DEEP LEARNING

Lecture 13: Meta-Learning

Dr. Yang Lu

Department of Computer Science and Technology

luyang@xmu.edu.cn

厦門大學信息学院（特色化示范性软件学院）
School of Informatics Xiamen University (National Characteristic Demonstration Software School)

厦门大学 计算机科学与技术系
Department of Computer Science and Technology, Xiamen University

What is Meta?

- When meta is used as a single word, it refers to itself or to the conventions of its genre, self-referential.
  - The chinese translation is 元.

Image source: https://technews.tw/2021/10/29/a-social-technology-company-meta/

What is Meta?

- When meta is used as a prefix, meta-X means “beyond-X,” “after-X,” or “X about X”.
- Examples:
  - Metadata: data that describes other data.
  - Metafile: in computer graphics, define objects and images using a list of coordinates.
  - Metaphysics (形而上学): a branch of philosophy that examines the fundamental nature of reality.
  - Meta-analysis: a statistical analysis that combines the results of multiple scientific studies.
  - Metaverse: a virtual world supporting persistent online 3-D virtual environments.

Meta-Learning

- Meta-learning （元学习） also means "beyond learning", "above learning" or "learning about learning".
- It has another name:

Learning to learn

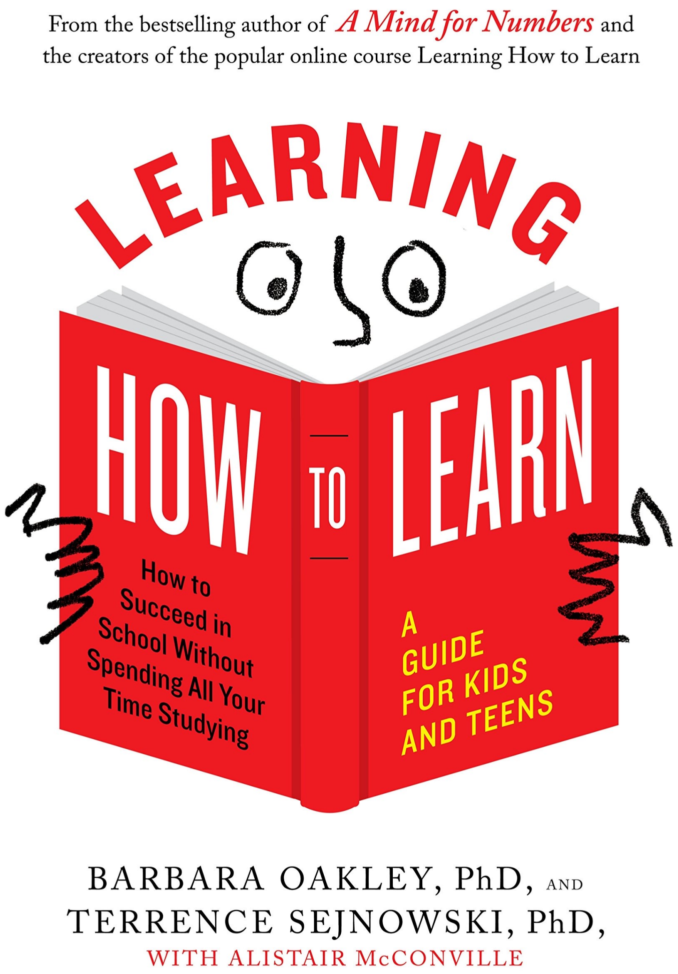


Image source: https://images-na.ssl-images-amazon.com/images/I/81j+JH7WshL.jpg

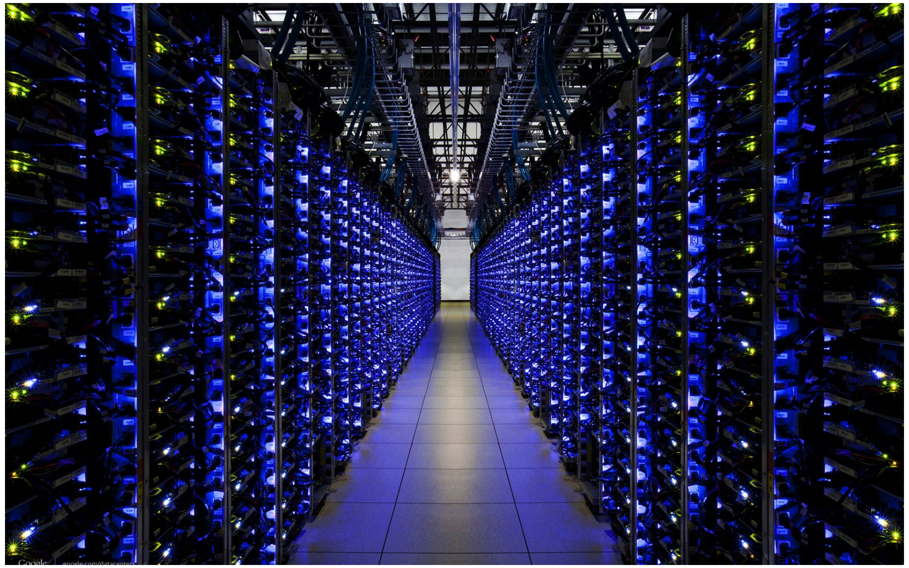
Meta-Learning

- Historically, the success of machine learning was driven by the choice of hand-engineered features with model learning.
  - Problem: how to select feature…
- Deep learning realized the promise of joint feature and model learning, providing a huge improvement in performance for many tasks.
  - Problem: how to select algorithm/hyperparameter…
- Meta-learning in neural networks can be seen as aiming to provide the next step of integrating joint feature, model, and algorithm learning.

Algorithm Learning

- Learning aspects:
  - Model learning: select the best model for the task.
  - Feature learning: select the best feature for the task.
  - Algorithm learning: select the best algorithm for the task.
- How do we select an algorithm to train a model?
  - Manually try different algorithms.
  - Manually try an algorithm with different hyperparameters.

Algorithm Selection vs. Algorithm Learning

Industry	Academia
	
Using 1000 GPUs to try 1000 sets of hyperparameters	Randomly choose a set by imagination and claim that it is the best!

Meta-learning automates this procedure by end-to-end neural networks.

Image source: https://www.extremetech.com/wp-content/uploads/2015/01/Servers.jpg
https://media.bastillepost.com/wp-content/uploads/hongkong/2016/05/20160526_HIST_%E7%85%89%E4%B8%B9_%E7%A7%A6%E5%A7%8B%E7%9A%87F.jpg

Relation to AutoML

- AutoML aims to automate parts of the machin learning process that are typically manual.
  - Such as data preparation, algorithm selection, hype parameter tuning, and architecture search.
- AutoML sometimes makes use of end-to-en optimization.
  - Meta-learning can be seen as a specialization of AutoML.

Google Cloud AutoML

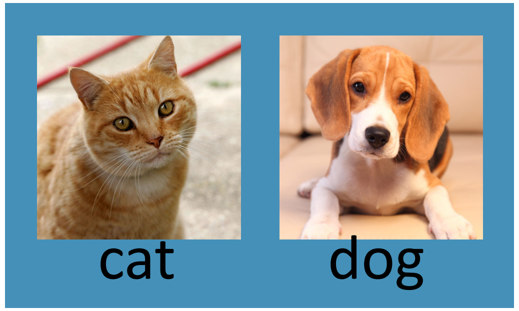
- Meta-learning is about algorithm learning, rather than algorithm tuning.

Image source: https://www.datanami.com/2018/01/17/google-automate-machine-learning-automl-service/

From Machine Learning to Meta-Learning

- In machine learning, we select an algorithm F, train it by optimizing the parameter θ, and obtain model f_θ.

Training data

$F($ 

$) \rightarrow f_\theta$

Test data

$f_\theta($  $) \rightarrow \text{cat}$

By the idea of learning to learn, can we learn F?

Image source: Meta Learning: Learn to Learn, Hung-Yi Lee

From Machine Learning to Meta-Learning

- Now, our goal is to find the best algorithm F for the task, just like the best model f for the data.

$A($  cat  dog $) \rightarrow F_{\omega}$

Training data

$F_{\omega}($  cat  dog $) \rightarrow f_{\theta}$

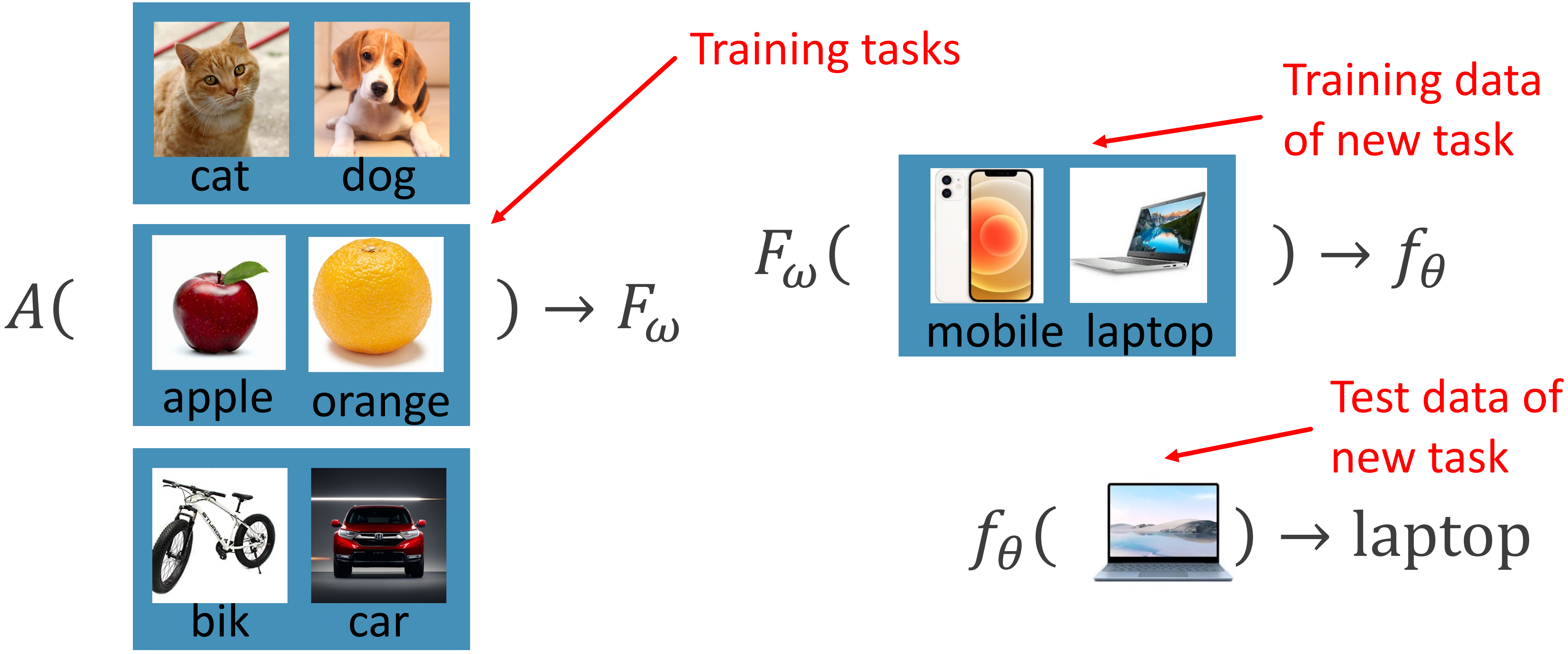
Test data

$f_{\theta}($  $) \rightarrow \text{cat}$

ω is learnable algorithm parameters. It is usually called meta-knowledge.

From Machine Learning to Meta-Learning

- When we deal with multiple tasks, we can also train F_ω to be good for all tasks. In this way, it is capable of generalizing new task.



Single-Task and Multi-Task Meta-Learning

- Single-task and multi-task meta-learning actually deals with different problems.
  - Single-task meta-learning aims at learning the most suitable algorithm for this task.
  - Multi-task meta-learning aims at learning the most suitable algorithm for all tasks, and be capable of dealing with new task.

Notice the difference between multi-task meta-learning and multi-task learning.

How to Learn Model Parameter?

- Now, the problem is **how to learn algorithm parameter ω**?
- Recall that how we learn model parameter θ.

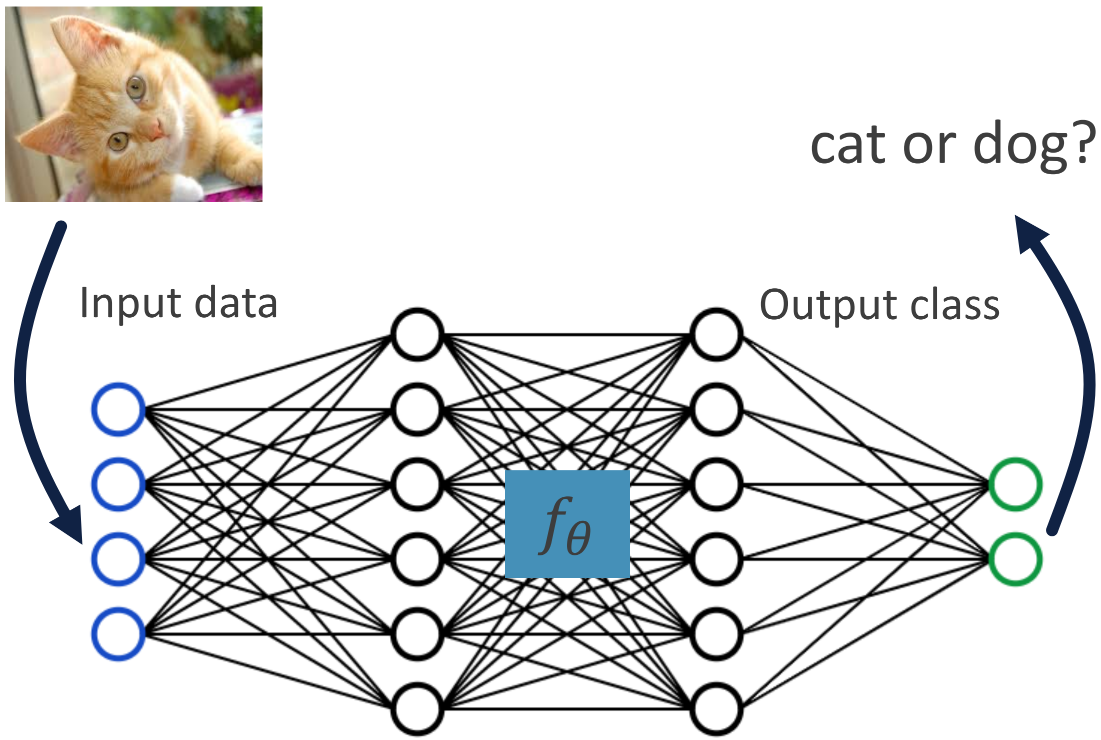
Step 1: Model with unknown parameters

↓

Step 2: Define loss function

↓

Step 3: Optimization

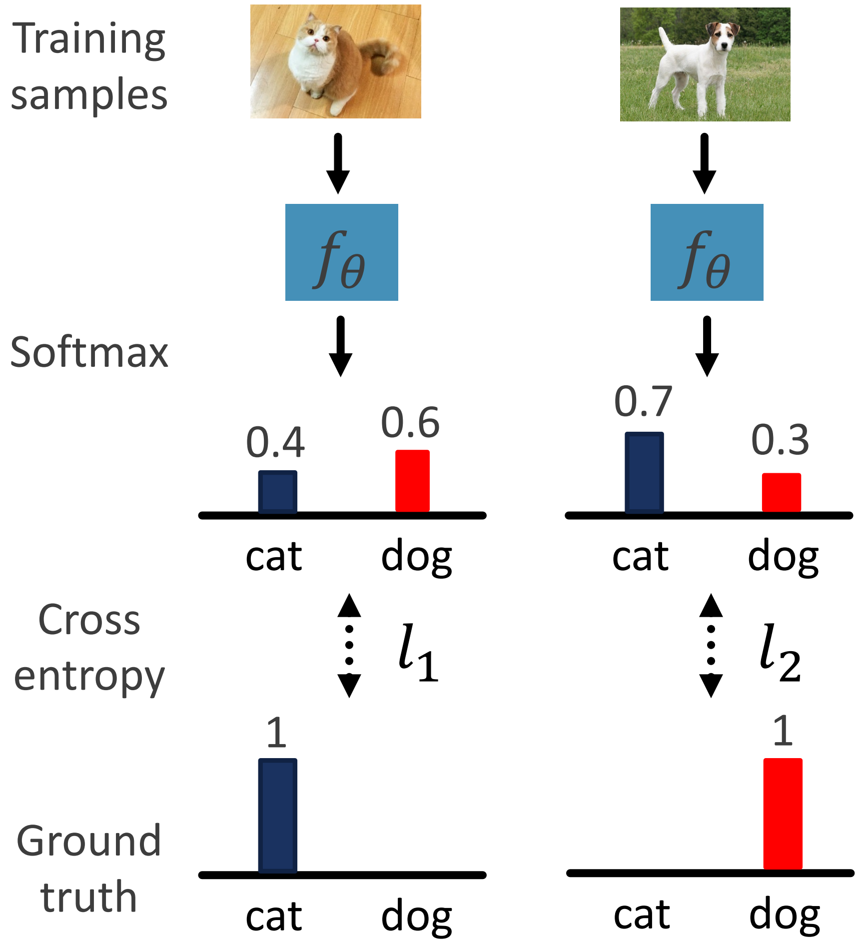


How to Learn Model Parameter?

Step 1: Model with unknown parameters

Step 2: Define loss function

Step 3: Optimization



How to Learn Model Parameter?

Step 1: Model with unknown parameters

Step 2: Define loss function

Step 3: Optimization

loss: $L(\theta) = \sum_{i=1}^{|D|} l_i$ sum over training examples

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

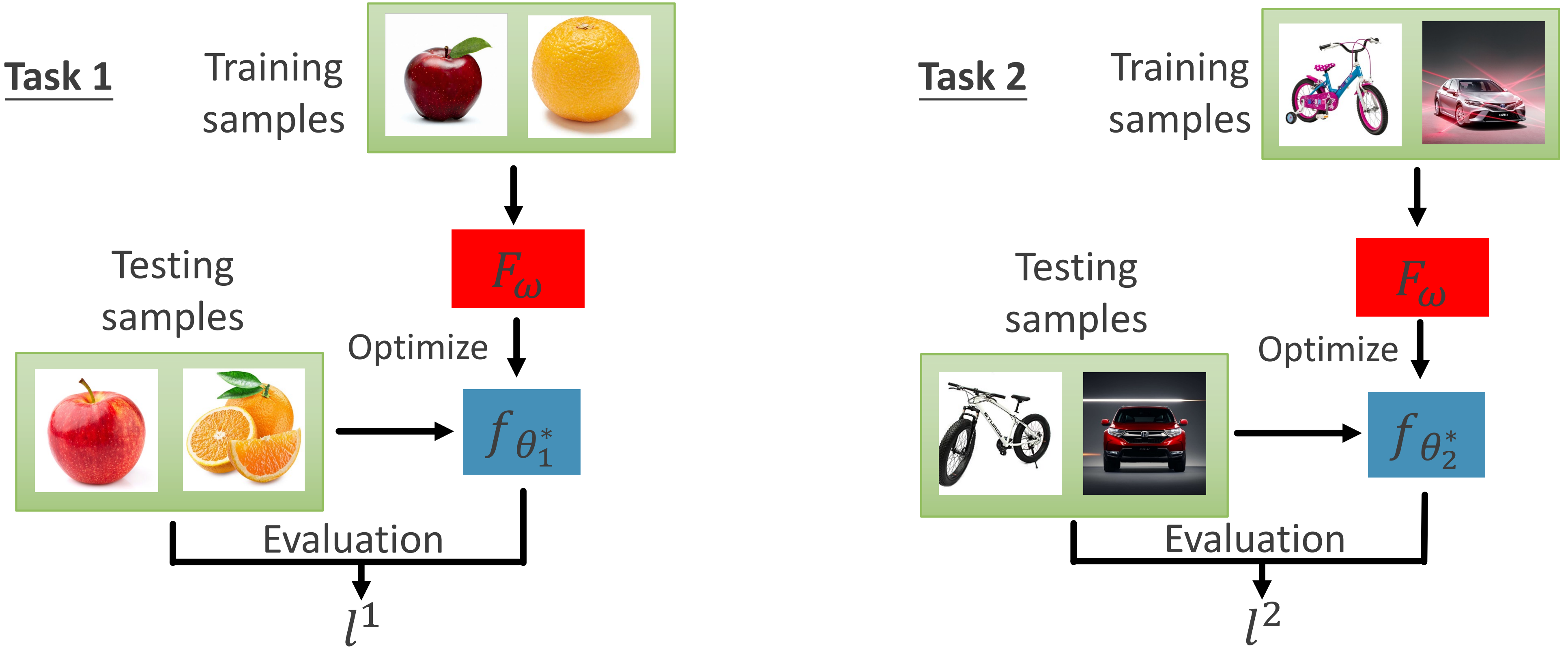
done by gradient descent

f_{θ^*} is the model learned by a learning algorithm from data.

How to Learn Algorithm Parameter?

- Learning algorithm parameter is exactly same as learning model parameter.
- Instead of generalizing over data, it generalizes over tasks.

How to Learn Algorithm Parameter?



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

How to Learn Algorithm Parameter?

- In typical machine learning, the loss is computed based on training examples.
- In meta-learning, the loss is computed based on testing examples.
- Is there any problem here?

You dare use testing data during training!

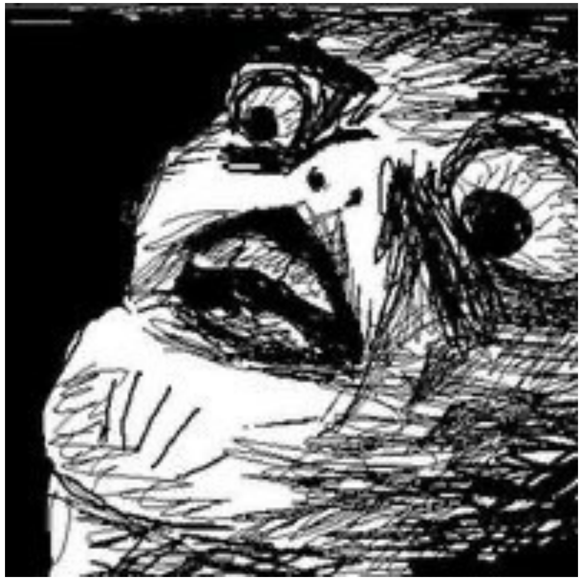
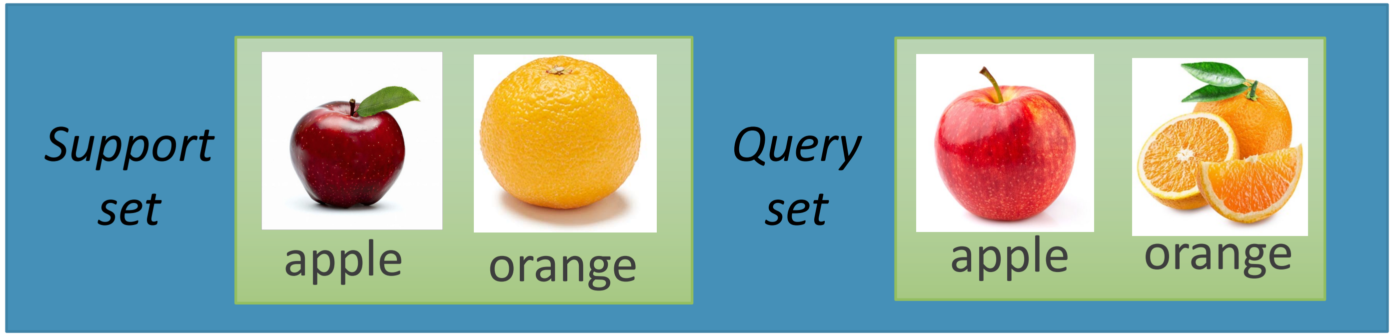


Image source: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcT4B8MvT431E_J0unZHdgWqRxEuJUzwDQGb2nvTWSoYhTdFcLqEIz9chcEI4_4bzur_DD4&usqp=CAU

How to Learn Algorithm Parameter?

- However, we have to use “testing data” to evaluate how the algorithm parameter performs.
- We can cut a part from the training data, just like validation data.

Training data



Used for optimizing model parameter θ

Used for optimizing algorithm parameter ω

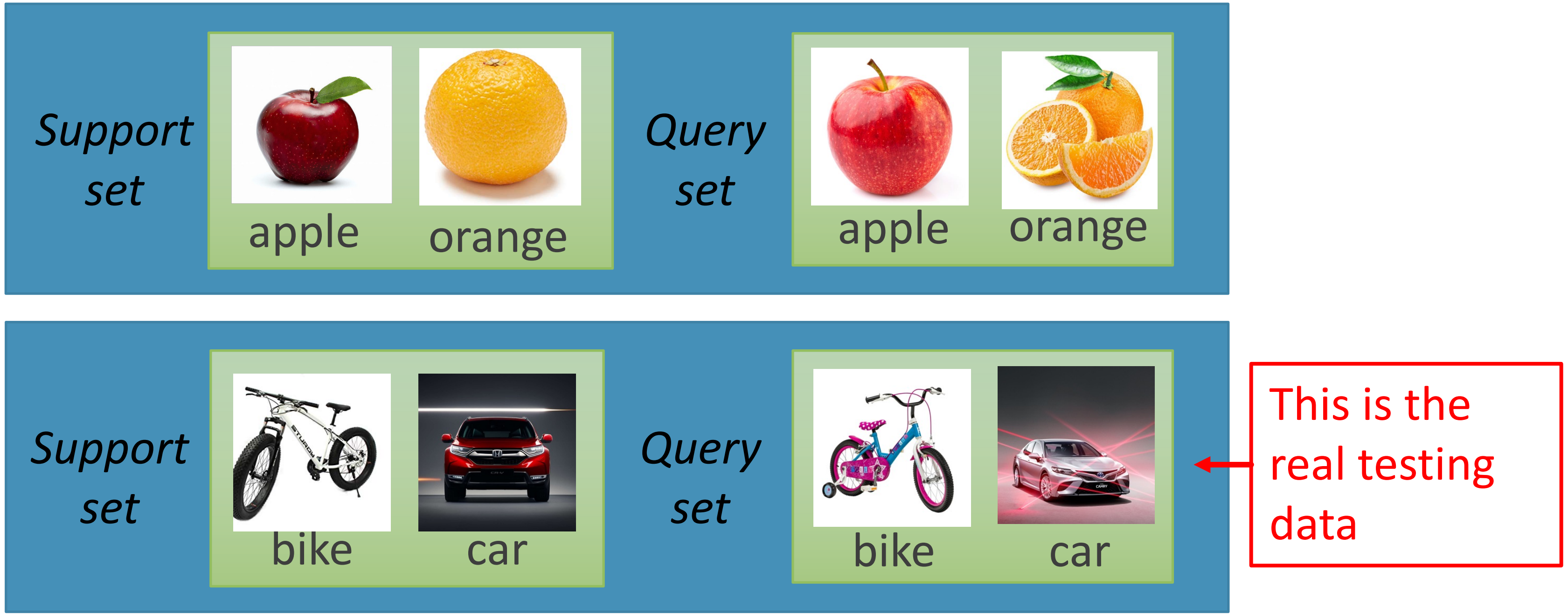
Query set is nothing but “validation data for training”

How to Learn Algorithm Parameter?

- Similarly, we also split testing data into support and query set.
- The label of testing support set is available during testing.
- The testing query set is the real testing data to evaluate the performance of meta-learning algorithm.

Training data

Testing data



Formalizing Meta-Learning

- Training and testing phases in meta-learning is called meta-training and meta-testing.
  - Meta-training can be formalized as bi-level optimization:

Loss to evaluate algorithm

Outer loop
$$\omega^* = \arg\min_{\omega} \sum_{i=1}^{M} L^{meta}\left(\theta^{*(i)}(\omega), D_{train}^{query(i)}\right)$$

Inner loop
$$s.t.\ \ \theta^{*(i)}(\omega) = \arg\min_{\theta} L^{task}\left(\theta, \omega, D_{train}^{support(i)}\right)$$

Loss to evaluate model

  - Meta-testing for new task j can be formalized as:

$$\theta^* = \arg\min_{\theta} L^{task}\left(\theta, \omega^*, D_{test}^{support(j)}\right)$$

And finally we use f_{θ^*} to predict samples in $D_{test}^{query(j)}$.

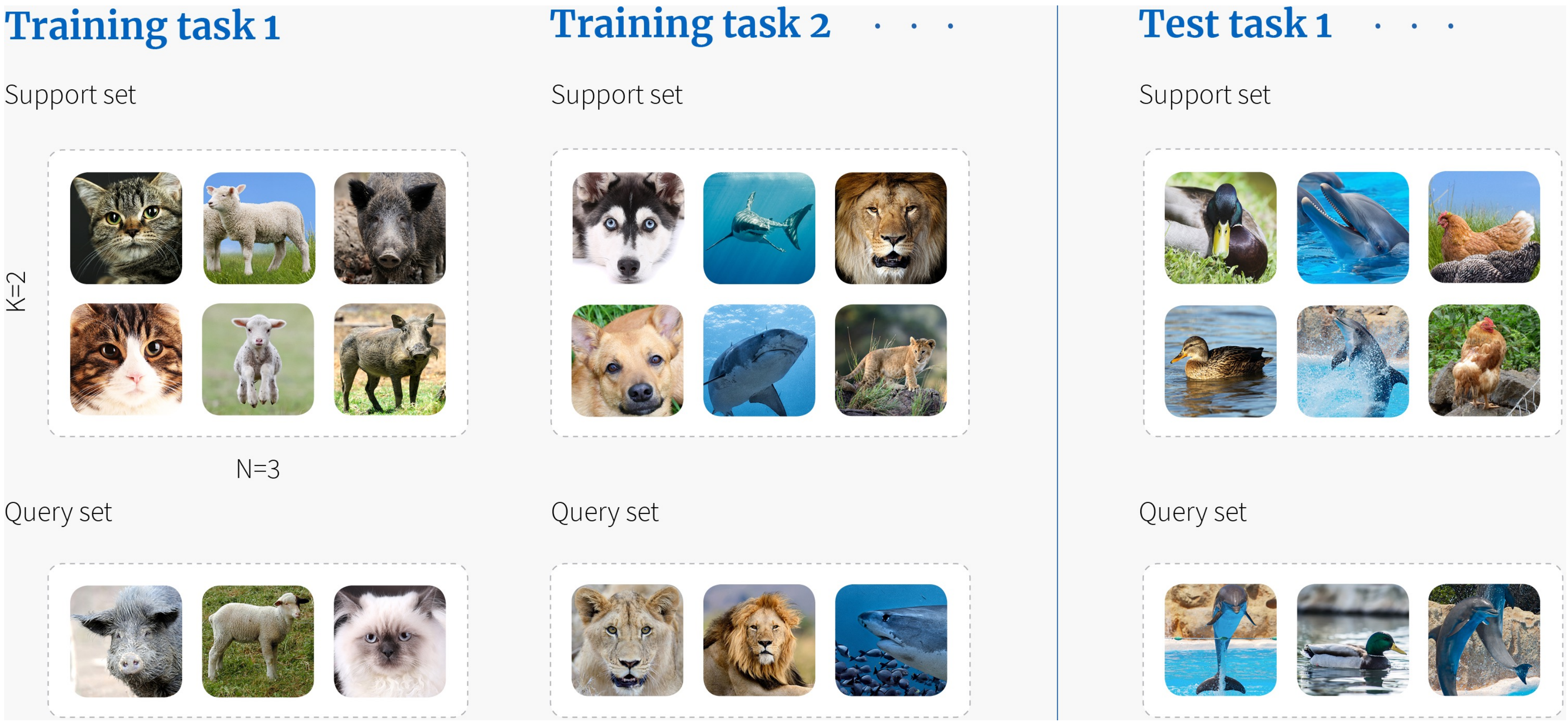
Few-Shot Learning

- One direct application of multi-task meta-learning is few-shot learning (小样本学习).
- Do we human beings need great amount of training data to recognize image category?
- No! We have the learning ability. We know how to learn!
  - When we deal with new task, our experience help us learn with only a few samples.
  - We are experts of "learning to learn".

- Have you seen before an okapi?
- Can you learn to recognize it from only this image?

Image source: https://cdn.britannica.com/80/146380-050-9144819E/Okapi.jpg

Few-Shot Learning



A typical setting for few-shot learning: n-way-k-shot

k is usually set at 1 or 5

Image source: https://www.borealisai.com/en/blog/tutorial-2-few-shot-learning-and-meta-learning-i/

Setting of Few-Shot Learning

- Few-shot learning has its own benchmark datasets.
  - MiniImageNet, Fewshot-CIFAR100, Omniglot, etc.
- A typical dataset split for MiniImageNet is: 64 training classes, 12 validation classes, and 24 test classes.
  - No class overlap among training, validating and test.
- We run a lot of episodes for training. In each episode, we randomly select N classes with $k+1$ samples.
  - k support samples and 1 query sample.

Few-Shot Classification Leaderboard

*mini*ImageNet Leaderboard (5-class)

Edit this leaderboard

Search:

Method	Venue	Year	Backbone	Setting	1-shot	5-shot	Code	Reported by
EASY	arXiv	2022	3xResNet012	Transductive	84.04 ± 0.23	89.14 ± 0.11	[PyTorch]	[Source]
iLPC	ICCV	2021	WRN-28-10	Semi-supervised	83.58±0.79	89.68±0.37	[PyTorch]	[Source]
iLPC	ICCV	2021	WRN-28-10	Transductive	83.05±0.79	88.82±0.42	[PyTorch]	[Source]
PT+MAP	arXiv	2021	WRN	Transductive	82.92 ± 0.26	88.82 ± 0.13	[PyTorch]	[Source]
PTN	AAAI	2021	WRN-28-10	Semi-supervised	82.66 ± 0.97	88.43 ± 0.67	None	[Source]
EASY	arXiv	2022	2xResNet-12(1/√2)	Transductive	82.31 ± 0.24	88.57 ± 0.12	[PyTorch]	[Source]
Simple CNAPS	CVPR	2020	ResNet18 (pre-trained on ImageNet)	Inductive	82.16	89.80	[PyTorch]	[Source]
Oblique Manifold	ICCV	2021	WRN-28-10	Transductive	80.64±0.34	89.39±0.39	[PyTorch]	[Source]
ICA + MSP	ECCV	2020	DenseNet	Semi-supervised	80.11 ± 0.25	85.78 ± 0.13	None	[Source]
EPNet	ECCV	2020	WRN-28-10	Semi-supervised	79.22 ± 0.92	88.05 ± 0.51	[PyTorch]	[Source]

Meta-Knowledge

- Talking so much about the learnable algorithm F_ω.
- What exactly are those algorithm parameters ω?

Hyperparameters

Initial parameters

Parameters of a model generation network

Embeddings

Model architecture

Loss function

…

Meta-Knowledge Taxonomy

- Optimization-based Method
- Model-based Method
- Metric-based Method

OPTIMIZATION-BASED METHOD

Optimization-Based Method

- The meta-knowledge ω is related to the optimization process.
  - Learning to optimize
  - Learning to initialize
  - Learning to weight
  - Learning to reward
  - Learning to augment
  - Dataset distillation
  - Neural architecture search

Learning to Optimize

Learning to learn by gradient descent by gradient descent

Marcin Andrychowicz[1], Misha Denil[1], Sergio Gómez Colmenarejo[1], Matthew W. Hoffman[1], David Pfau[1], Tom Schaul[1], Brendan Shillingford[1,2], Nando de Freitas[1,2,3]

[1]Google DeepMind [2]University of Oxford [3]Canadian Institute for Advanced Research

marcin.andrychowicz@gmail.com
{mdenil,sergomez,mwhoffman,pfau,schaul}@google.com
brendan.shillingford@cs.ox.ac.uk, nandodefreitas@google.com

M. Andrychowicz, M. Denil, S. G. Colmenarejo, and M. W. Hoffman, “Learning to learn by gradient descent by gradient descent,” in NIPS, 2016, pp. 1–17.

Learning to Optimize

- Conventionally, when we do optimization

$$\theta_{t+1} \leftarrow \theta_t + \lambda g\big(\nabla_\theta L(\theta_t)\big)$$

  we select the optimizer g such as SGD, momentum, AdaGrad, ADAM, etc.

  - g can be seen as a hand-crafted function of the gradients.

- In meta-learning, we can learn proper optimization function by meta-knowledge ω:

$$\theta_{t+1} \leftarrow \theta_t + g_\omega\big(\nabla_\theta L(\theta_t)\big)$$

M. Andrychowicz, M. Denil, S. G. Colmenarejo, and M. W. Hoffman, “Learning to learn by gradient descent by gradient descent,” in NIPS, 2016, pp. 1–17.

Learning to Optimize

Support set

Query set

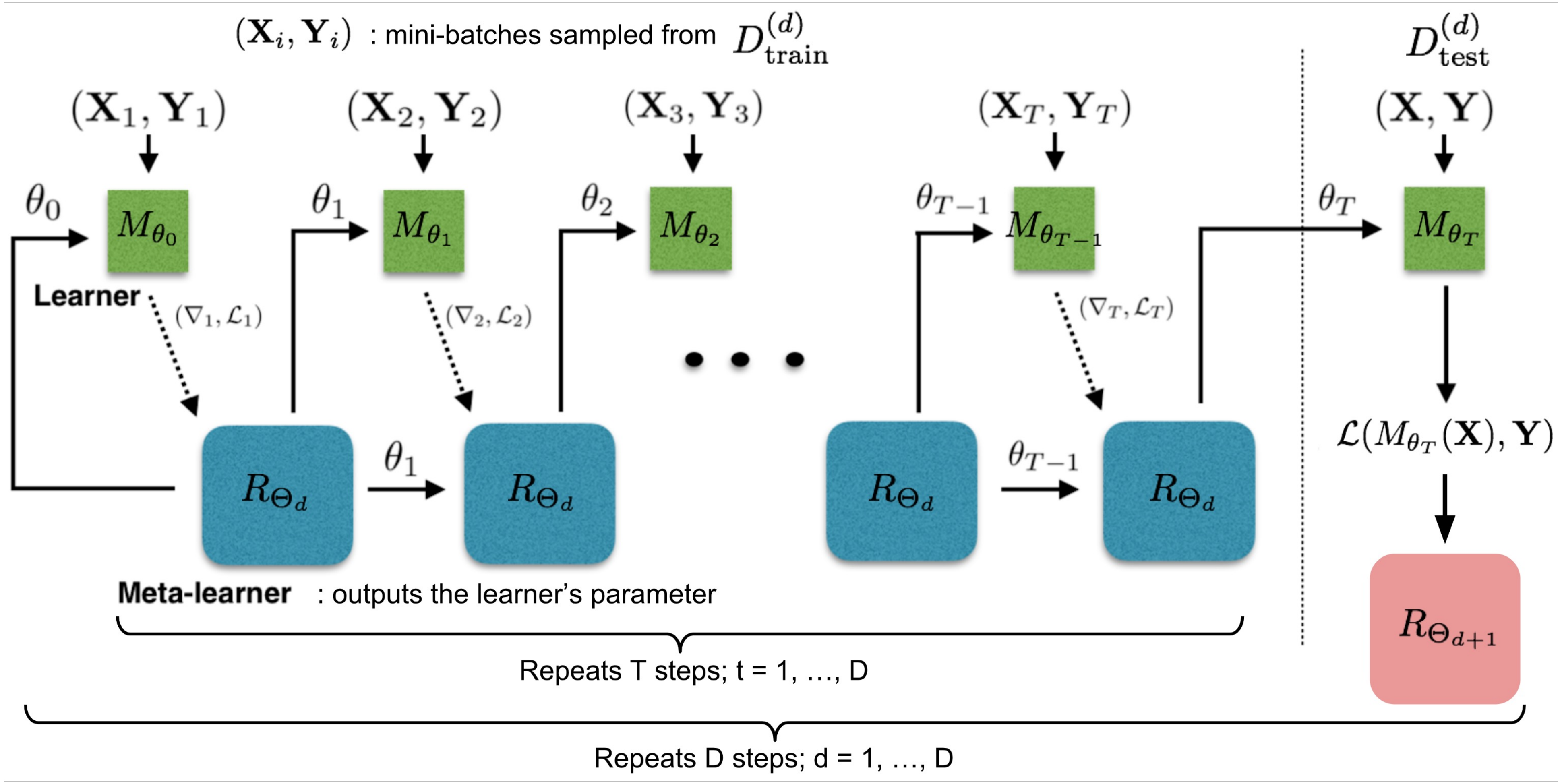


Image source: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html

Original image: S. Ravi and H. Larochelle, "Optimization as a Model for Few-Shot Learning," in ICLR, 2017, pp. 1–11.

Learning to Optimize

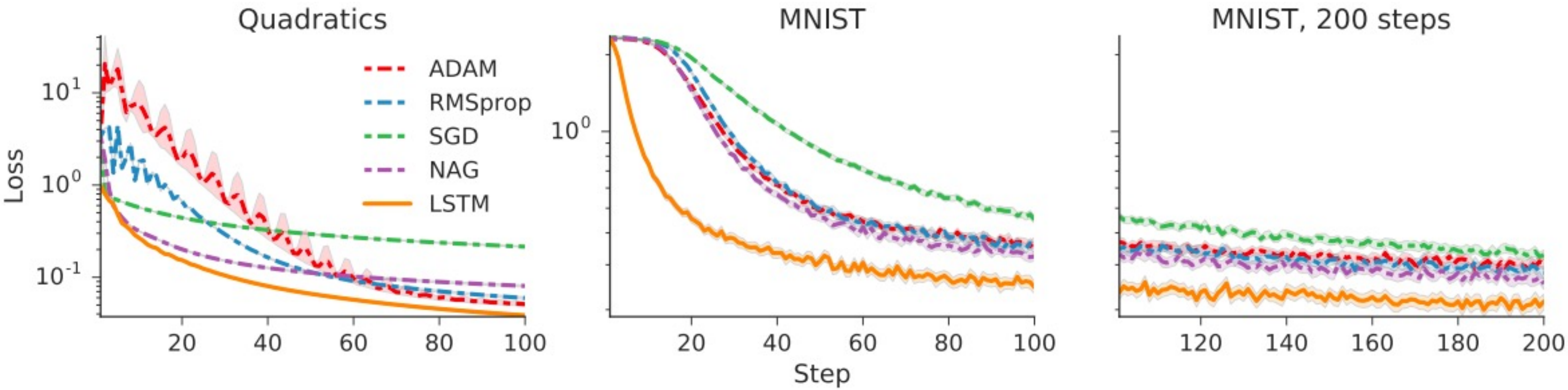
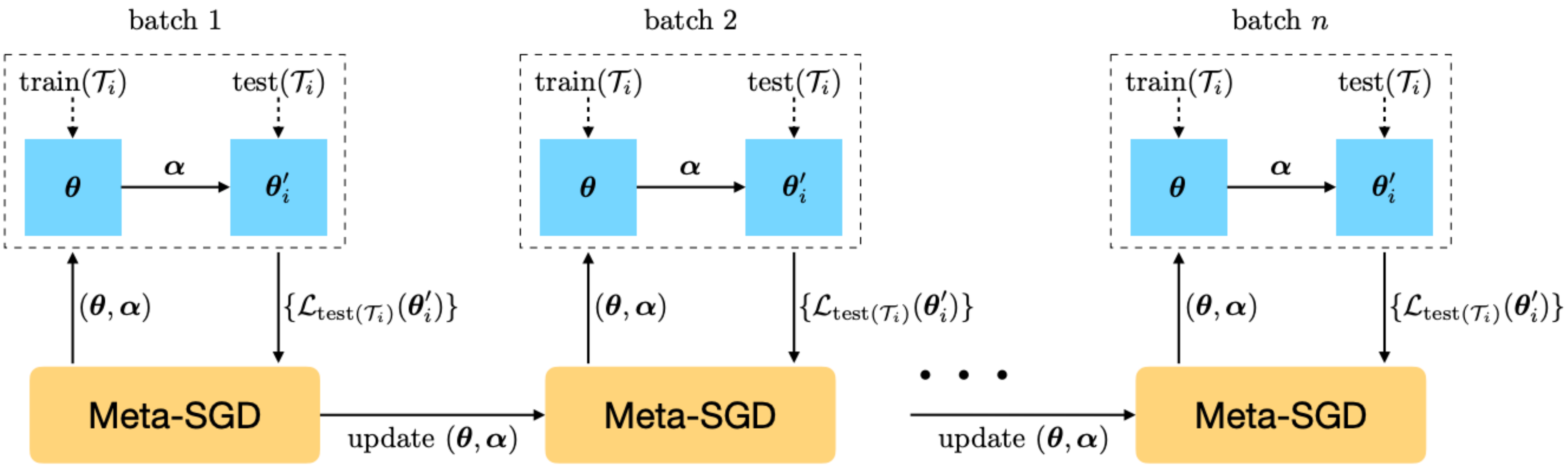


Image source: M. Andrychowicz, M. Denil, S. G. Colmenarejo, and M. W. Hoffman, “Learning to learn by gradient descent by gradient descent,” in NIPS, 2016, pp. 1–17.

Learning to Optimize



Learn update direction and learning rate separately

Image source: Z. Li, F. Zhou, F. Chen, and H. Li, “Meta-SGD : Learning to Learn Quickly for Few-Shot Learning,” in arXiv, 2017.

Learning to Initialize

Model-agnostic meta-learning for fast adaptation of deep networks
C Finn, P Abbeel, S Levine - … on machine **learning**, 2017 - proceedings.mlr.press
… for **meta-learning** that is **model-agnostic**, in the sense that it is compatible with any **model** trained with gradient descent and applicable to a variety of different **learning** problems, …
☆ Save 99 Cite Cited by 10953 Related articles All 14 versions

- In Model-Agnostic Meta-Learning (MAML), ω is the initialized model parameter θ.
- The goal is to find a good θ, such that only a few steps optimization can obtain good model for a task.

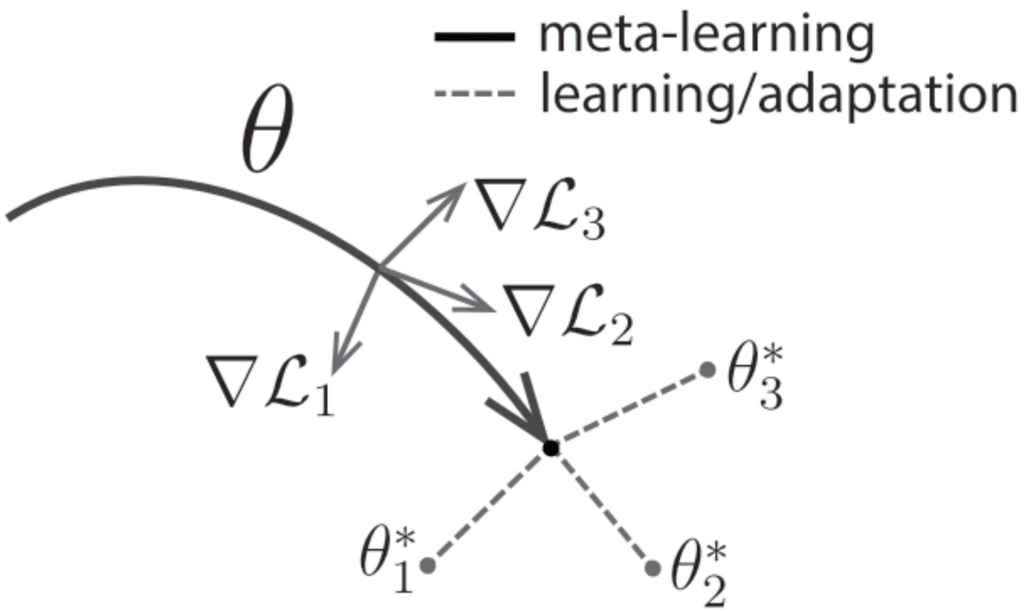


Image source: C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in ICML, 2017, vol. 3, pp. 1856–1868.

MAML

- The optimization of MAML follows:

$$\theta_i' \leftarrow \theta - \beta \nabla_\theta L\left(\theta, D_{train}^{support(i)}\right)$$

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{i=1}^{M} L\left(\theta_i', D_{train}^{query(i)}\right)$$

  - ω is the model parameter θ itself.
  - The loss function is same: $L = L^{meta} = L^{task}$.
- It is call model-agnostic because there is no specified meta-learning model for ω.
  - Any model can apply MAML.

MAML

Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do** — Outer loop
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do** — Inner loop
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta'_i = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$
end while

Inner loop updates model parameter for each task i.

Outer loop updates θ by evaluating each $\theta_i{}'$ on query set.

$\theta_i{}'$ is obtained from θ. Therefore evaluating $\theta_i{}'$ implicitly evaluates θ.

Image source: C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in ICML, 2017, vol. 3, pp. 1856–1868.

MAML vs. pre-trained model

- MAML looks for a good initialization to generalize new task.
- Pre-trained model transfers knowledge from a well-learned model on source tasks to a target task by finetuning.

What is the difference here?

MAML vs. Pre-Trained Model

- MAML doesn't require the initialized model θ perform well on each task, but the one-step optimized $\theta_i{}'$:

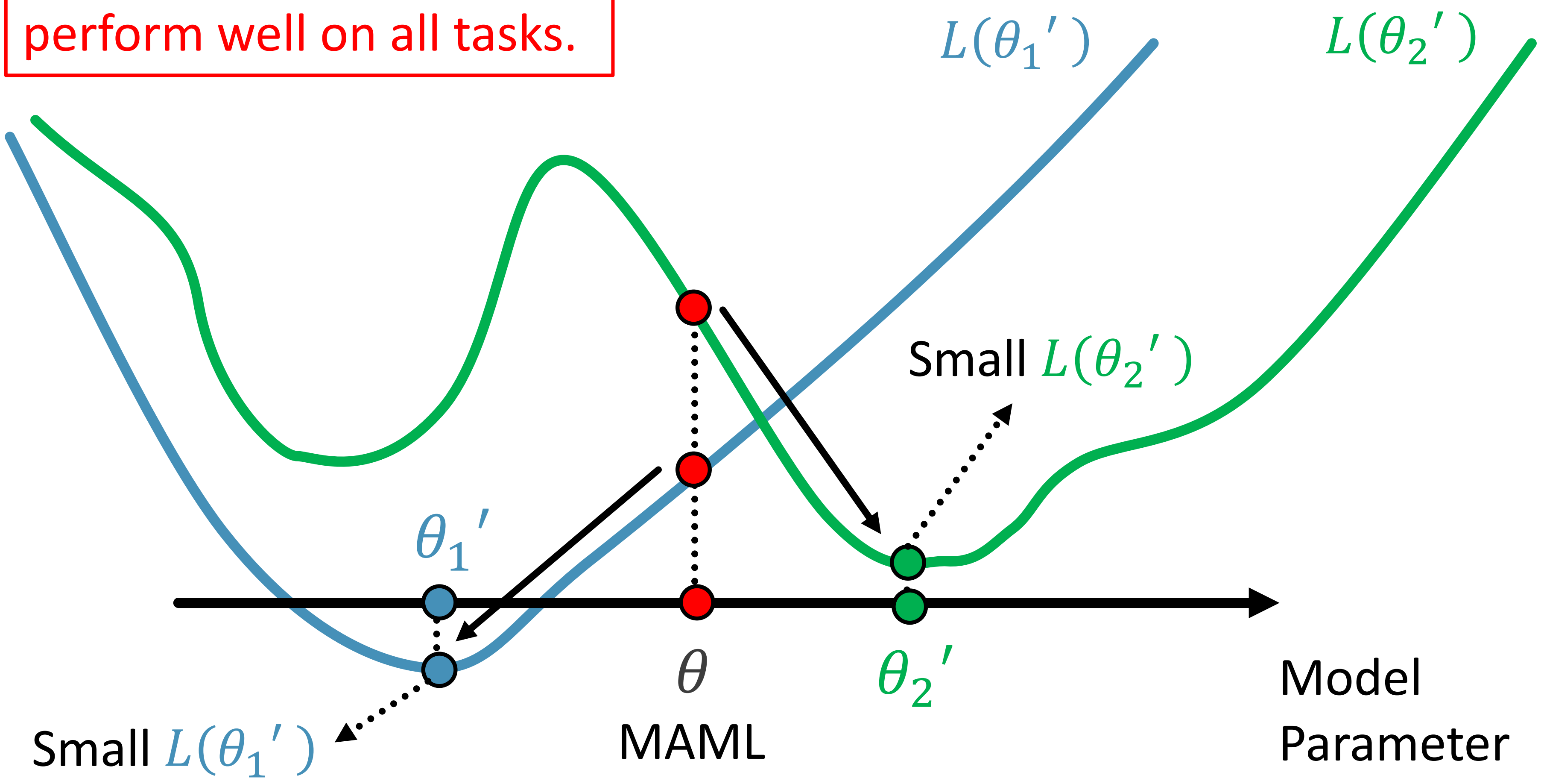
$$L(\theta) = \sum_{i=1}^{M} L\left(\theta_i', D_{train}^{query(i)}\right)$$

- Pre-trained model usually require the initialized model θ perform well on each task:

$$L(\theta) = \sum_{i=1}^{M} L\left(\theta, D_{train}^{(i)}\right)$$

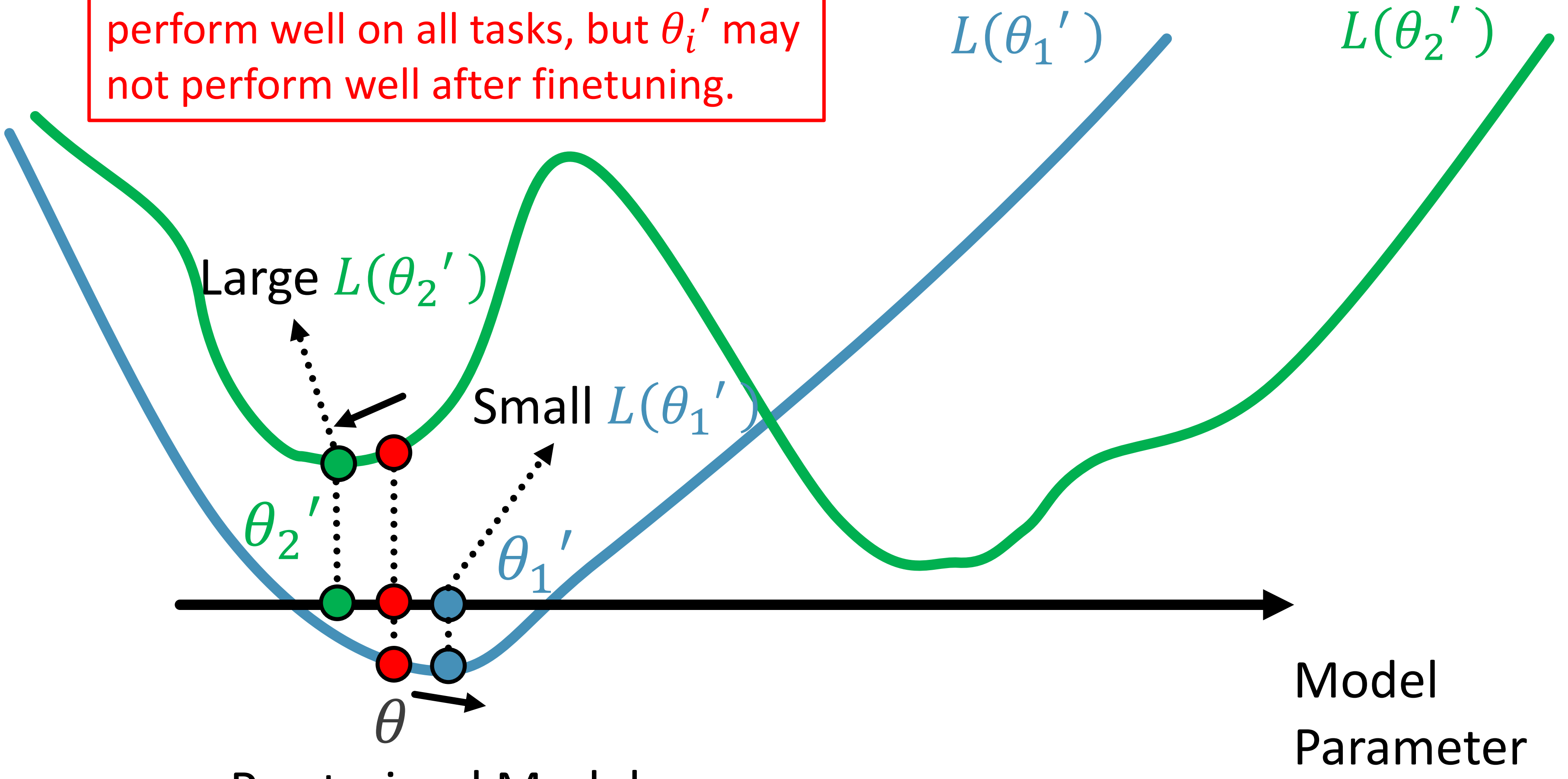
MAML vs. Pre-Trained Model

MAML doesn't require θ perform well on all tasks.



MAML vs. Pre-Trained Model

Pre-trained model requires θ perform well on all tasks, but θ_i' may not perform well after finetuning.

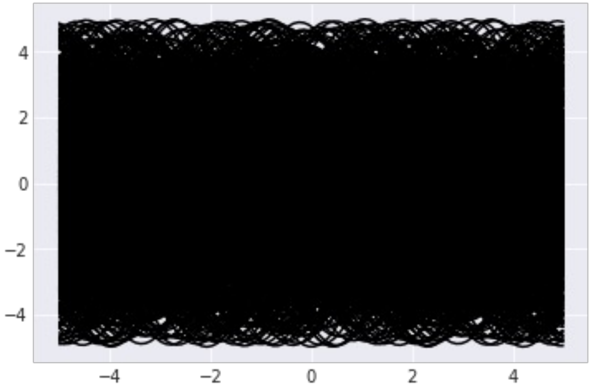


Pre-trained Model

MAML vs. Pre-Trained Model

- Toy example: try to learn sine function $y = a\sin(x + b)$.
- Each combination of a and b is a task.
- The goal is to fit a new sin function based on only a few points.

3 random tasks



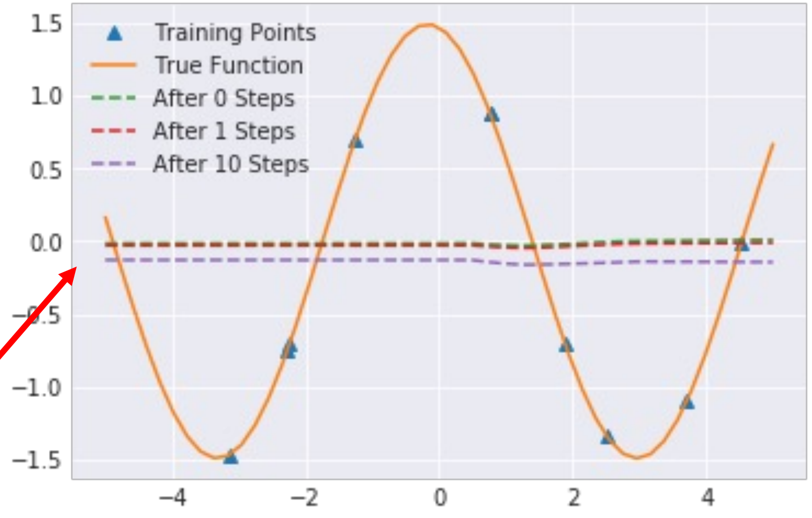
1000 random tasks

Image source: https://towardsdatascience.com/paper-repro-deep-metalearning-using-maml-and-reptile-fd1df1cc81b0

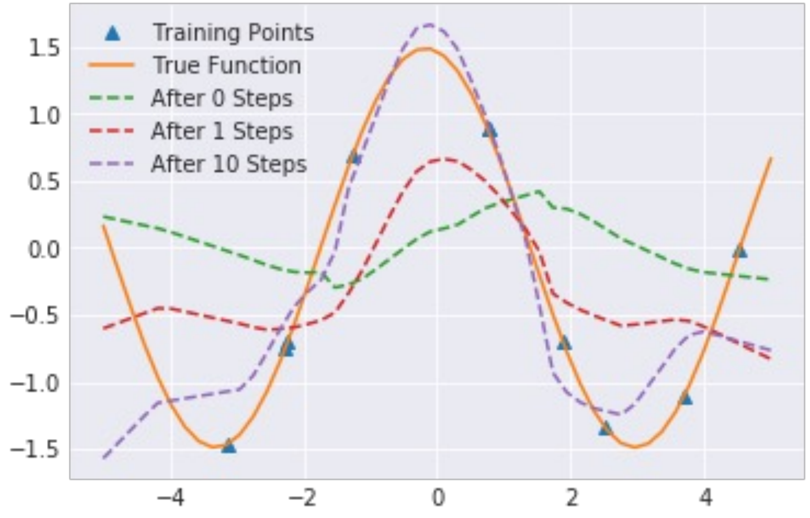
MAML vs. Pre-Trained Model

- MAML is able to quickly adapt with only a few datapoints.
- MAML trained model f_θ has learned to model the periodic nature of the sine wave!

Pre-trained model learns a straight line because it has to minimize the error for all tasks.



Pre-trained model



MAML

Image source: https://towardsdatascience.com/paper-repro-deep-metalearning-using-maml-and-reptile-fd1df1cc81b0

First-Order MAML

- MAML takes partial derivatives on θ at both inner and outer loop.
- Therefore, the outer loop actually calculates the second-order derivatives, i.e. Hessian.

$$\nabla_\theta \sum_{i=1}^{M} L(\theta_i')$$

$$= \sum_{i=1}^{M} \nabla_\theta L(\theta_i')$$

$$= \sum_{i=1}^{M} \begin{bmatrix} \partial L(\theta_i')/\partial\theta_1 \\ \partial L(\theta_i')/\partial\theta_2 \\ \vdots \\ \boxed{\partial L(\theta_i')/\partial\theta_j} \\ \vdots \end{bmatrix}$$

$$\partial L(\theta_i{}')/\partial\theta_j$$

$$= \sum_k \frac{\partial L(\theta_i')}{\partial\theta_k'} \boxed{\frac{\partial\theta_k'}{\partial\theta_j}}$$

$$\frac{\partial\theta_k'}{\partial\theta_j} = \begin{cases} 1 - \beta \dfrac{\partial L(\theta)}{\partial\theta_k\partial\theta_j}, & k = j \\ -\beta \dfrac{\partial L(\theta)}{\partial\theta_k\partial\theta_j} & k \neq j \end{cases}$$

$$\boxed{\theta_k' = \theta - \beta\nabla_\theta L(\theta)}$$

First-Order MAML

- The second order derivative $\frac{\partial L(\theta)}{\partial \theta_k \partial \theta_j}$ is the element of Hessian matrix $H_\theta(L)$.
- We can rewrite the outer loop gradient as:

$$\nabla_\theta L(\theta_i{}') = \big(I - \beta H_\theta(L)\big) \nabla_{\theta_i'} L(\theta_i')$$

- First-Order MAML (FOMAML) calculates the approximation by simply setting the Hessian matrix at 0:

$$\nabla_\theta L(\theta_i{}') \approx \nabla_{\theta_i'} L(\theta_i')$$

First-Order MAML

8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Replacing θ by θ_i' highly improves the efficiency without loss of much accuracy.

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy 1-shot	5-way Accuracy 5-shot
fine-tuning baseline	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
nearest neighbor baseline	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
matching nets (Vinyals et al., 2016)	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
meta-learner LSTM (Ravi & Larochelle, 2017)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML, first order approx. (ours)	$\mathbf{48.07 \pm 1.75\%}$	$\mathbf{63.15 \pm 0.91\%}$
MAML (ours)	$\mathbf{48.70 \pm 1.84\%}$	$\mathbf{63.11 \pm 0.92\%}$

Image source: C. Finn, P. Abbeel, and S. Levine, “Model-agnostic meta-learning for fast adaptation of deep networks,” in ICML, 2017, vol. 3, pp. 1856–1868.

Reptile

- Reptile further simplify FOMAML.

k time update instead of 1

Algorithm 1 Reptile (serial version)

Initialize ϕ, the vector of initial parameters
for iteration $= 1, 2, \ldots$ **do**
  Sample task τ, corresponding to loss L_τ on weight vectors $\widetilde{\phi}$
  Compute $\widetilde{\phi} = U_\tau^k(\phi)$, denoting k steps of SGD or Adam
  Update $\phi \leftarrow \phi + \epsilon(\widetilde{\phi} - \phi)$
end for

Simply use the direction instead of calculating gradient

Image source: A. Nichol, J. Achiam, and J. Schulman, “On first-order meta-learning algorithms,” arXiv, pp. 1–15, 2018.

MAML and Reptile

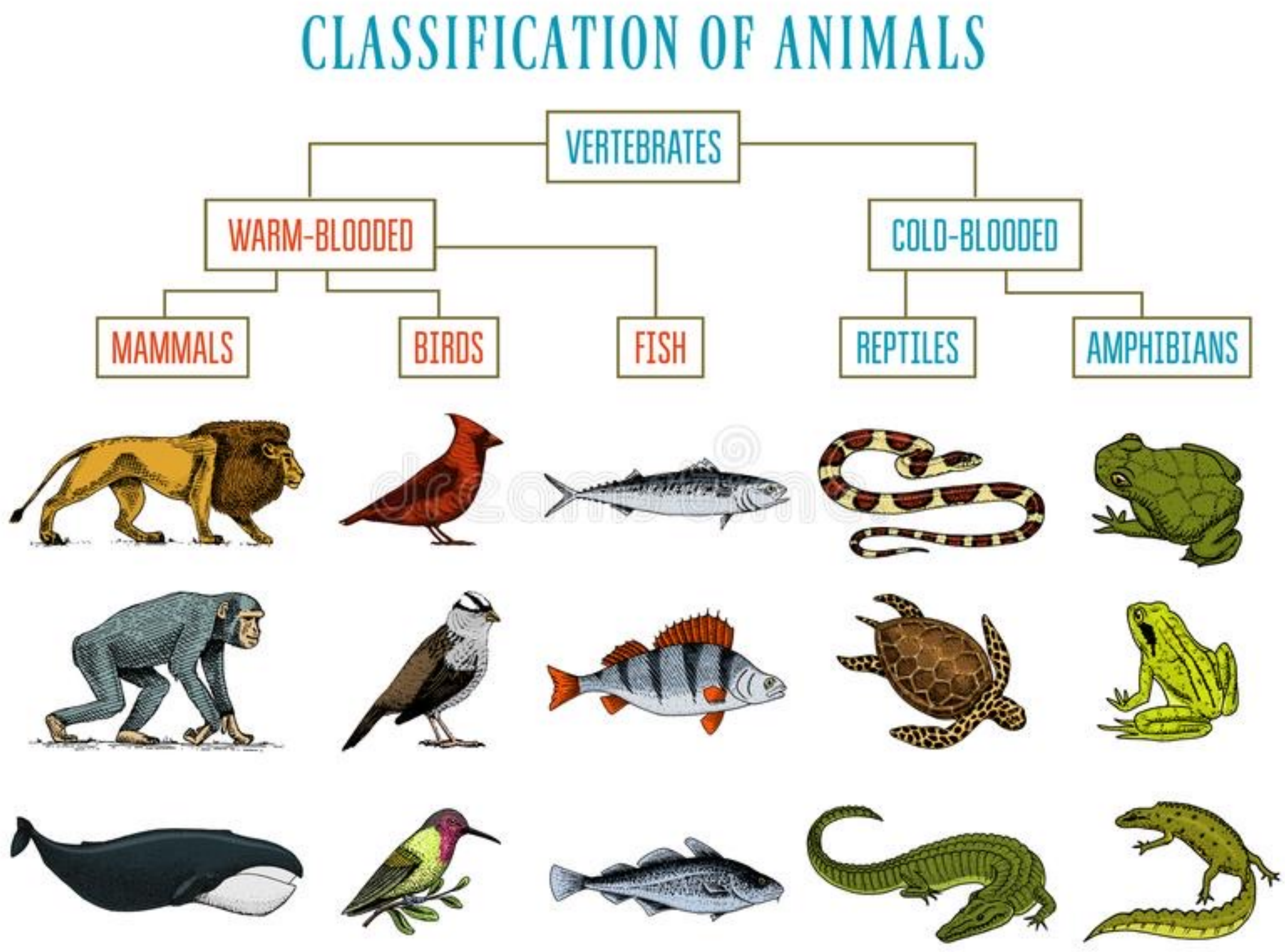


Image source: https://thumbs.dreamstime.com/b/classification-animals-reptiles-amphibians-mammals-birds-crocodile-fish-lion-whale-snake-frog-education-diagram-biology-116831567.jpg

Learning to Weight

- During optimization, we may assign different weights to different training samples, according to its learning difficulty.

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{i=1}^{N} w(\boldsymbol{x}_i) l(\theta, \boldsymbol{x}_i)$$

- In this manner, difficult (frequently misclassified) samples are assigned higher weights.
- For example, Focal loss assigns weight by:

Hand-crafted design!

$$w(\boldsymbol{x}_i) = \left(1 - p_{y_i}\right)^{\gamma}$$

where p_{y_i} is the probability belonging to its ground truth y_i.

Source: T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in ICCV, 2017, pp. 2980–2988.

Learning to Weight

- Can we learn a mapping function from the sample $\boldsymbol{x}_i$ to its weight $w(\boldsymbol{x}_i)$?
- Of course! Simply train an MLP to learn the relationship:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \sum_{i=1}^{N} F_\omega\big(l(\theta, \boldsymbol{x}_i)\big) l(\theta, \boldsymbol{x}_i)$$

where $F_\omega\big(l(\theta, \boldsymbol{x}_i)\big)$ takes the training loss as input and output the corresponding weight.

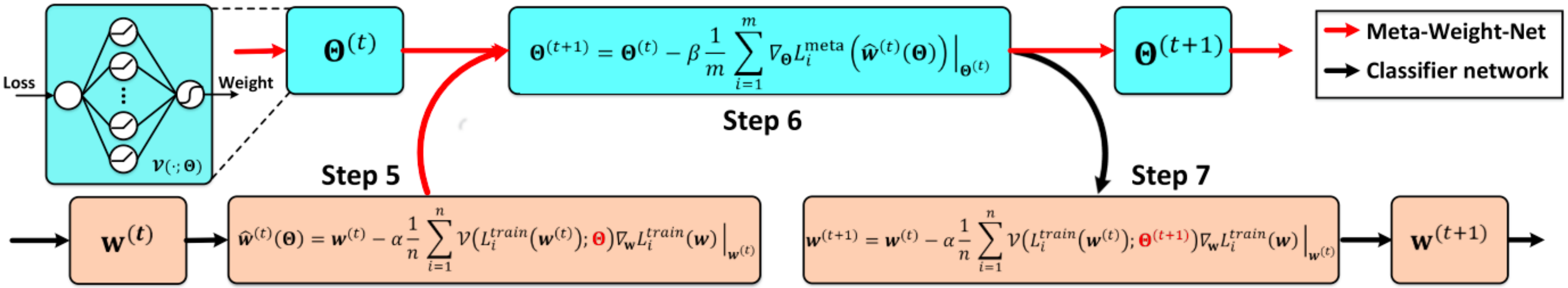
Source: J. Shu et al., “Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting,” in NIPS, 2019, pp. 1–12.

Learning to Weight

Meta-weight-net: Learning an explicit mapping for sample weighting
J Shu, Q Xie, L Yi, Q Zhao, S Zhou… - Advances in neural …, 2019 - proceedings.neurips.cc
… In contrast, with the explicit yet simple **Meta-Weight-Net**, our method can learn the weight in a more stable way, as shown in Fig. 6, and can be easily generalized from a certain task to …
☆ Save 99 Cite Cited by 683 Related articles All 10 versions

- Again, we use $D_{train}^{support}$ ($w^{(t)}$ in the figure) to optimize θ, and use D_{train}^{query} ($\Theta^{(t)}$ in the figure) to optimize ω.

Update meta-knowledge



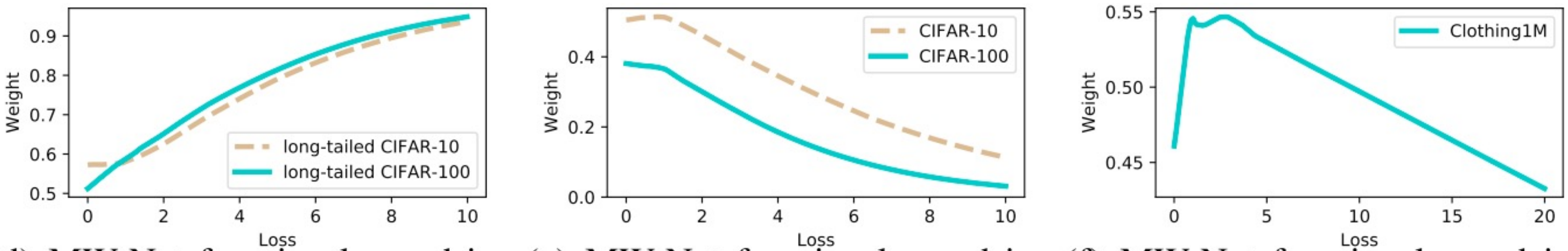
Temporarily update model parameter

Formally update model parameter using updated meta-knowledge

Image source: J. Shu et al., "Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting," in NIPS, 2019, pp. 1–12.

Learning to Weight

- The weighting function is learned based on the distribution of the dataset.



(d) MW-Net function learned in class imbalance case

(e) MW-Net function learned in corrupter labels case

(f) MW-Net function learned in real Clothing1M dataset

Image source: J. Shu et al., “Meta-Weight-Net: Learning an Explicit Mapping For Sample Weighting,” in NIPS, 2019, pp. 1–12.

Learning to Reward

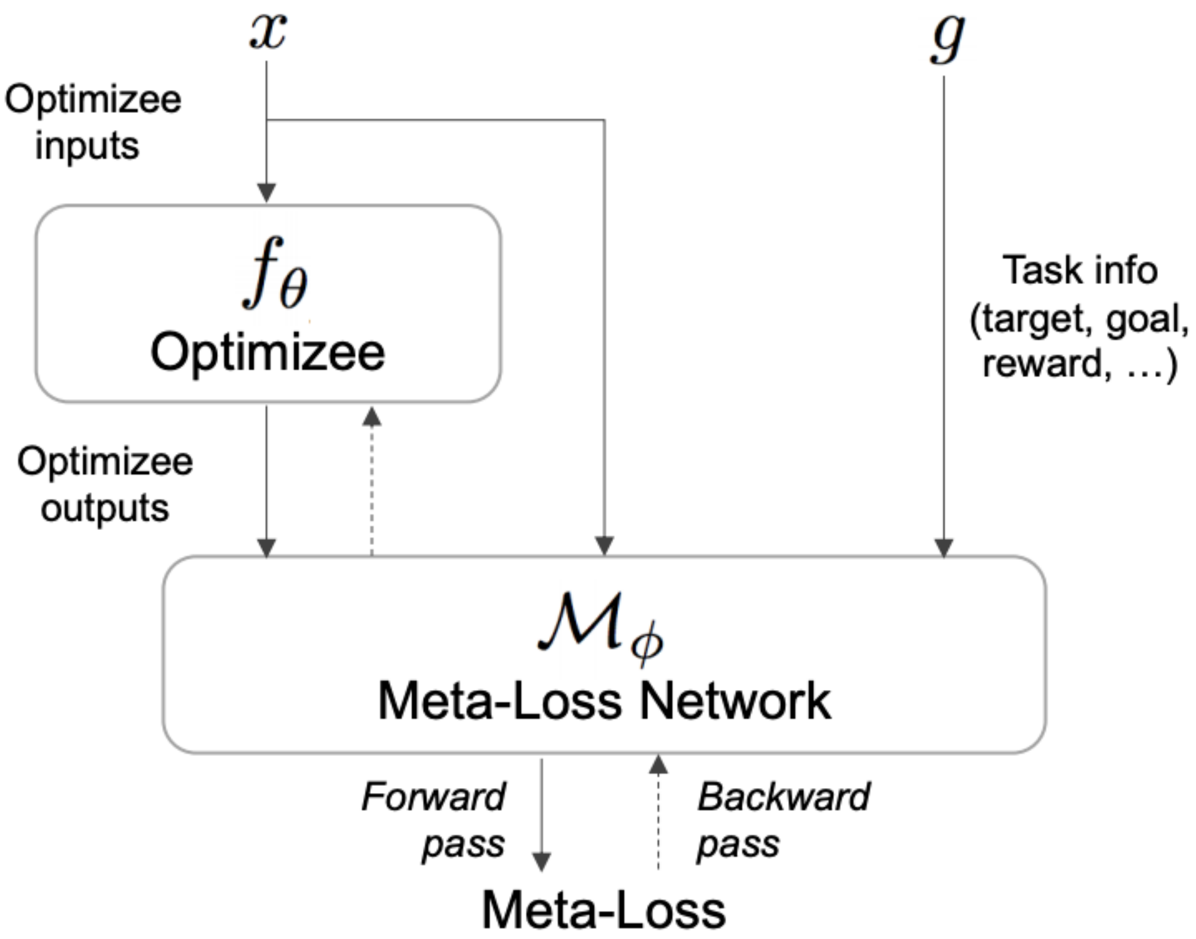


Image source: S. Bechtle, "Meta Learning via Learned Loss," in arXiv.

Learning to Augment

- The data augmentation operation is wrapped up in inner optimization, which is conventionally hand-designed.
  - E.g. crop, zoom, flip, rotate, etc.
- When ω defines the data augmentation strategy, it can be learned by the outer optimization, in order to maximize validation performance.

Learning to Augment

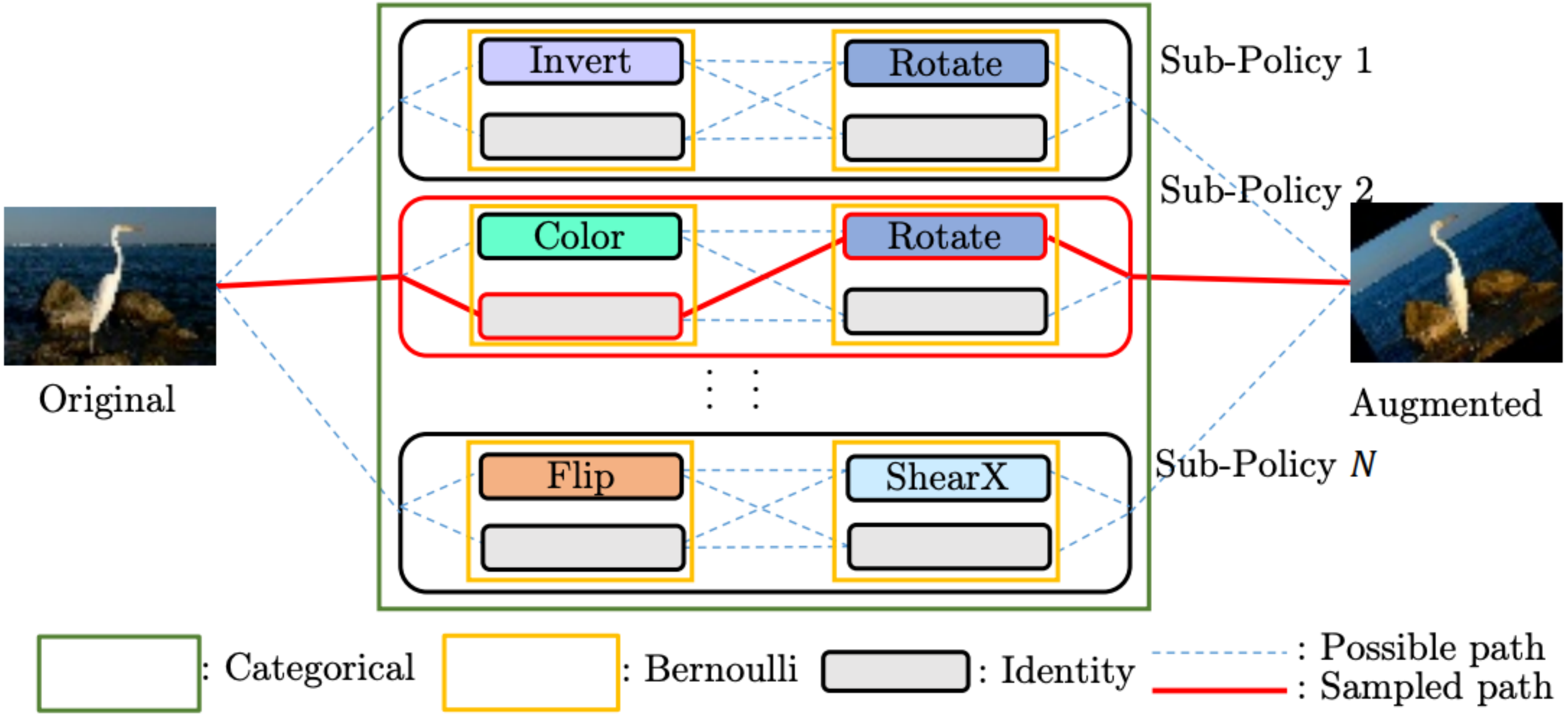


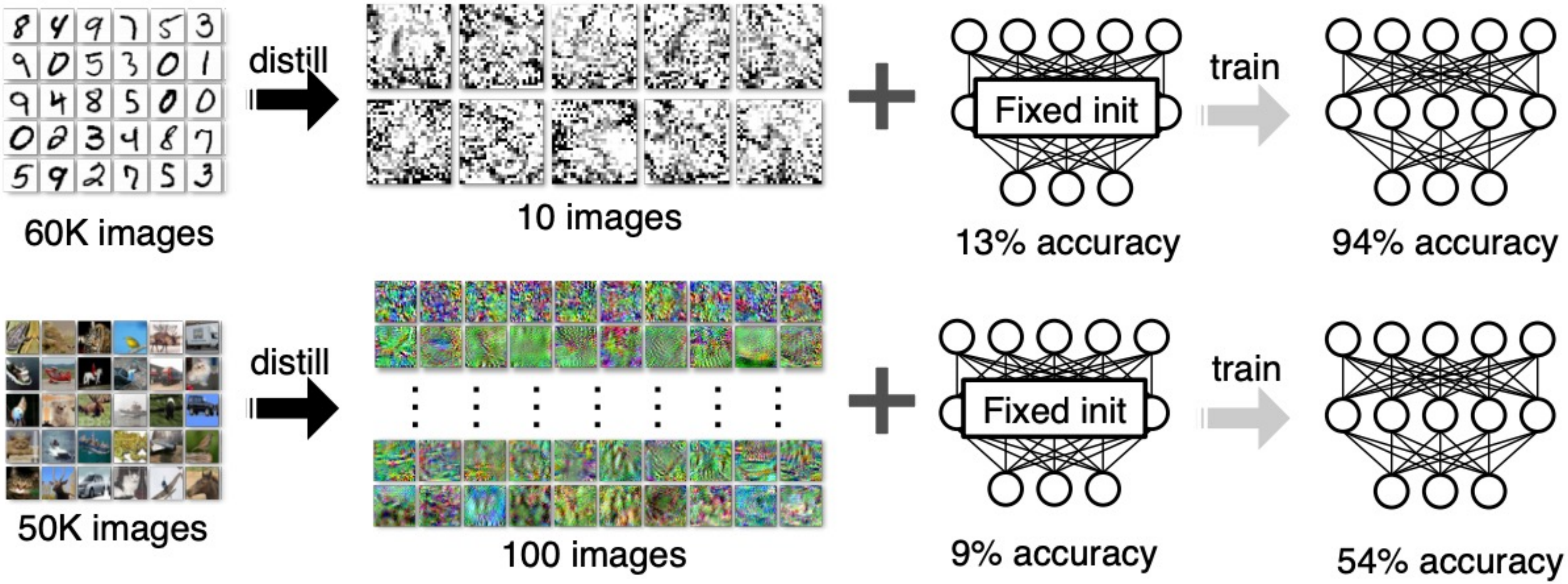
Image source: Y. Li, G. Hu, Y. Wang, T. Hospedales, N. M. Robertson, and Y. Yang, “DADA: Differentiable Automatic Data Augmentation,” in arXiv, 2020, pp. 1–16.

Dataset Distillation

- In bi-level optimization, we always use the same support data to optimize model parameter θ.
- Can the support data itself be the meta-knowledge ω?
  - Select the most significant samples to train the model.
  - Only a few selected samples can achieve high performance.

Dataset Distillation

99% when fully trained



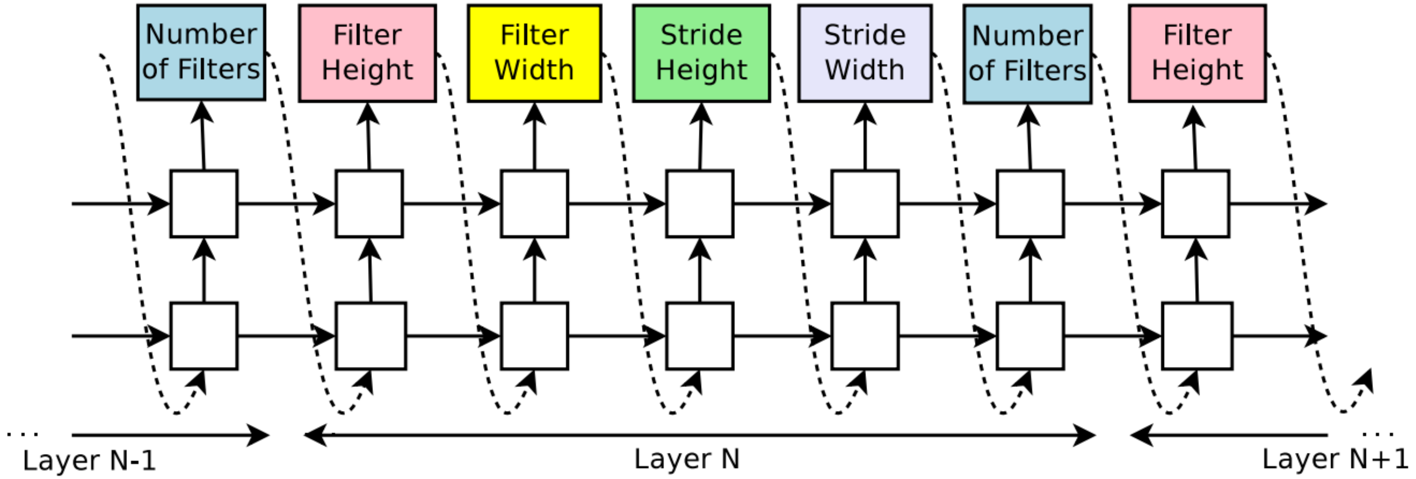
(a) Dataset distillation on MNIST and CIFAR10

80% when fully trained

Image source: T. Wang, J.-Y. Zhu, A. Torralba, and A. A. Efros, "Dataset Distillation," in CoRR, 2018, pp. 1–14.

Neural Architecture Search

- ω specifies the architecture of a neural network.
  - E.g. number of filters, filter size, stride and pooling size, activation functions, shortcut connections, etc.
- The search space is usually hard to define, and optimize.
  - Most search spaces are broad, and the space of architectures is not trivially differentiable.



Reinforcement learning is usually adopted to search the spaces.

Image source: Zoph, Barret, and Quoc V. Le. "Neural architecture search with reinforcement learning." arXiv preprint arXiv:1611.01578 (2016).

MODEL-BASED METHOD

Model-Based Method

- The optimization of model network f_θ in all optimization-based methods are still based on gradient descend.
  - In the inner loop, given ω, we optimize θ.

$$\theta^*(\omega) = \underset{\theta}{\operatorname{argmin}}\, L^{task}(\theta, \omega, D_{train}^{support})$$

- Can we omit this optimization step and directly obtain θ^*?

$$\theta^*(\omega) = g_\omega(D_{train}^{support})$$

Model-based methods adopt the meta-knowledge ω to directly generate a model.

Memory-Augmented Neural Networks

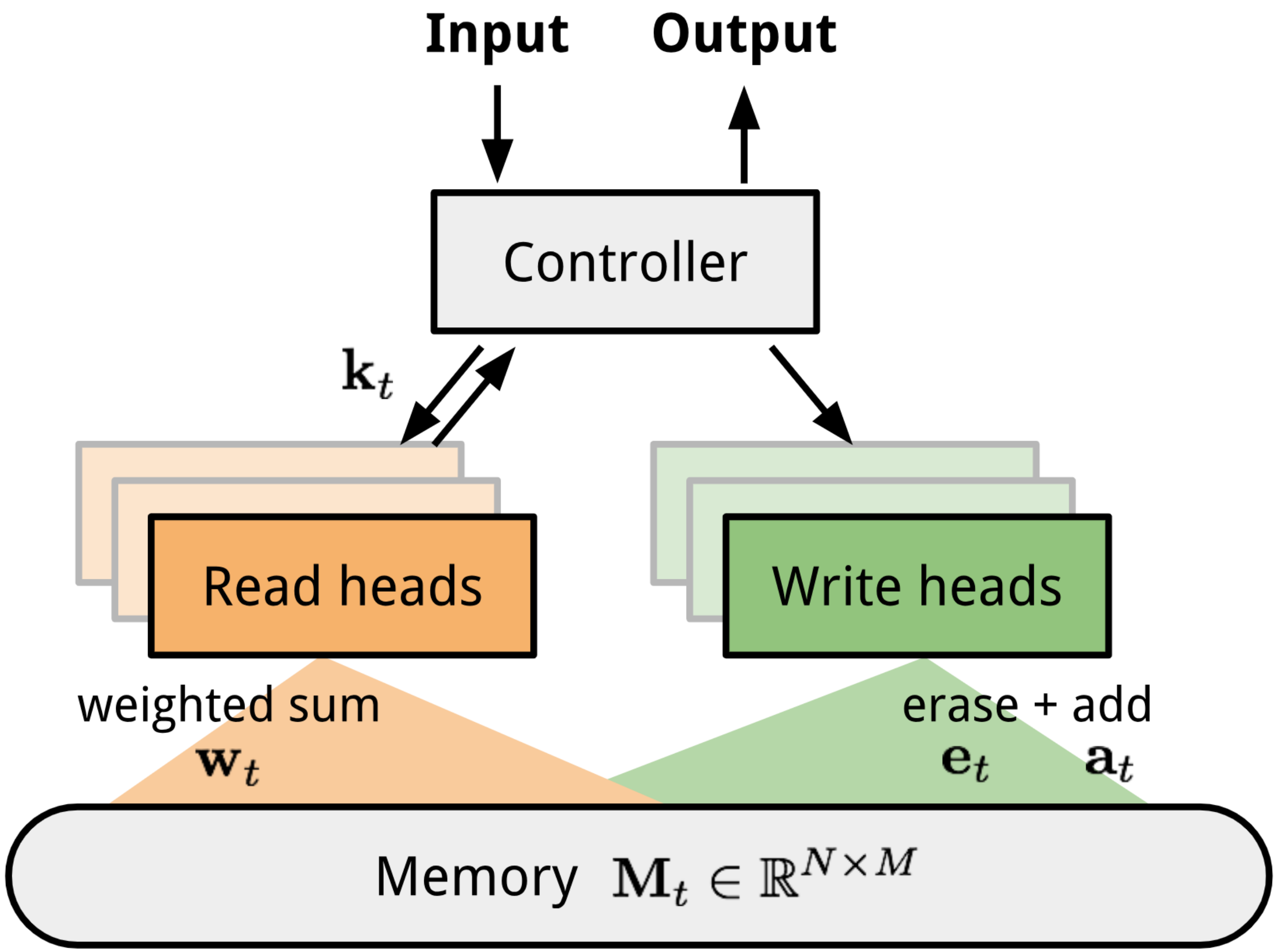
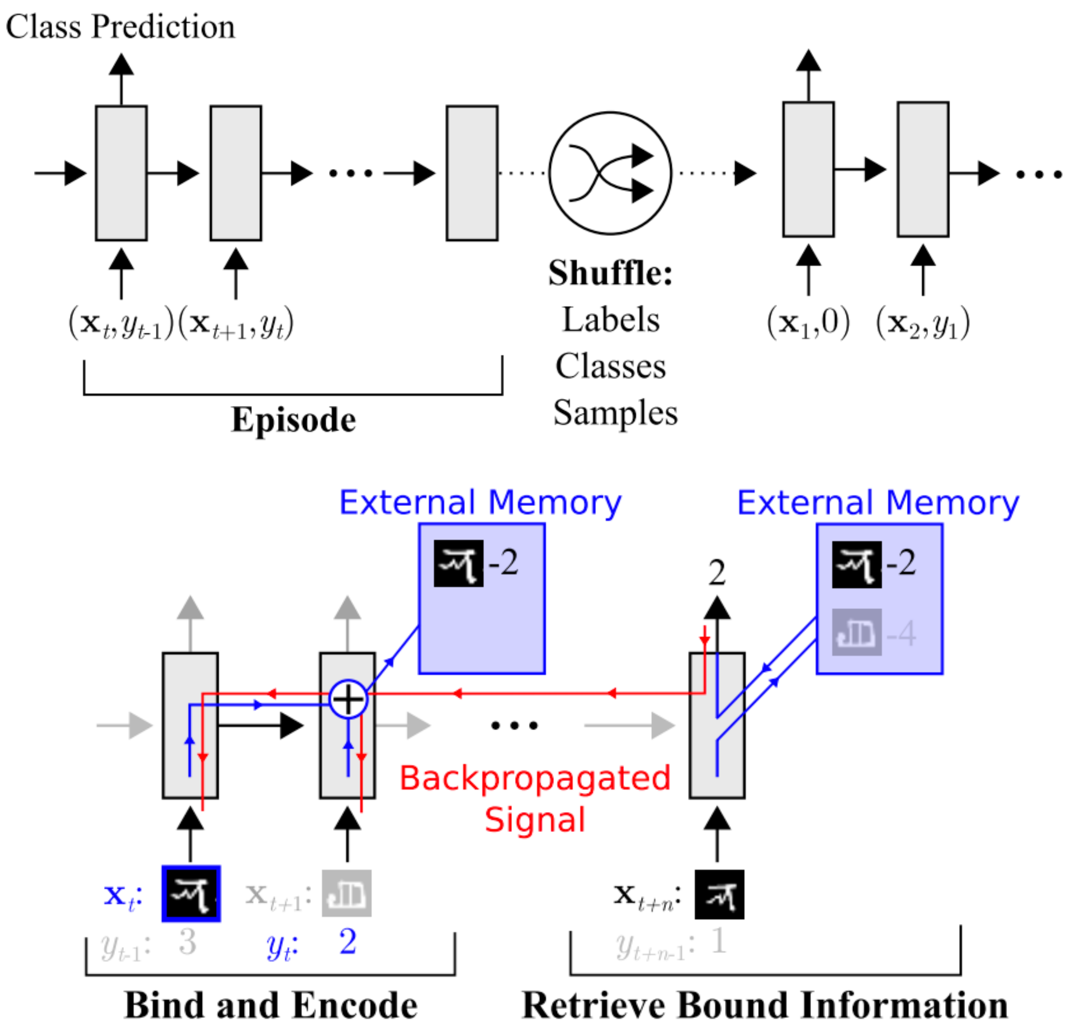


Image source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html#neural-turing-machines

Memory-Augmented Neural Networks



Meta-knowledge ω learns proper representation to read and write with memory.

Image source: A. Santoro, M. Botvinick, T. Lillicrap, G. Deepmind, and W. G. Com, "Meta-Learning with Memory-Augmented Neural Networks," in ICML, 2016, vol. 48.

Meta Networks

- Slow weights: weights that are learned from an optimization process like SGD.
- Fast weights: weights that are directly generated by another network.
- In MetaNet, loss gradients are used as meta information to populate models that learn fast weights.
  - Slow and fast weights are combined to make predictions in neural networks.

Meta Networks

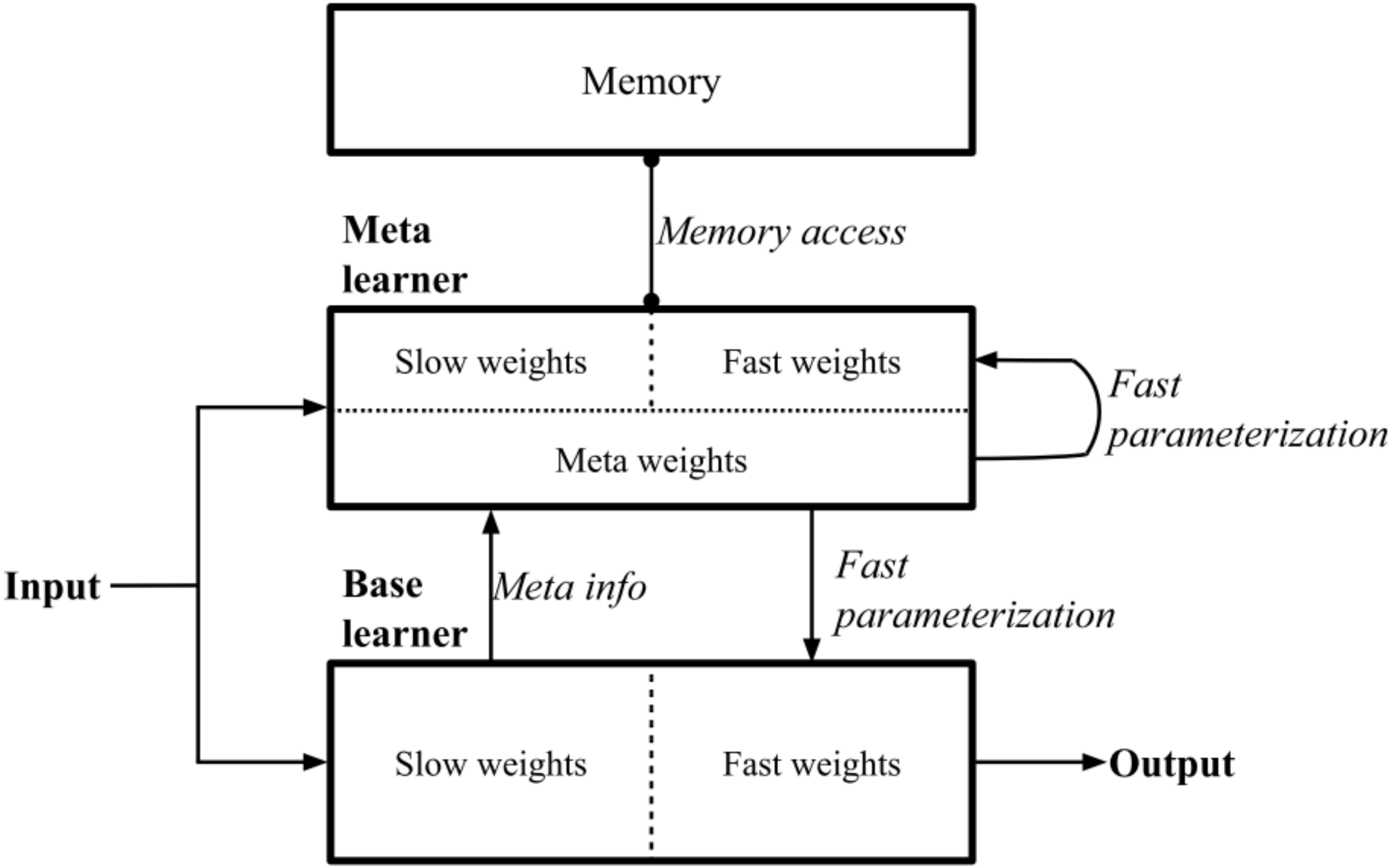
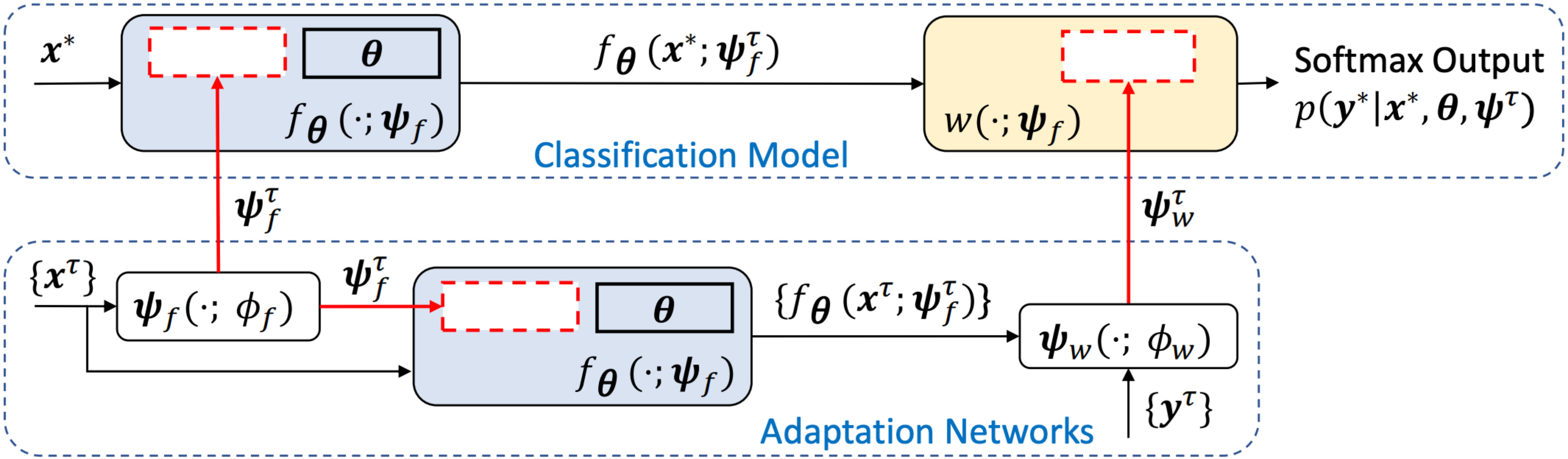


Image source: T. Munkhdalai and H. Yu, "Meta Networks," in ICML, 2017.

CNAPs



The model is not generated from scratch. We generate the adaptation model to new tasks, instead of optimize the model to new tasks.

Image source: J. Requeima, J. Gordon, S. Nowozin, and R. E. Turner, "Conditional Neural Adaptive Processes," in NIPS, 2019.

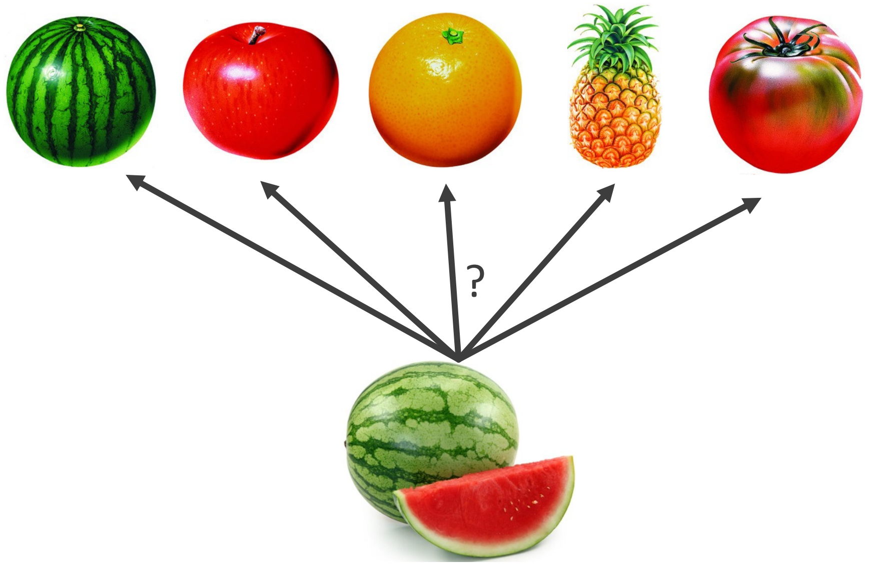
METRIC-BASED METHOD

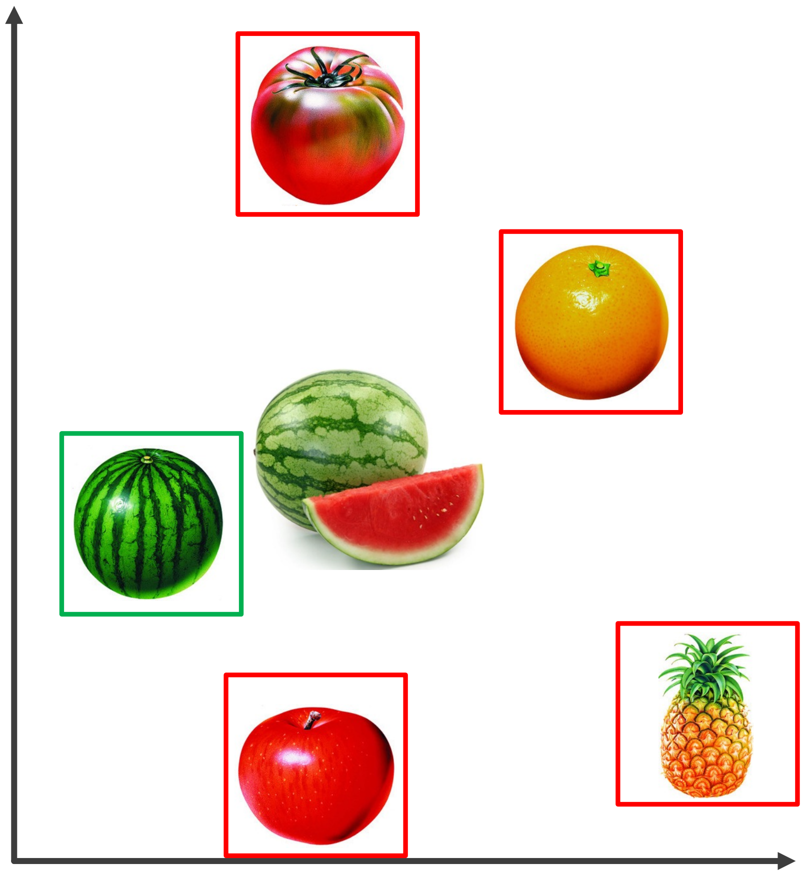
Metric-Based Method

- Previously, we always have a model f_θ to output class score $f_\theta(\boldsymbol{x})$, no matter f_θ is optimized by gradient descend with meta knowledge or directly generated by meta model.
- Do we have to use a model to do prediction? Is there any machine learning method that doesn't have a model?

kNN

Metric-Based Method





Metric-Based Method

- Metric-based methods learn an embedding network F_ω.
- The learned representation is suitable for recognition by simple similarity comparison between query and support instances.
- Take one-shot learning as an example, at testing phase, we simply calculating the similarity between:

$$F_\omega(\boldsymbol{x}_{test}^{query}) \quad \text{and} \quad F_\omega(\boldsymbol{x}_{test}^{support(j)})$$

- It can also be treated as model-based method with only one linear layer.

Metric-Based Method

- Siamese networks
- Matching networks
- Prototypical networks
- Relation networks
- Graph networks

Siamese Networks

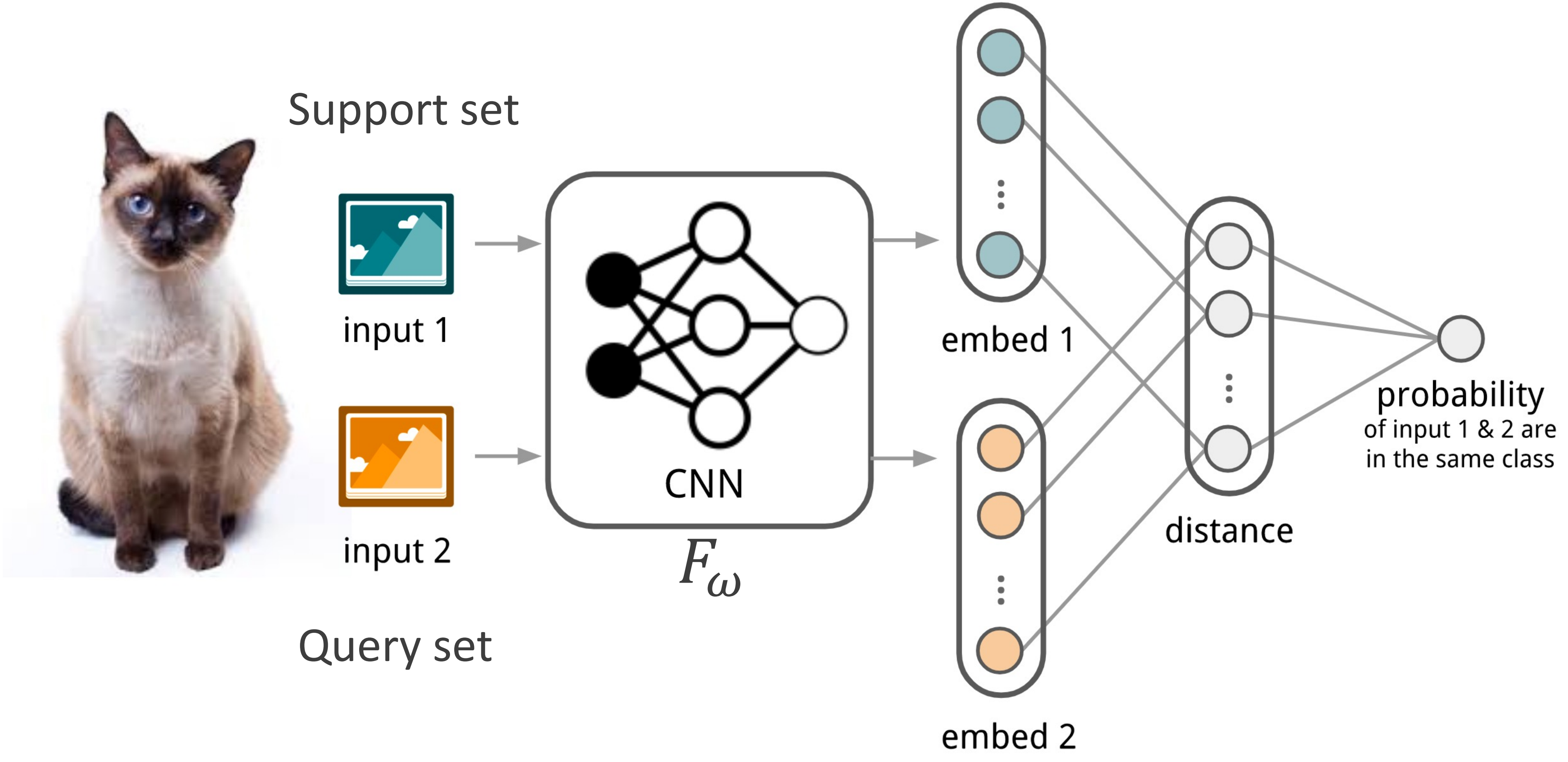


Image source: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.html#convolutional-siamese-neural-network

Matching Networks

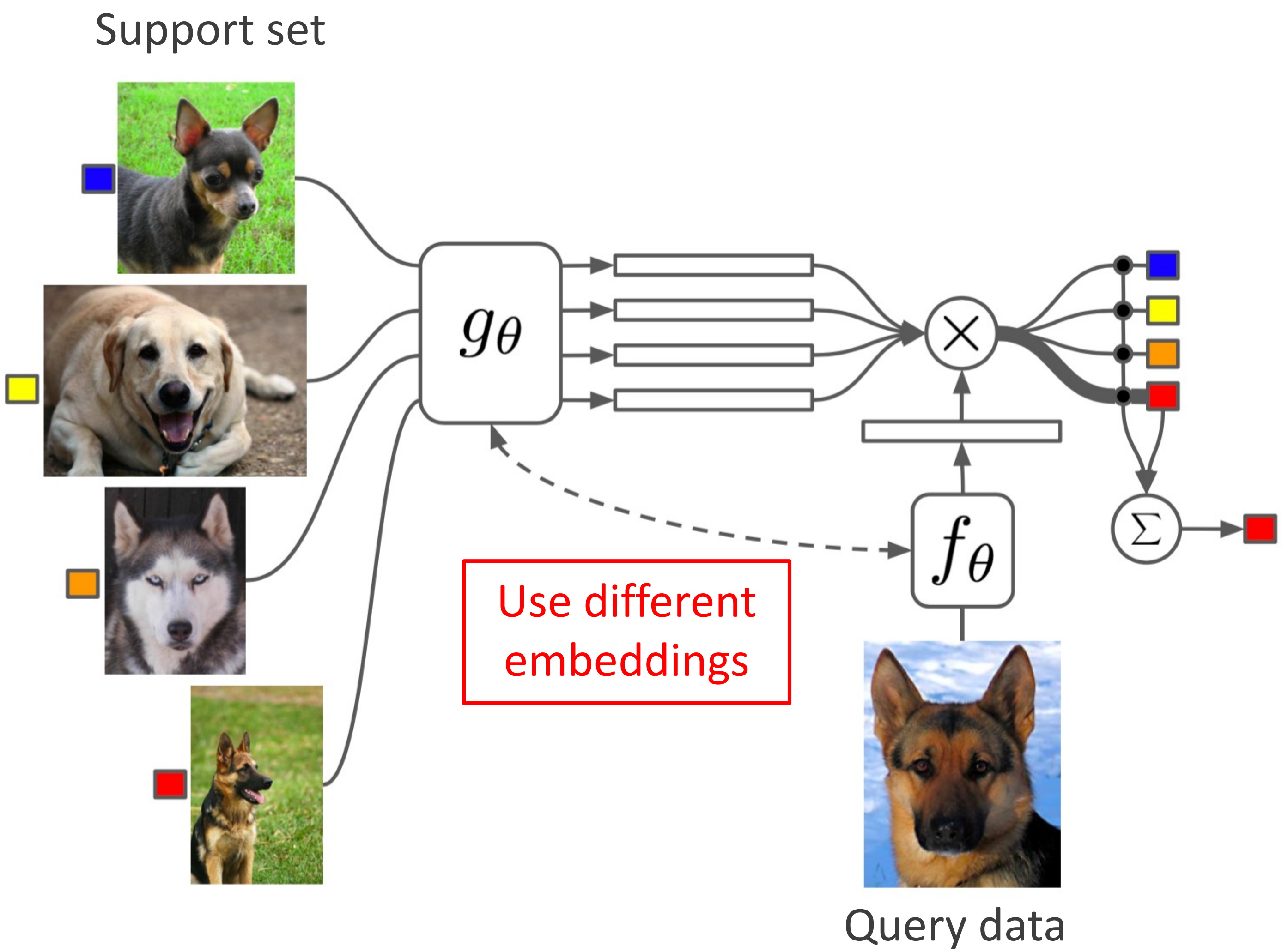


Image source: O. Vinyals, C. Blundell, and T. Lillicrap, “Matching Networks for One Shot Learning,” in NIPS, 2016.

Matching Networks

- Simple version:

$$a(\hat{\boldsymbol{x}}, \boldsymbol{x}_i) = \exp\left(cos\left(F_\omega(\hat{\boldsymbol{x}}), G_\omega(\boldsymbol{x}_i)\right)\right) / \sum_{j=1}^{k} \exp\left(cos\left(F_\omega(\hat{\boldsymbol{x}}), G_\omega(\boldsymbol{x}_j)\right)\right)$$

- Full context version:

$$\begin{aligned}
\hat{\boldsymbol{h}}_k, \boldsymbol{c}_k &= \mathrm{LSTM}(F(\hat{\boldsymbol{x}}), [\boldsymbol{h}_{k-1}, \boldsymbol{r}_{k-1}], \boldsymbol{c}_{k-1}) \\
\boldsymbol{h}_k &= \hat{\boldsymbol{h}}_k + F(\hat{\boldsymbol{x}}) \\
\boldsymbol{r}_{k-1} &= \sum_{i=1}^{|S|} a\left(\boldsymbol{h}_{k-1}, G(\boldsymbol{x}_i)\right) G(\boldsymbol{x}_i)
\end{aligned}$$

$$a\left(\boldsymbol{h}_{k-1}, G(\boldsymbol{x}_i)\right) = \exp\left(\boldsymbol{h}_{k-1}^T G(\boldsymbol{x}_i)\right) / \sum_{j=1}^{|S|} \exp\left(\boldsymbol{h}_{k-1}^T G(\boldsymbol{x}_j)\right)$$

source: O. Vinyals, C. Blundell, and T. Lillicrap, "Matching Networks for One Shot Learning," in NIPS, 2016.

Relation Networks

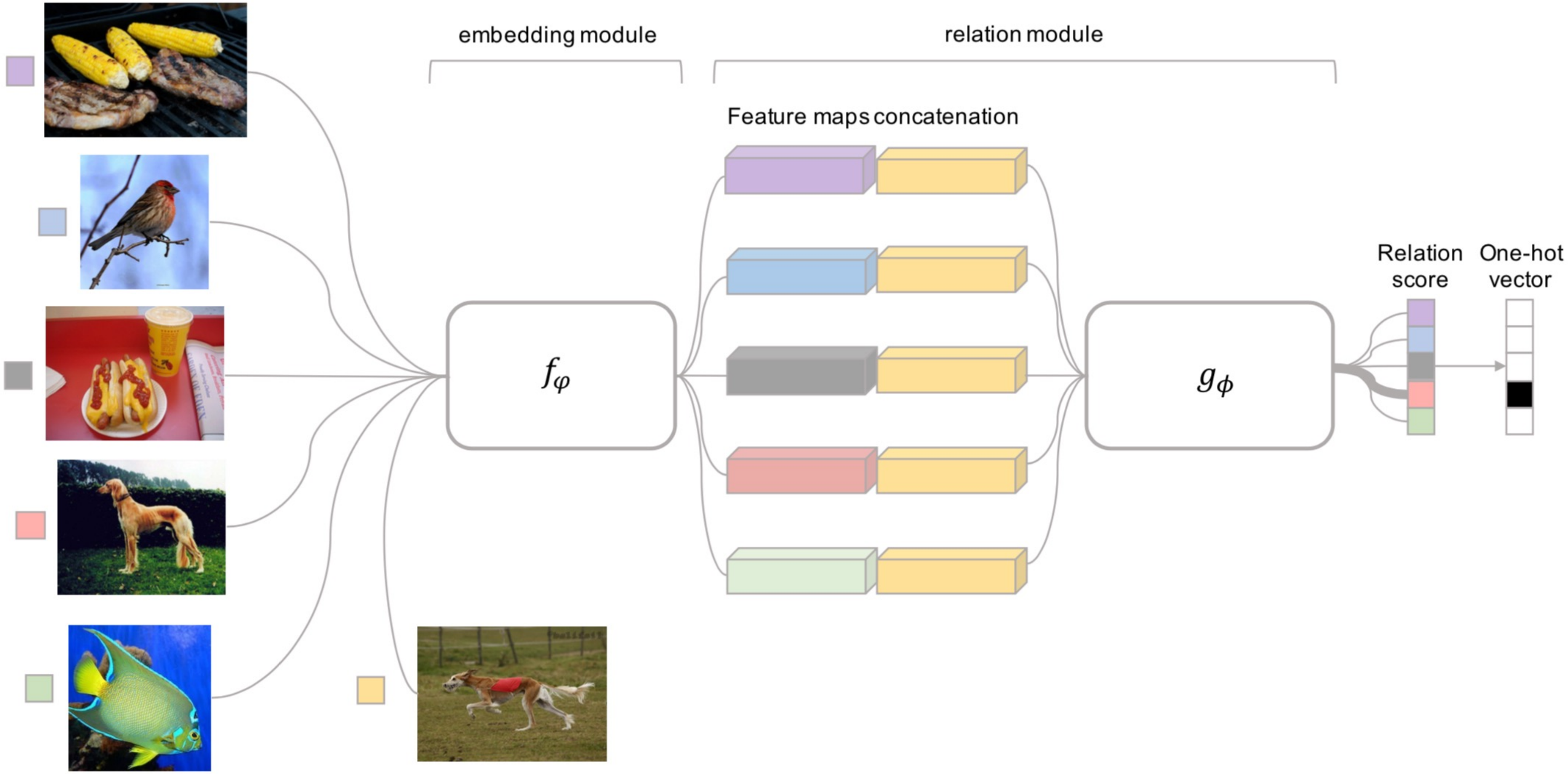
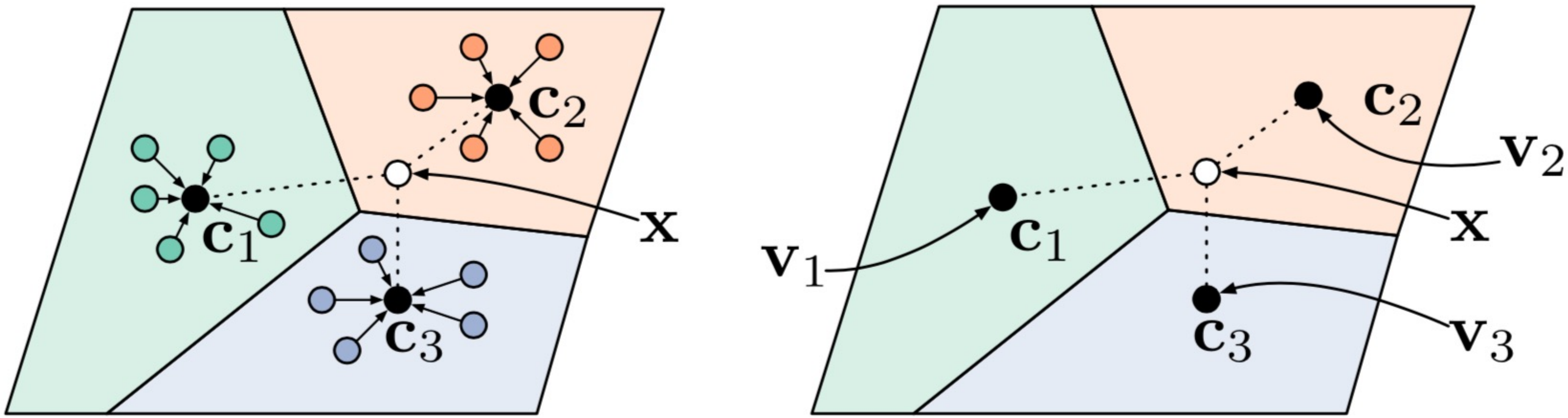


Image source: F. Sung, Y. Yang, and L. Zhang, “Learning to Compare : Relation Network for Few-Shot Learning Queen Mary University of London,” in CVPR, 2018, pp. 1199–1208.

Prototypical Networks



(a) Few-shot (b) Zero-shot

Image source: J. Snell, K. Swersky, and R. Zemel, “Prototypical Networks for Few-shot Learning,” in NIPS, 2017.

Zero-Shot Learning

- One-shot is difficult enough. How can zero-shot work?
- Zero-shot learning differs from few-shot learning in that instead of being given a support set of training points, we are given a class meta-data vector $\boldsymbol{v}_k$ for each class.
  - For example, $\boldsymbol{v}_k$ can be a sentence embedding for text description of the image.
  - The "zero" in zero-shot is for the labelled support set, but we can utilize other information instead.
- We can simply build a mapping between meta-data vector $\boldsymbol{v}_k$ to its prototype:

$$\boldsymbol{c}_k = g_\omega(\boldsymbol{v}_k).$$

Graph Networks

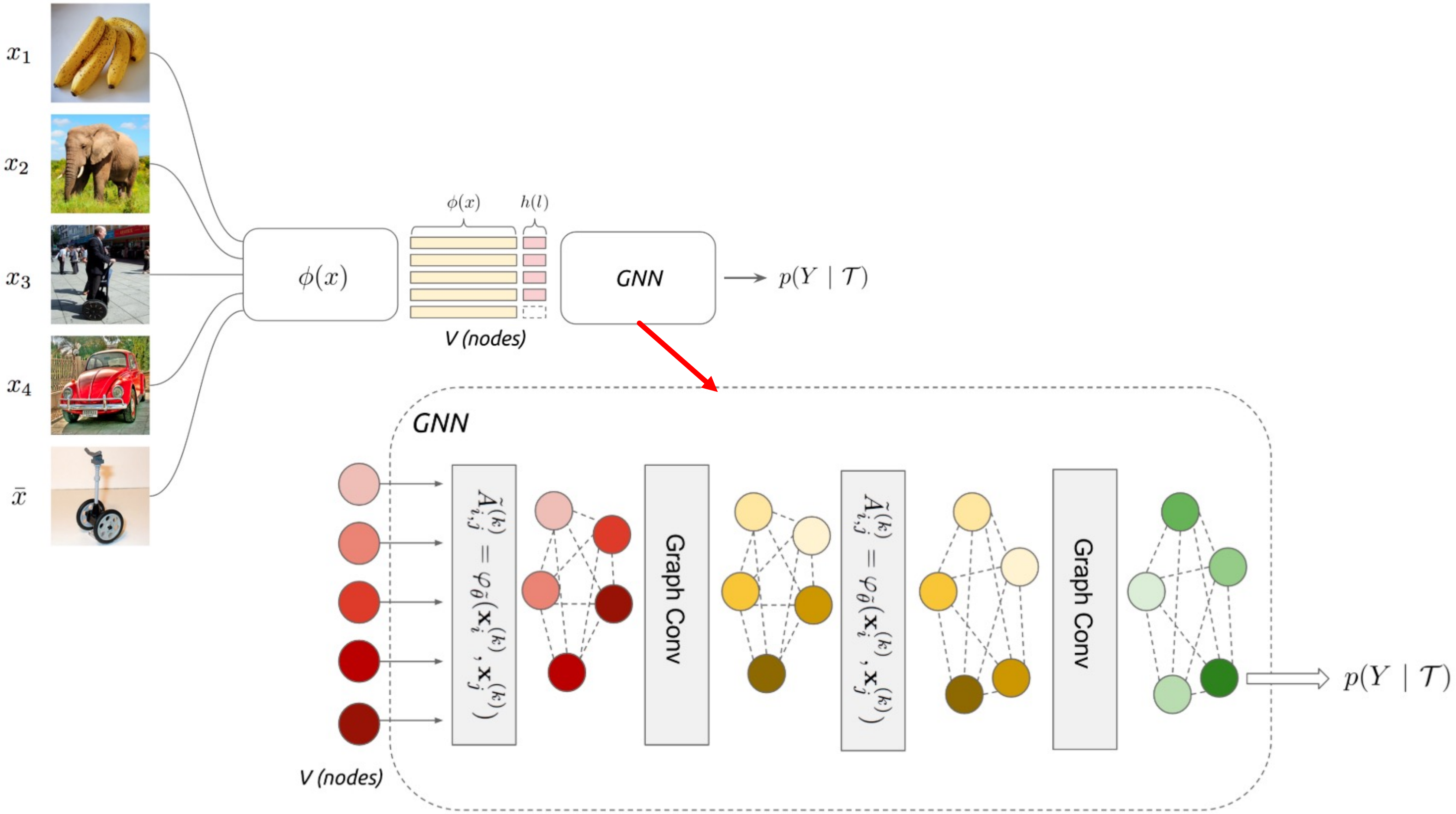


Image source: V. Garcia and J. Bruna, “Few-Shot Learning with Graph Neural Networks,” in ICLR, 2018, pp. 1–13.

Metric-Based Method

Generally, two steps:

- Build an embedding network F_ω.
- Model the relation between $F_\omega(\hat{\boldsymbol{x}})$ and $F_\omega(\boldsymbol{x}_i)$, and output the probabilities.

Conclusion

After this lecture, you should know:

- What is meta-learning and learning to learn.
- How can we utilize meta-knowledge.
- What is few-shot learning and N-way-k-shot.
- What is the difference between optimization-based, model-based and metric-based meta-learning.

Reference & Suggested Reading

- Meta Learning: Learn to Learn
- CS 330: Deep Multi-Task and Meta Learning
- Meta-Learning: Learning to Learn Fast
- T. Hospedales, A. Antoniou, P. Micaelli, and A. Storkey, “Meta-Learning in Neural Networks: A Survey,” in *arXiv*, 2020.

Thank you!

- Any question?
- Don’t hesitate to send email to me for asking questions and discussion. ☺