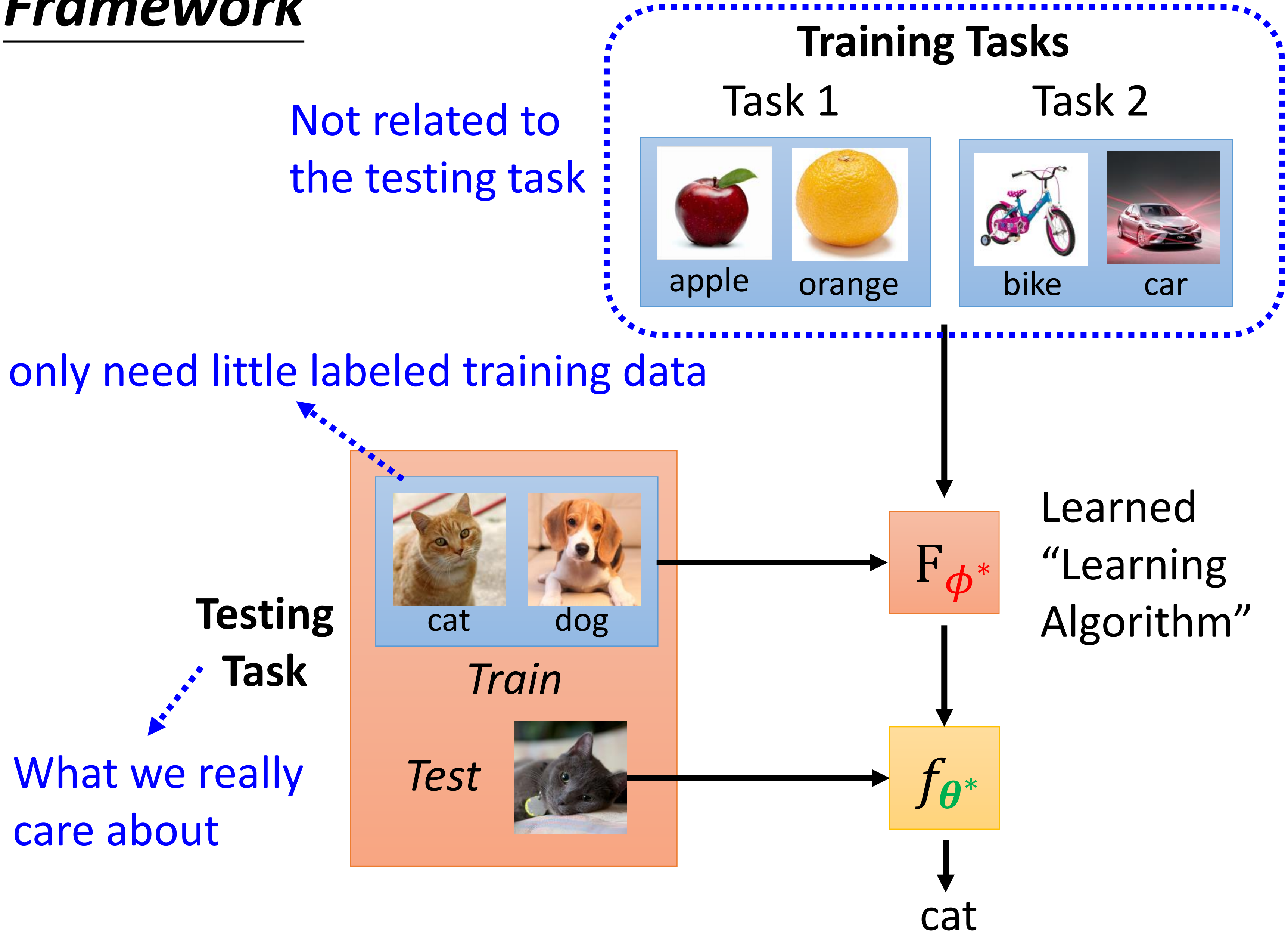
Gradient descent is your friend.

What if $L(\phi)$ is not differentiable?

Reinforcement Learning / Evolutionary Algorithm

Now we have a learned “learning algorithm” F_{ϕ^*}

Framework



Training Tasks
Task 1
Task 2
apple
orange
bike
car
Not related to the testing task
only need little labeled training data
Testing Task
cat
dog
Train
Test
What we really care about
F_{ϕ^*}
Learned "Learning Algorithm"
f_{θ^*}
cat

ML v.s. Meta

Goal

<u>**Machine Learning**</u> ≈ find a function f

Dog-Cat Classification

$f($  $) =$ "cat"

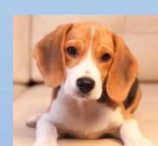
<u>**Meta Learning**</u>

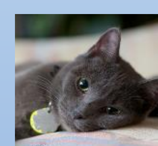
≈ find a function F that finds a function f

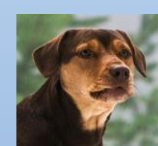
Learning Algorithm

$F($


cat


dog


cat


dog

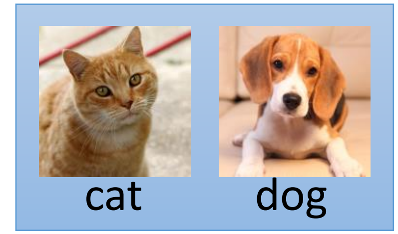
Training Examples

$) = f$

Training Data

Machine Learning

One task



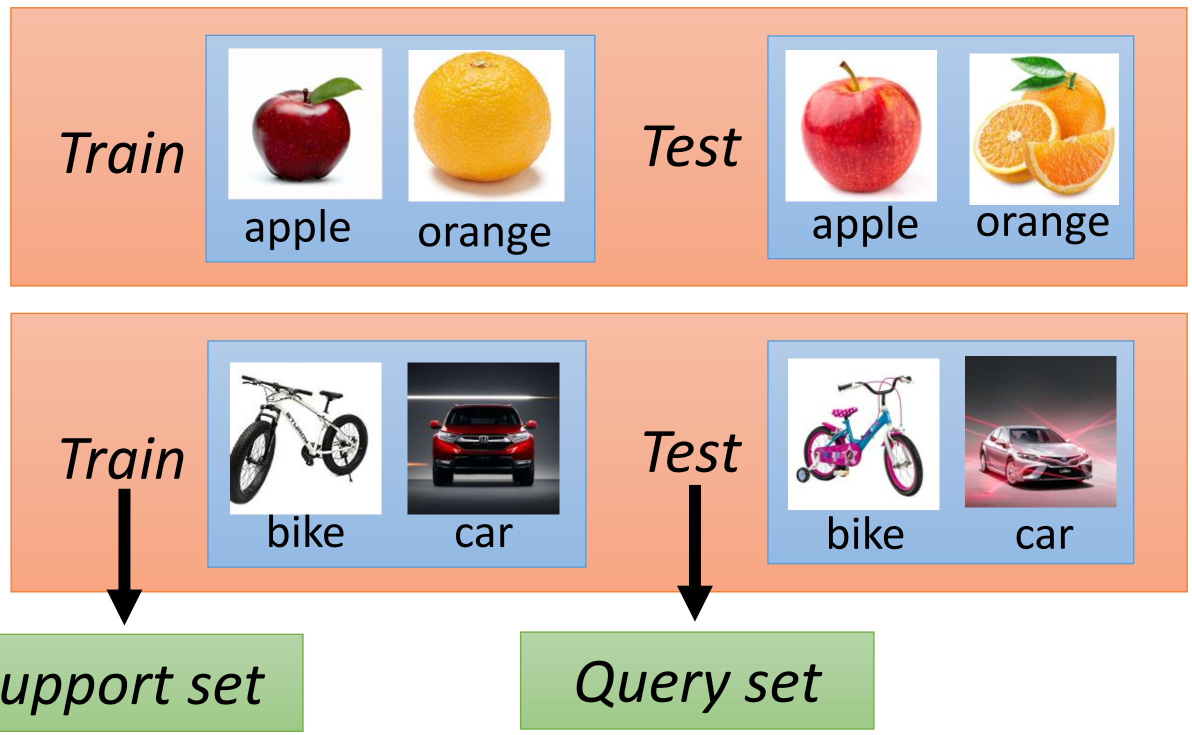
Train

Meta Learning

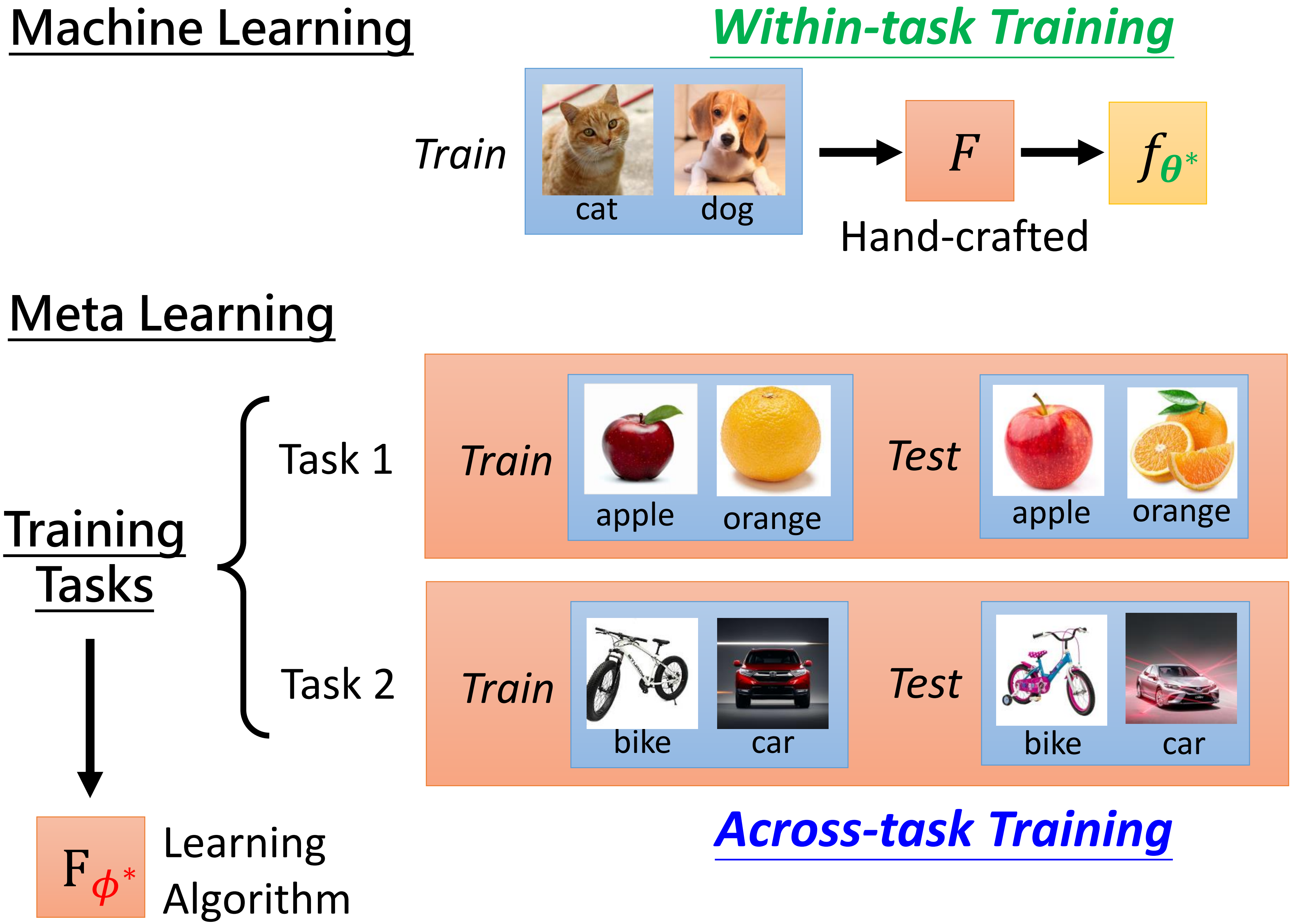
Training tasks

Task 1
Apple & Orange

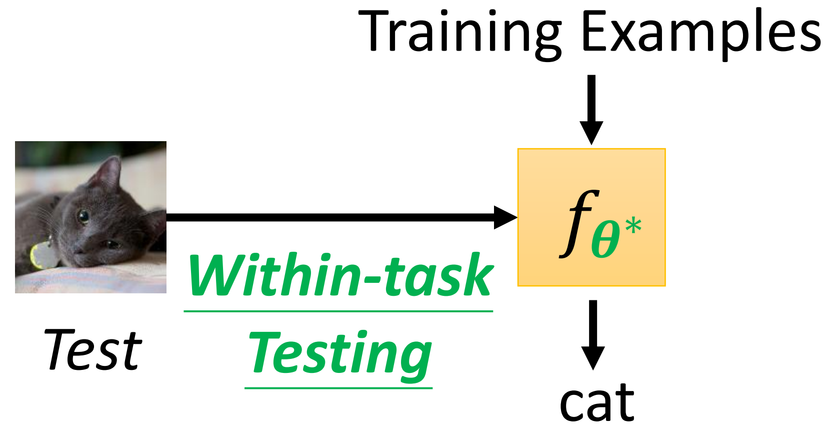
Task 2
Car & Bike



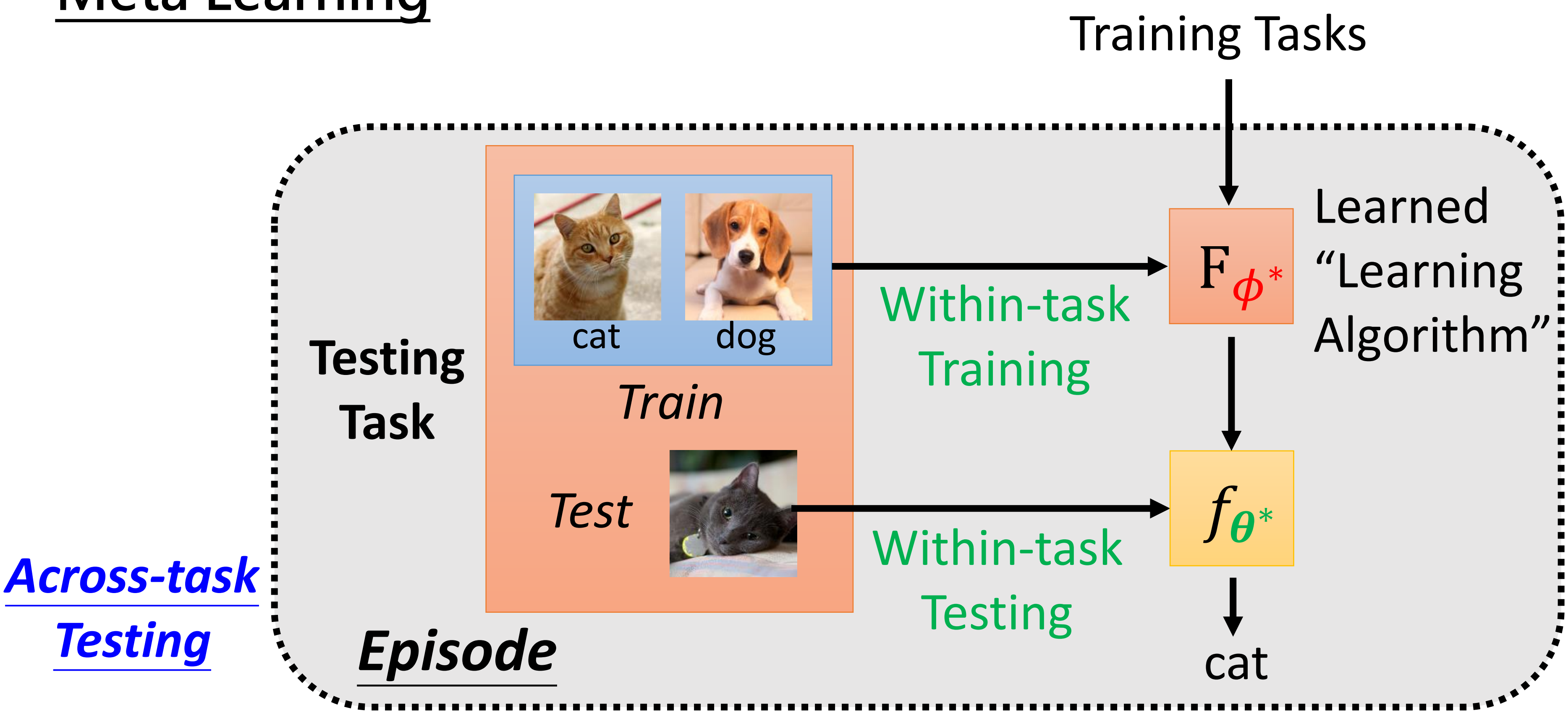
(in the literature of "*learning to compare*")


Machine Learning
Within-task Training
Train
cat
dog
F
f_{θ^*}
Hand-crafted
Meta Learning
Training Tasks
Task 1
Train
apple
orange
Test
apple
orange
Task 2
Train
bike
car
Test
bike
car
F_{ϕ^*}
Learning Algorithm
Across-task Training

Machine Learning



Meta Learning



Loss

<u>Machine Learning</u>

$$L(\boldsymbol{\theta}) = \boxed{\sum_{k=1}^{K} e_k}$$

Sum over training examples in one task

<u>Meta Learning</u>

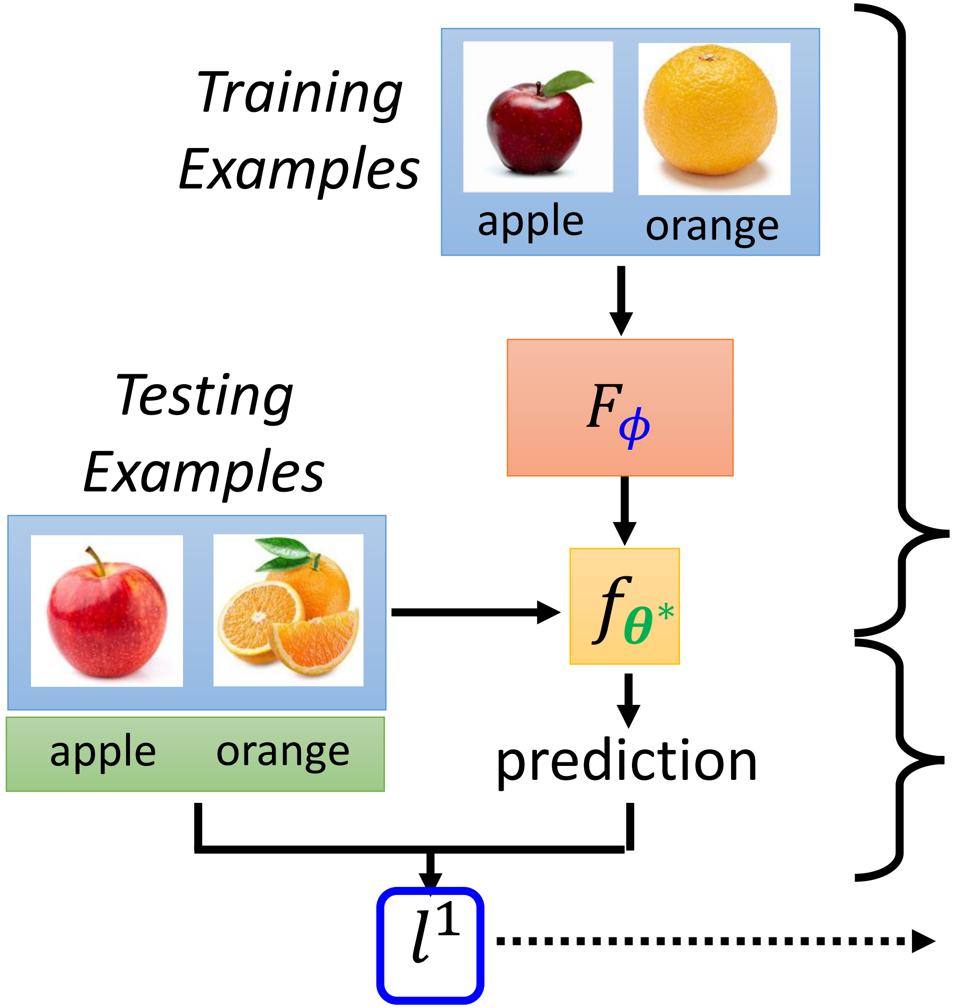
$$L(\phi) = \boxed{\sum_{n=1}^{N} \boxed{l^n}}$$

Sum over testing examples in one task

Sum over training tasks

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

If your optimization method needs to compute $L(\phi)$



Outer Loop in *"Learning to initialize"*

Across-task training includes **within-task training and testing**

Inner Loop in *"Learning to initialize"*

Within-task Training

Within-task Testing

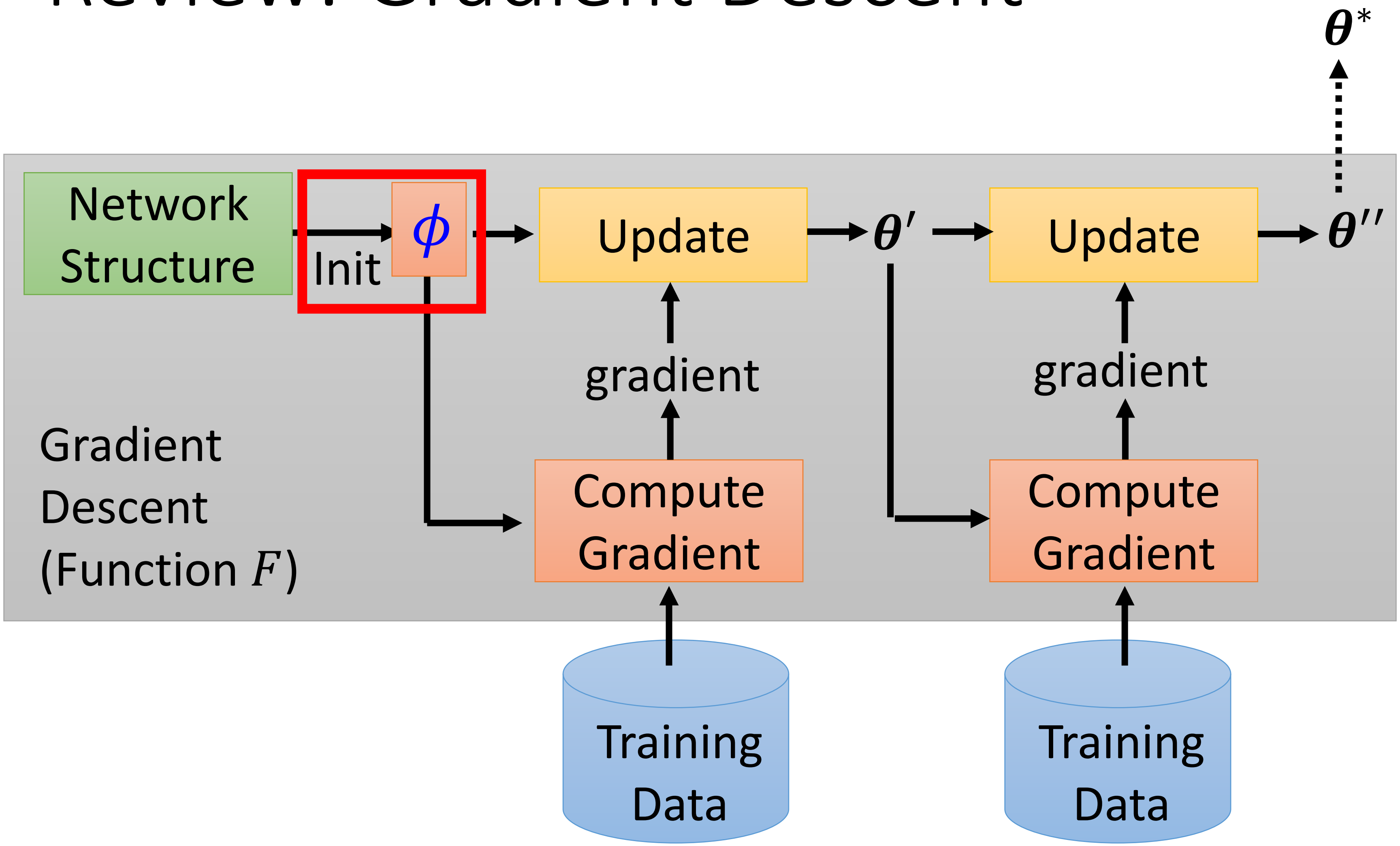
To compute the loss

Meta Learning v.s ML

- What you know about ML can usually apply to meta learning
  - Overfitting on training tasks
  - Get more training tasks to improve performance
  - Task augmentation
  - There are also hyperparameters when learning a learning algorithm ……
  - Development task ☺

What is learnable in a learning algorithm?

Review: Gradient Descent



Learning to initialize

- Model-Agnostic Meta-Learning (MAML)

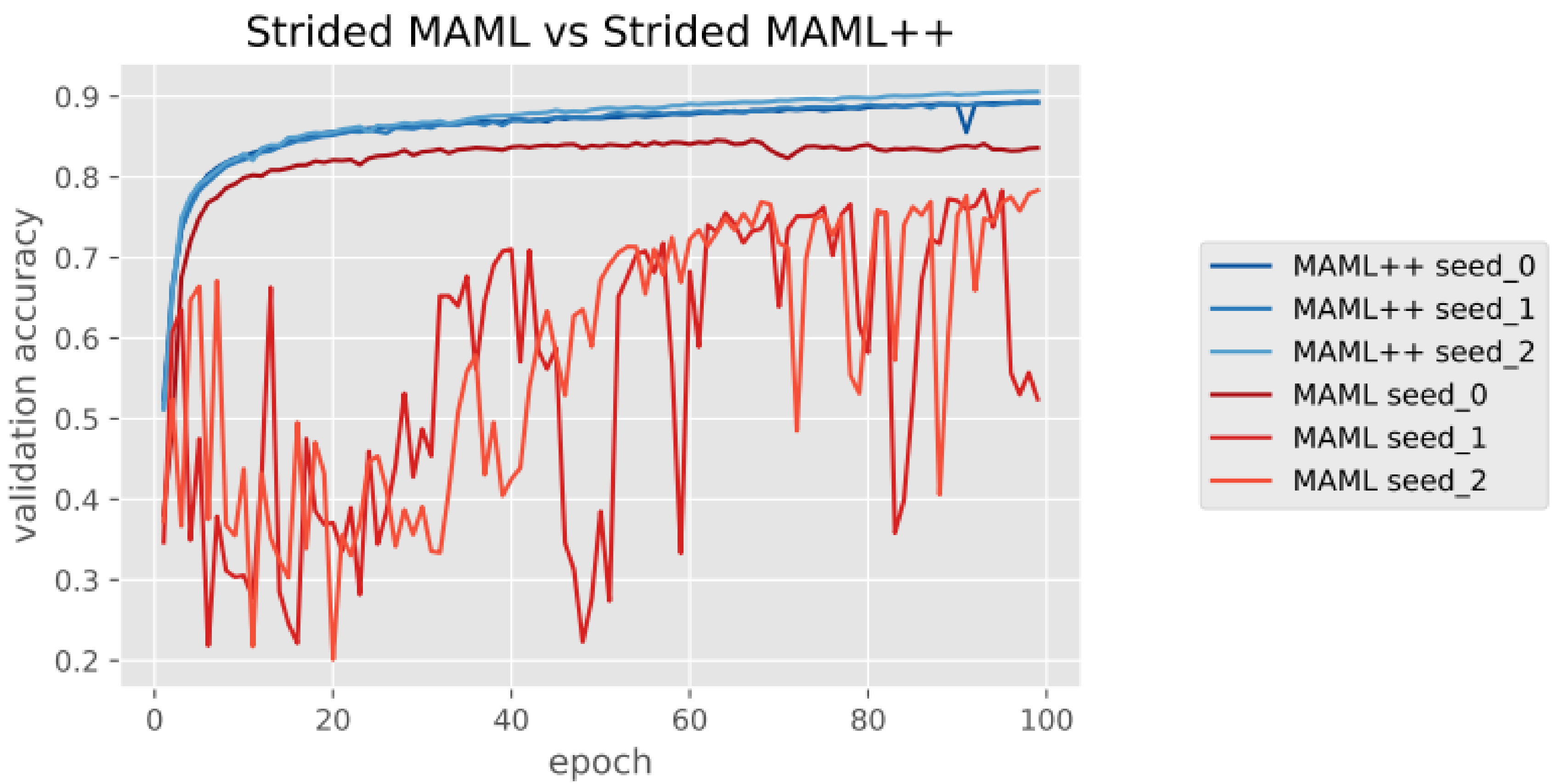
Mammals

Chelsea Finn, Pieter Abbeel, and Sergey Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”, ICML, 2017

- Reptile

https://arxiv.org/abs/1803.02999

How to train your ~~Dragon~~ MAML



Antreas Antoniou, Harrison Edwards, Amos Storkey, How to train your MAML, ICLR, 2019

MAML

Task 1

cat dog

Task 2

cat dog

find good init

Testing Task

cat dog

Pre-training (Self-supervised Learning)

Trained by proxy tasks
(fill-in the blanks, etc.)

find good init

cat dog

MAML

Isn't it domain adaptation / transfer learning?

Task 1 Task 2




find good init


cat dog cat dog cat dog

Pre-training (more typical ways)




find good init


cat dog cat dog cat dog

Use data from different tasks to train a model

Also known as multi-task learning (baseline of meta)

MAML v.s. Pre-training

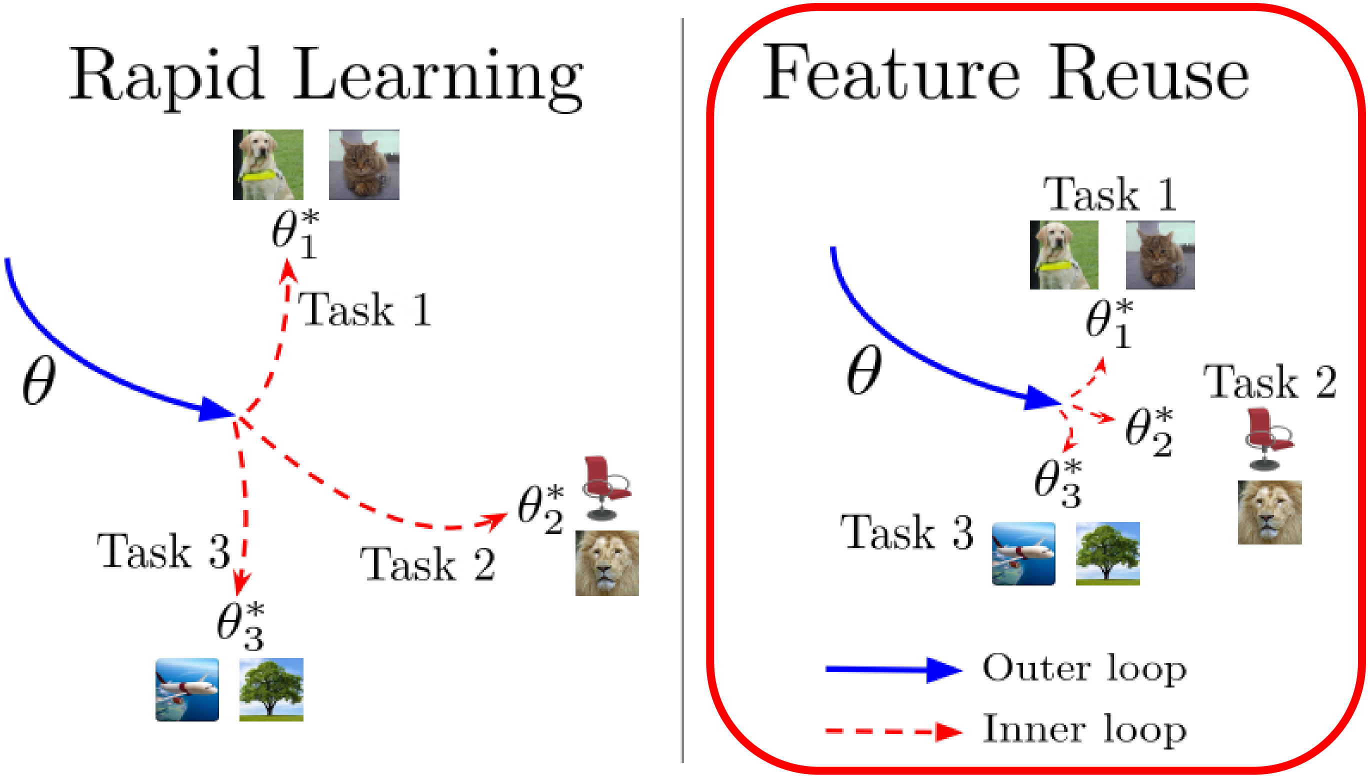
- https://youtu.be/vUwOA3SNb_E

影片中有防不勝防
的業配

這就是 “meta 業配”

MAML is good because ……

- ANIL (Almost No Inner Loop)



Aniruddh Raghu, Maithra Raghu, Samy Bengio, Oriol Vinyals, Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML, ICLR, 2020

More about MAML

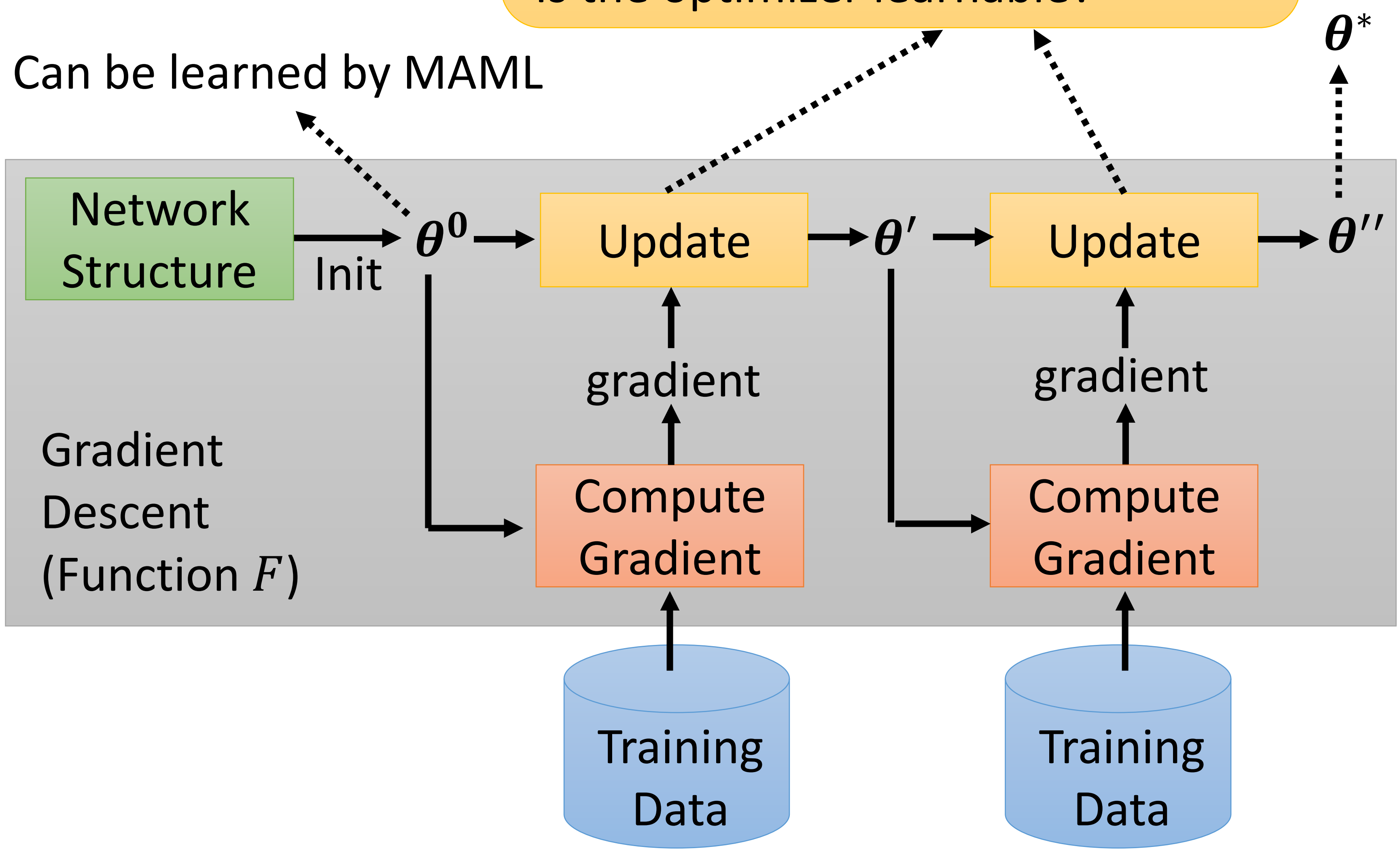
- More mathematical details behind MAML
  - https://youtu.be/mxqzGwP_Qys
- First order MAML (FOMAML)
  - https://youtu.be/3z997JhL9Oo
- Reptile
  - https://youtu.be/9jJe2AD35P8

Optimizer

Basis form: $\boldsymbol{\theta}^{t+1} \leftarrow \boldsymbol{\theta}^{t} - \lambda \boldsymbol{g}^{t}$ ϕ

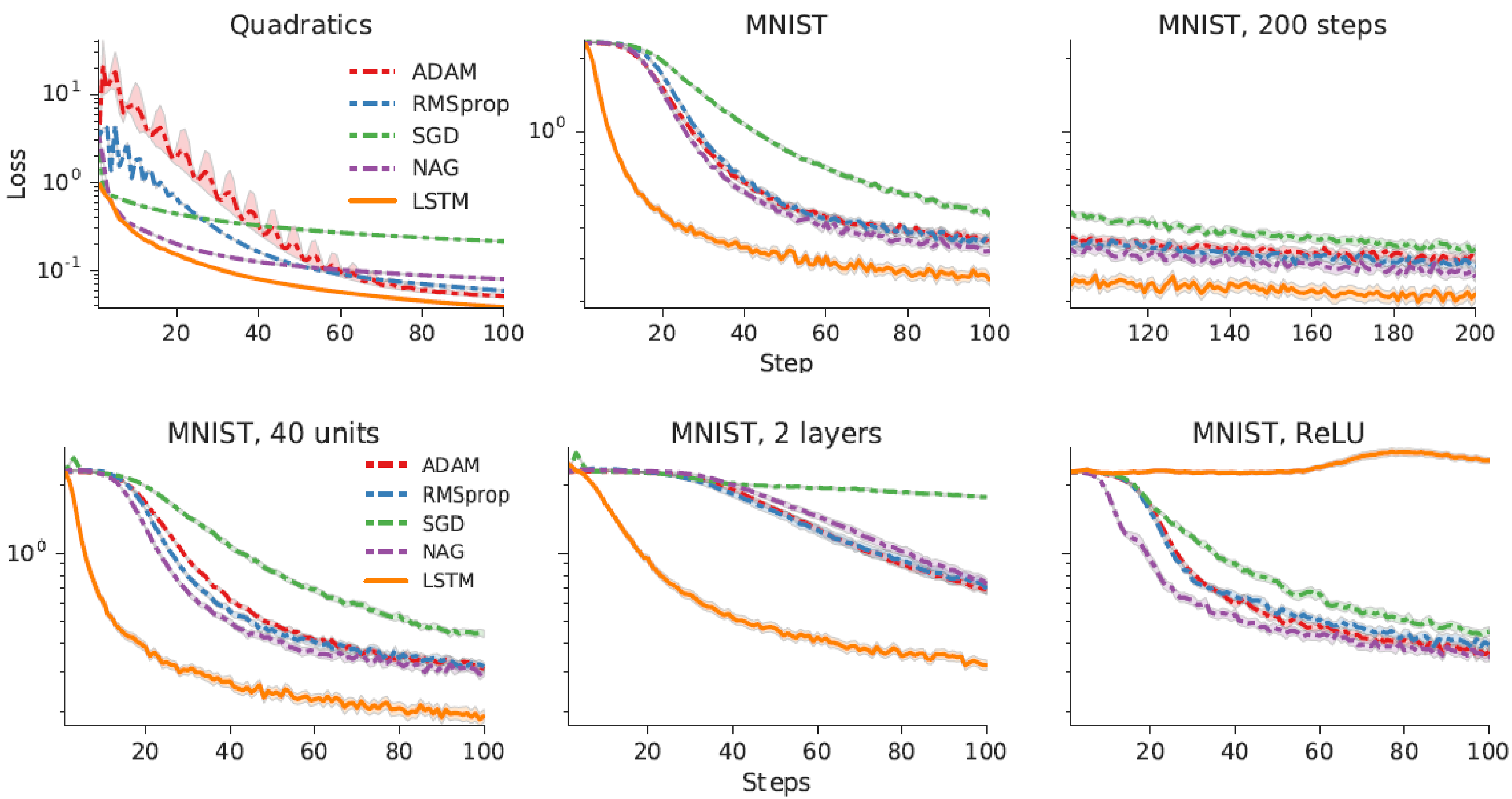
Adagrad, RMSprop, NAG, Adam ……

Is the optimizer learnable?

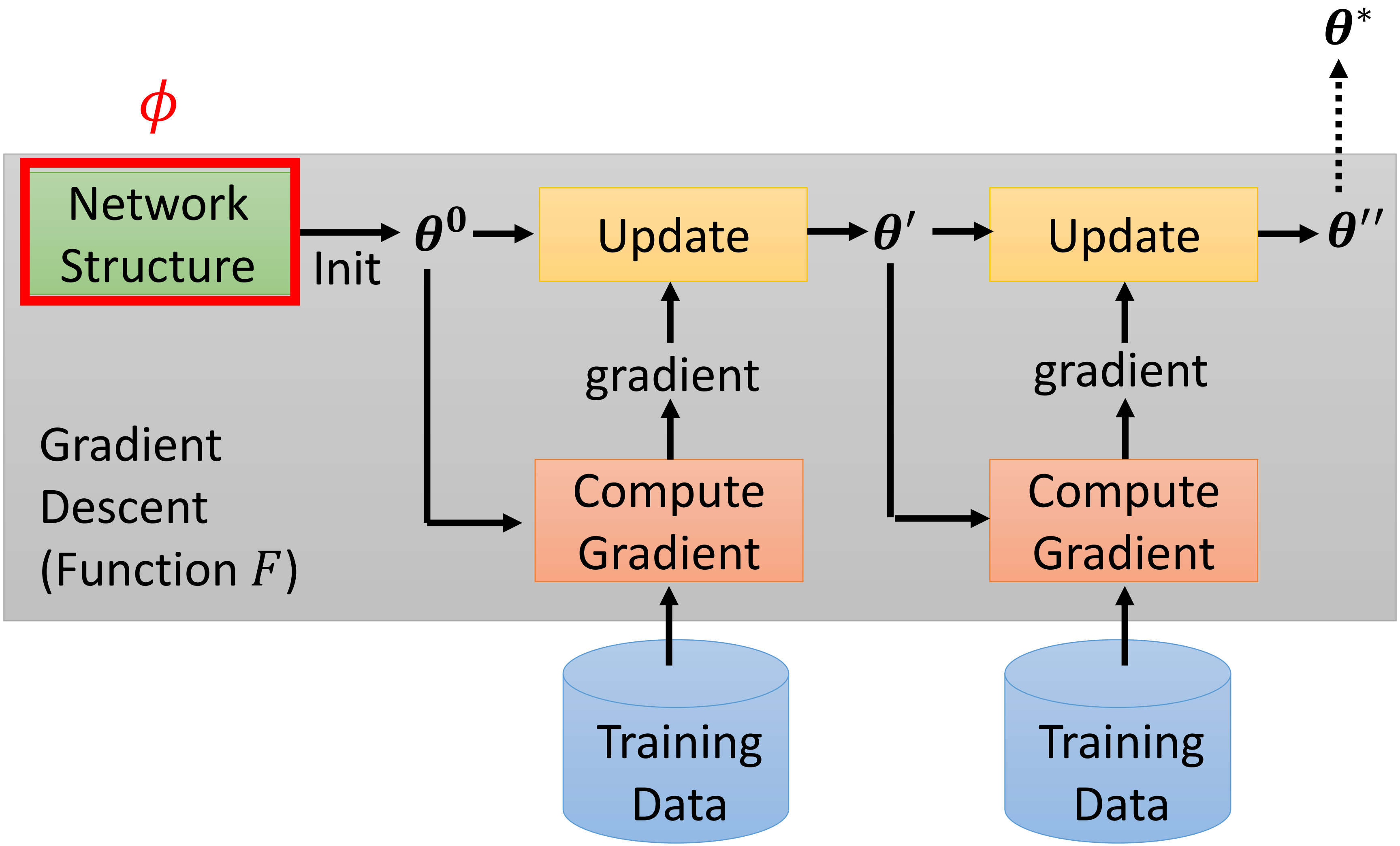


Optimizer

Marcin Andrychowicz, et al., Learning to learn by gradient descent by gradient descent, NIPS, 2016



Network Architecture Search (NAS)



Network Architecture Search (NAS)

$$\hat{\phi} = arg\min_{\phi} L(\phi) \qquad \nabla_{\phi} L(\phi) = ?$$

Network Architecture

- Reinforcement Learning
  - Barret Zoph, et al., Neural Architecture Search with Reinforcement Learning, ICLR 2017
  - Barret Zoph, et al., Learning Transferable Architectures for Scalable Image Recognition, CVPR, 2018
  - Hieu Pham, et al., Efficient Neural Architecture Search via Parameter Sharing, ICML, 2018

An agent uses a set of actions to determine the network architecture.

ϕ: the agent's parameters

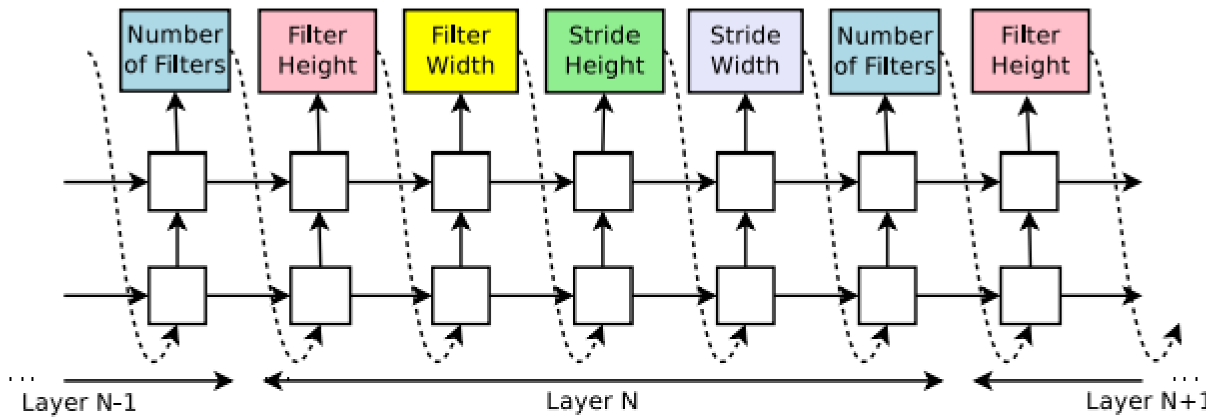
$-L(\phi)$

Reward to be maximized

Network Architecture Search (NAS)

Across-task Training

Update ϕ to maximize reward $-L(\phi)$

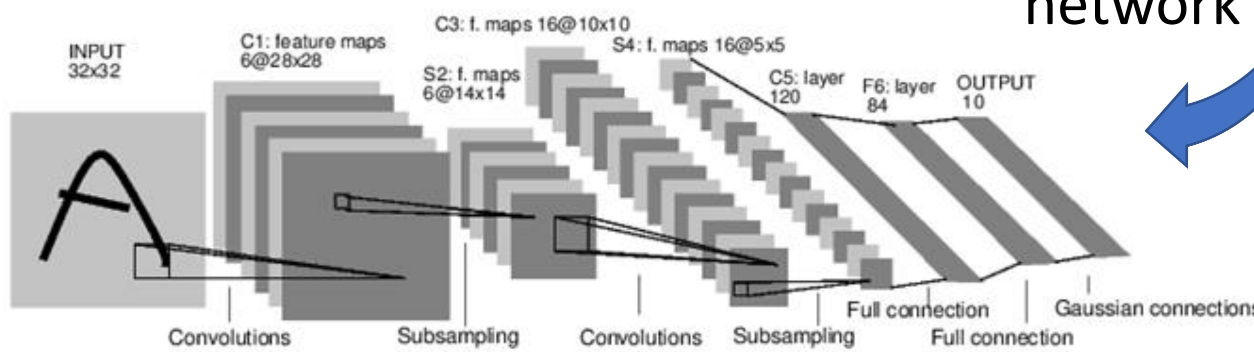


agent ϕ (RNN)

form a network

$-L(\phi)$

Accuracy of the network



A Full Convolutional Neural Network (LeNet)

Train the network

Within-task Training

Network Architecture Search (NAS)

$$\hat{\phi} = \arg\min_{\phi} L(\phi) \qquad \nabla_{\phi} L(\phi) = ?$$

Network Architecture

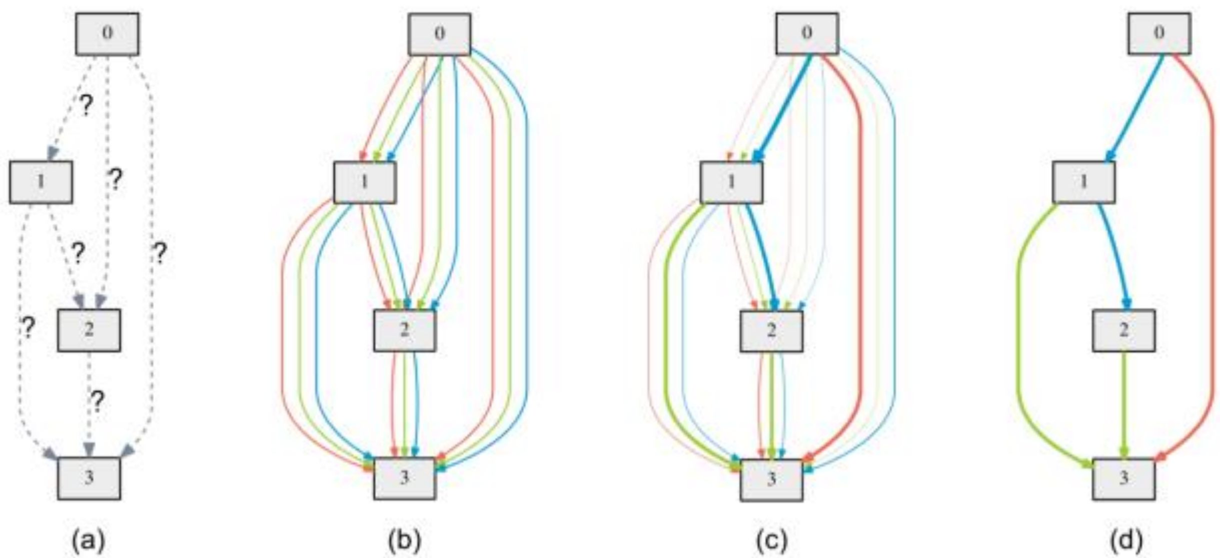
- Reinforcement Learning
  - Barret Zoph, et al., Neural Architecture Search with Reinforcement Learning, ICLR 2017
  - Barret Zoph, et al., Learning Transferable Architectures for Scalable Image Recognition, CVPR, 2018
  - Hieu Pham, et al., Efficient Neural Architecture Search via Parameter Sharing, ICML, 2018
- Evolution Algorithm
  - Esteban Real, et al., Large-Scale Evolution of Image Classifiers, ICML 2017
  - Esteban Real, et al., Regularized Evolution for Image Classifier Architecture Search, AAAI, 2019
  - Hanxiao Liu, et al., Hierarchical Representations for Efficient Architecture Search, ICLR, 2018

Network Architecture Search (NAS)

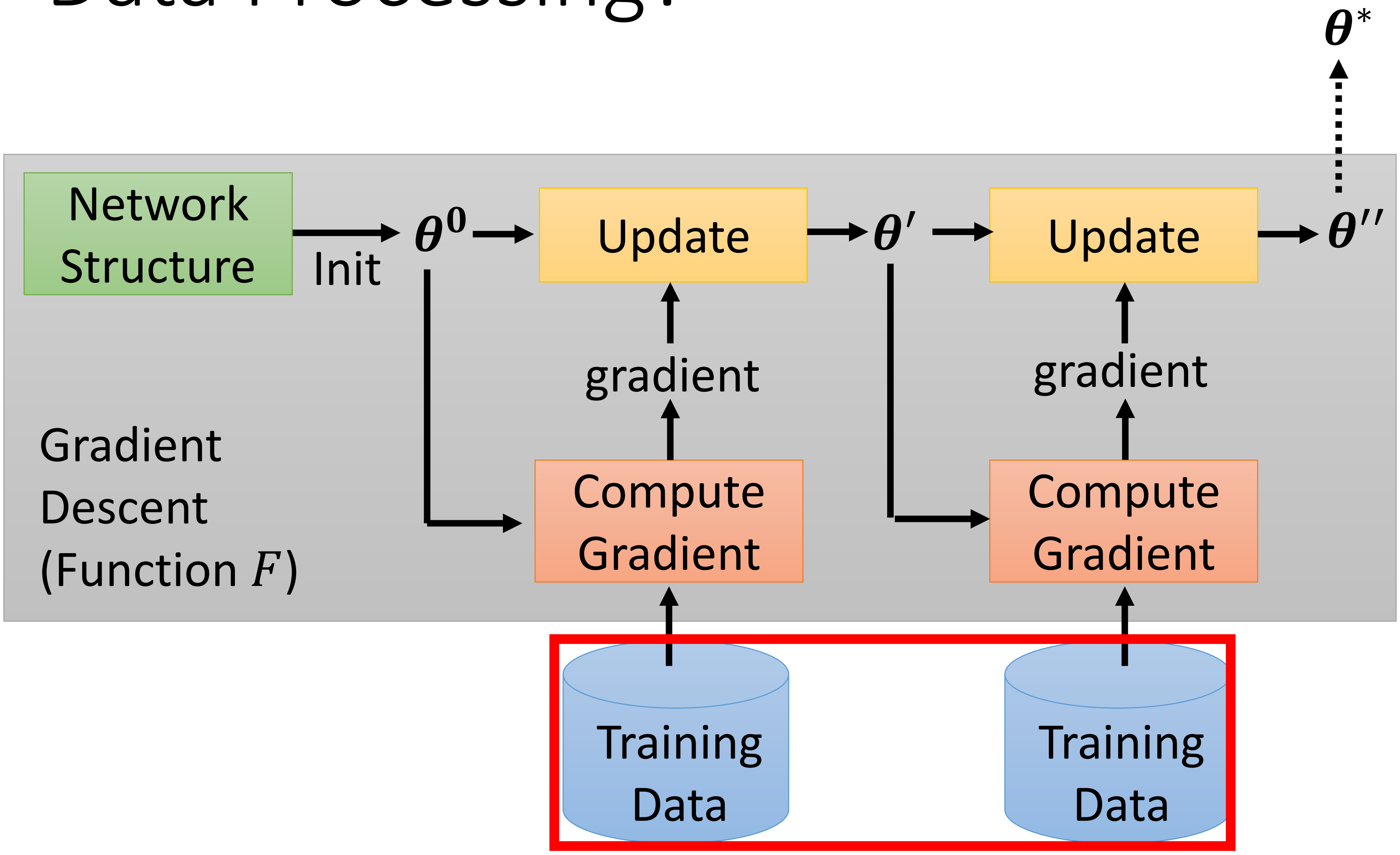
$$\hat{\phi} = \arg\min_{\phi} L(\phi) \qquad \nabla_{\phi} L(\phi) = ?$$

Network Architecture

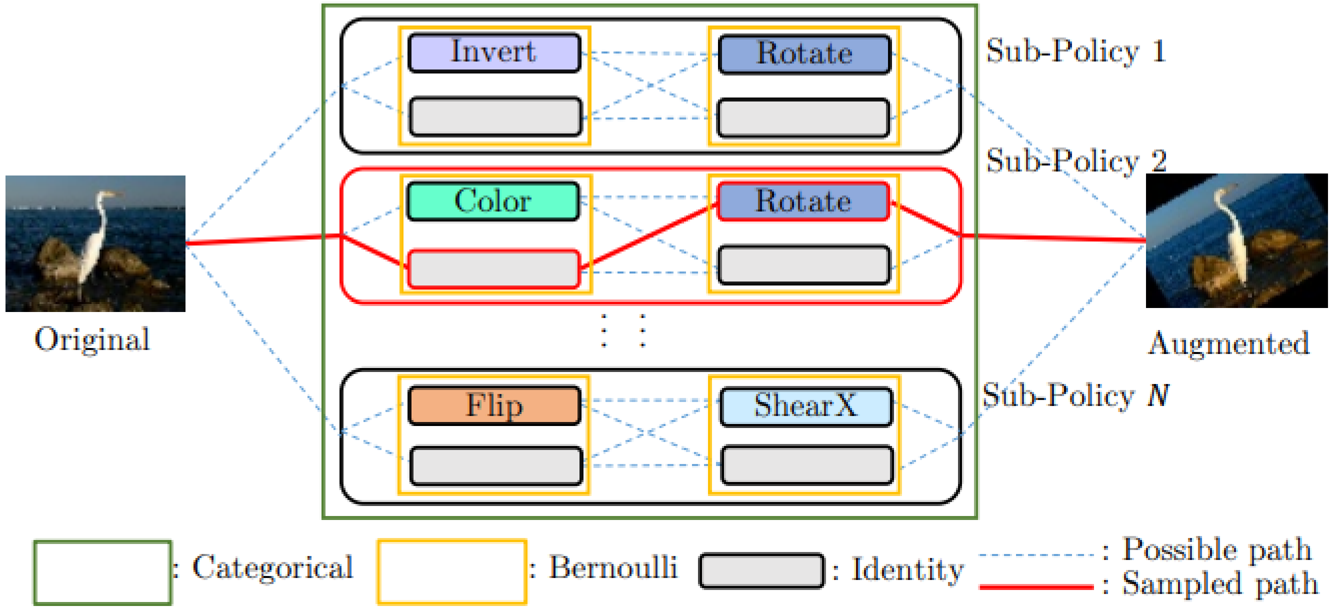
- DARTS Hanxiao Liu, et al., DARTS: Differentiable Architecture Search, ICLR, 2019



Data Processing?



Data Augmentation



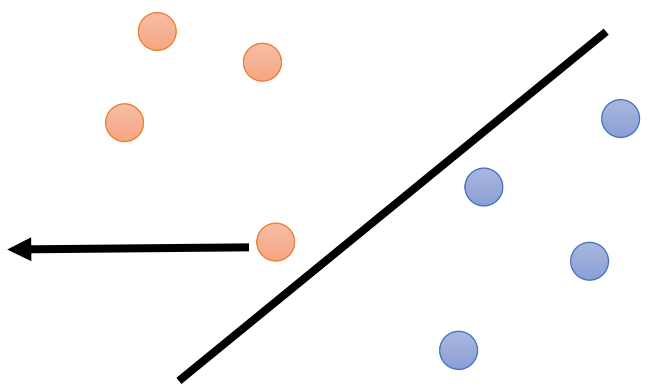
Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M. Robertson, Yongxin Yang, DADA: Differentiable Automatic Data Augmentation, ECCV, 2020
Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, Xi Chen, Population Based Augmentation: Efficient Learning of Augmentation Policy Schedules, ICML, 2019
Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le, AutoAugment: Learning Augmentation Policies from Data, CVPR, 2019

Sample Reweighting

- Give different samples different weights

Larger weights (focus on tough examples)?

Smaller weights (the labels are noisy)?

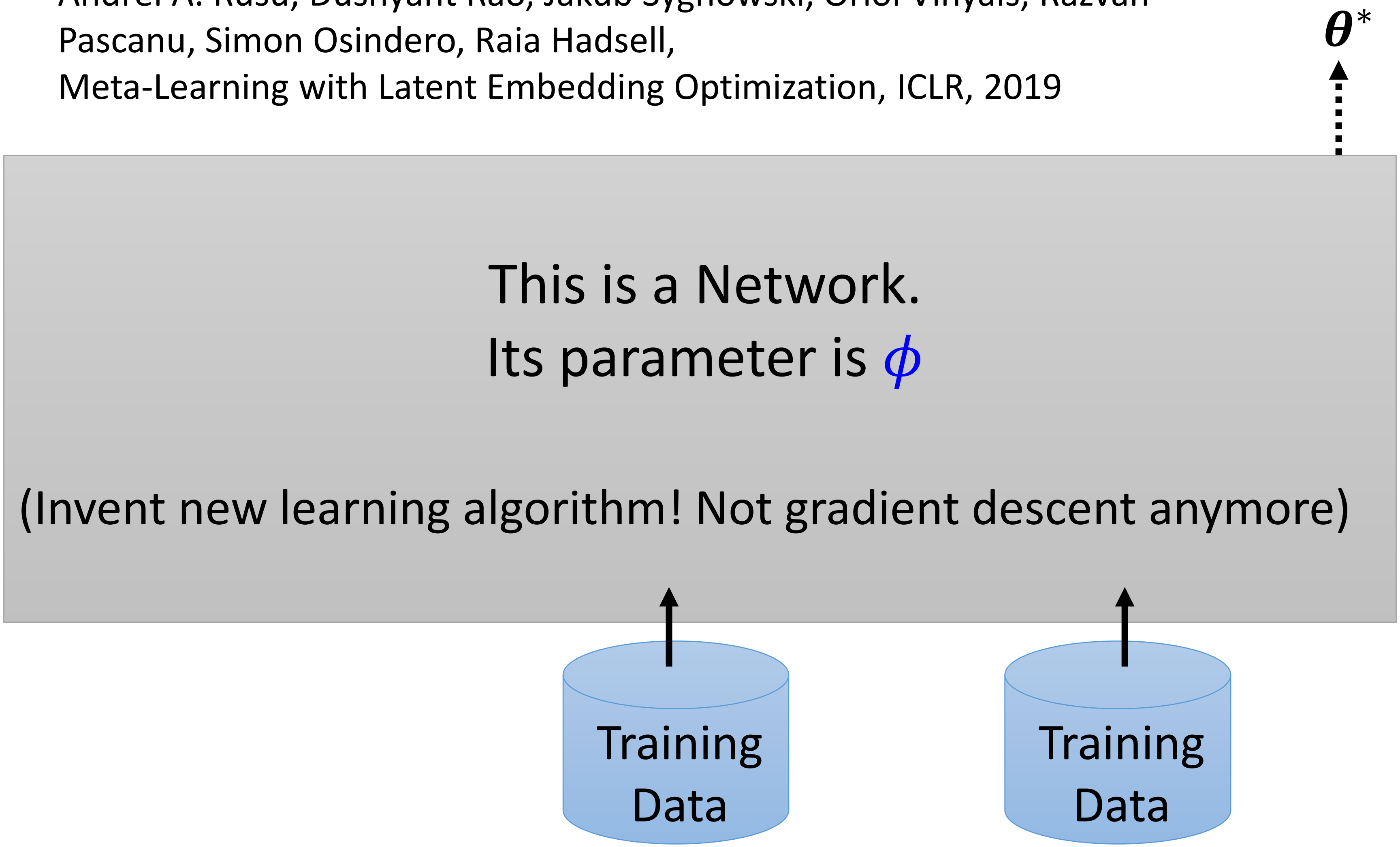


Sample Weighting Strategies ➡ Learnable ϕ

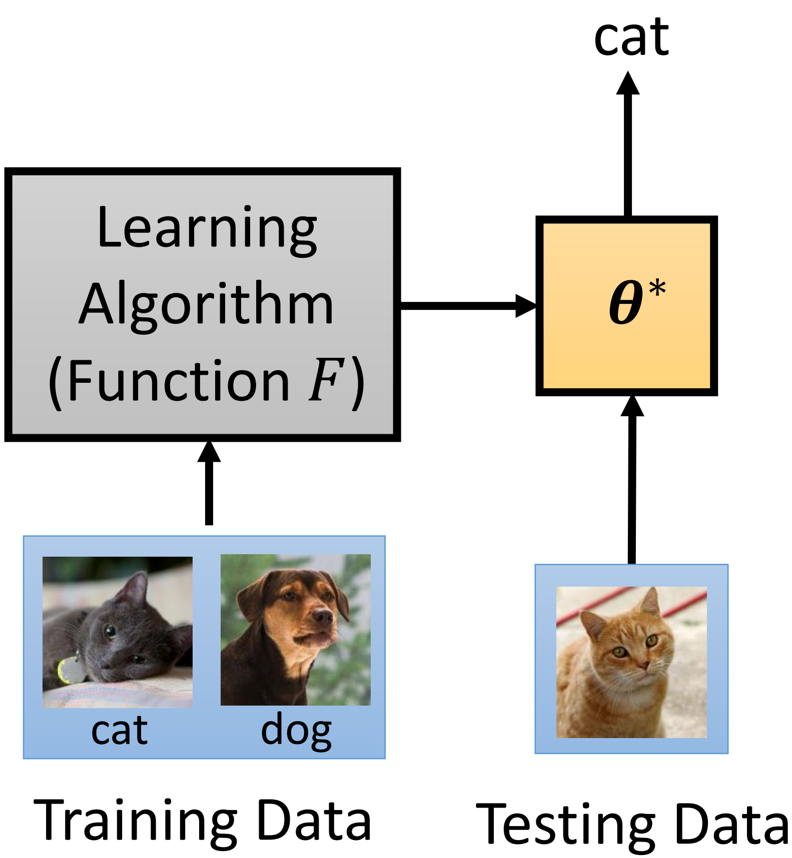
Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, Deyu Meng, Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting, NeurIPS, 2019
Mengye Ren, Wenyuan Zeng, Bin Yang, Raquel Urtasun, Learning to Reweight Examples for Robust Deep Learning, ICML, 2018

Beyond Gradient Descent

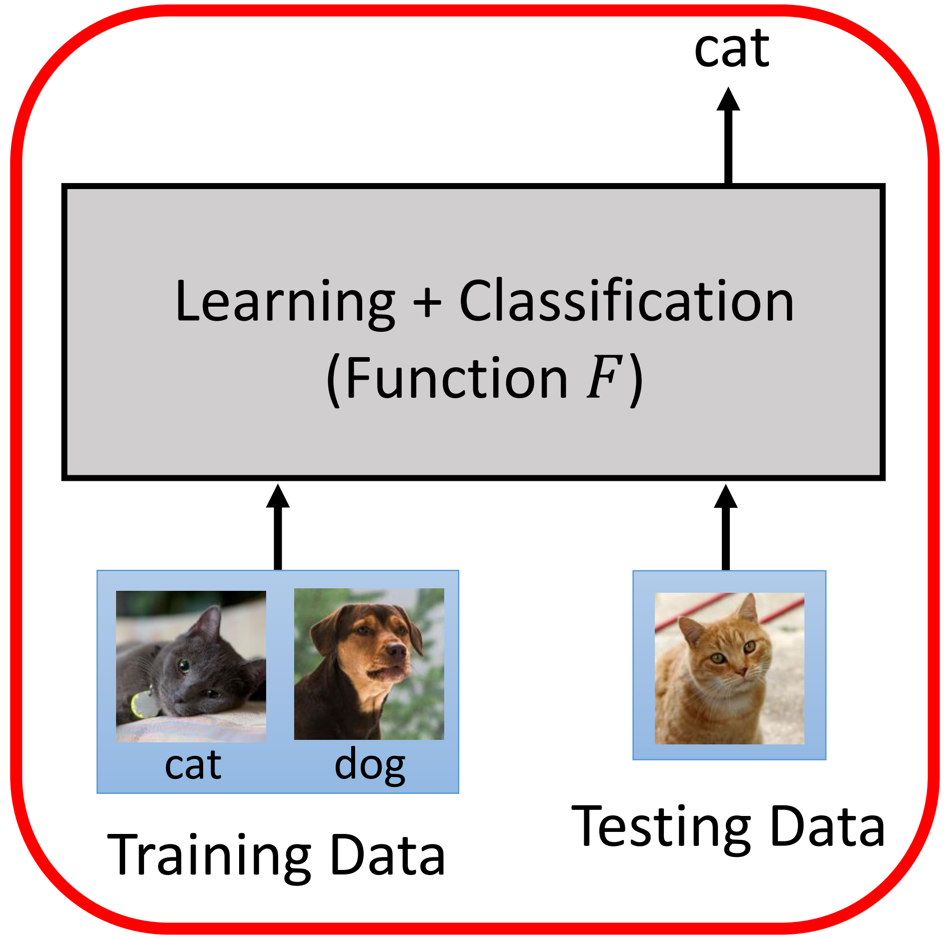
Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, Raia Hadsell,
Meta-Learning with Latent Embedding Optimization, ICLR, 2019



Until now ……



How about?



<u>Learning to compare</u>

(metric-based approach)

https://youtu.be/yyKaACh_j3M

https://youtu.be/scK2EIT7klw

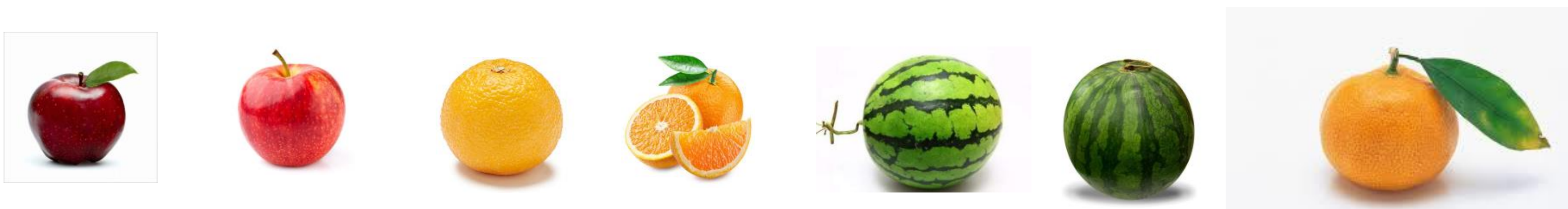
https://youtu.be/semSxPP2Yzg

https://youtu.be/ePimv_k-H24

Applications

Few-shot Image Classification

- Each class only has a few images.



Class 1 Class 1 Class 2 Class 2 Class 3 Class 3 Which class?

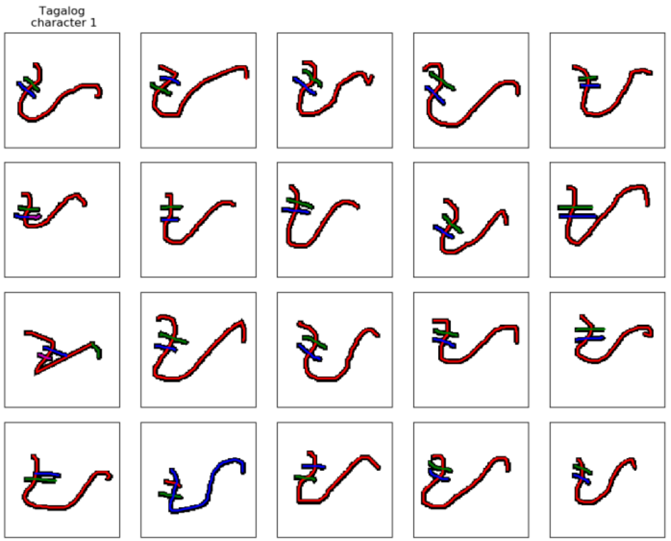
3-ways 2-shot

- N-ways K-shot classification: In each task, there are N classes, each has K examples.
- In meta learning, you need to prepare many N-ways K-shot tasks as training and testing tasks.

Omniglot

https://github.com/brendenlake/omniglot

- 1623 characters
- Each has 20 examples



Omniglot

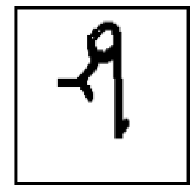
Demo:
https://openai.com/blog/reptile/

20 ways

1 shot

Each character represents a class

Training set (Support set)



Testing set (Query set)

- Split your characters into training and testing characters
  - Sample N training characters, sample K examples from each sampled characters $\rightarrow$ one training task
  - Sample N testing characters, sample K examples from each sampled characters $\rightarrow$ one testing task

	(A) **Learning to initialize**	(B) **Learning to compare**	(C) **Other**
Sound Event Detection	(Shi et al., 2020)	(Shimada et al., 2020a) (Chou et al., 2019) (Wang et al., 2020) (Shimada et al., 2020b) (Shi et al., 2020)	Network architecture search: (Li et al., 2020)
Keyword Spotting	(Chen et al., 2020a)	(Huh et al., 2020)	Net2Net: (Veniat et al., 2019) Network architecture search: (Mazzawi et al., 2019) Network architecture search: (Mo et al., 2020)
Text Classification	(Dou et al., 2019) (Bansal et al., 2019)	(Yu et al., 2018) (Tan et al., 2019) (Geng et al., 2019) (Sun et al., 2019)	Learning the learning algorithm: (Wu et al., 2019)
Voice Cloning			Learning the learning algorithm: (Chen et al., 2019b) (Serrà et al., 2019)
Sequence Labelng	(Wu et al., 2020)	(Hou et al., 2020)	
Machine Translation	(Gu et al., 2018) (Indurthi et al., 2020)		
Speech Recognition	(Hsu et al., 2020) (Klejch et al., 2019) (Winata et al., 2020a) (Winata et al., 2020b)		Learning to optimize: (Klejch et al., 2018) Network architecture search: (Chen et al., 2020b) (Baruwa et al., 2019)
Knowledge Graph	(Obamuyide and Vlachos, 2019) (Bose et al., 2019) (Lv et al., 2019) (Wang et al., 2019)	(Ye and Ling, 2019) (Chen et al., 2019a) (Xiong et al., 2018) (Gao et al., 2019)	
Dialogue / Chatbot	(Qian and Yu, 2019) (Madotto et al., 2019) (Mi et al., 2019)		Learning to optimize: (Chien and Lieow, 2019)
Parsing	(Guo et al., 2019) (Huang et al., 2018)		
Word Embedding	(Hu et al., 2019)	(Sun et al., 2018)	
Multi-model		(Eloff et al., 2019)	Learning the learning algorithm: (Surís et al., 2019)

http://speech.ee.ntu.edu.tw/~tlkagk/meta_learning_table.pdf