

An Introduction to Meta Learning

Yen-Yu Lin
National Chiao Tung University

About Dr. Yen-Yu Lin

- Work experience
 - Professor, CS, NCTU, 2019 ~
 - Joint appointment research fellow, CITI, Academia Sinica, 2020 ~
 - Associate research fellow, CITI, Academia Sinica, 2015 ~ 2019
 - Assistant research fellow, CITI, Academia Sinica, 2011 ~ 2015
- Research interests
 - Computer Vision (**CV**):
Let computers see, recognize, and interpret the world like humans
 - Machine Learning (**ML**):
Provide a statistical way to learn how human visual system works
 - Goal: Design **ML** methods to facilitate **CV** applications

Slide credit

- Some slides are collected/modified from Hung-yi Lee, Chelsea Finn, Sergey Levine, HyeongJoo Hwang et al., Yifang Sun, Yee Whye Teh

Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

Deep learning for AI research

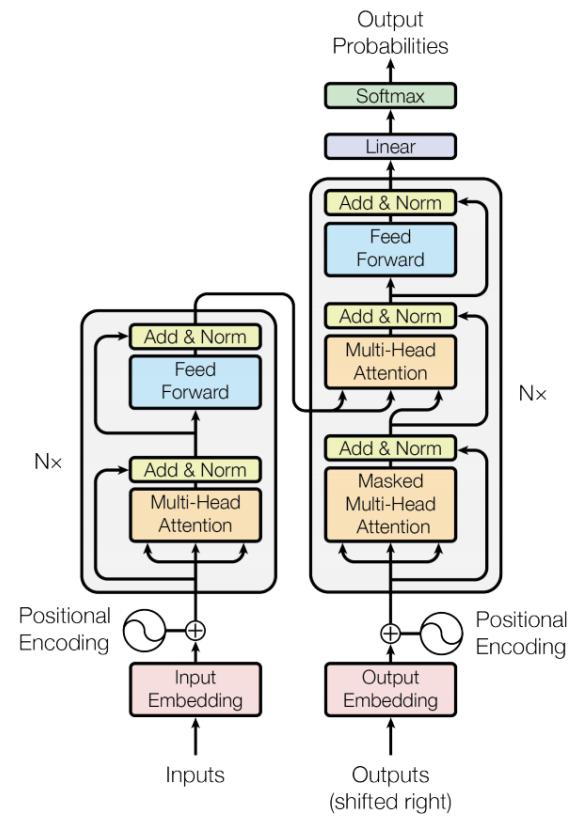
- Deep learning at the core of the current AI revolution
 - Affordable GPU hardware
 - Large annotated datasets
- Deep models with hundreds of layers and millions of parameters are able to
 - Extract knowledge from a huge set of data
 - Map an input to the desire outputs
 - Greatly improve existing applications
 - Even initiate new applications

Deep learning for AI research

- Large deep models from huge training data result in broad generalization



<https://jalammar.github.io/illustrated-gpt2/>



Vaswani et al. NIPS'17

Difficulties

- Supervised deep models need/have
 - Large amounts of labeled training data
 - Many optimization iterations to train a large set of parameters
 - Severe demands on the computing resources
- Applications with small data
 - Medical imaging, robotics, personalized data, recommendations, rare object categories...
- These issues limit the scalability of deep models to rare events, newly emerging domains, and new data categories

An example

training data

Braque



Cezanne



test datapoint



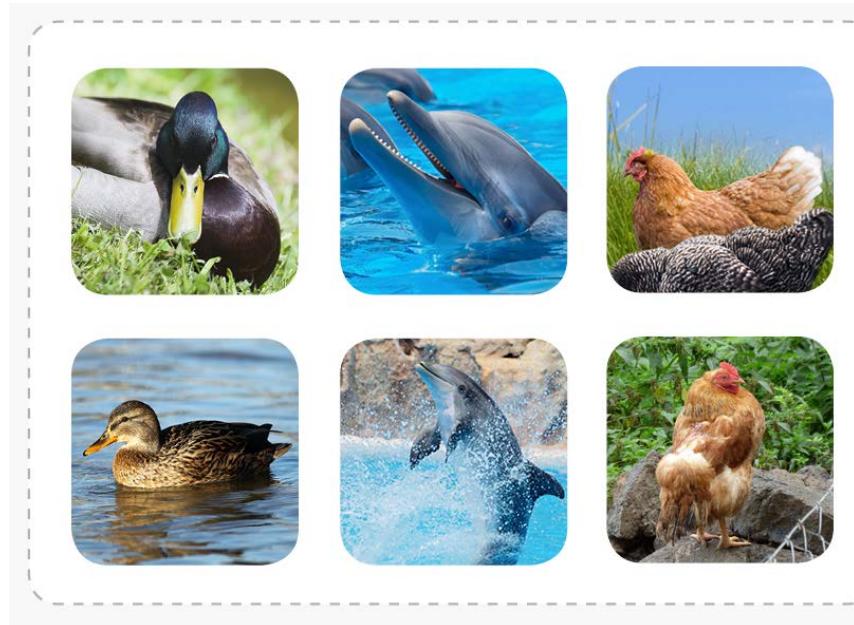
By Braque or Cezanne?



國立交通大學
National Chiao Tung University

Few shot learning

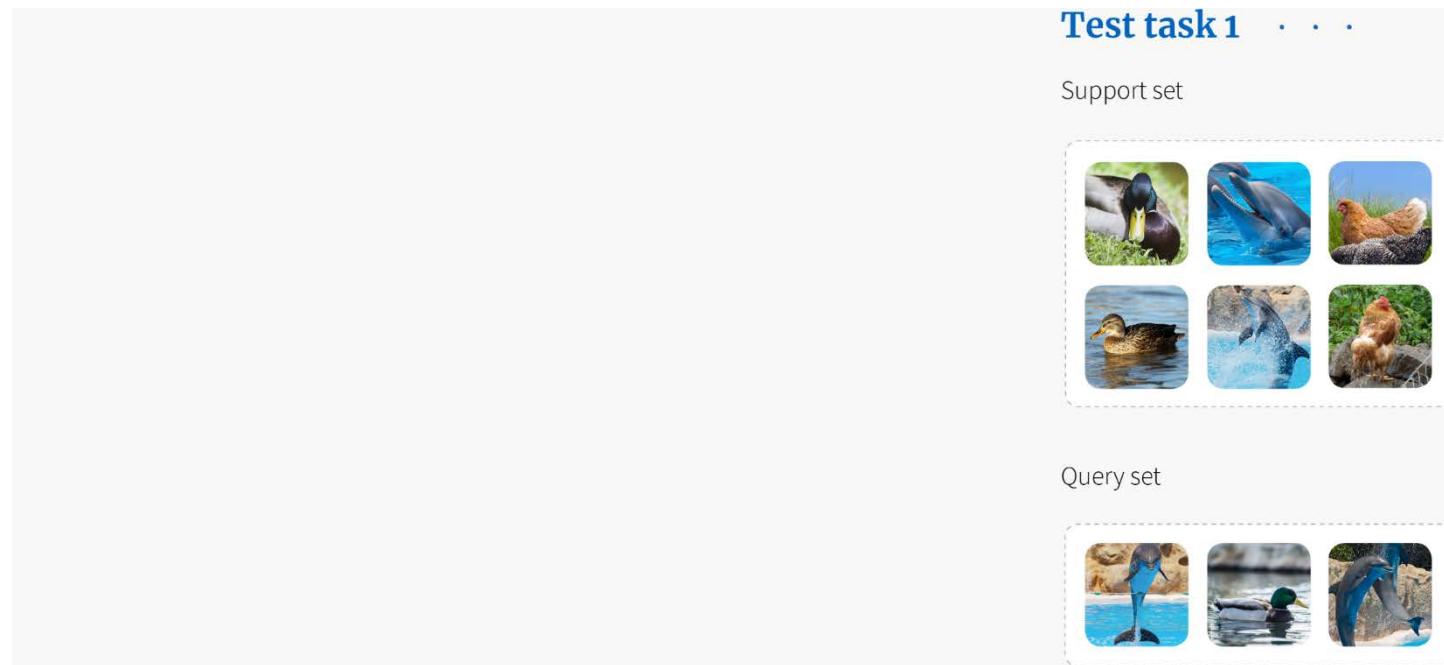
- **N-way K-shot** task: A machine learning task deals with **N classes**, each of which has **K data examples**
- An example of 3-way 2-shot classification:



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

Idea of meta learning

- Can we explicitly learn priors from previous experience that lead to efficient downstream learning?
 - Can we learn to learn?



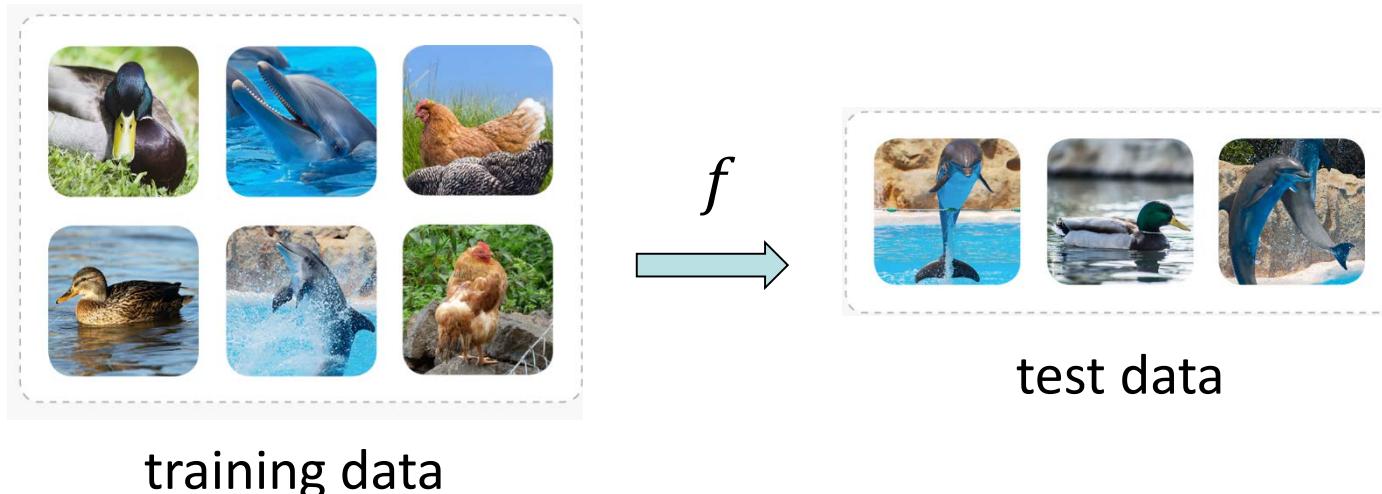
<https://www.borealisai.com/en/blog/tutorial-2-few-shot-learning-and-meta-learning-i/>

Meta learning

- Meta learning implements the **learning-to-learn** strategy
 - It learns some transferrable knowledge from a set of auxiliary tasks, which helps learn a target task with a few training data quickly and without suffering from overfitting

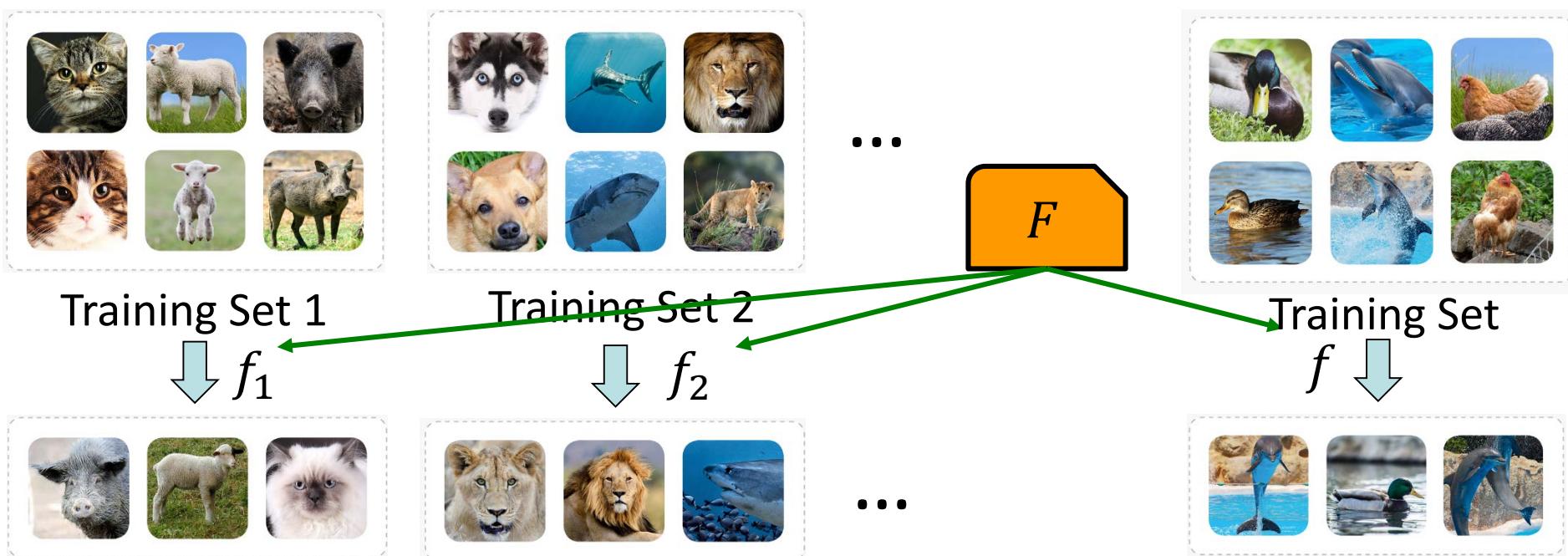
Meta learning vs. Supervised learning

- Supervised learning
 - Given a set of training data $\{(\mathbf{x}_i \in X, \mathbf{y}_i \in Y)\}_{i=1}^N$
 - We learn a model $f: X \rightarrow Y$
 - So that this model generalizes well to another set of test data
 - Limited performance if no sufficient training data are available



Meta learning vs. Supervised learning

- Given a set of learning tasks
- Derive a **meta learner** F that outputs a model f_i for each **training** task i
- So that each f_i works well on the respective test set
- The meta learner F can then be applied to a new **test** task



Support set and Query set in a task

Training Task 1



Training Set 1

Training Task 2



Training Set 2

...

Testing Task



Training Set

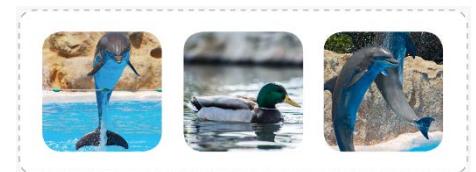


Test Set 1



Test Set 2

...



Test Set

Support set and Query set in a task

Training Task 1



Support Set 1

Training Task 2



Support Set 2

Testing Task



Support Set



Query Set 1



Query Set 2

...

...



Query Set

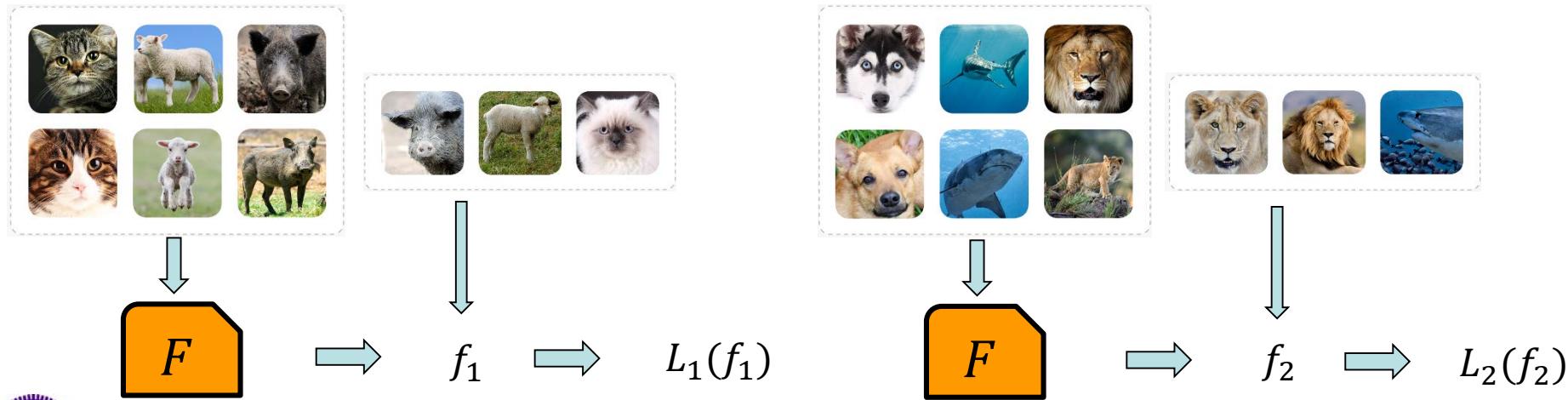


Meta training/Meta-Train/Meta-training

- Goal of **meta training** is to derive a meta learner F
- Training objective function

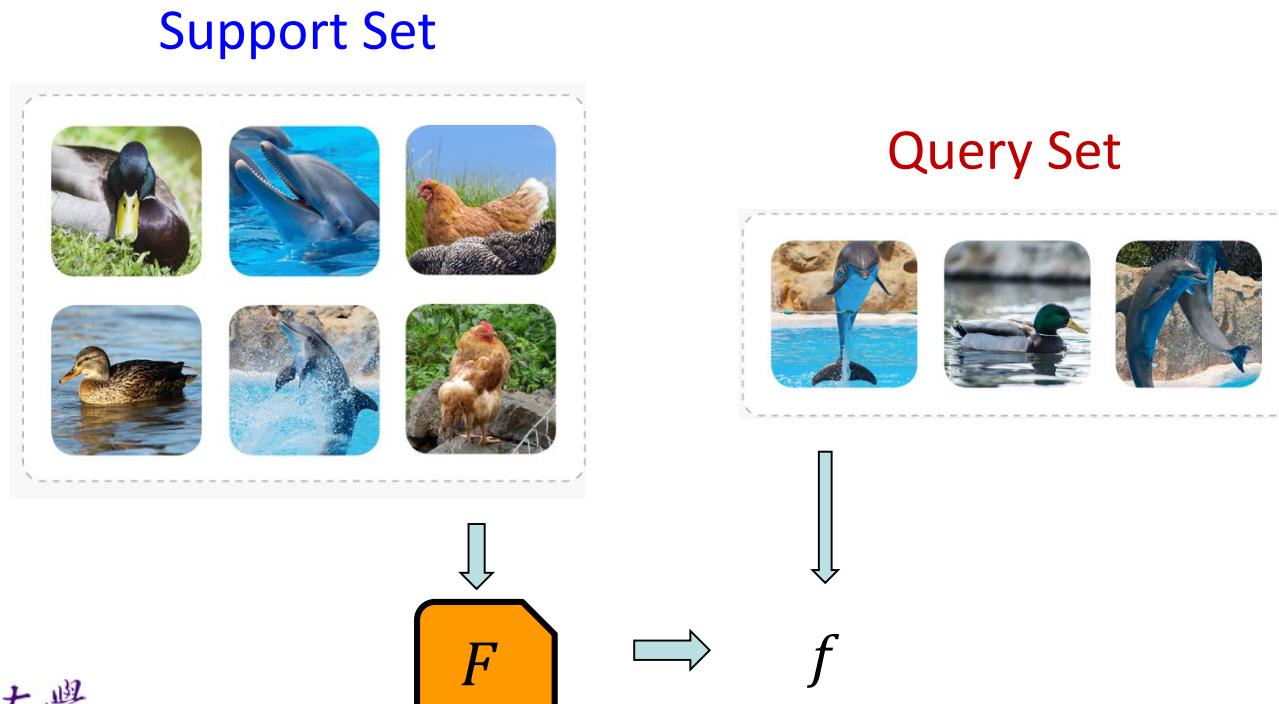
$$L(F) = \sum_{t=1}^T L_t(f_t)$$

- $L(F)$: Loss of F
- $L_t(f_t)$: Loss of f_t on training task t



Meta testing/Meta-Test/Meta testing

- During **meta testing**, the meta learner F is applied to the support set of a new testing task and outputs a model f
- The meta learner F is evaluated based on the performance of f on the query set of this task



Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

MAML (Model-Agnostic Meta-Learning)

[Finn et al. ICML 2017]

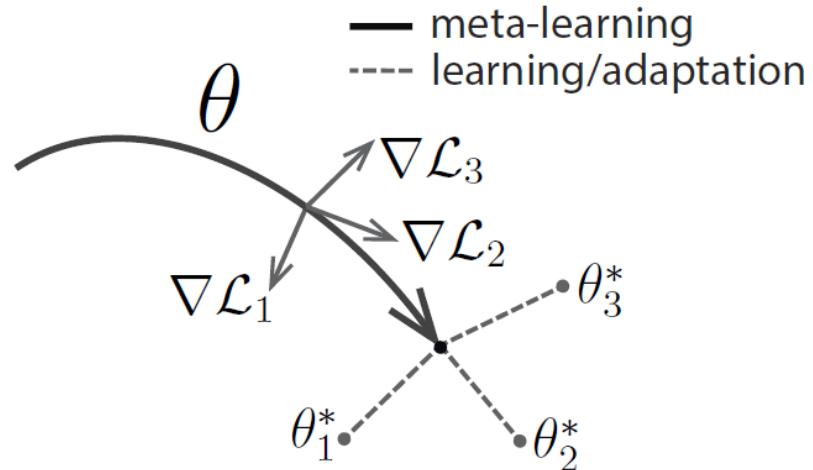
- MAML is a meta-learning algorithm for fast deep network adaptation
- Goal: Quickly adapt to new tasks with only **a small amount of data** and with only **a few gradient steps**, even **one gradient step**
- **Meta learner:** It learns a **generalized parameter initialization** of a model
- **Learner:** It learns a new task by using **a single gradient step**

Characteristics of MAML

- The MAML learner's weights are **updated using the gradient**
 - Not need additional parameters nor require a particular learner architecture
- **Fast adaptability** through good parameter initialization
 - It explicitly optimizes to learn an internal representation for all training tasks, i.e., representation suitable for these tasks
- **Model-agnostic**
 - The model should be parameterized. No other assumptions.
- **Task-agnostic**
 - Classification, regression, reinforcement learning, ...

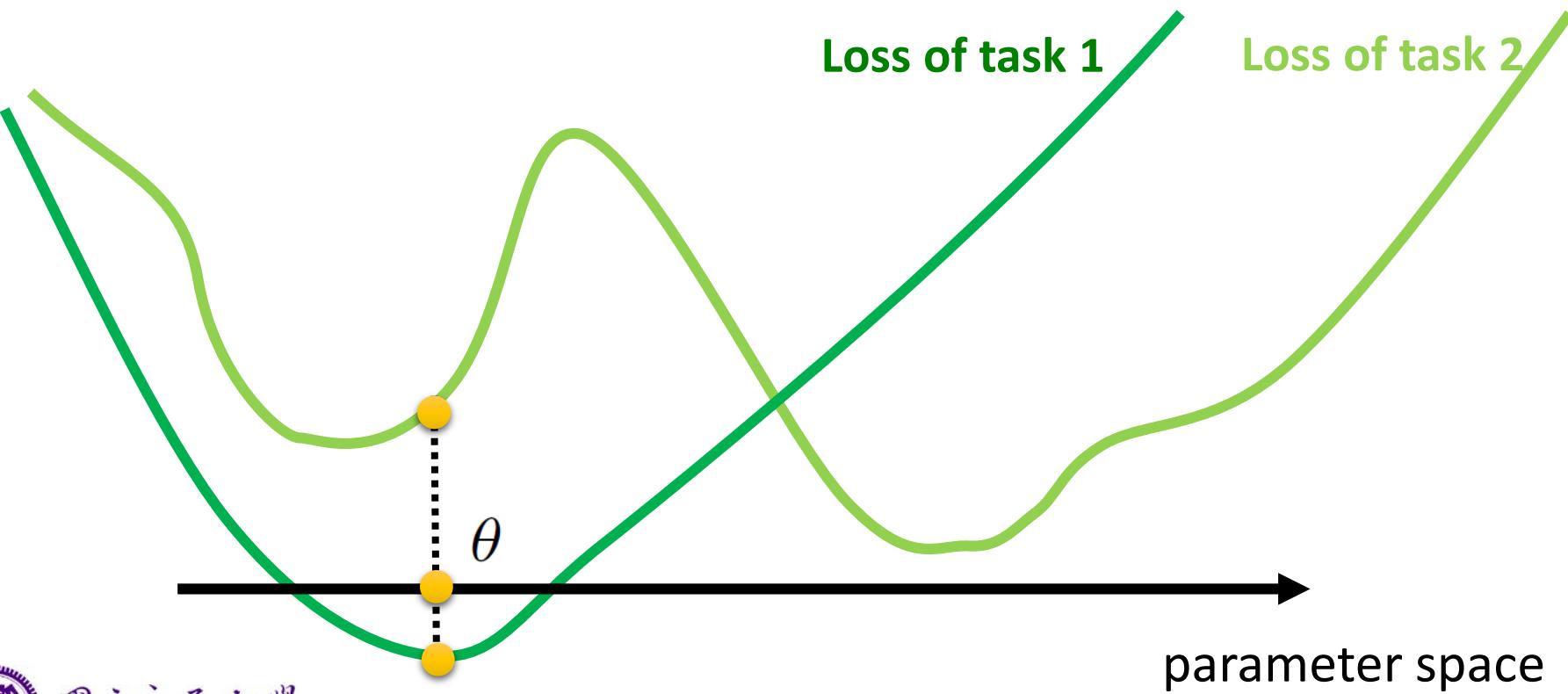
Idea of MAML

- Some internal representations are **more transferrable than others**
- f_θ : deep network model parametrized by θ
- Desire a model parameter set θ that:
 - Applying one (or a few) gradient step to θ on a new task produces maximally effective behavior
- Namely, θ commonly decreases loss of each task **after adaptation**



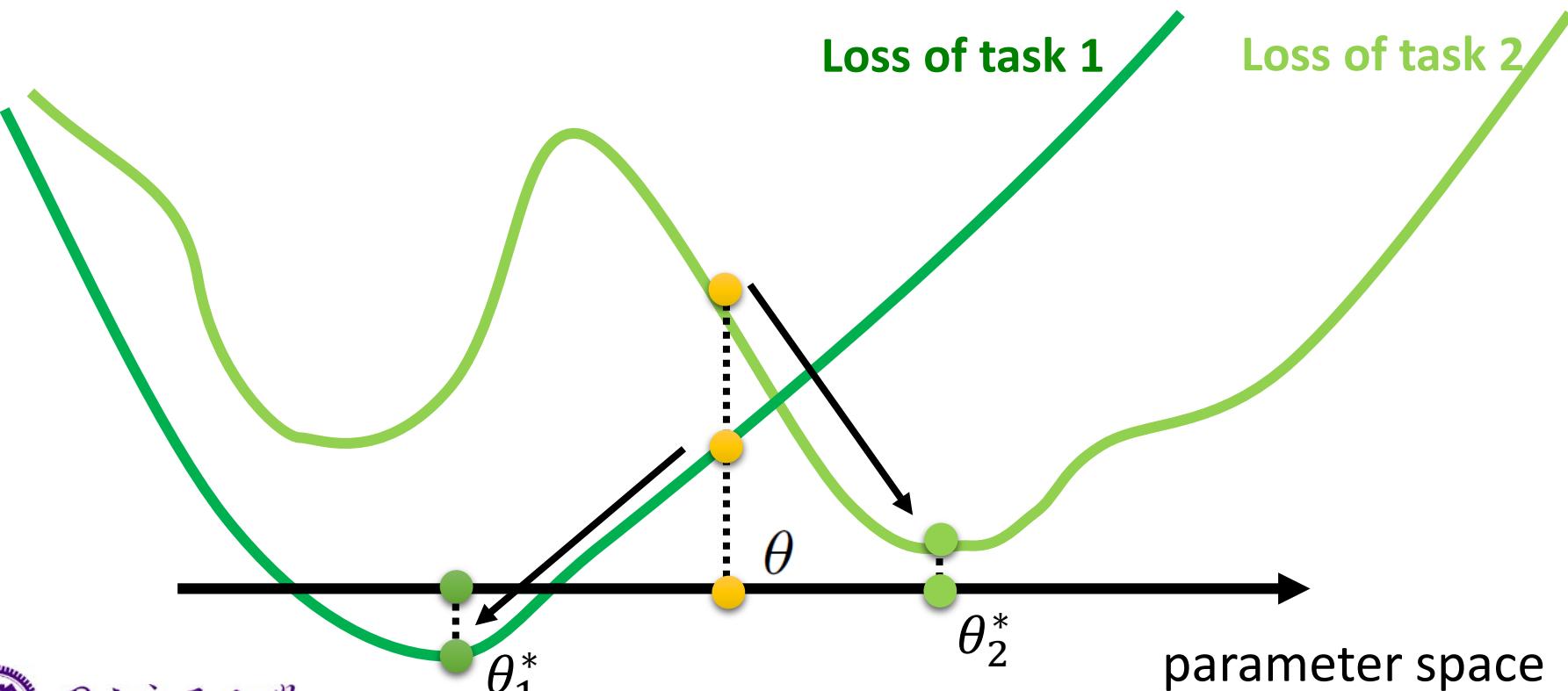
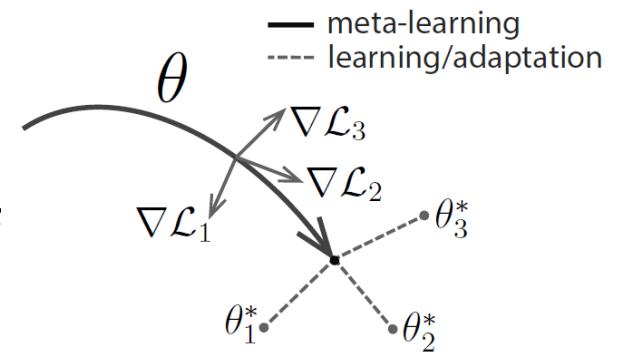
Good initialization of θ

- Option 1: θ that minimizes the losses of all tasks
 - Model pre-training



Good initialization of θ

- Option 1: θ that minimizes the losses of
 - Model pre-training
- Option 2: θ that minimizes the losses of all tasks **after adaptation**
 - MAML adopts



Supervised meta learning

- Notations
 - Model: f_θ
 - Task distribution: $p(\mathcal{T})$
- For each task: $T = \{L(x_1, y_1, \dots, x_K, y_K), (x, y) \sim q\}$
 - Data distribution: q (K samples are drawn from q)
 - Loss function: L
 - ◆ MSE (mean squared error) for regression
 - ◆ Cross entropy for classification
 - ◆ Any other **differentiable** loss functions can be used

MAML Algorithm

- Sampling training tasks
- Compute the gradient based on the support set and update parameters for each sampled task
- Update the initial parameters based on the query set

Algorithm 2 MAML for Few-Shot Supervised Learning

Require: $p(\mathcal{T})$: distribution over tasks

Require: α, β : step size hyperparameters

- 1: randomly initialize θ
- 2: **while** not done **do**
- 3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
- 4: **for all** \mathcal{T}_i **do**
- 5: Sample K datapoints $\mathcal{D} = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from \mathcal{T}_i
- 6: Evaluate $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$ using \mathcal{D} and $\mathcal{L}_{\mathcal{T}_i}$
- 7: Compute adapted parameters with gradient descent:
$$\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta})$$
- 8: Sample datapoints $\mathcal{D}'_i = \{\mathbf{x}^{(j)}, \mathbf{y}^{(j)}\}$ from \mathcal{T}_i for the meta-update
- 9: **end for**
- 10: Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})$ using each \mathcal{D}'_i and $\mathcal{L}_{\mathcal{T}_i}$
- 11: **end while**

1st order approximation

- The update rule

$$\begin{aligned}\theta &\leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) \\&= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}) \\&= \theta - \beta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} (\nabla_{\theta} \theta'_i) \nabla_{\theta'_i} \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i})\end{aligned}$$

Hessian matrix is required.

MAML suggests 1st order approximation:
Consider it as an identity matrix

Experiments on classification: Omniglot dataset

- The Omniglot dataset
 - 1623 characters (tasks)
 - 20 instances for each character were drawn by 20 people
 - 1200 for training, 423 for test.



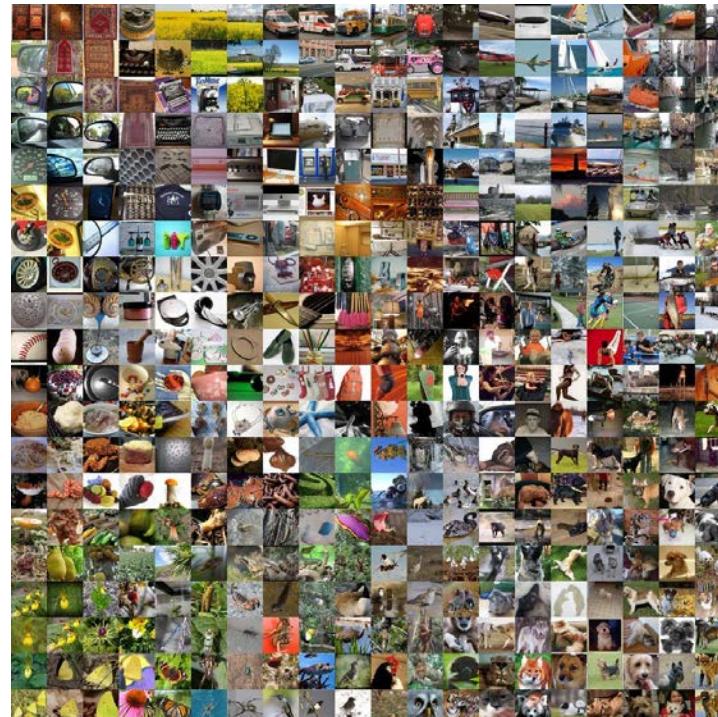
Experiments on classification: Omniglot dataset

- MAML outperforms the state-of-the-arts

	5-way Accuracy		20-way Accuracy	
	1-shot	5-shot	1-shot	5-shot
Omniglot (Lake et al., 2011)				
MANN, no conv (Santoro et al., 2016)	82.8%	94.9%	–	–
MAML, no conv (ours)	$89.7 \pm 1.1\%$	$97.5 \pm 0.6\%$	–	–
Siamese nets (Koch, 2015)	97.3%	98.4%	88.2%	97.0%
matching nets (Vinyals et al., 2016)	98.1%	98.9%	93.8%	98.5%
neural statistician (Edwards & Storkey, 2017)	98.1%	99.5%	93.2%	98.1%
memory mod. (Kaiser et al., 2017)	98.4%	99.6%	95.0%	98.6%
MAML (ours)	$98.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$

Experiments on classification: Mini-ImageNet dataset

- Mini-Imagenet (Ravi & Larochelle, 2017)
 - 100 classes sampled from the ImageNet dataset
 - 64 classes for training, 12 classes for validation, and 24 classes fro testing



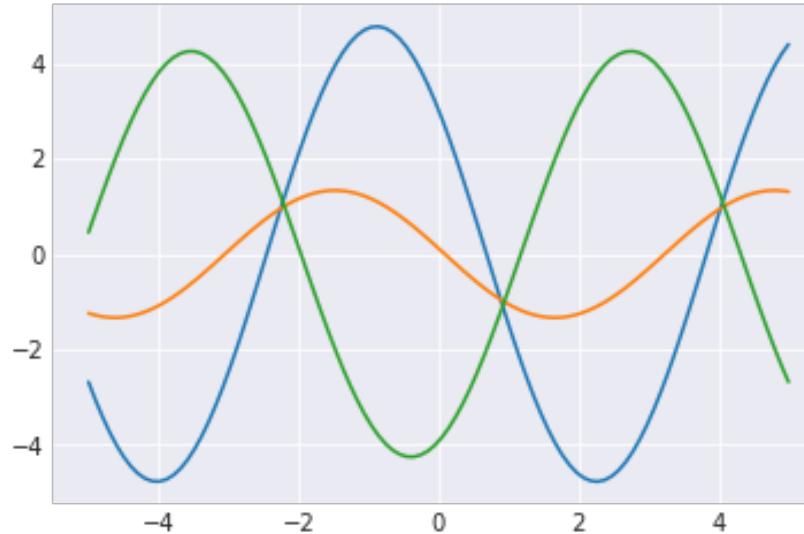
Experiments on classification: Mini-ImageNet dataset

- MAML outperforms the state-of-the-arts
- 1st order approximation does not hurt the performance, but saves the computation cost

MiniImagenet (Ravi & Larochelle, 2017)	5-way Accuracy	
	1-shot	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)	$48.07 \pm 1.75\%$	$63.15 \pm 0.91\%$
MAML (ours)	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$

Experiments on regression

- A regression task
 - Given a target sine function $y = a \sin(x + b)$
 - ◆ Amplitude a and phase b are varied between tasks
 - ◆ a in $[0.1, 0.5]$ and b in $[0, \pi]$
 - Sample K points from the target function
 - Use the samples to estimate the target function

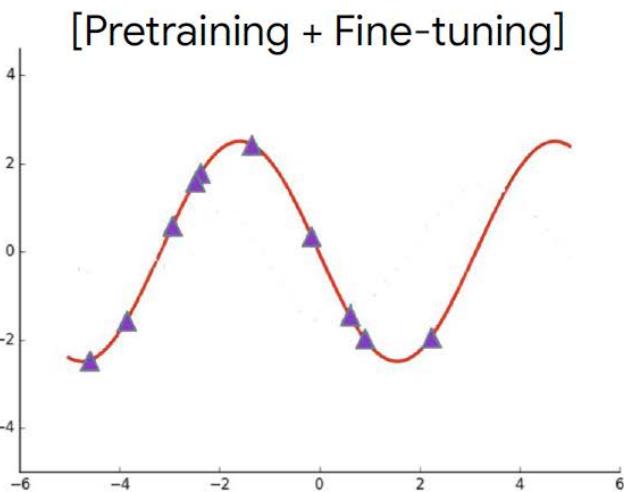
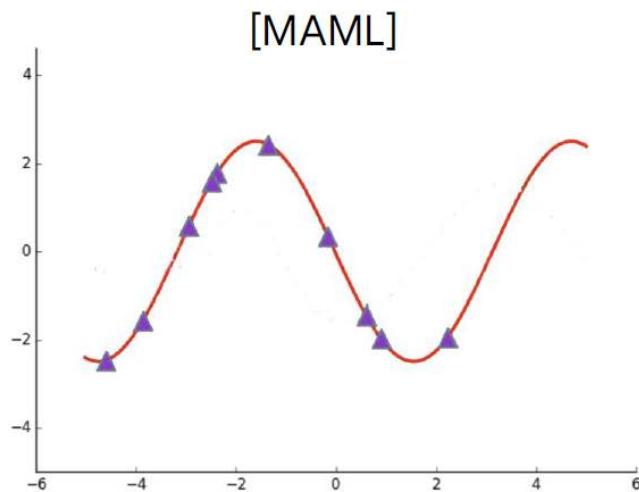


Experiments on regression

- Loss function: Mean Squared Error (MSE)
- Regressor: 2 hidden layers with 40 units and ReLU
- Training
 - Use only 1 gradient step for learner
 - $K=5$ or 10 example (5-shot learning or 10-shot learning)

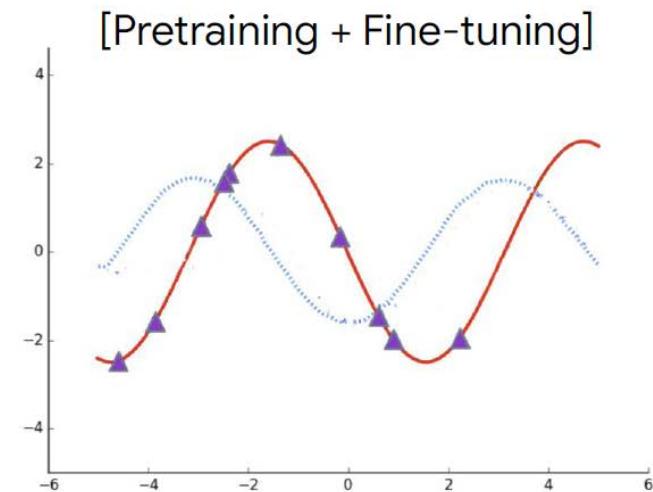
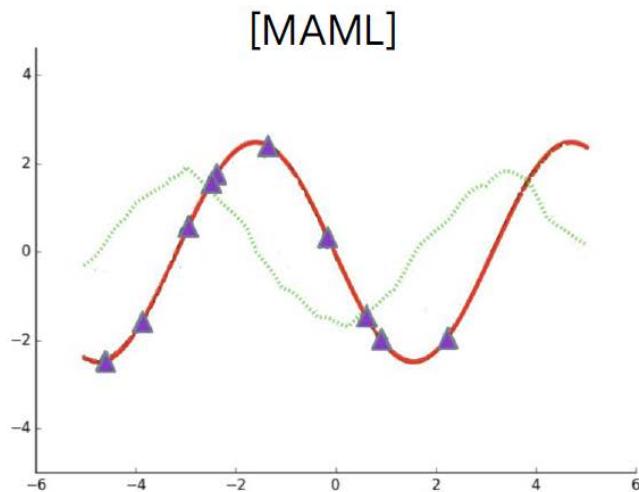
Experiments on regression

- 10 points and ground truth



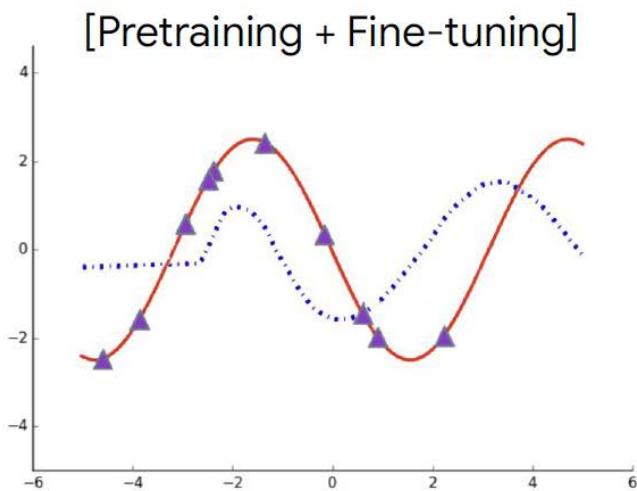
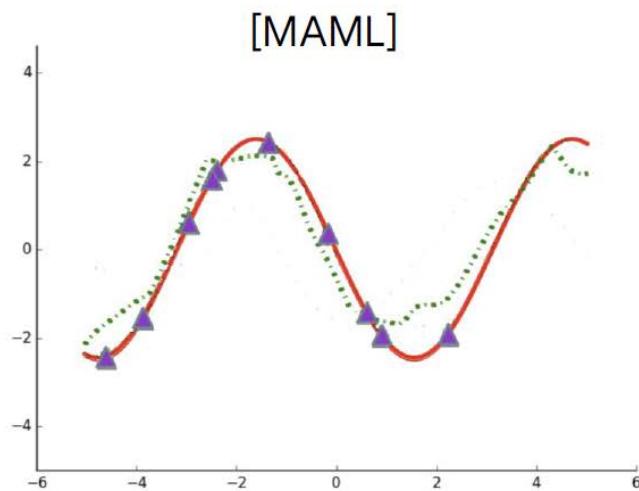
Experiments on regression

- Initialization by using MAML vs. Pre-training



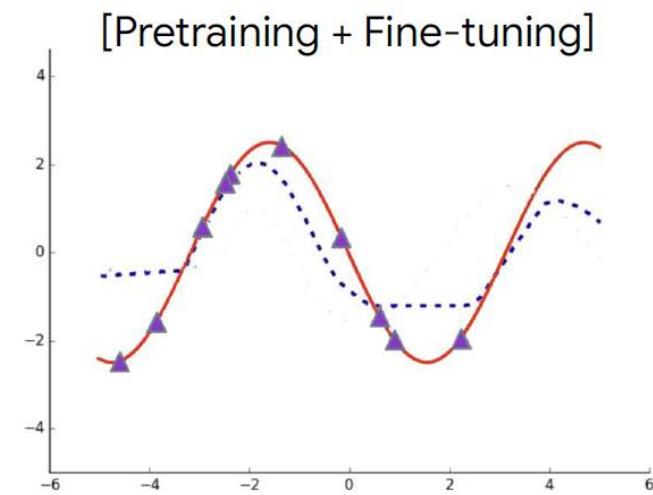
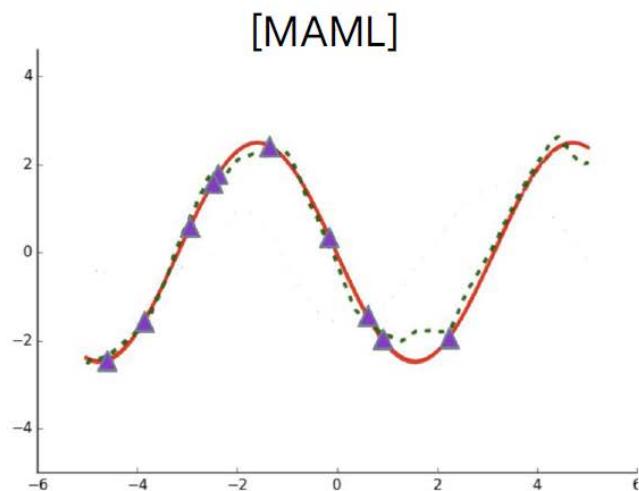
Experiments on regression

- After one gradient step update



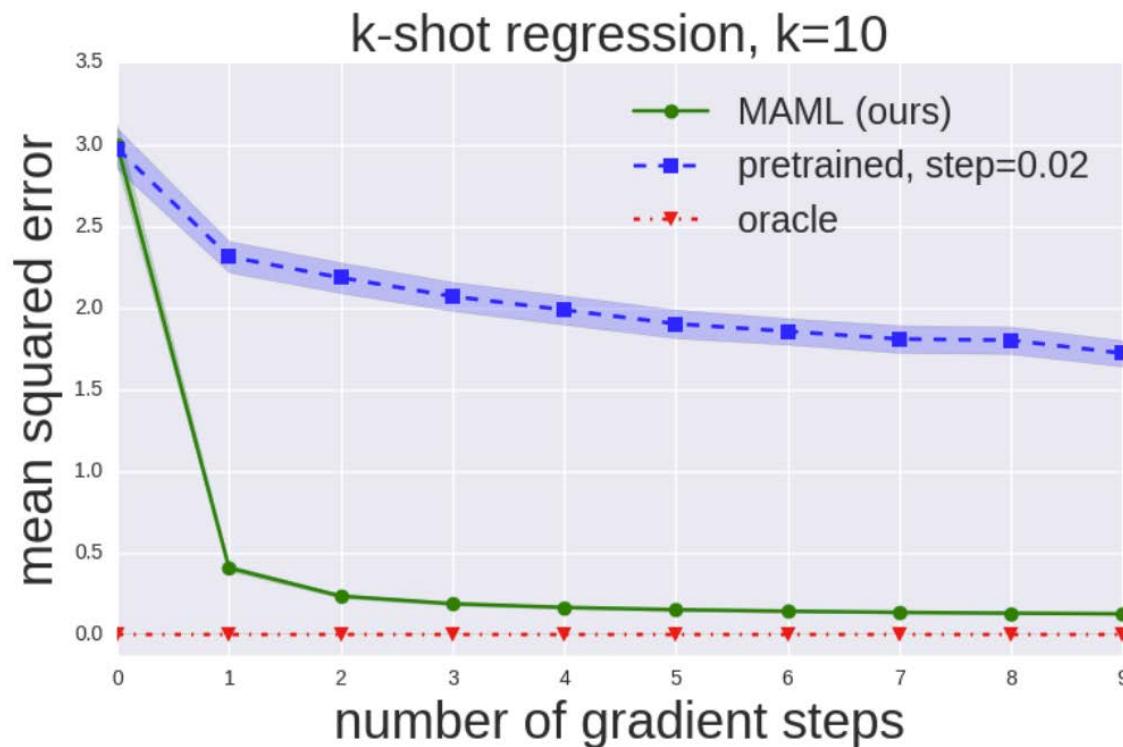
Experiments on regression

- After 10 gradient step update



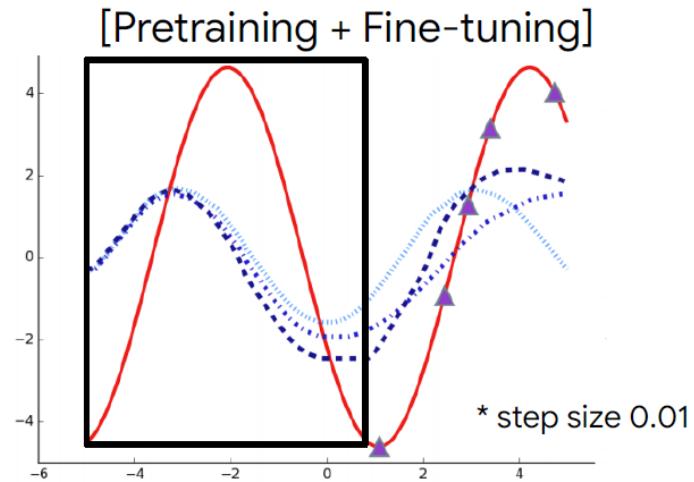
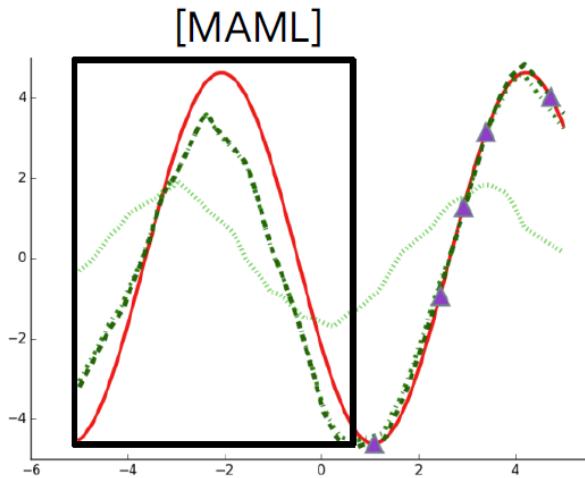
Experiments on regression

- Quantitative results



Experiments on regression

- 5-shot case
- Good predictions are also made for ranges not particularly seen



Conclusions

- Simple: MAML learns easily adaptable model parameters via gradient descent
- Compact: MAML does not introduce any learned parameters for meta learning
- General: MAML is amenable to gradient-based models, any differentiable objective functions, and various tasks

Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

LSTM-based meta-learner

[Ravi and Larochelle, ICLR 2017]

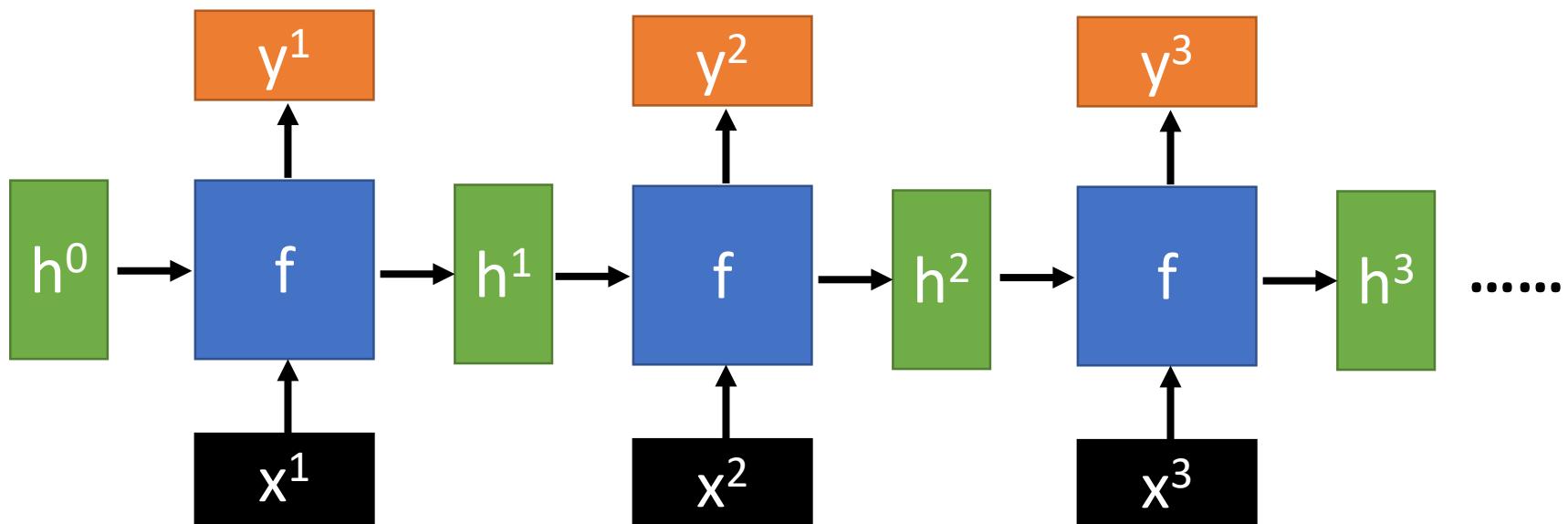
- Deep neural networks show great success in large data domains, and generally perform poorly on few-shot tasks
- This paper presents an **LSTM-based meta-learner**, which is optimized to train **another learner neural network classifier** in the few-shot regime
- The LSTM-based meta learner captures
 - Short-term knowledge within a task
 - Long-term knowledge common among all tasks
- The meta-learner model learns a task-common **initialization** for the learner classifier



Recurrent neural networks

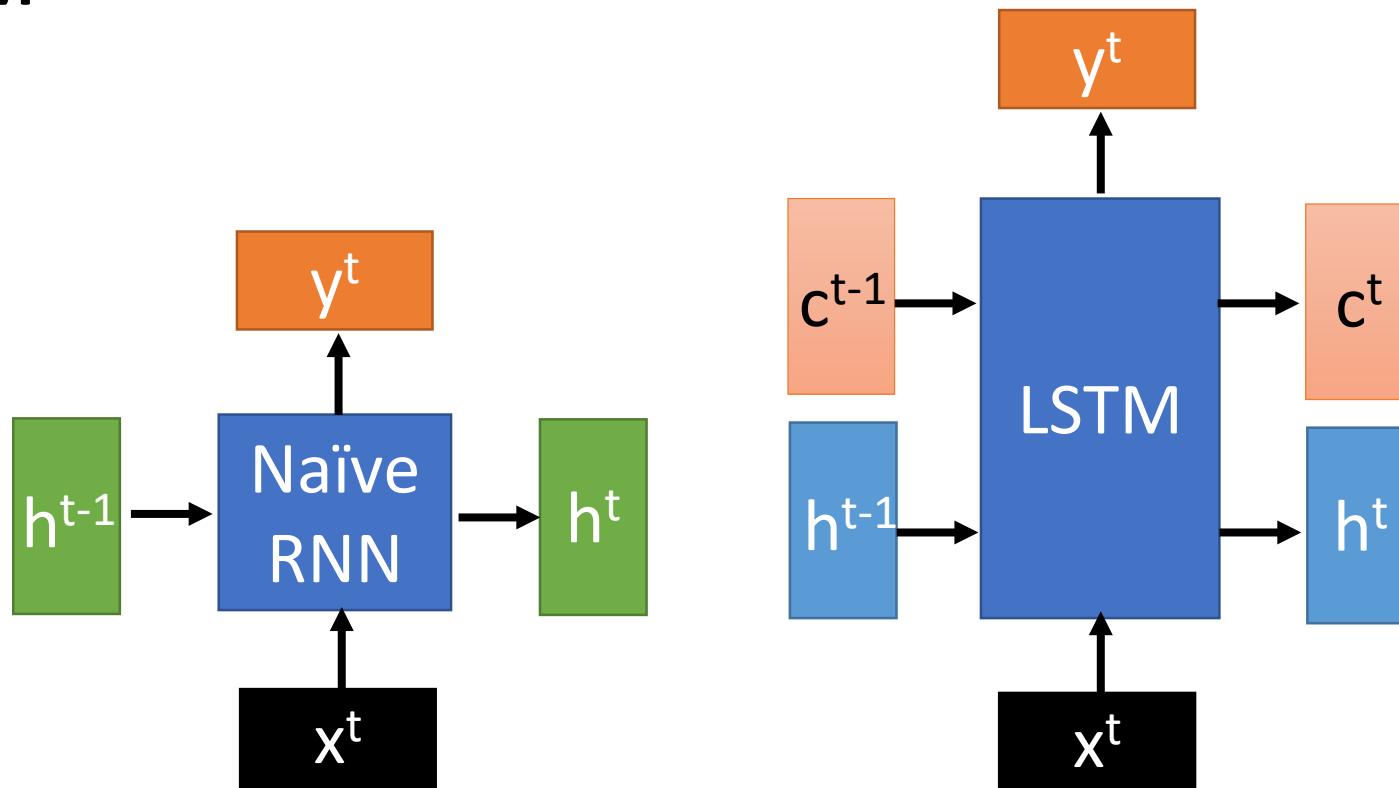
- Given function $f: (h', y) = f(h, x)$

h and h' are vectors with the same dimension



No matter how long the input/output sequence is,
we only need one function f

LSTM

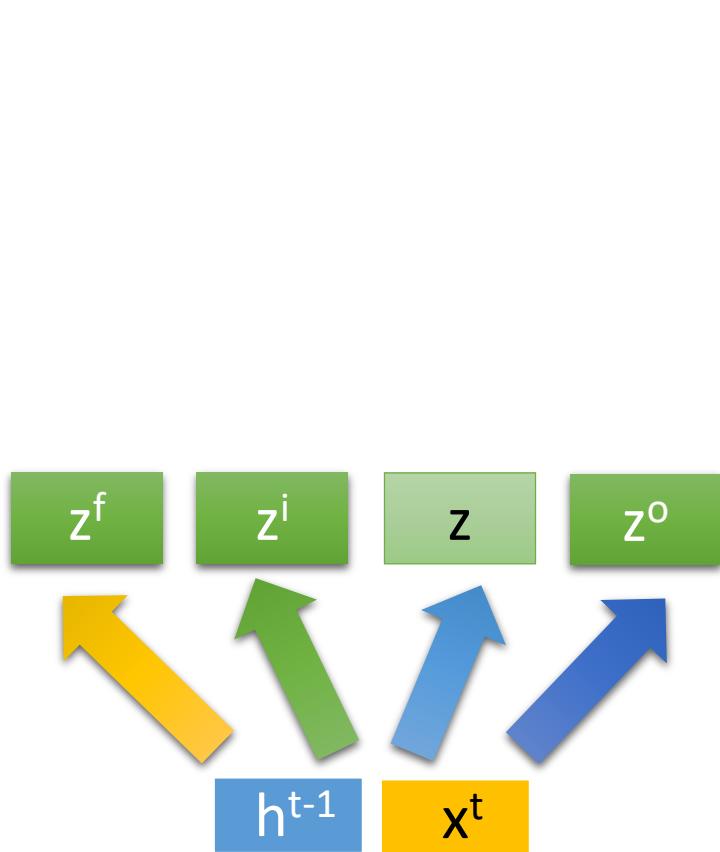


c changes slowly $\rightarrow c^t$ is c^{t-1} added by something

h changes faster $\rightarrow h^t$ and h^{t-1} can be very different



LSTM



$$z = \tanh(W h^{t-1} + x^t)$$

input

$$z^i = \sigma(W^i h^{t-1} + x^t)$$

forget

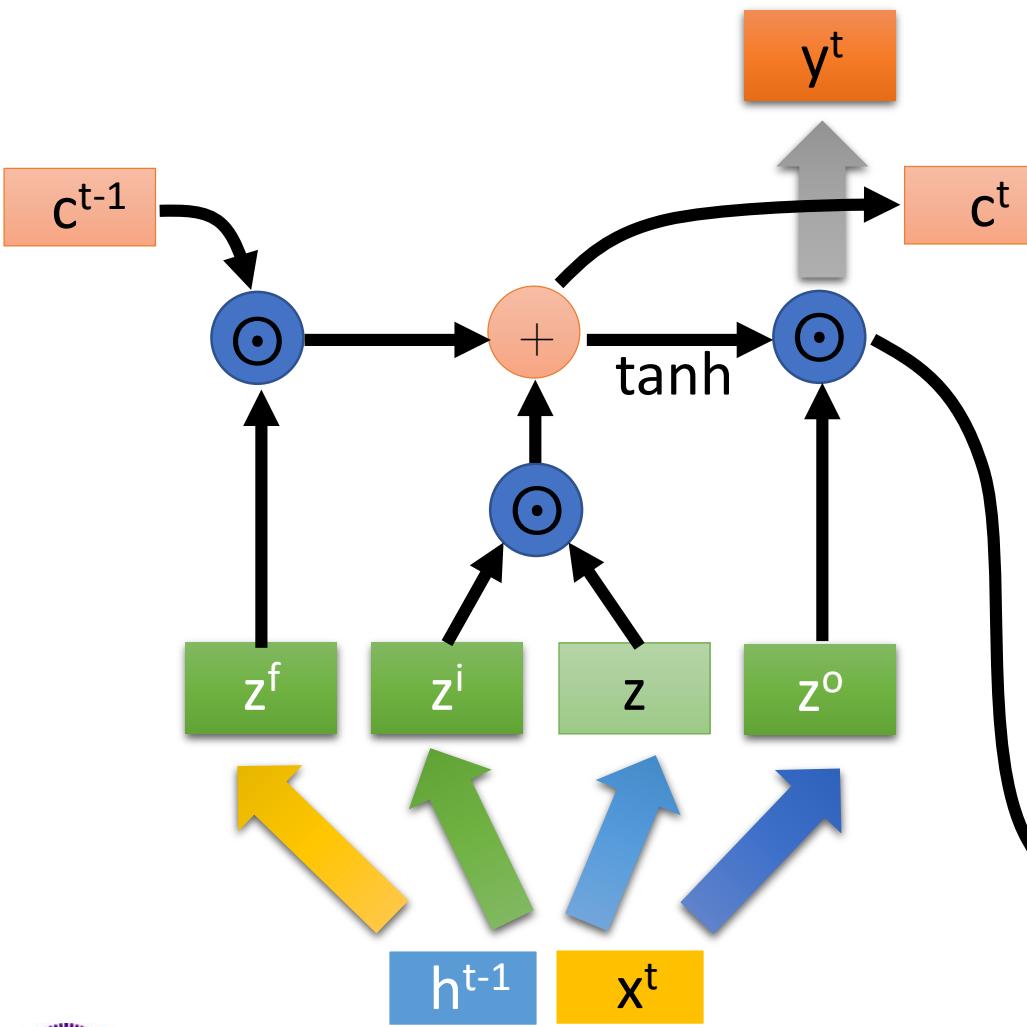
$$z^f = \sigma(W^f h^{t-1} + x^t)$$

output

$$z^o = \sigma(W^o h^{t-1} + x^t)$$



LSTM



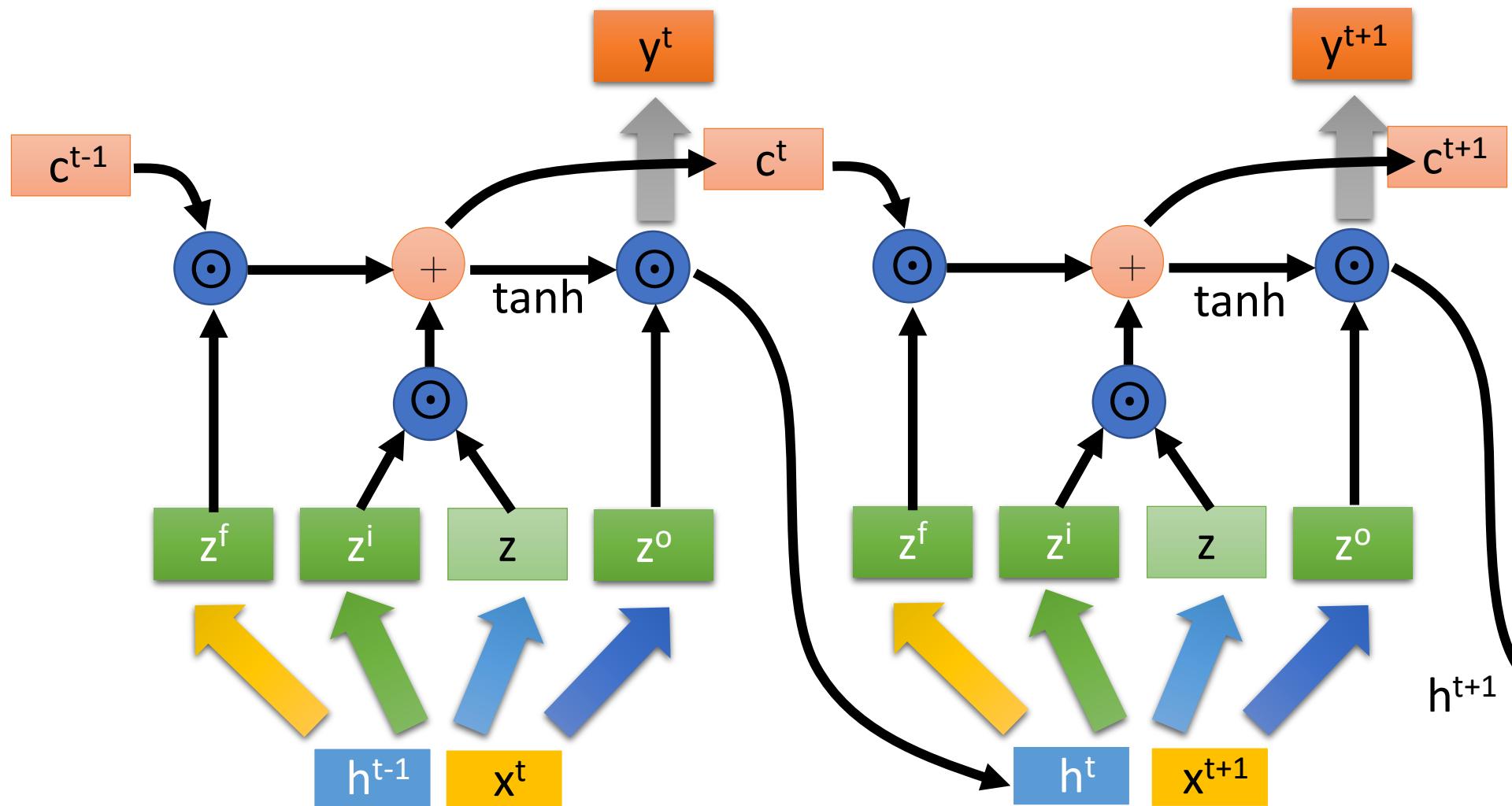
$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot \tanh(c^t)$$

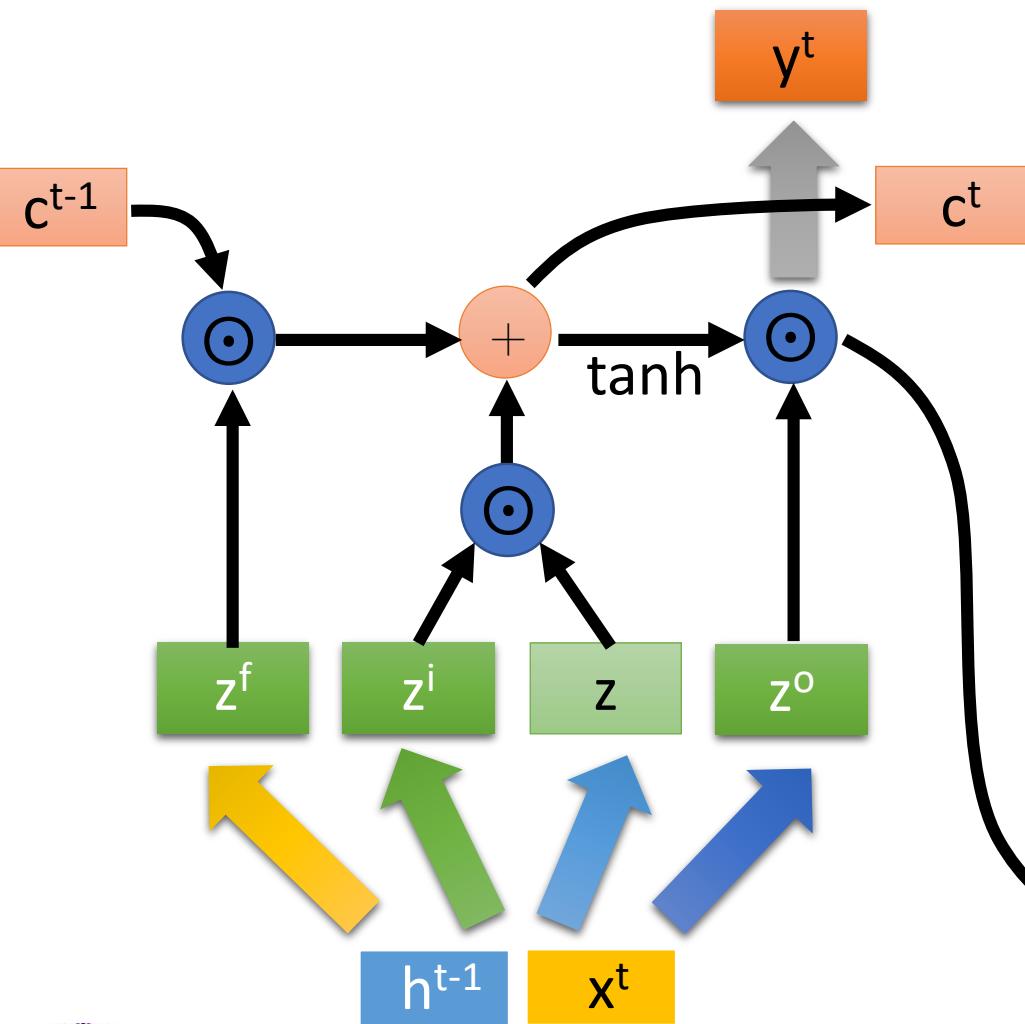
$$y^t = \sigma(W' h^t)$$



LSTM



LSTM



$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot \tanh(c^t)$$

$$y^t = \sigma(W' h^t)$$



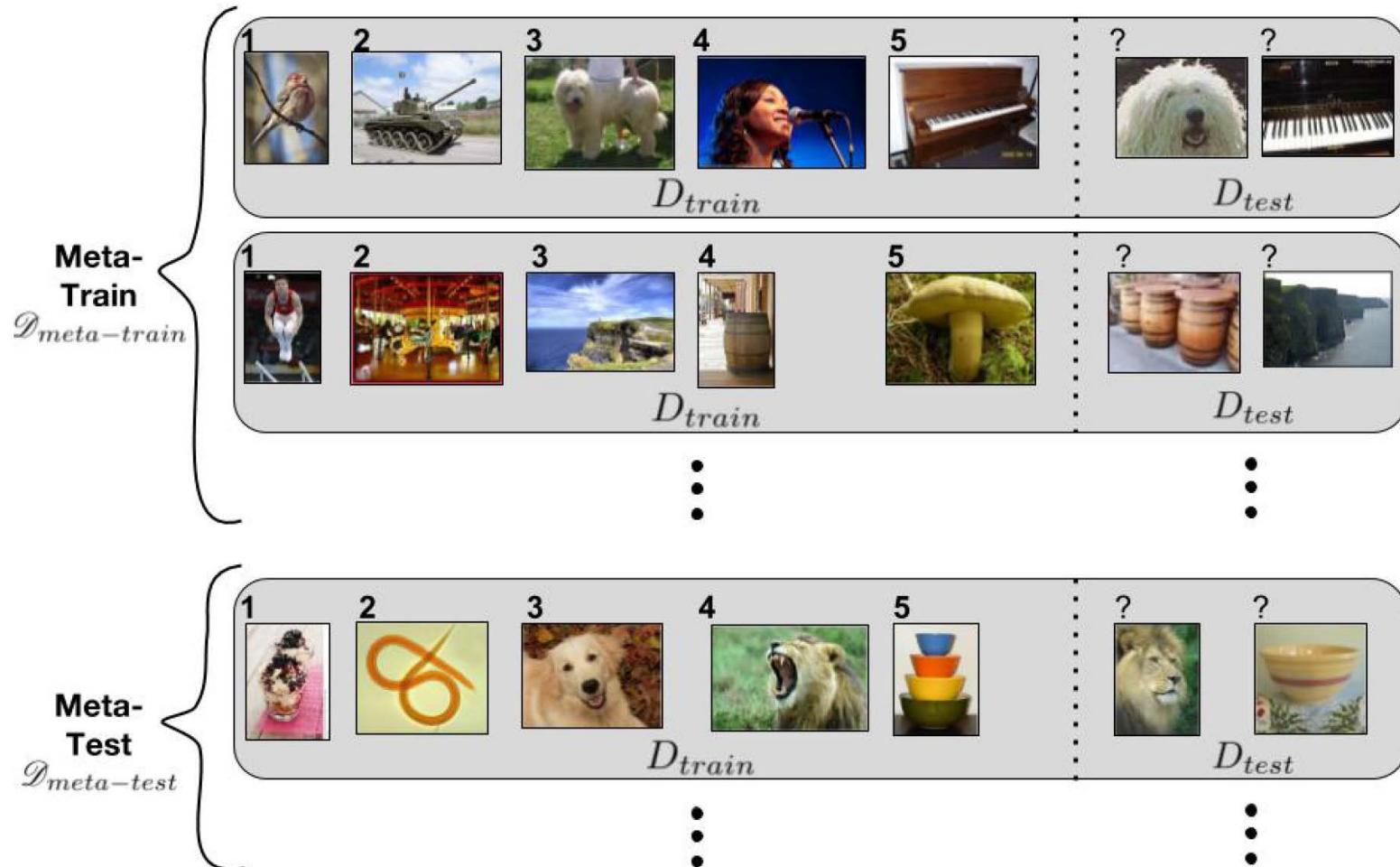
Problem statement

- Regular machine learning task
 - Given a dataset D , we split it into two disjoint sets D_{train} and D_{test}
 - We optimize a model with parameters θ on the training set D_{train}
 - And evaluate its generalization on the test set D_{test}
- In meta learning, we are dealing with a **meta-set** \mathcal{D} containing multiple regular datasets
 - For each dataset $D \in \mathcal{D}$, it has a split of D_{train} and D_{test}
 - Consider a k -shot, C -way classification task for each D . D_{train} has kC training examples while D_{test} has a number of examples for evaluation

Problem statement

- We have different meta-sets $\mathcal{D}_{meta-train}$, $\mathcal{D}_{meta-validation}$, and $\mathcal{D}_{meta-test}$ for meta-training, meta-validation, and meta-testing respectively
 - **Meta-training:** Produce a learning procedure (meta-learner), which takes D_{train} as input while outputs a classifier (learner) that has a good generalization on D_{test}
 - **Meta-validation:** Perform hyper-parameter selection of the meta-learner
 - **Meta-testing:** Evaluate the performance of the meta-learner on $\mathcal{D}_{meta-test}$

Problem statement



Observation and idea

- For $D \in \mathcal{D}$, the learner, a neural net classifier, with parameters θ is optimized by gradient descent, i.e.,

$$\theta^t = \theta^{t-1} - \eta \nabla_{\theta} l$$

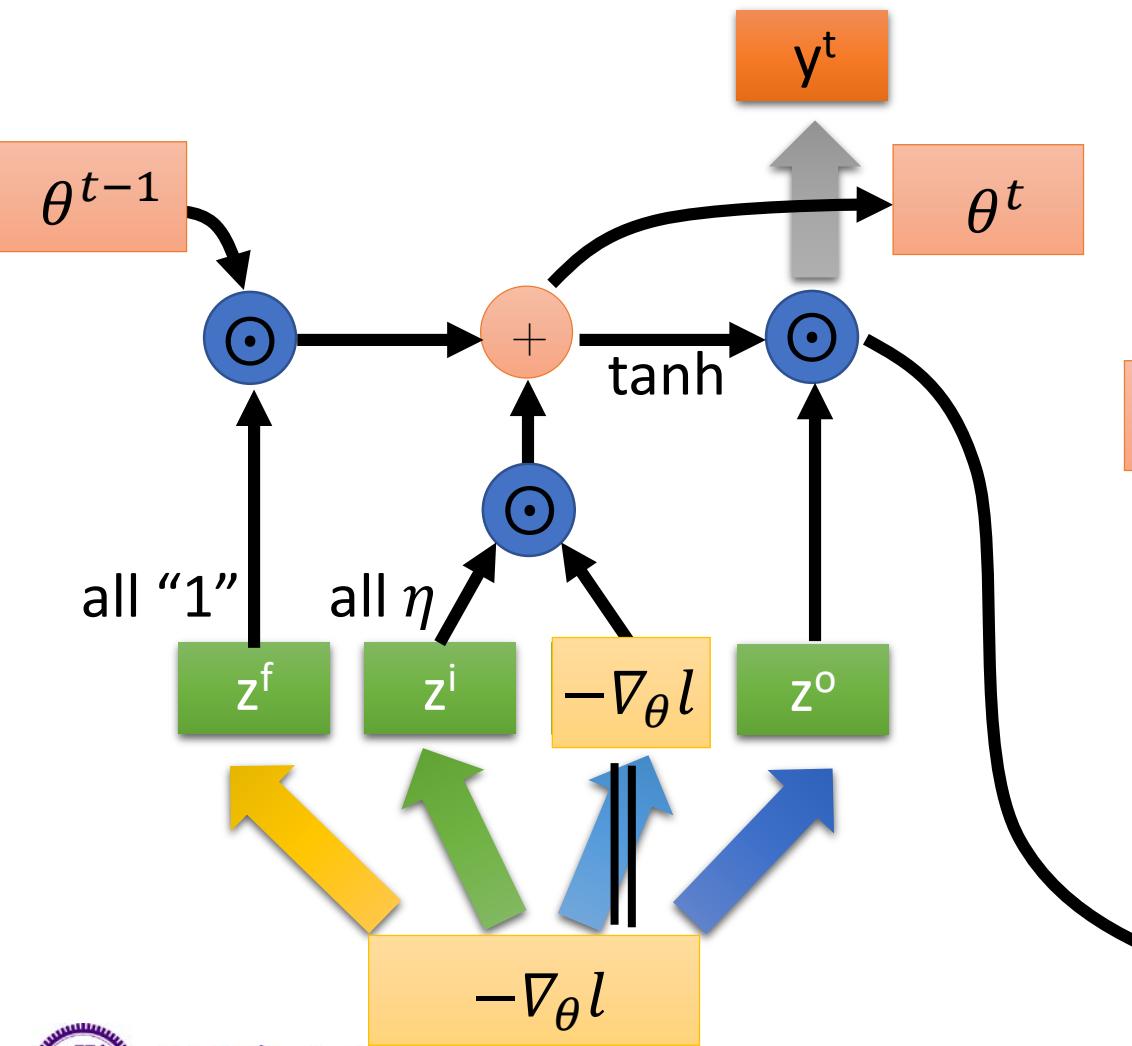
- The update above is similar the cell state in an LSTM

$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

- $\theta^t \rightarrow c^t$
- $\theta^{t-1} \rightarrow c^{t-1}$
- $\eta \rightarrow z^i$
- $-\eta \nabla_{\theta} l \rightarrow z$

LSTM as meta learner

$$\theta^t = \theta^{t-1} - \eta \nabla_{\theta} l$$



$$\theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_{\theta} l$$

$$\begin{bmatrix} 1 \\ 1 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \eta \\ \eta \\ \vdots \end{bmatrix}$$

$$h^t = z^o \odot \tanh(c^t)$$

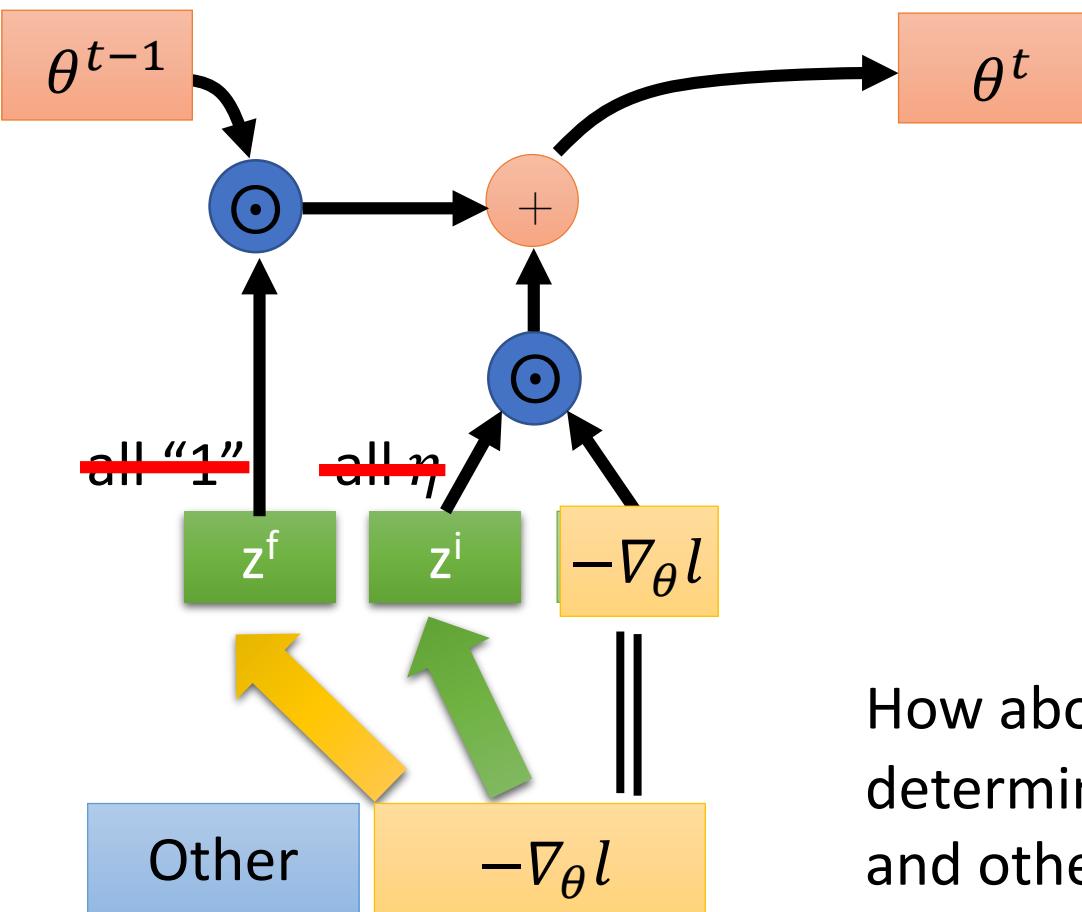
$$y^t = \sigma(W' h^t)$$

h^t



LSTM as meta learner

$$\theta^t = \theta^{t-1} - \eta \nabla_{\theta} l$$



Something like regularization

$$\theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_{\theta} l$$

Dynamic learning rate

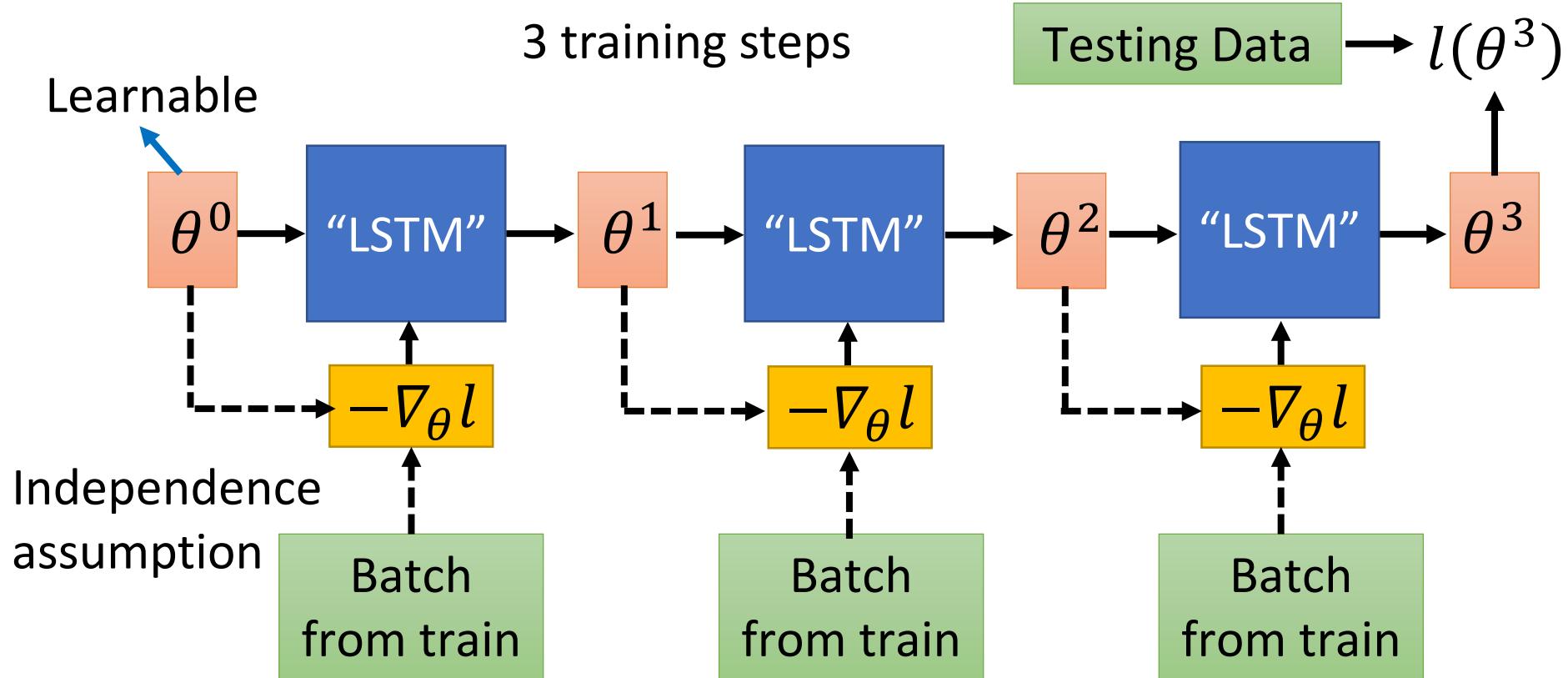
How about machine learn to determine z^f and z^i from $-\nabla_{\theta} l$ and other information?



Optimization

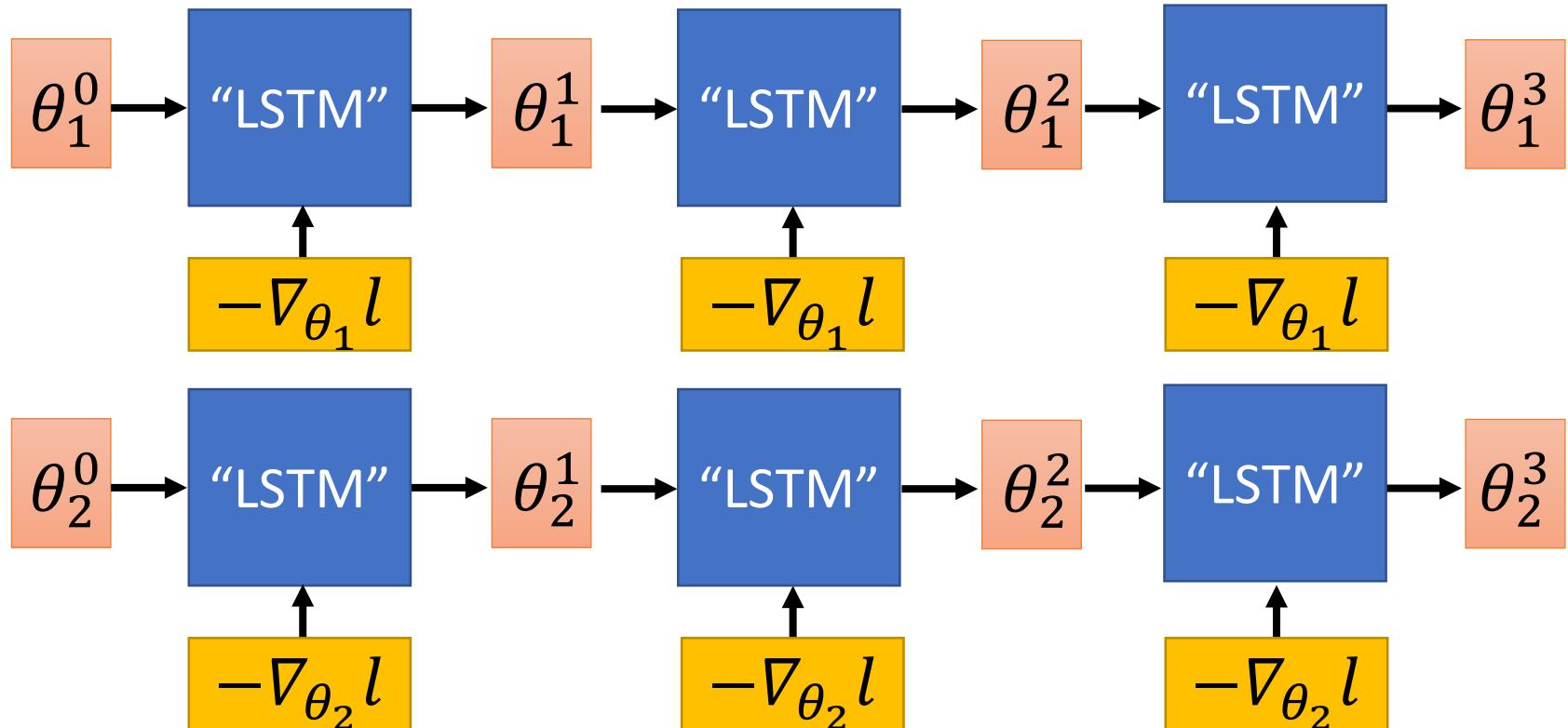
$$\theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_{\theta} l$$

Learn to minimize



Implementation

- The LSTM used only has one cell. Share across all parameters



Algorithm 1 Train Meta-Learner

Input: Meta-training set $\mathcal{D}_{meta-train}$, Learner M with parameters θ , Meta-Learner R with parameters Θ .

```
1:  $\Theta_0 \leftarrow$  random initialization
2:
3: for  $d = 1, n$  do
4:    $D_{train}, D_{test} \leftarrow$  random dataset from  $\mathcal{D}_{meta-train}$ 
5:    $\theta_0 \leftarrow c_0$                                  $\triangleright$  Initialize learner parameters
6:
7:   for  $t = 1, T$  do
8:      $\mathbf{X}_t, \mathbf{Y}_t \leftarrow$  random batch from  $D_{train}$ 
9:      $\mathcal{L}_t \leftarrow \mathcal{L}(M(\mathbf{X}_t; \theta_{t-1}), \mathbf{Y}_t)$            $\triangleright$  Get loss of learner on train batch
10:     $c_t \leftarrow R((\nabla_{\theta_{t-1}} \mathcal{L}_t, \mathcal{L}_t); \Theta_{d-1})$        $\triangleright$  Get output of meta-learner using Equation 2
11:     $\theta_t \leftarrow c_t$                                           $\triangleright$  Update learner parameters
12:  end for
13:
14:   $\mathbf{X}, \mathbf{Y} \leftarrow D_{test}$ 
15:   $\mathcal{L}_{test} \leftarrow \mathcal{L}(M(\mathbf{X}; \theta_T), \mathbf{Y})$            $\triangleright$  Get loss of learner on test batch
16:  Update  $\Theta_d$  using  $\nabla_{\Theta_{d-1}} \mathcal{L}_{test}$                    $\triangleright$  Update meta-learner parameters
17:
18: end for
```

Experiments on minilmageNet

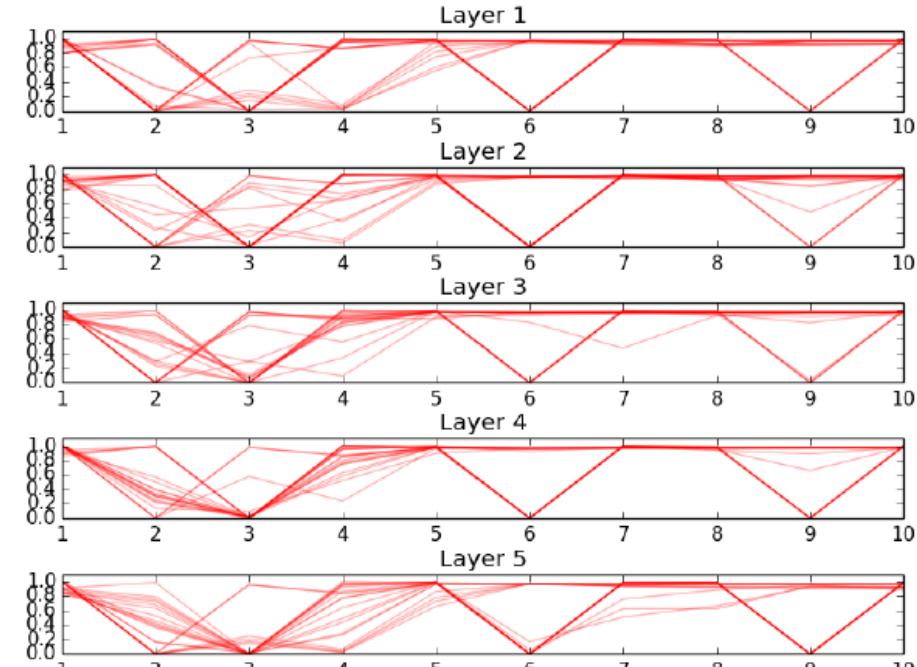
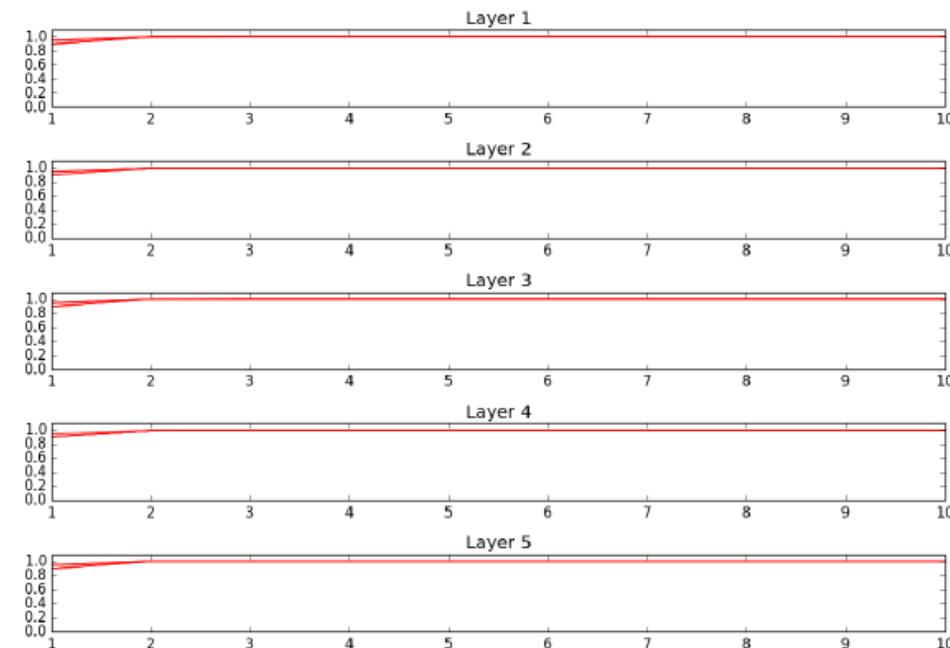
- The results

Model	5-class	
	1-shot	5-shot
Baseline-finetune	$28.86 \pm 0.54\%$	$49.79 \pm 0.79\%$
Baseline-nearest-neighbor	$41.08 \pm 0.70\%$	$51.04 \pm 0.65\%$
Matching Network	$43.40 \pm 0.78\%$	$51.09 \pm 0.71\%$
Matching Network FCE	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
Meta-Learner LSTM (OURS)	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$

Experiments on minilmageNet

- The values of the forget gate and the input gate

$$\theta^t = z^f \odot \theta^{t-1} + z^i \odot -\nabla_{\theta} l$$



(a) Forget gate values for 1-shot meta-learner

(b) Input gate values for 1-shot meta-learner

Conclusions

- An LSTM-based model for meta learning
- The LSTM meta-learner uses its state to represent the learning updates of the parameters of a learner
- It discovers both a good initialization for a learner's parameters and an effective mechanism for updating the learner's parameters
 - It works well for new classes with few training data

Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

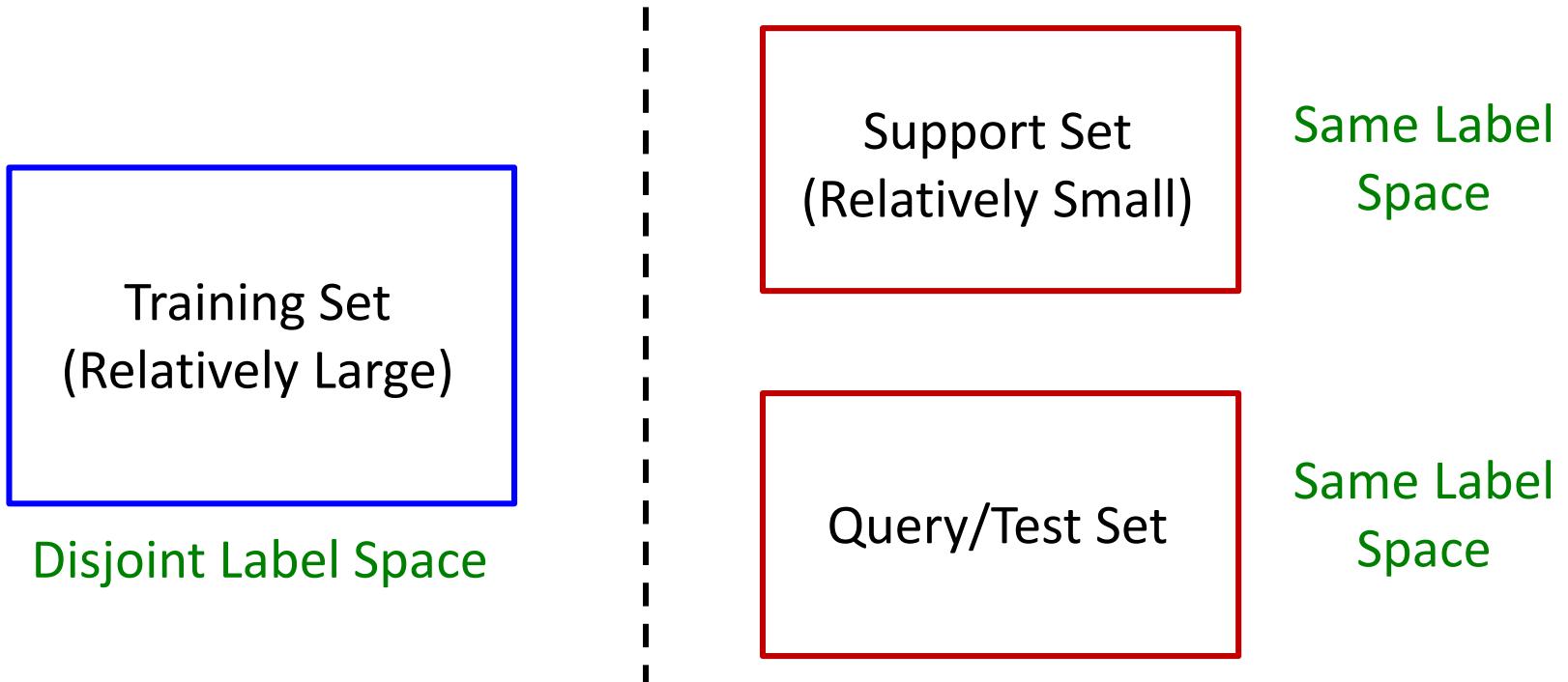
Relation network for meta learning

[Sung et al. CVPR 2018]

- Relation network: A simple, flexible, and general framework for few-shot learning
- It learns to recognize new classes given only few examples from each
- It is trained end-to-end from scratch
- During meta-training, it learns a deep distance metric to compare a small number of images within episodes
- During meta-testing, it classifies images by computing relation scores between query and support images
- It can be extended to zero-shot learning

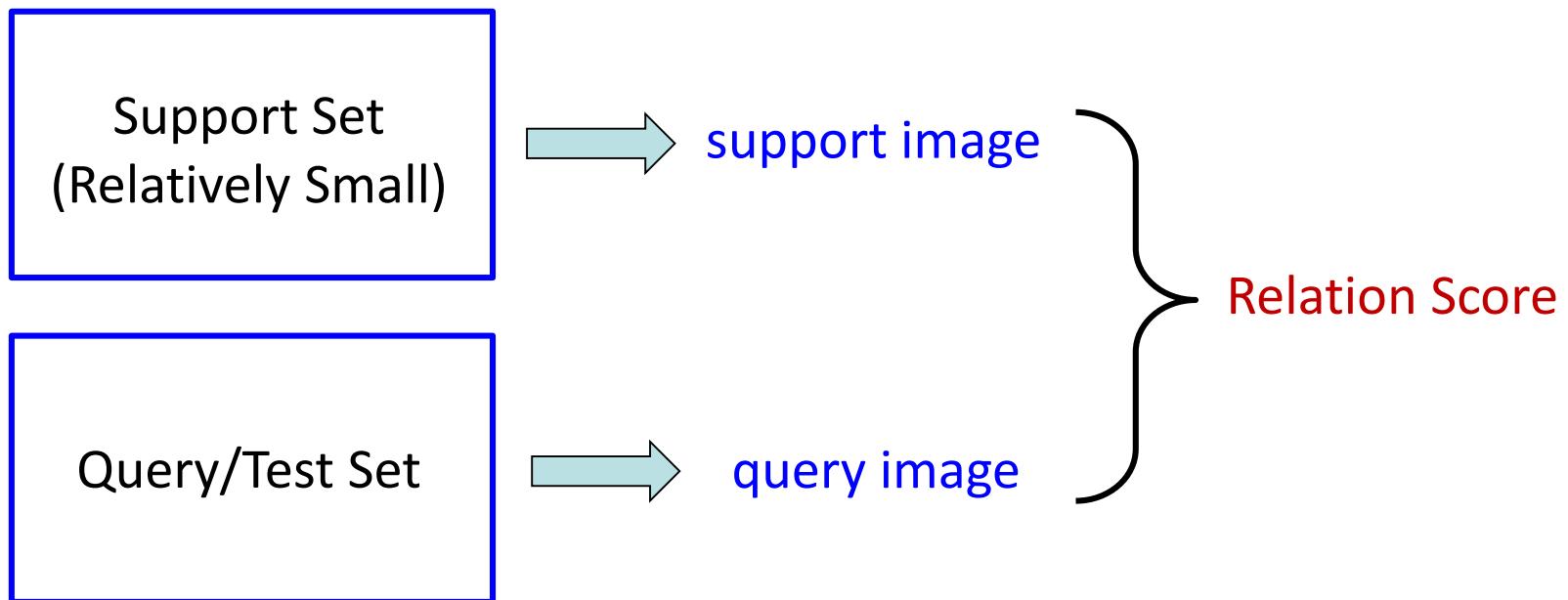
Meta learning for few shot learning

- Goal: Train a classifier to recognize new classes given few examples of each class
- Problem setting



Relation network: Recognition by comparison

- Relation network compares **support images** and **query images** and makes classification according to the returned **relation scores**

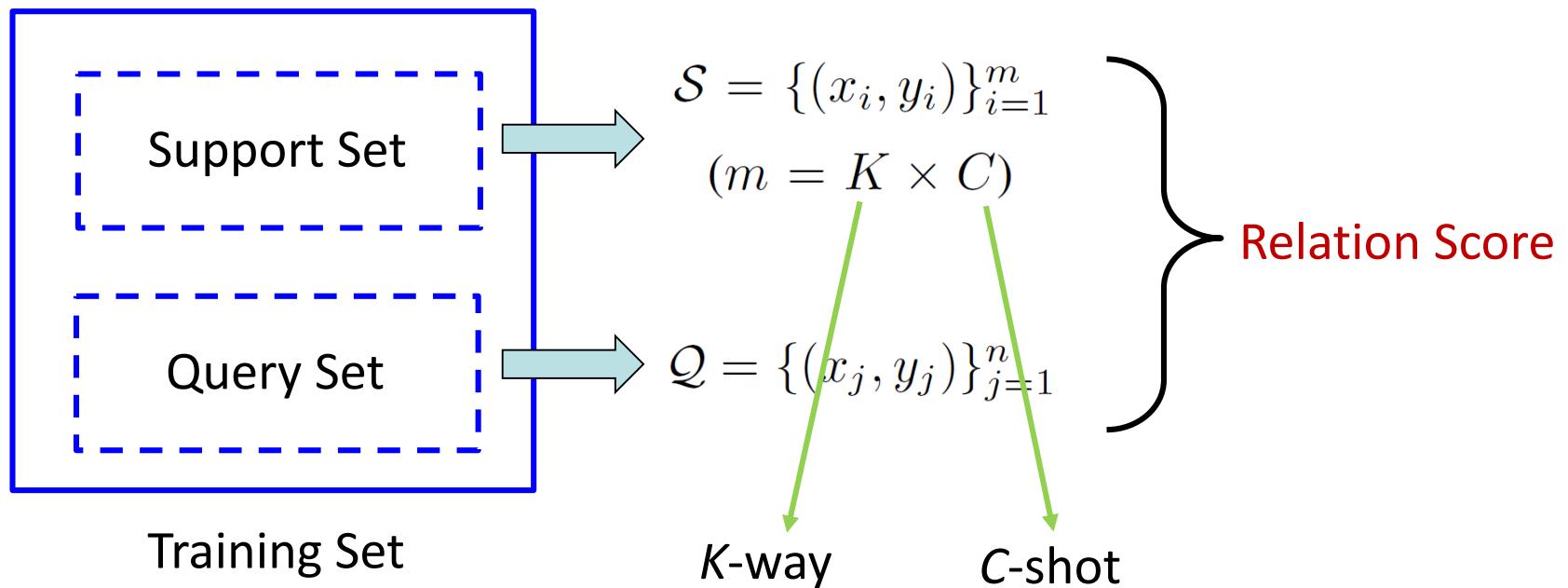


- Formulate a recognition task as several verification comparisons



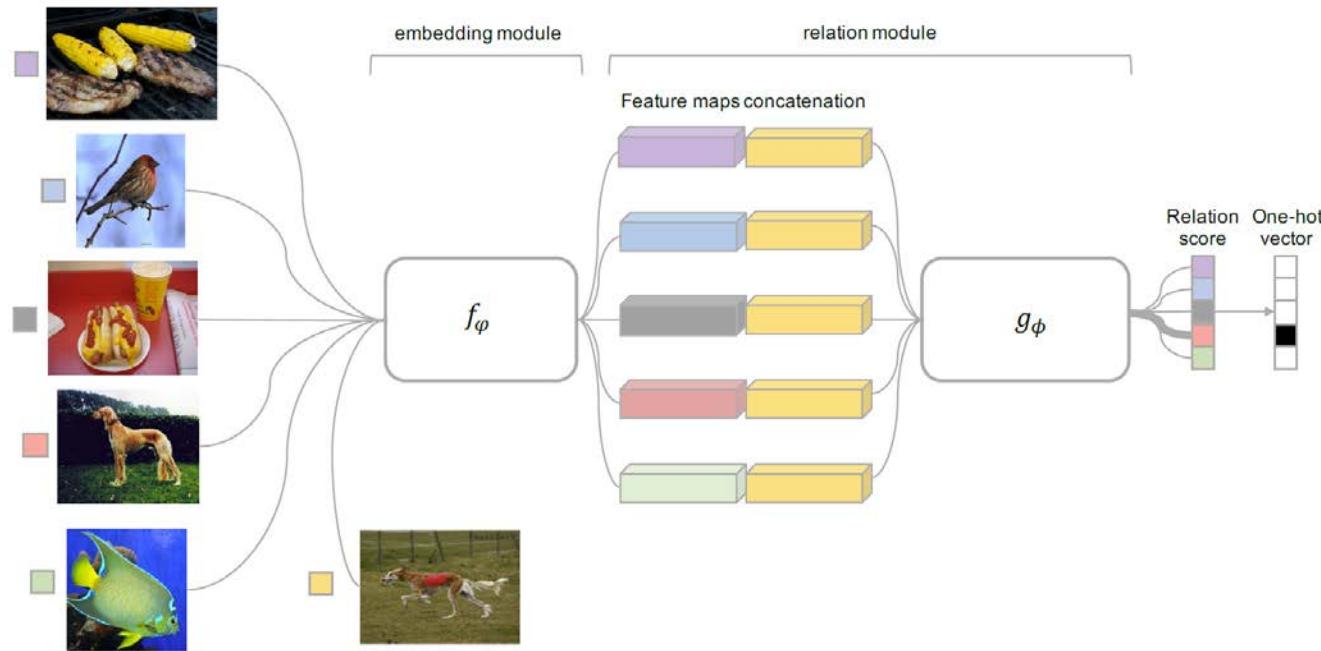
Relation network: Recognition by comparison

- Relation network learns to compare using meta learning: It mimics the comparison procedure on the training set and derives the model for comparison



Network architecture

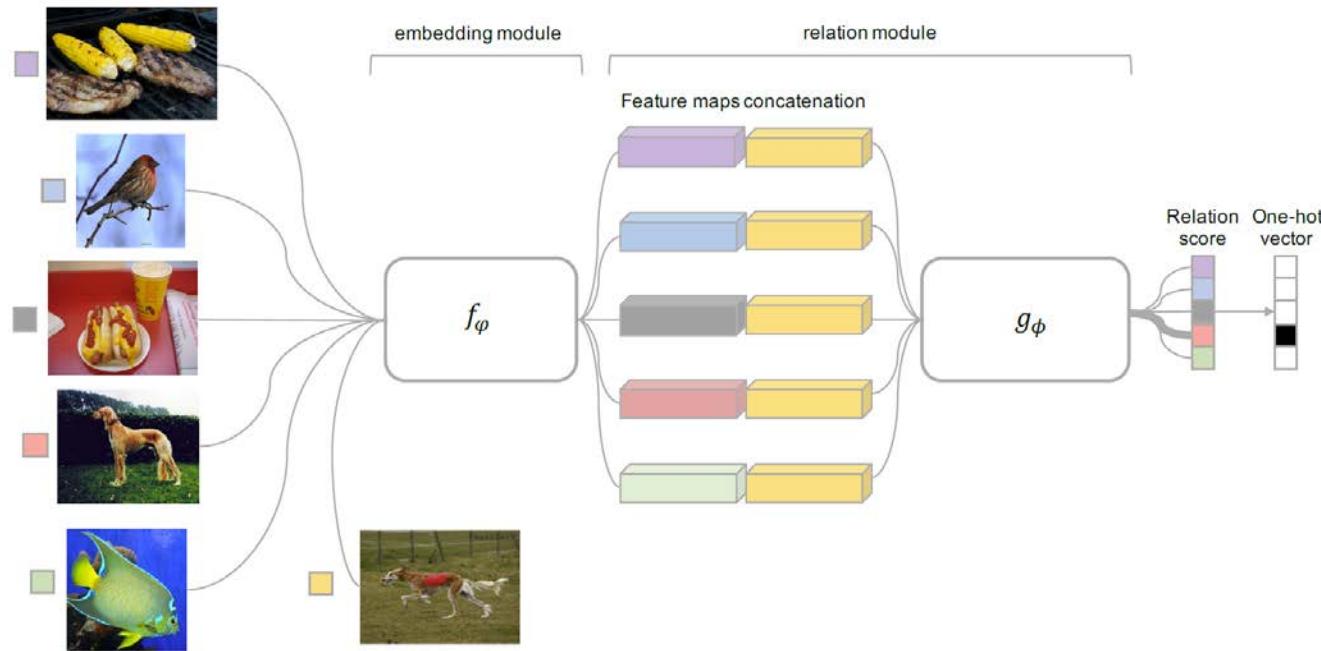
- An example of the 5-way 1-shot setting



- The embedding module computes the feature maps of each input image

Network architecture

- An example of the 5-way 1-shot setting

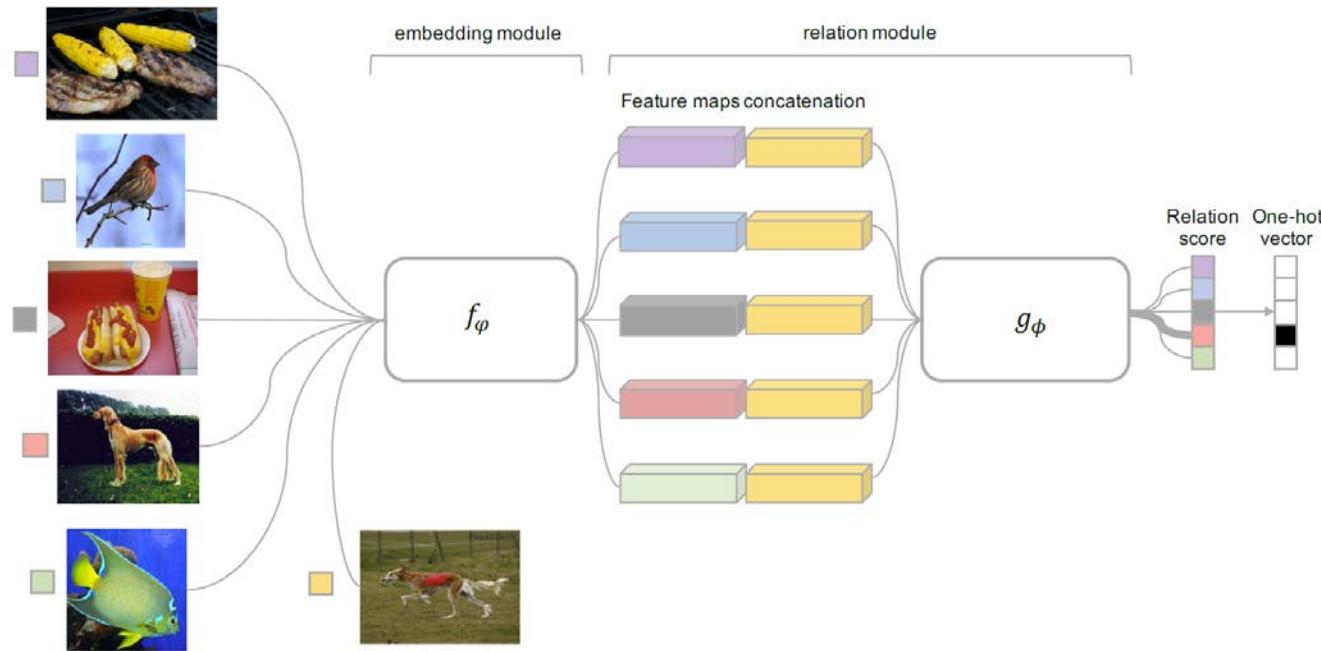


- The relation module, working on the concatenation of a support image and a query image, returns a similarity score in range of 0 and 1



Network architecture

- An example of the 5-way 1-shot setting



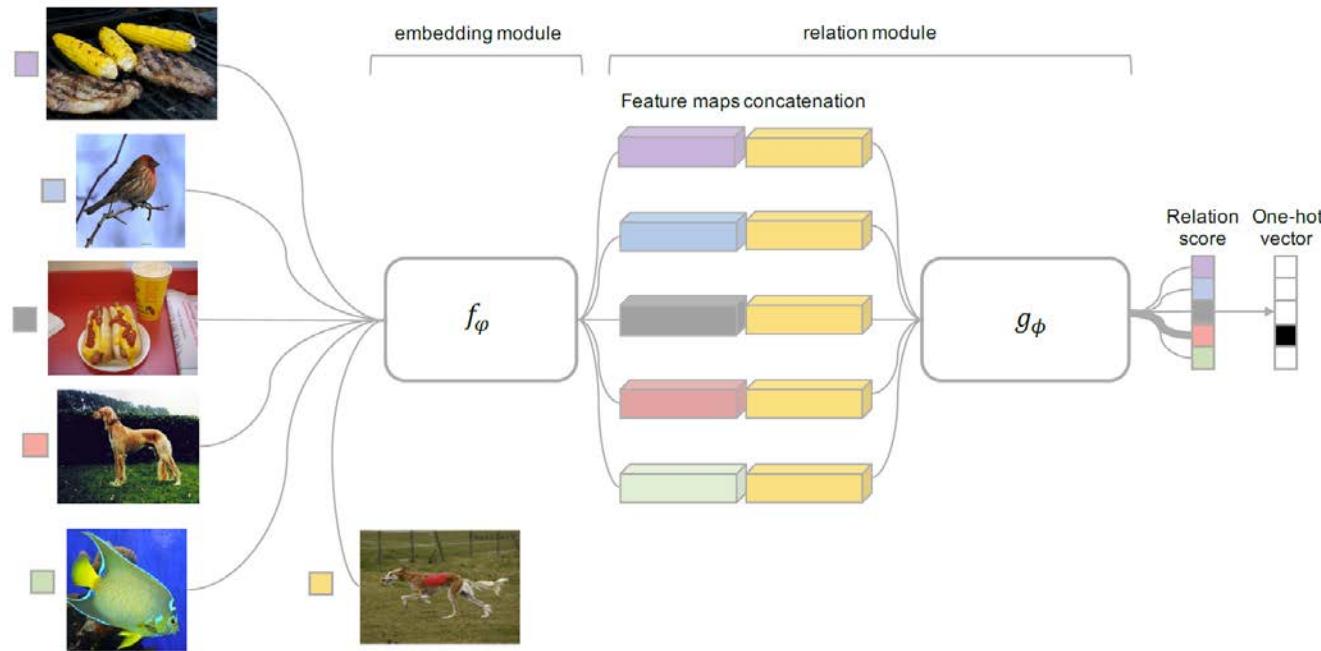
- The formulation

$$r_{i,j} = g_\phi(\mathcal{C}(f_\varphi(x_i), f_\varphi(x_j))), \quad i = 1, 2, \dots, C$$



Network architecture

- An example of the 5-way 1-shot setting

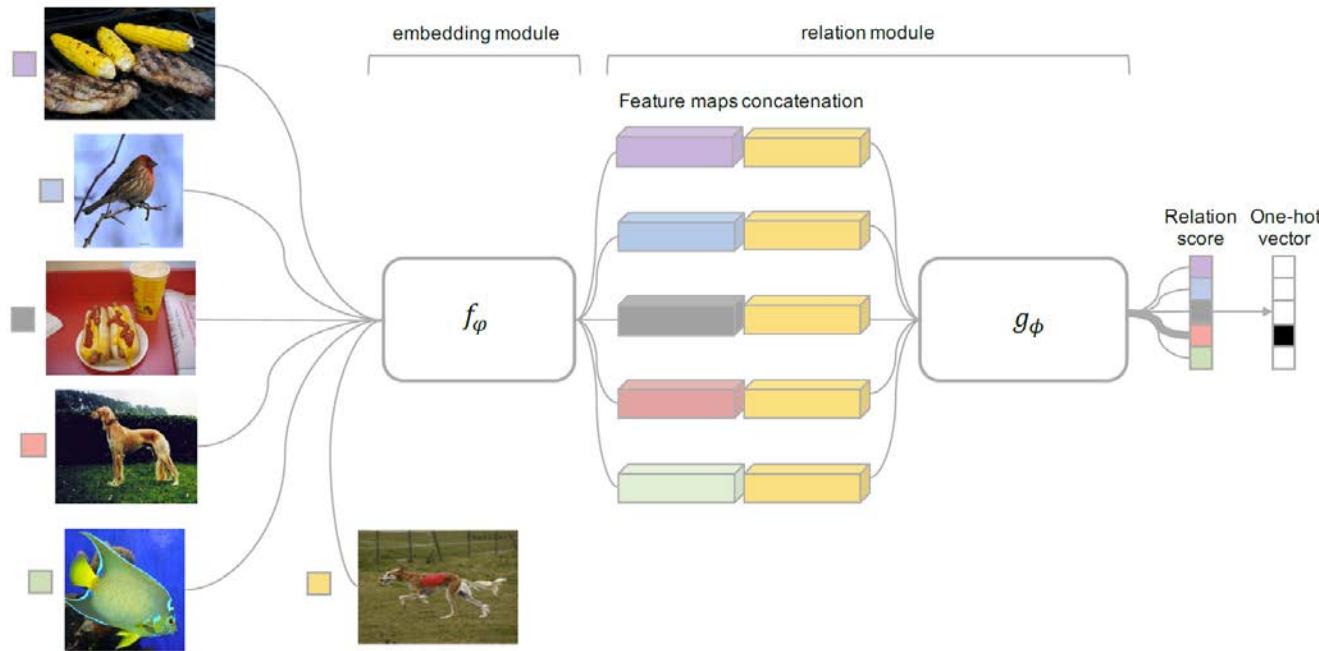


- Training objective

$$\varphi, \phi \leftarrow \operatorname{argmin}_{\varphi, \phi} \sum_{i=1}^m \sum_{j=1}^n (r_{i,j} - \mathbf{1}(y_i == y_j))^2$$

Network architecture

- An example of the 5-way 1-shot setting



- For K -shot extension, we element-wise sum over the embedding module outputs of all samples of each class to form the feature maps of this class



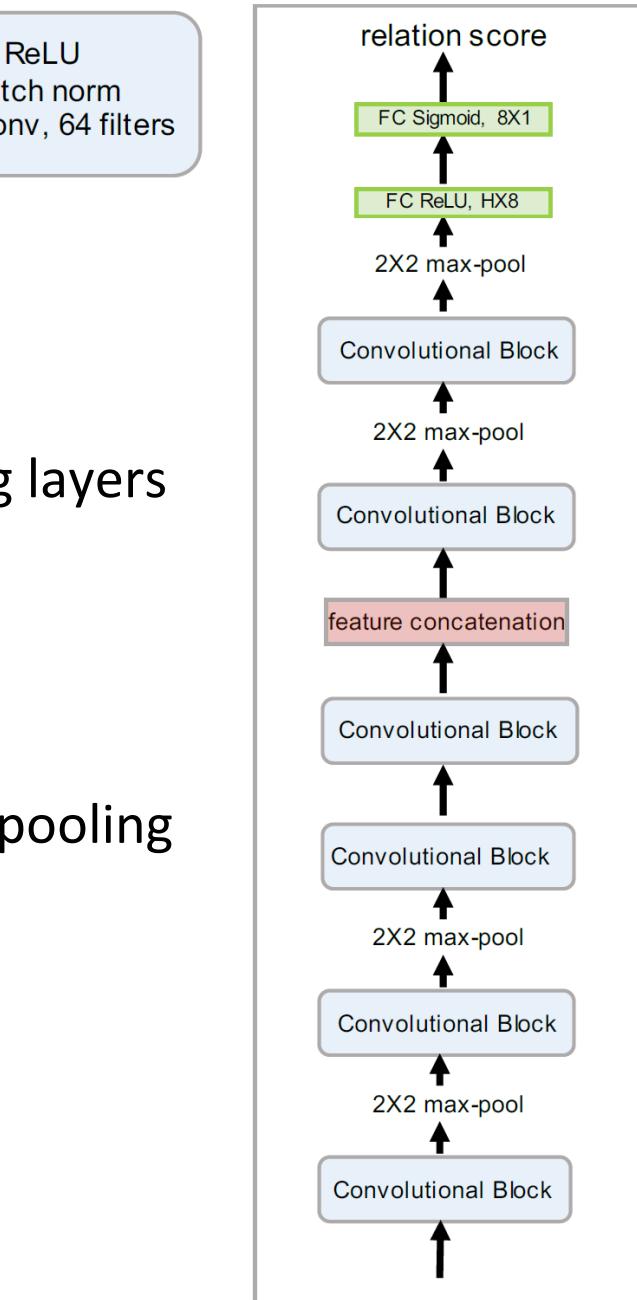
Implementations

- Embedding module
 - 4 conv. blocks
 - Each block contains a 64 3x3 filters
 - First two blocks have 2x2 max-pooling layers
- Relation module
 - 2 conv. blocks
 - Both blocks are followed by 2x2 max-pooling
 - Two fully connected layers
 - The size of H depends on the dataset

(a) Convolutional Block

ReLU
batch norm
3X3 conv, 64 filters

(b) RN for few-shot learning



Experiments: Datasets and settings

- Datasets for few shot learning
 - Omniglot
 - ◆ 1623 characters, each of which has 20 samples
 - ◆ 1200 characters for training and 423 for testing
 - minilmagenet
 - ◆ 100 classes, each of which has 600 samples
 - ◆ 64, 16, and 20 classes are used for training, validation, and testing respectively
- Settings
 - Way: 5-way and 20-way
 - Shot: 1-shot and 5-shot

Experimental results: Omniglot

- Relation network achieves the state-of-the-art performance on this dataset
- No fine-tuning during the meta-test phase is required

Model	Fine Tune	5-way Acc.		20-way Acc.	
		1-shot	5-shot	1-shot	5-shot
MANN [32]	N	82.8%	94.9%	-	-
CONVOLUTIONAL SIAMESE NETS [20]	N	96.7%	98.4%	88.0%	96.5%
CONVOLUTIONAL SIAMESE NETS [20]	Y	97.3%	98.4%	88.1%	97.0%
MATCHING NETS [39]	N	98.1%	98.9%	93.8%	98.5%
MATCHING NETS [39]	Y	97.9%	98.7%	93.5%	98.7%
SIAMESE NETS WITH MEMORY [18]	N	98.4%	99.6%	95.0%	98.6%
NEURAL STATISTICIAN [8]	N	98.1%	99.5%	93.2%	98.1%
META NETS [27]	N	99.0%	-	97.0%	-
PROTOTYPICAL NETS [36]	N	98.8%	99.7%	96.0%	98.9%
MAML [10]	Y	$98.7 \pm 0.4\%$	$99.9 \pm 0.1\%$	$95.8 \pm 0.3\%$	$98.9 \pm 0.2\%$
RELATION NET	N	$99.6 \pm 0.2\%$	$99.8 \pm 0.1\%$	$97.6 \pm 0.2\%$	$99.1 \pm 0.1\%$

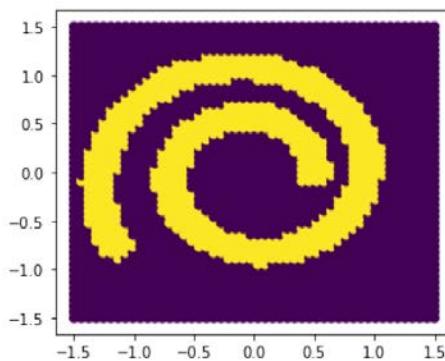
Experimental results: minilmagenet

- The performance on the minilmagenet dataset

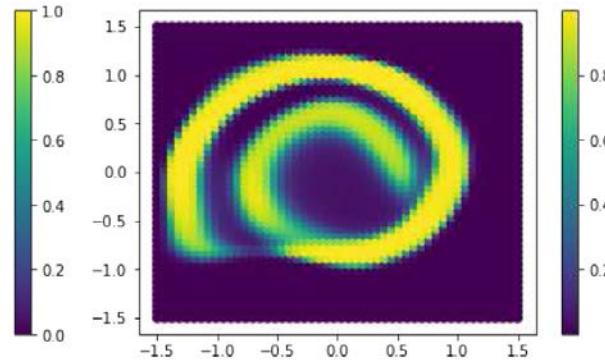
Model	FT	5-way Acc.	
		1-shot	5-shot
MATCHING NETS [39]	N	$43.56 \pm 0.84\%$	$55.31 \pm 0.73\%$
META NETS [27]	N	$49.21 \pm 0.96\%$	-
META-LEARN LSTM [29]	N	$43.44 \pm 0.77\%$	$60.60 \pm 0.71\%$
MAML [10]	Y	$48.70 \pm 1.84\%$	$63.11 \pm 0.92\%$
PROTOTYPICAL NETS [36]	N	$49.42 \pm 0.78\%$	$68.20 \pm 0.66\%$
RELATION NET	N	$50.44 \pm 0.82\%$	$65.32 \pm 0.70\%$

Experimental results: Visualization

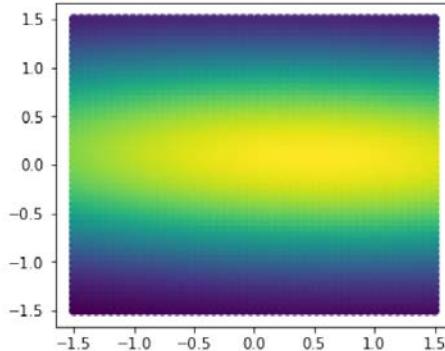
- Relation network learns a deep **embedding** and learning a deep non-linear **metric**



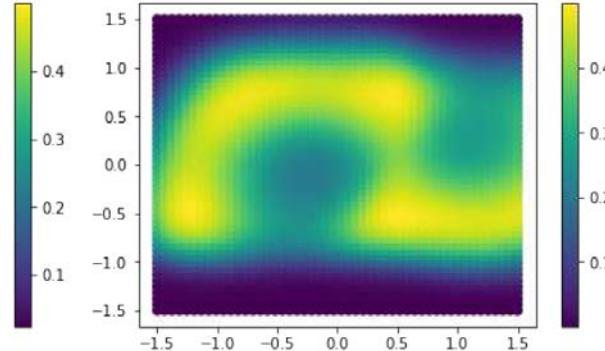
(a) Ground Truth



(b) Relation Network



(c) Metric Learning



(d) Metric + Embedding



Experimental results: Visualization

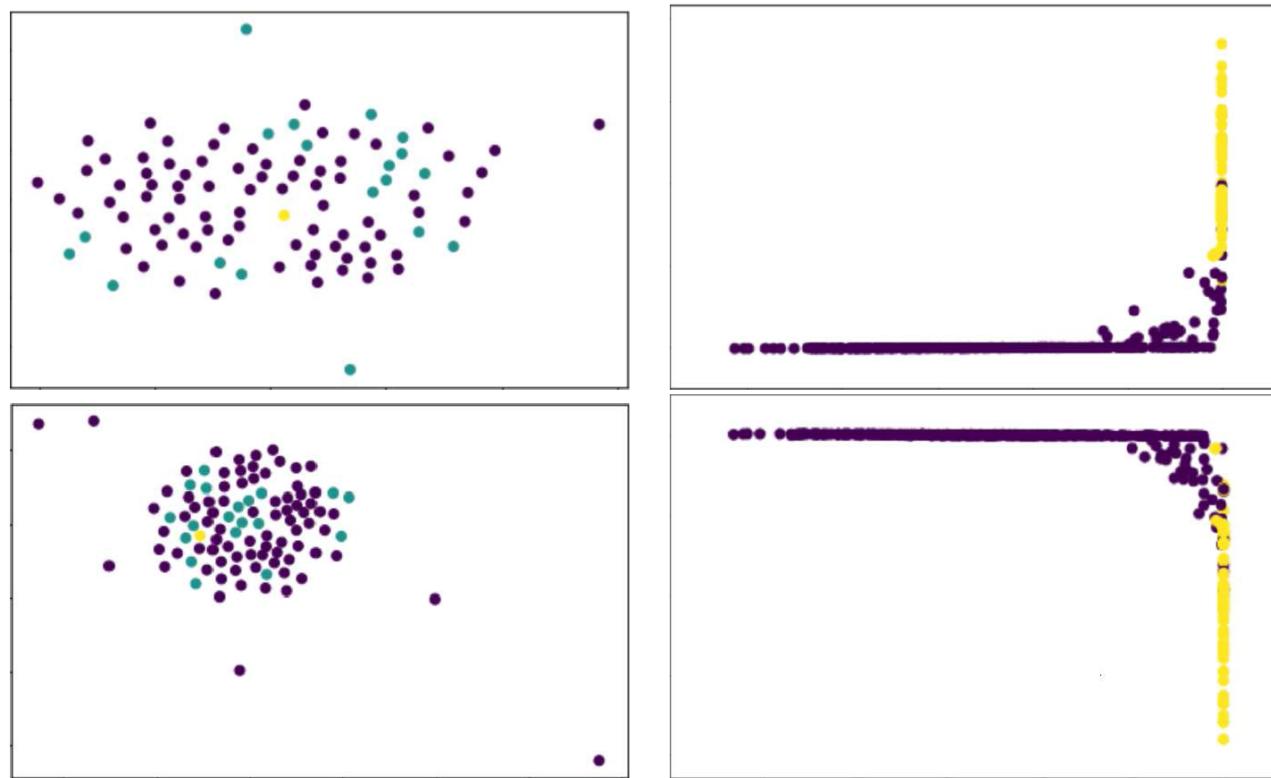


Figure 5: Example Omniglot few-shot problem visualisations. Left: Matched (cyan) and mismatched (magenta) sample embeddings for a given query (yellow) are not straightforward to differentiate. Right: Matched (yellow) and mismatched (magenta) relation module pair representations are linearly separable.



Conclusions

- A simple metric-learning method for few-shot learning and meta learning
- Relation network learns an embedding and a deep non-linear distance metric
- It is end-to-end trainable
- It produces the state-of-the-art performance without fine-tuning

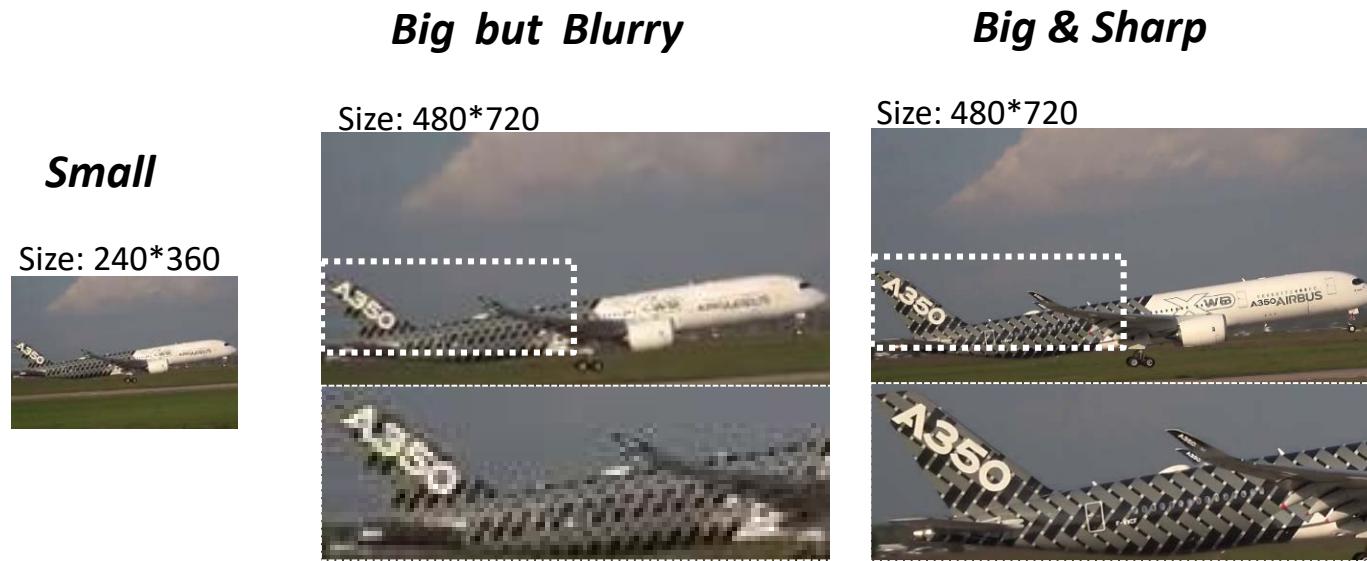
Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

Meta learning for fast adaption to super-resolution

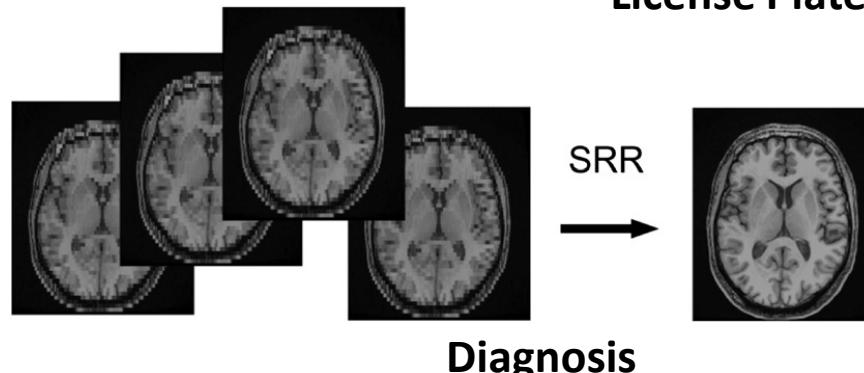
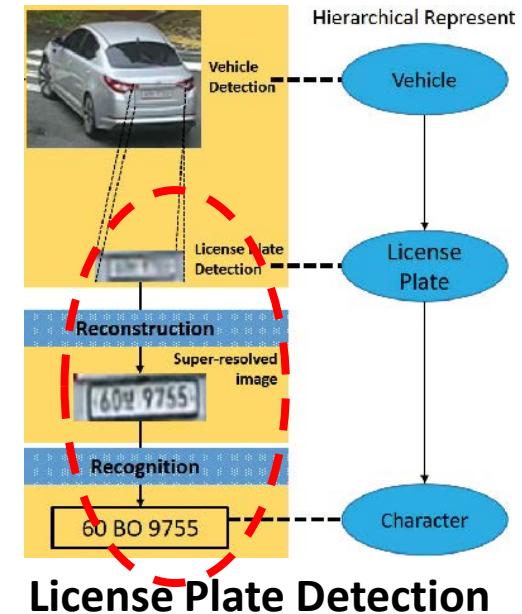
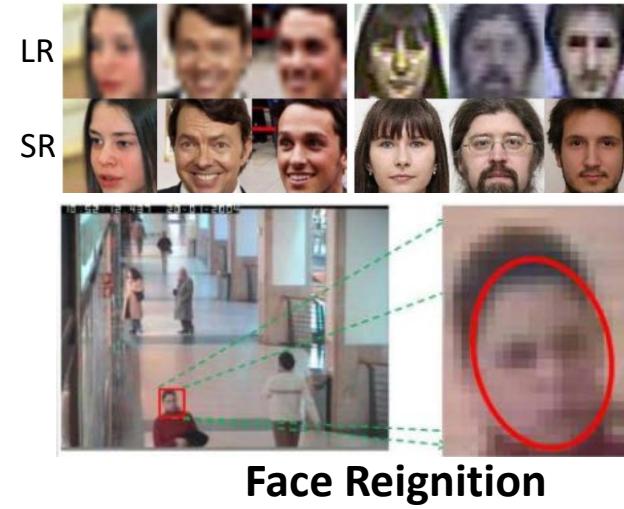
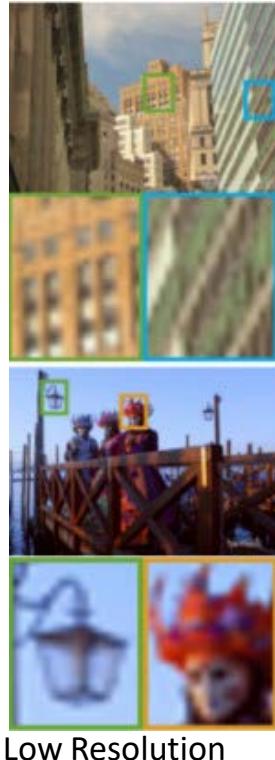
[Park et al. ECCV 2020]

- Goal of image super-resolution
 - Enlarge an image with details recovered
 - Provide sharper contours

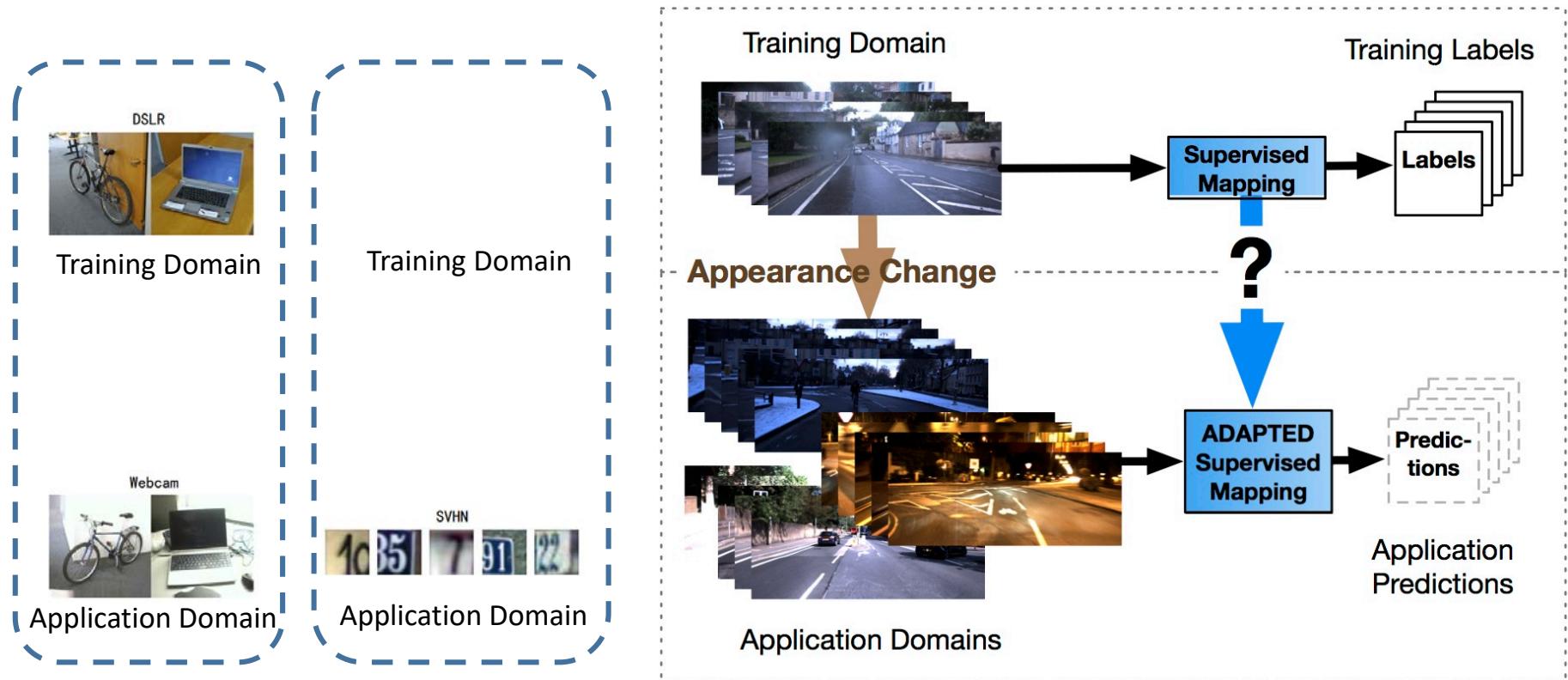


Why super-resolution

- It is essential to many applications



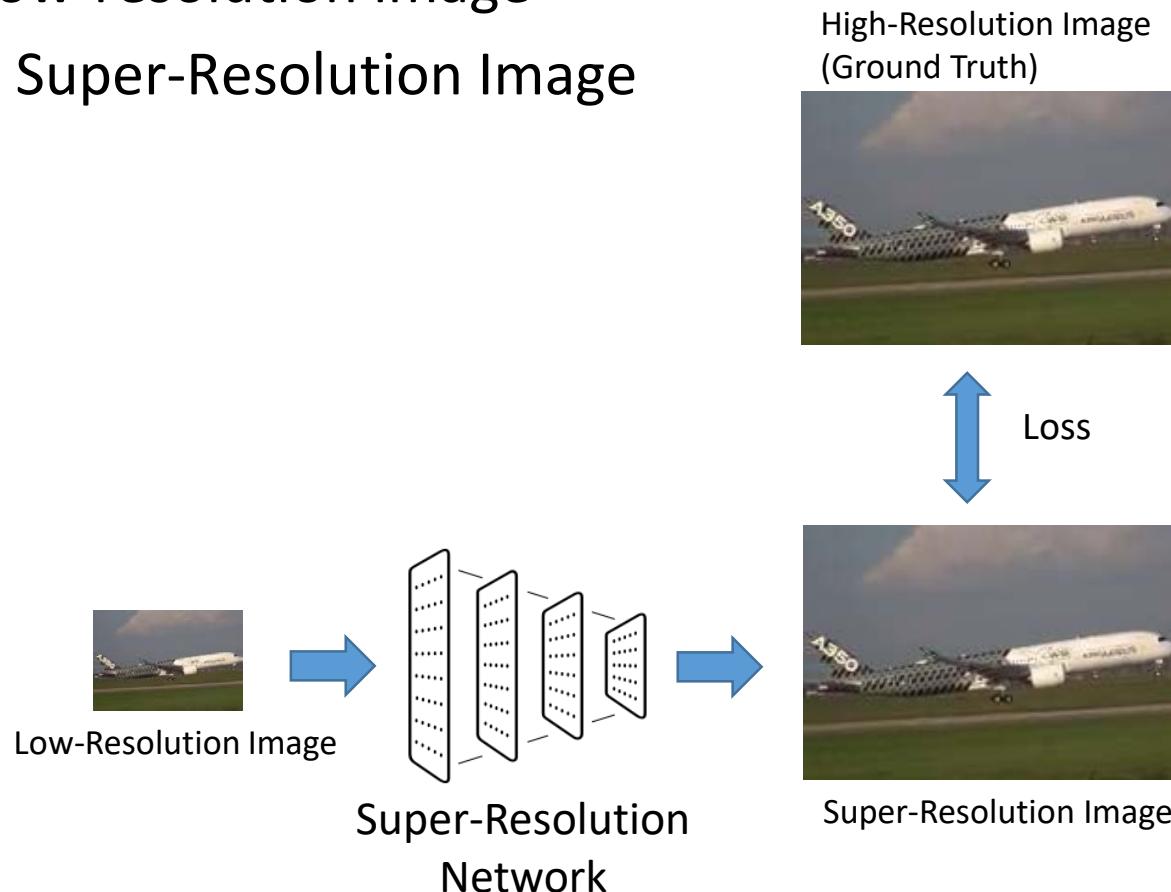
Adaptation for bridging the domain gap



Wulfmeier et al., Addressing Appearance Change in Outdoor Robotics with Adversarial Domain Adaptation.
In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

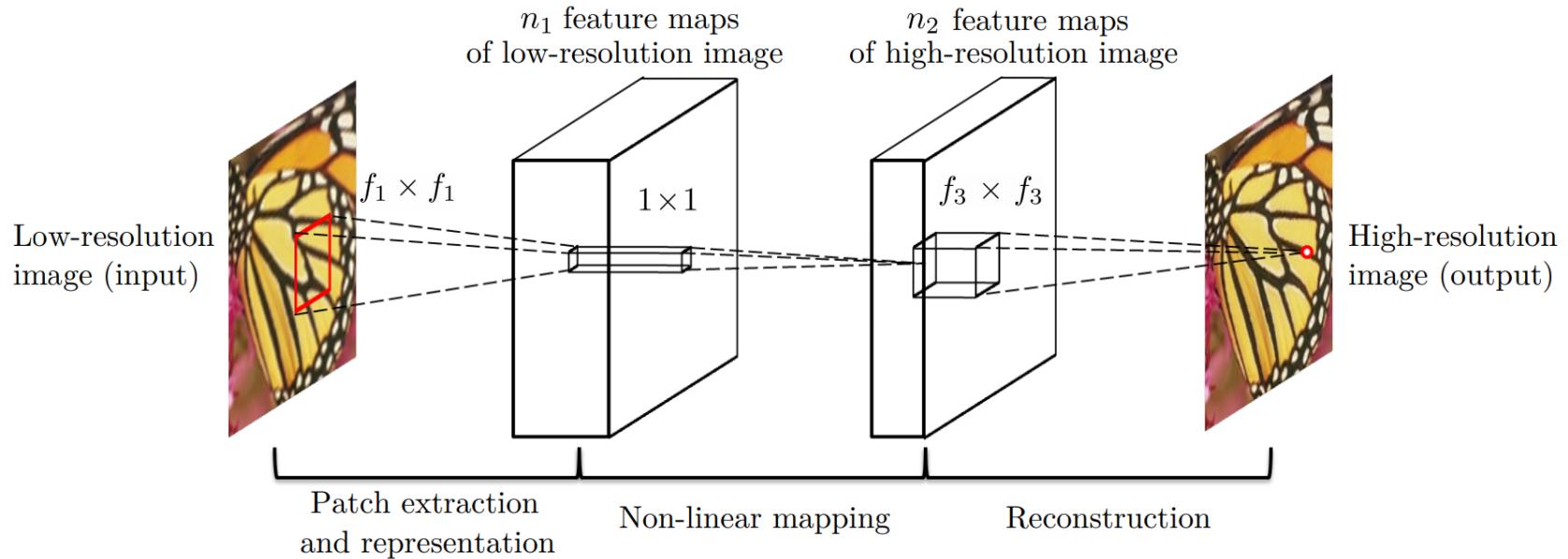
Deep learning for super-resolution

- Input: Low-resolution image
- Output: Super-Resolution Image



SRCNN: Super-Resolution CNN

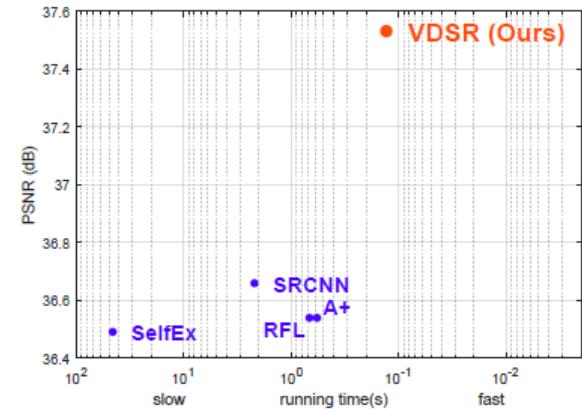
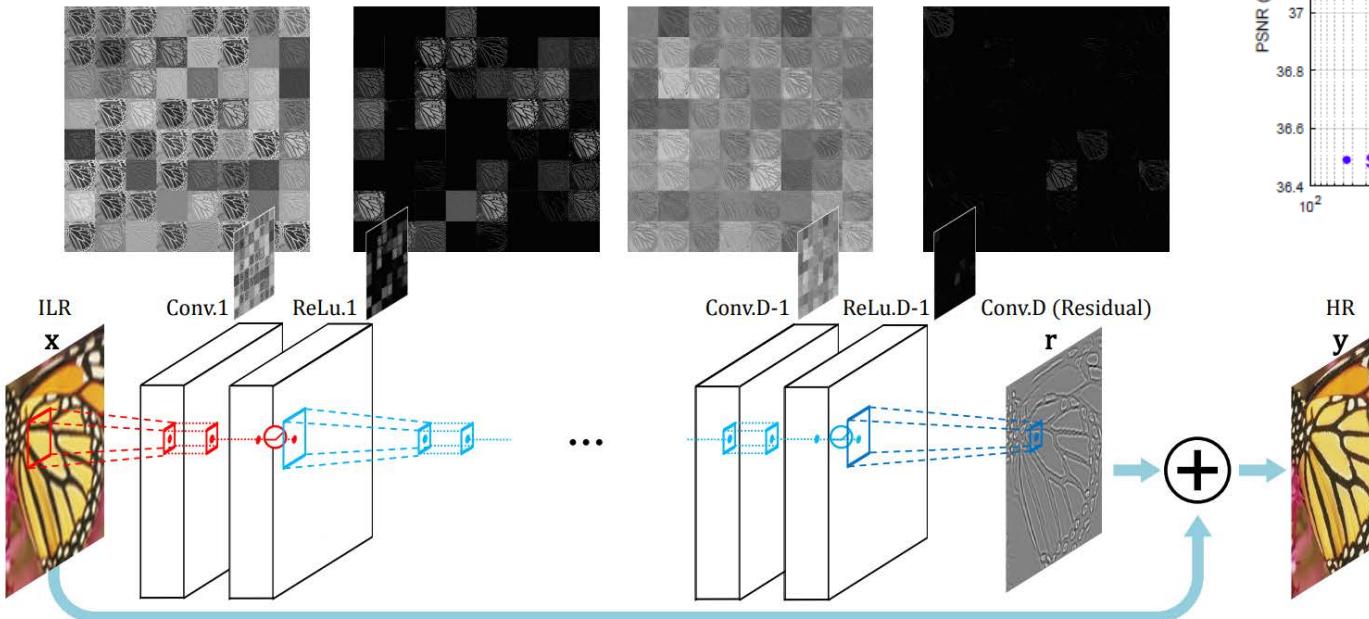
- 3-layer CNN



Dong et al., Learning a deep convolutional network for image super-resolution. In ECCV 2014.

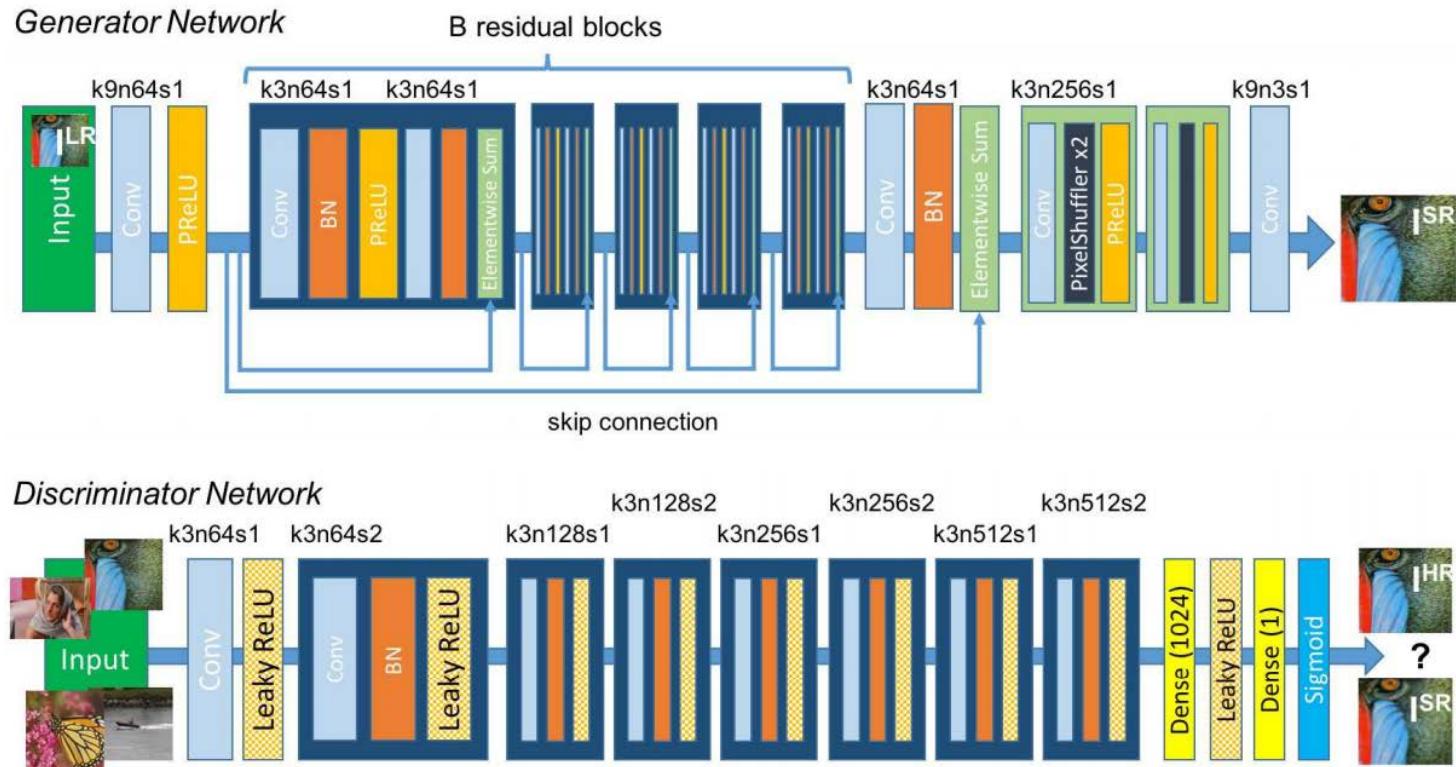
VDSR: Very Deep CNN for Super-Resolution

- Deep residual learning for super-resolution



Kim et al., Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In CVPR 2016.

SRGAN: Super-Resolution using Generative Adversarial Network

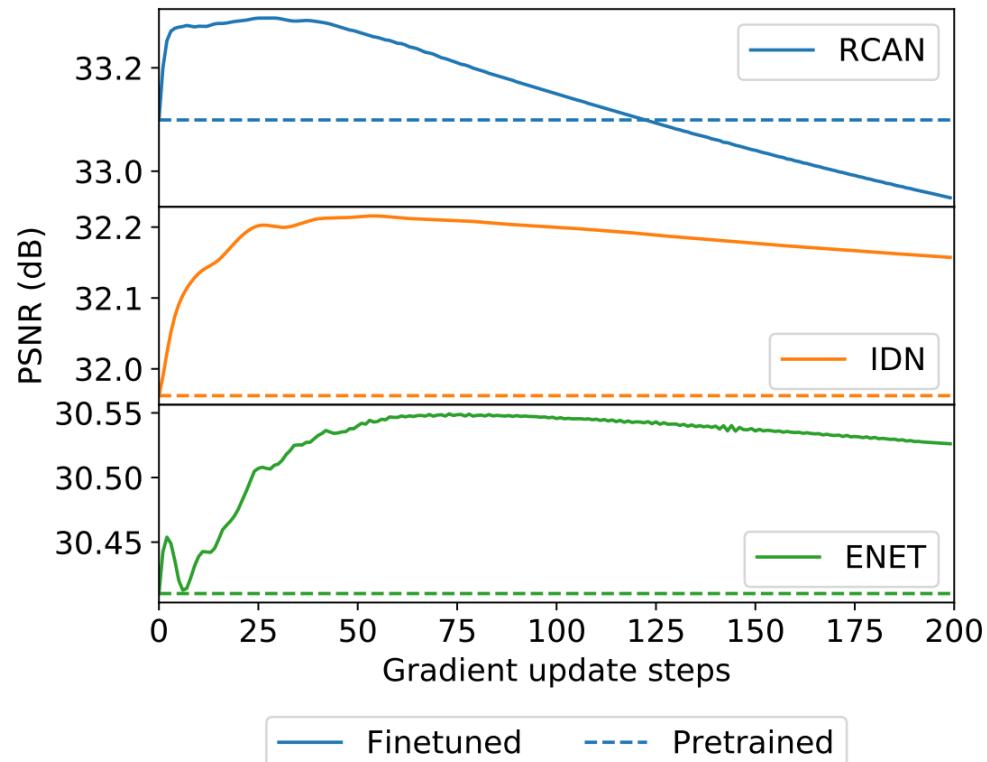
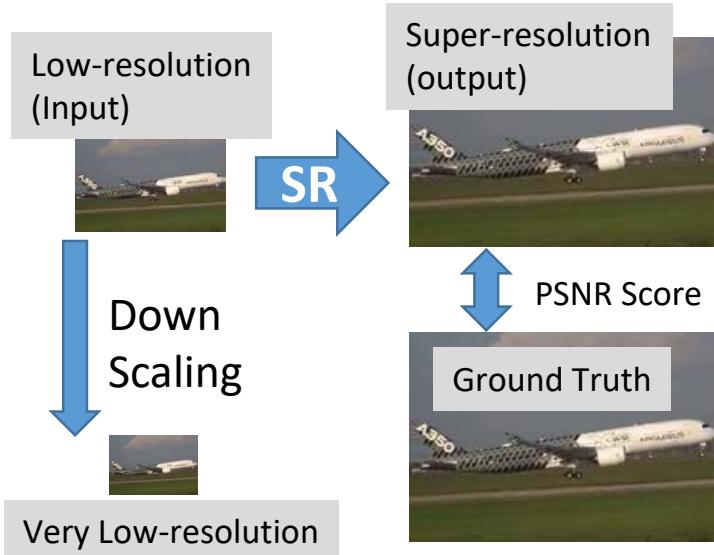


Ledig et al., Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In CVPR 2017.



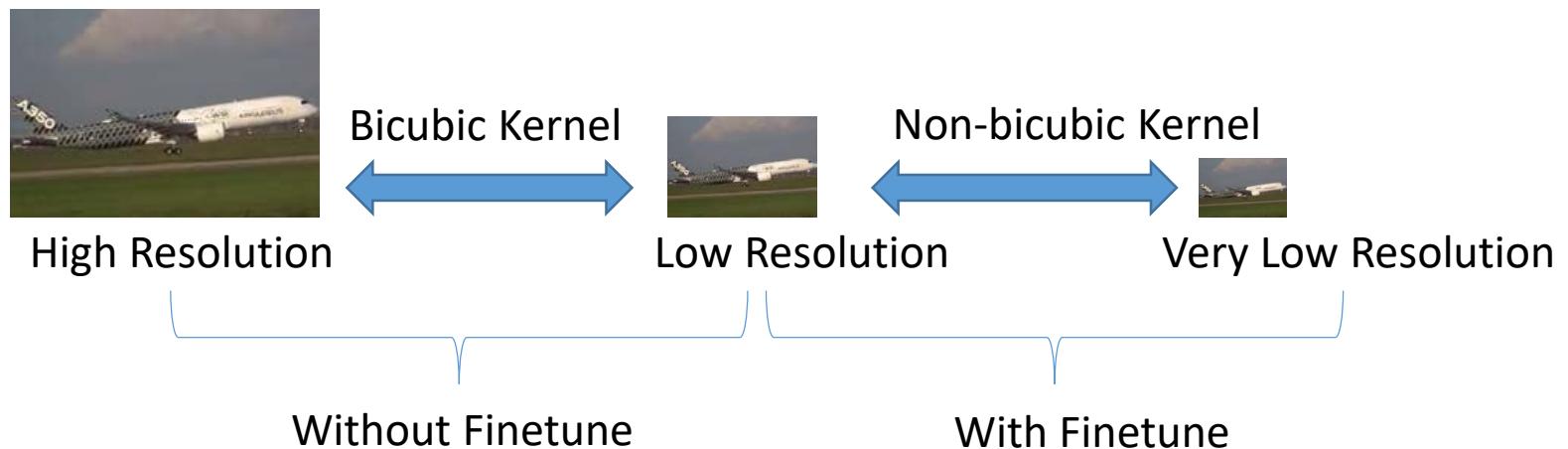
Exploiting patch-recurrence

- Learning-base SR Network
 - Residual Channel Attention Network
 - Information Distillation Network
 - EnhanceNet

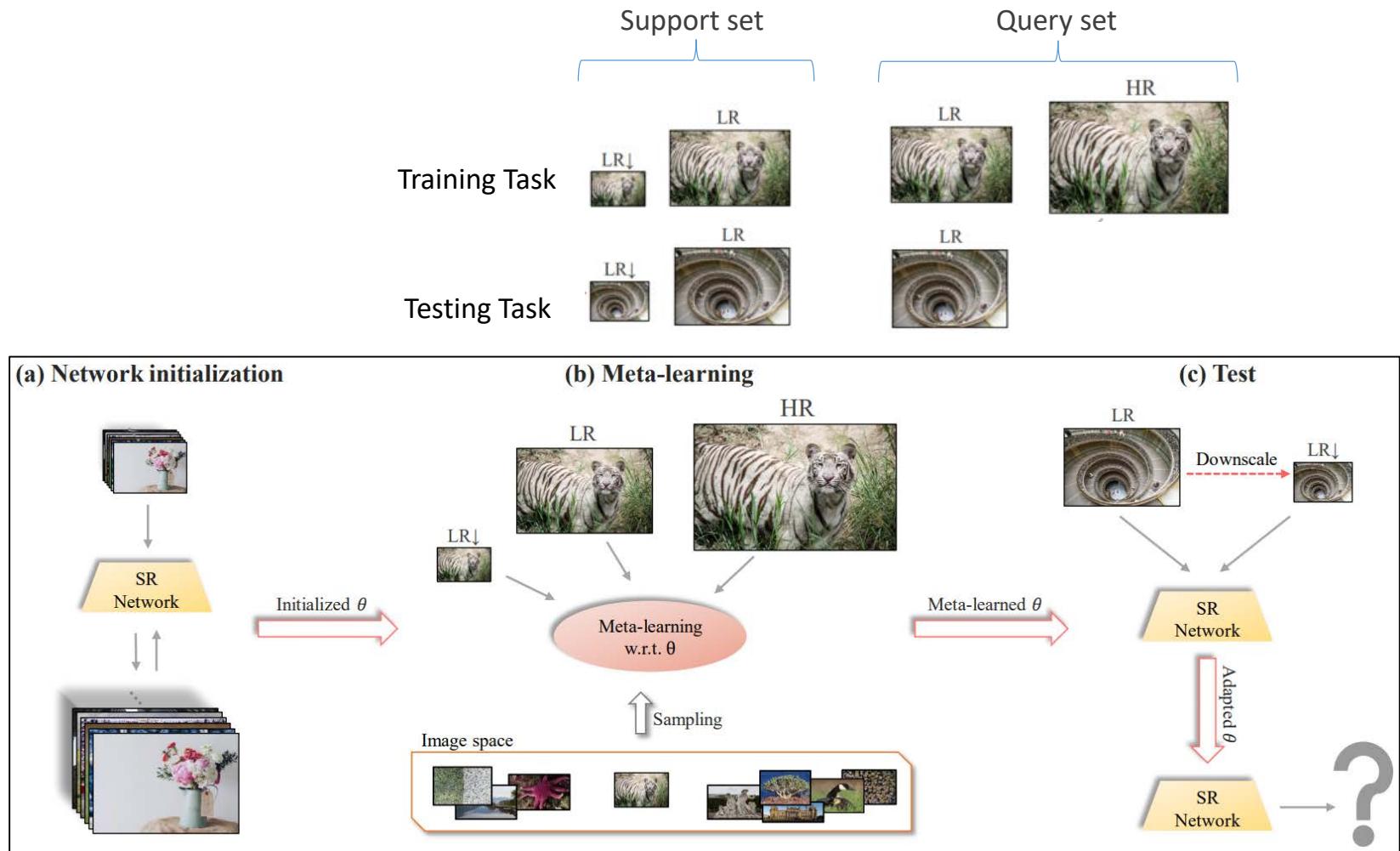


Handling unknown SR kernel

	Set5	Set14	BSD100	Urban100
IDN [18]	28.24	26.29	26.28	23.49
IDN-Finetune	31.52	28.91	28.47	25.93



Overview



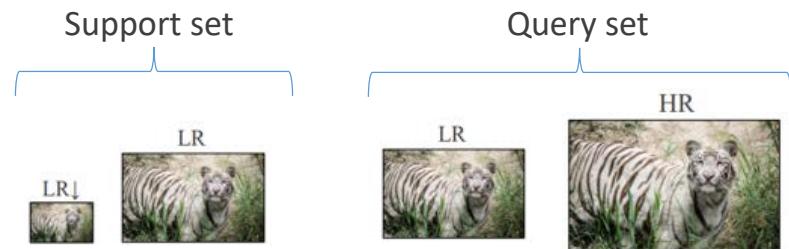
Meta training

Algorithm 1: MLSR training algorithm

Require: $p(I)$: Distribution (e.g. uniform) over images

Require: α, β : Hyper-parameters (step-size)

- 1 Initialize θ
 - 2 **while** not converged **do**
 - 3 Sample a batch of images $\{I_i\} \sim p(I)$
 - 4 Generate $\{HR_i\}$, $\{LR_i\}$, $\{LR_i \downarrow\}$ from $\{I_i\}$
 - 5 **foreach** i **do**
 - 6 Evaluate $\nabla_{\theta}\mathcal{L}(f_{\theta}(LR_i \downarrow), LR_i)$ using \mathcal{L}
 - 7 Compute adapted parameters with SGD:
 $\theta_i \leftarrow \theta - \alpha \nabla_{\theta}\mathcal{L}(f_{\theta}(LR_i \downarrow), LR_i)$
 - 8 Update $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_i \mathcal{L}(f_{\theta_i}(LR_i), HR_i)$
-



Meta inference

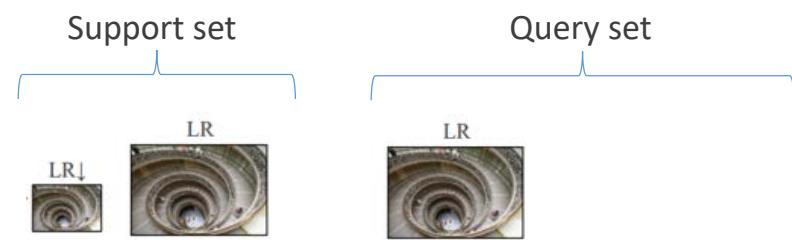
Algorithm 2: MLSR inference algorithm

Require: I : Given image

Require: α : Hyper-parameter (step-size)

Require: n : Number of gradient updates

- 1 Initialize θ with meta-trained parameter
 - 2 Generate $LR, LR \downarrow$ from I
 - 3 $i \leftarrow 0$
 - 4 **while** $i < n$ **do**
 - 5 Compute adapted parameters with SGD:
 $\theta \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(f_{\theta}(LR \downarrow), LR)$
 - 6 $i \leftarrow i + 1$
 - 7 Compute $f_{\theta}(LR)$
-



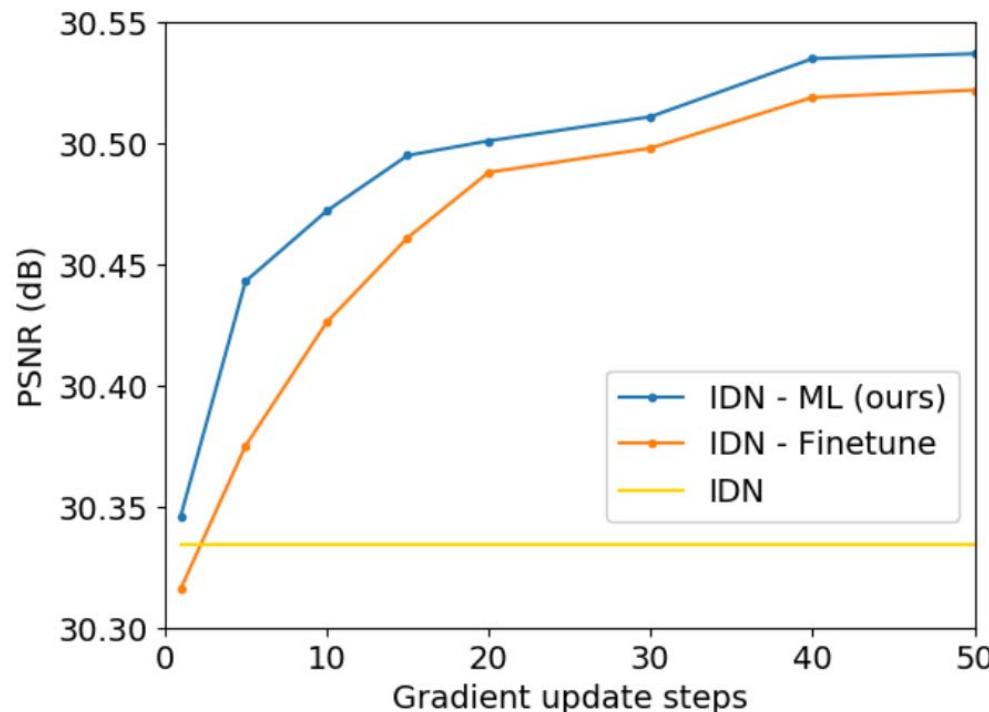
Experimental setup

- SR networks
 - SRCNN [ECCV 2014]
 - ENET [ICCV 2017]
 - IDN [CVPR 2018]
 - RCAN [ECCV 2018]
- Comparison
 - With/Without meta learning
- Loss Function
 - L2 loss

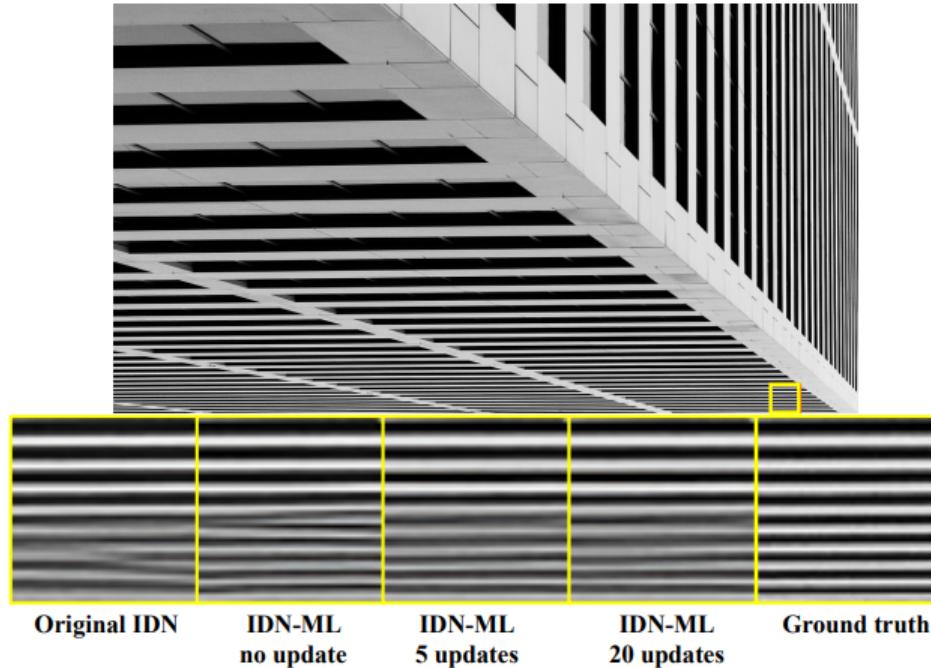
Experimental results

Algorithm	Iteration	DIV2K		BSD100		Urban100	
		PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
VDSR [21]	-	-	-	31.90	0.8960	30.76	0.9140
DRRN [33]	-	-	-	32.05	0.8973	31.23	0.9188
MemNet [34]	-	-	-	32.08	0.8978	31.31	0.9195
SRCNN [5]		34.11	0.9272	31.13	0.8852	29.39	0.8927
SRCNN-ML (ours)	5	34.14	0.9274	31.15	0.8855	29.42	0.8931
	20	34.18	0.9276	31.19	0.8857	29.48	0.8936
	100	34.23	0.9281	31.22	0.8860	29.54	0.8945
ENET [27]	-	34.59	0.9329	31.64	0.8935	30.38	0.9097
ENET-ML (ours)	5	34.62	0.9331	31.69	0.8936	30.46	0.9105
	20	34.64	0.9333	31.69	0.8936	30.49	0.9108
	100	34.67	0.9335	31.67	0.8934	30.52	0.9112
IDN [18]	-	35.24	0.9403	32.11	0.8994	31.95	0.9269
IDN-ML (ours)	5	35.36	0.9408	32.17	0.8996	32.06	0.9275
	20	35.38	0.9409	32.17	0.8996	32.17	0.9280
	100	35.40	0.9413	32.08	0.8988	32.23	0.9286
RCAN [38]	-	35.69	0.9451	32.38	0.9023	33.10	0.9369
RCAN-ML (ours)	5	35.72	0.9454	32.39	0.9023	33.27	0.9373
	20	35.75	0.9458	32.37	0.9022	33.32	0.9379
	100	35.48	0.9444	32.04	0.8982	33.26	0.9373

Fine-turning vs. meta learning



With and without adaptation



model name	before/after adaptation
IDN (trained w/ DIV2K)	32.19 / -
IDN (trained w/ Urban100)	32.28 / -
IDN-ML (trained w/ DIV2K)	32.19 / 32.37
IDN-ML (trained w/ Urban100)	32.13 / 32.43

Conclusions

- Meta learning for SR (MLSR) enables quick adaptation of SR parameters using only input LR image during the test phase
- MLSR can be combined with conventional SR networks without any architecture changes
- Boost PSNR performance of deep learning-based methods

Outline

- Problem statement of meta learning
- Representative meta learning algorithms
 - MAML [Finn et al., ICML 2017]
 - LSTM-based meta learner [Ravi and Larochelle, ICLR 2017]
 - Relation network [Sung et al. CVPR 2018]
- Meta learning applications
 - Super-resolution [Part et al., ECCV 2020]
 - Video frame interpolation [Choi et al. CVPR 2020]

Meta learning for video frame interpolation

[Choi et al. CVPR 2020]

- Frame interpolation: Upscale the temporal resolution of a video

1X

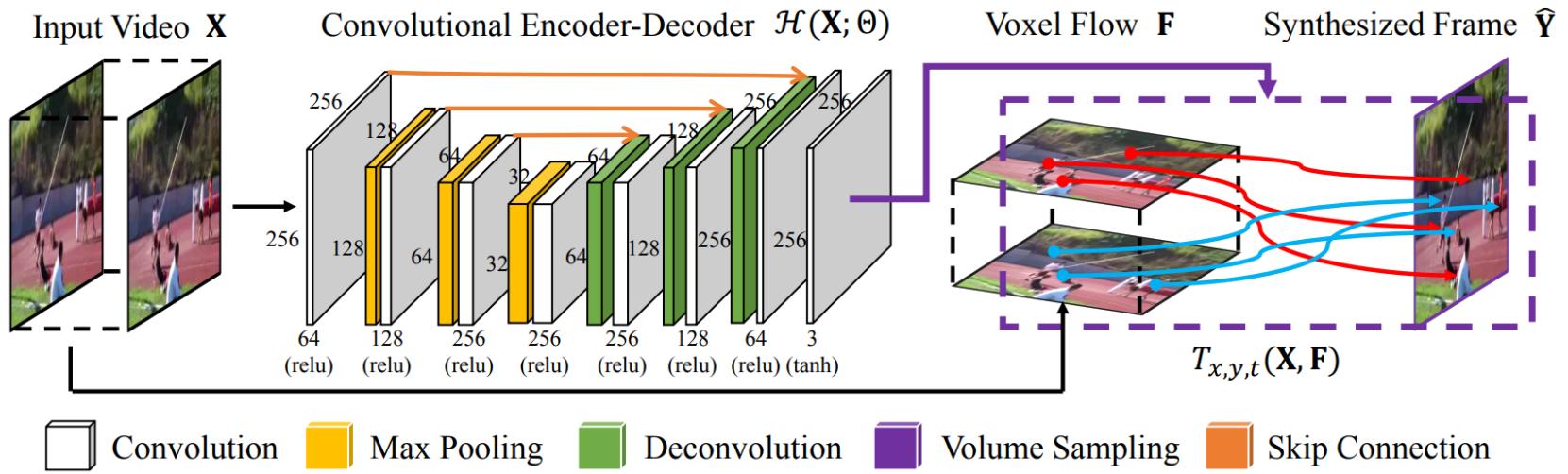


8X



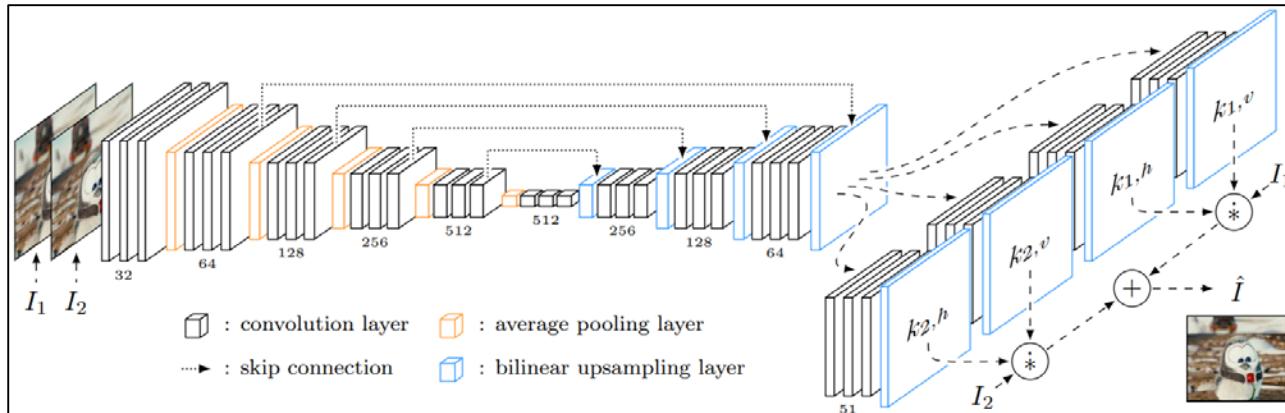
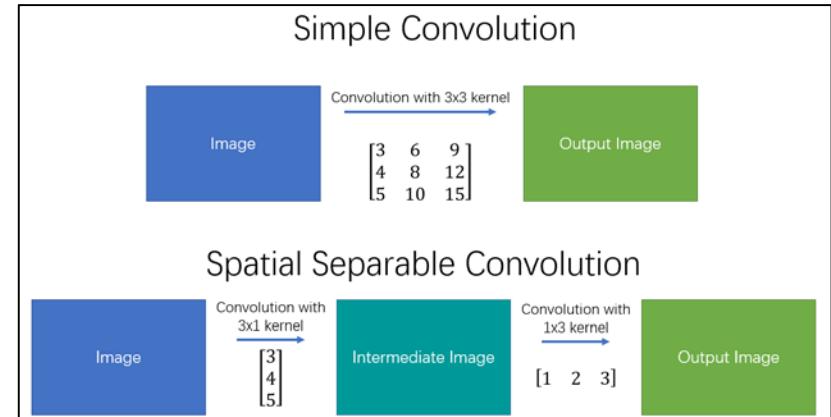
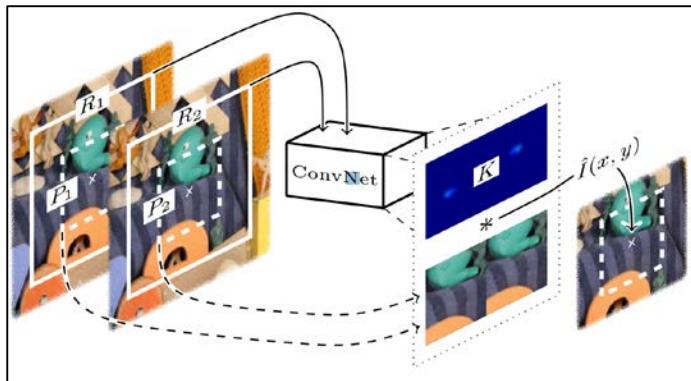
Liu et al., Deep Video Frame Interpolation using Cyclic Frame Generation. In AAAI 2019.

Deep voxel flow



Liu et al., Video Frame Synthesis Using Deep Voxel Flow. In ICCV 2017.

SepConv

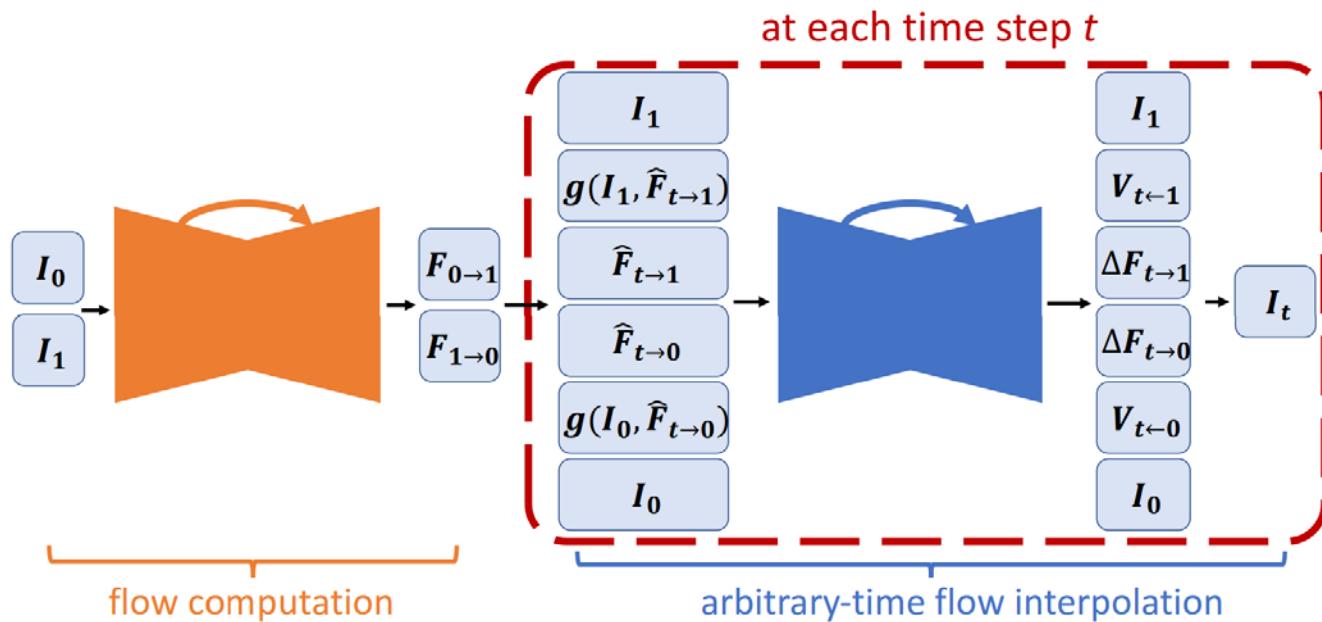


Niklaus et al., Video Frame Interpolation via Adaptive Convolution. In CVPR 2017.

Niklaus et al., Video Frame Interpolation via Adaptive Separable Convolution. In ICCV 2017.



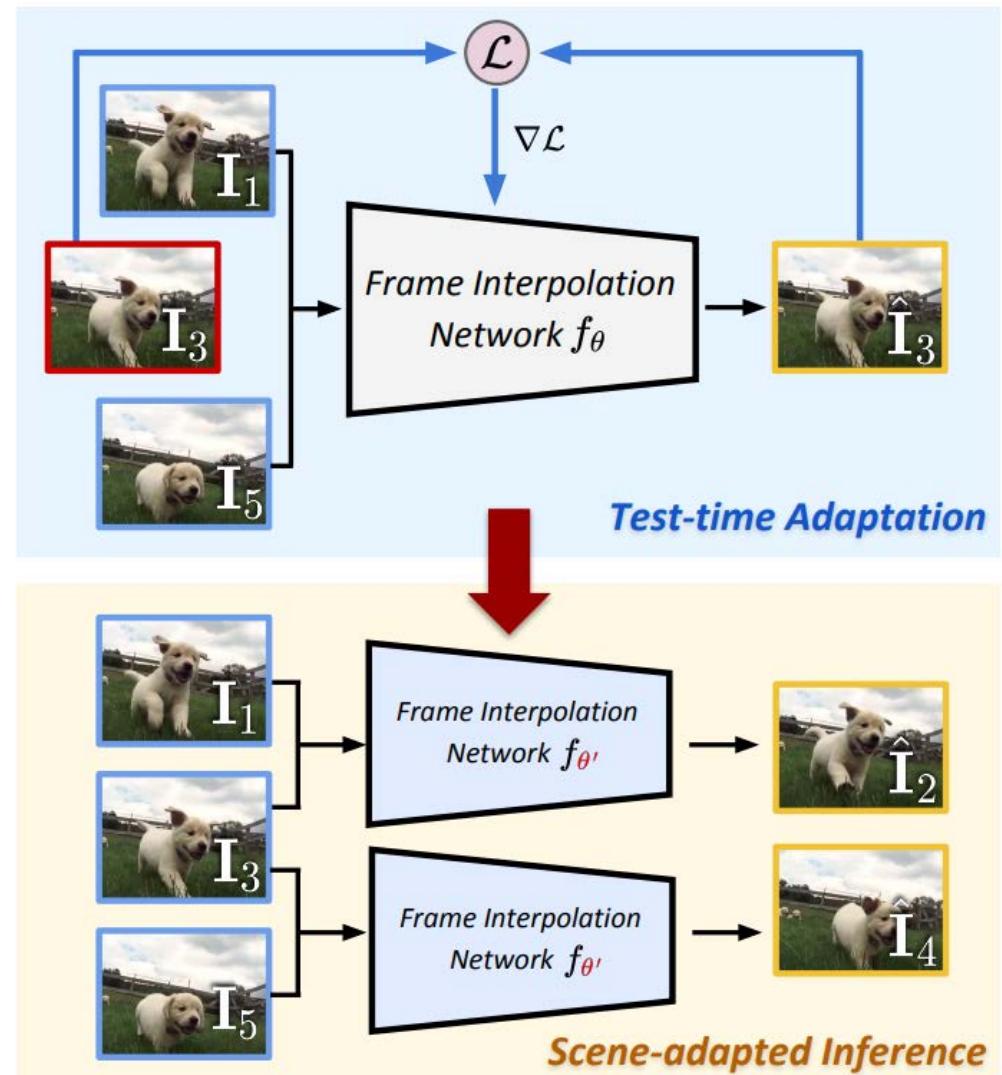
Super SloMo



Jiang et al., Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation. In CVPR 2018.

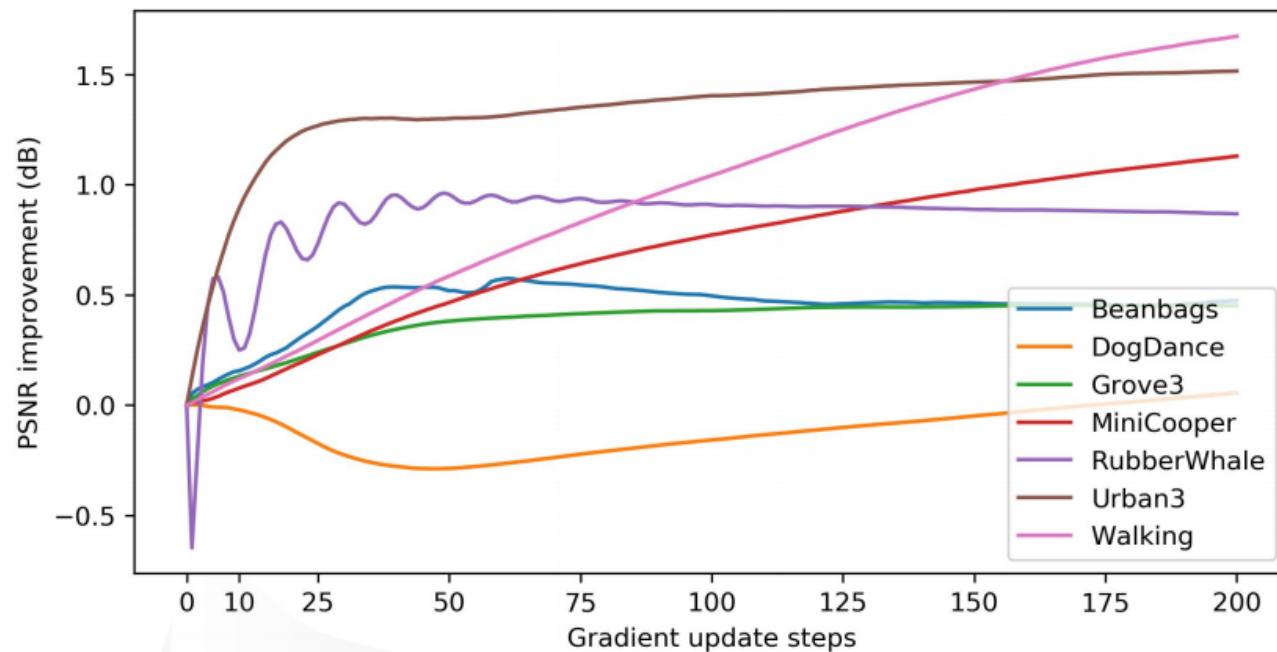
Overview

- Two-step approach:
 - Test-time adaptation
 - Scene-adapted inference



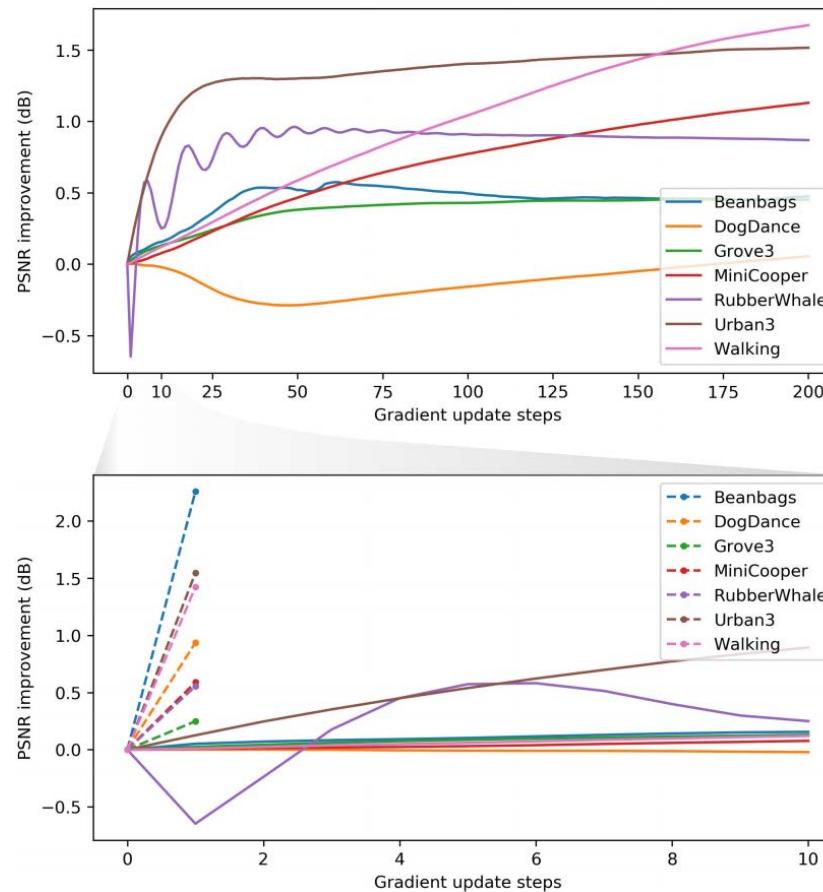
Feasibility test of fine-tuning

- Fine-tuning on several videos



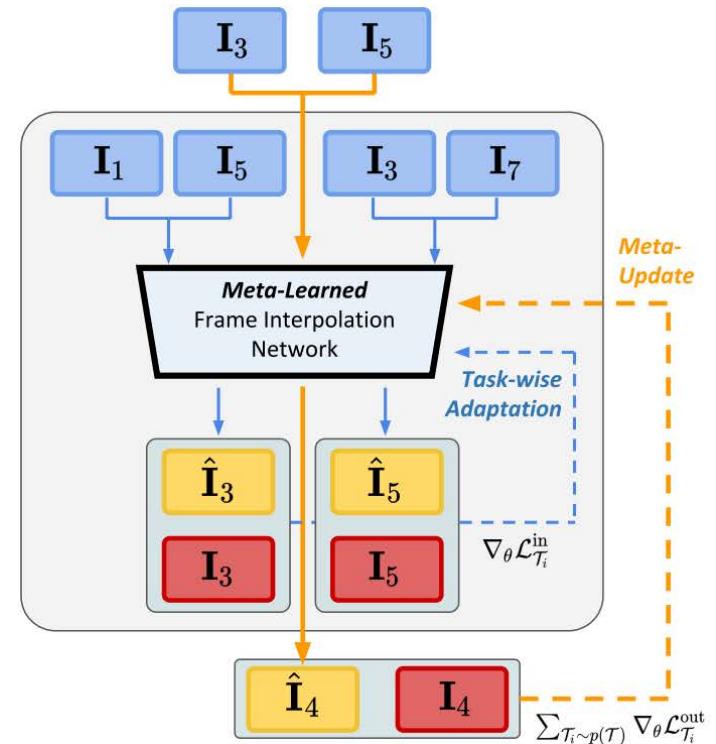
Meta learning for video frame interpolation

- Fine-tuning vs. meta learning



Meta training

- A task: video frame interpolation on a video



Meta training: Algorithm

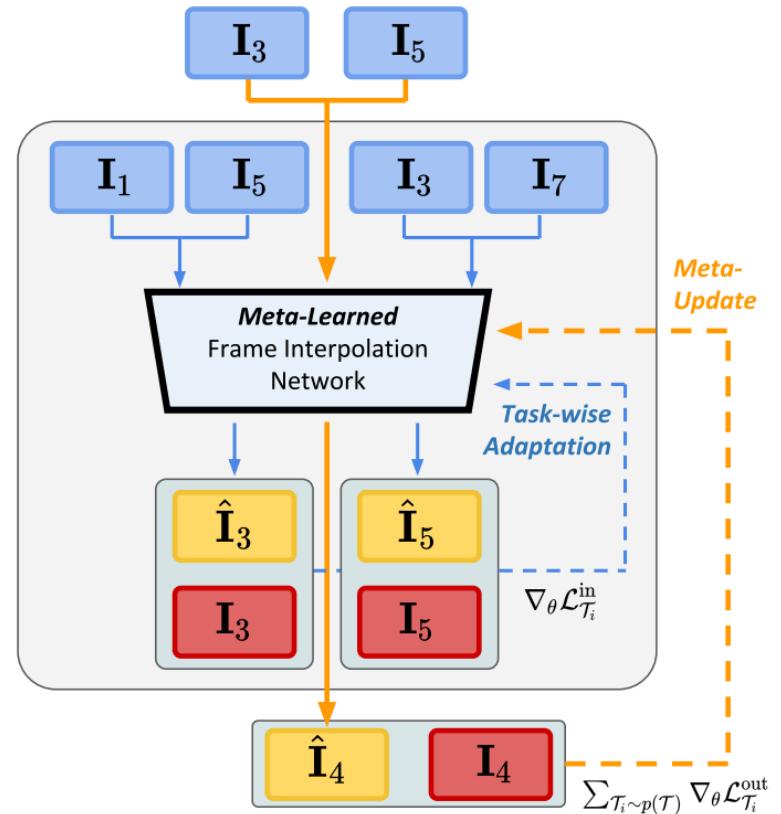
Algorithm 1: Scene-Adaptive Frame Interpolation

Require: $p(\mathcal{T})$: uniform distribution over sequences

Require: α, β : step size hyper-parameters

```

1 Initialize parameters  $\theta$ 
2 while not converged do
3   Sample batch of sequences  $\mathcal{T}_i \sim p(\mathcal{T})$ 
4   foreach  $i$  do
5     Generate triplets
6      $\mathcal{D}_{\mathcal{T}_i} = \{(\mathbf{I}_1, \mathbf{I}_3, \mathbf{I}_5), (\mathbf{I}_3, \mathbf{I}_5, \mathbf{I}_7)\}$  from  $\mathcal{T}_i$ 
7     Compute  $\hat{\mathbf{I}}_3, \hat{\mathbf{I}}_5$  in Eq. (3)
8     Evaluate  $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}^{\text{in}}(f_{\theta})$  using  $\mathcal{L}_{\mathcal{T}_i}$  in Eq. (4)
9     Compute adapted parameters with gradient
10    descent:  $\theta'_i = \theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}_i}^{\text{in}}(f_{\theta})$ 
11    Generate and save triplet  $\mathcal{D}'_{\mathcal{T}_i} = \{(\mathbf{I}_3, \mathbf{I}_4, \mathbf{I}_5)\}$ 
12    from  $\mathcal{T}_i$  for the meta-update
13  end
14  Update  $\theta \leftarrow \theta - \beta \nabla_{\theta} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{\text{out}}(f_{\theta'_i})$  using
15    each  $\mathcal{D}'_{\mathcal{T}_i}$  and  $\mathcal{L}_{\mathcal{T}_i}$  in Eq. (5)
16 end
```



$$\hat{\mathbf{I}}_3 = f_{\theta}(\mathbf{I}_1, \mathbf{I}_5), \quad \hat{\mathbf{I}}_5 = f_{\theta}(\mathbf{I}_3, \mathbf{I}_7), \quad (3)$$

$$\mathcal{L}_{\mathcal{T}_i}^{\text{in}}(f_{\theta}) = \mathcal{L}_{\mathcal{T}_i}(\hat{\mathbf{I}}_3, \mathbf{I}_3) + \mathcal{L}_{\mathcal{T}_i}(\hat{\mathbf{I}}_5, \mathbf{I}_5). \quad (4)$$

$$\mathcal{L}_{\mathcal{T}_i}^{\text{out}}(f_{\theta'_i}) = \mathcal{L}_{\mathcal{T}_i}(f_{\theta'_i}(\mathbf{I}_3, \mathbf{I}_5), \mathbf{I}_4), \quad (5)$$



Experiments: Datasets

- Vimeo90K-Septuplet
 - 91,701 samples
 - 7-frame sequences
 - 448×256 resolution
- Middlebury-OTHERS
 - 12 examples
 - ◆ Remove the sequence that only have two input frames
 - Maximum resolution of 640×480
- HD dataset
 - 1280×544 to 1920×1080
 - The length of the sequences in HD dataset is either 70 or 100

Experiments: Competing methods

- Frame interpolation networks
 - DVF [20], SuperSloMo [16], SepConv [31], and DAIN [3].
- Baseline:
 - Model with pre-trained parameters, provided by the authors.
- Re-trained:
 - Fine-tune each baseline model with VimeoSeptuplet training set.
- Meta-trained:
 - Proposed approach, starting from the baseline model parameters.

[20] Liu et al., Video Frame Synthesis Using Deep Voxel Flow. In ICCV 2017.

[31] Niklaus et al., Video Frame Interpolation via Adaptive Separable Convolution. In ICCV 2017.

[16] Jiang et al., Super SloMo: High Quality Estimation of Multiple Intermediate Frames for Video Interpolation. In CVPR 2018.

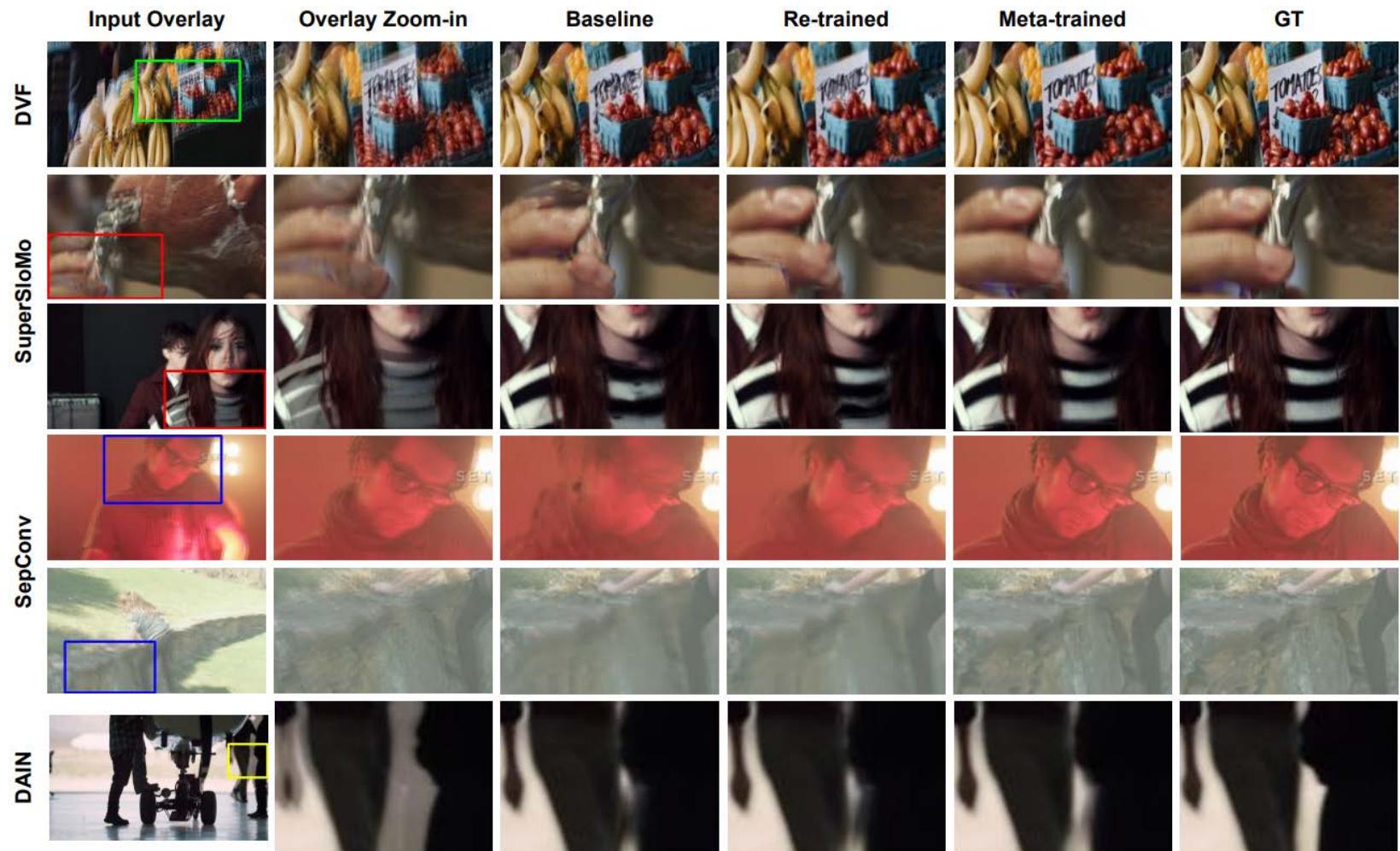
[3] Bao et al., Depth-Aware Video Frame Interpolation. In CVPR 2019.

Experimental results

- Consistent performance boost achieved by the meta-trained.
- Meta training is only done in VimeoSeptuplet dataset. It generalizes well to the other datasets.

Method	VimeoSeptuplet [48]			Middlebury-OTHERS [2]			HD [4]		
	Baseline	Re-trained	Meta-trained	Baseline	Re-trained	Meta-trained	Baseline	Re-trained	Meta-trained
DVF [20]	26.60	32.21 (+0.06)	32.27	26.70	29.51 (+0.19)	29.70	—	—	—
SuperSloMo [16]	30.85	32.76 (+0.36)	33.12	30.28	33.54 (+0.16)	33.70	26.05	29.66 (+0.15)	29.81
SepConv [31]	33.70	33.72 (+0.45)	34.17	35.14	34.90 (+0.91)	35.81	30.04	30.01 (+0.18)	30.19
DAIN [3]	34.73	34.86 (+0.08)	34.94	36.57	36.50 (+0.0)	36.50	30.35	30.45 (+0.06)	30.51

Experiments: Visualization



Ablation studies

- Effects on the number of inner loop updates

# gradient updates	0	1	2	3	5
Naive Fine-tune	34.90	34.90	34.95	34.99	35.03
Meta-trained	34.90	35.81	35.63	35.58	35.45
PSNR gain	—	+0.91	+0.68	+0.59	+0.42

- Effects on the inner loop learning rate

Learning rate α	0	10^{-6}	10^{-5}	10^{-4}
PSNR (dB)	33.72	34.10	34.17	34.15

Conclusions

- Fully utilize the additional information available at test time
- Meta learning algorithm can quickly adapt its parameters according to the input frames
- Scene-adaptive frame interpolation algorithm can be easily employed to any video frame interpolation network

問答題

References

- MAML
 - Chelsea Finn, Pieter Abbeel, and Sergey Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks,” ICML 2017.
- LSTM-based meta-learner
 - Sachin Ravi and Hugo Larochelle, “Optimization as A Model for Few-Shot Learning,” ICLR 2017.
- Relation Network
 - Sung et al., “Learning to Compare: Relation Network for Few-Shot Learning,” CVPR 2018.
- Meta learning for super-resolution
 - Park et al., “Fast Adaptation to Super-Resolution Networks via Meta-Learning,” ECCV 2020.
- Meta learning for video frame interpolation
 - Choi et al., “Scene-Adaptive Video Frame Interpolation vis Meta-Learning,” CVPR 2020.



Thank You for Your Attention!

THANK YOU FOR YOUR ATTENTION

Yen-Yu Lin (林彥宇)

Email: lin@cs.nctu.edu.tw

URL: <https://www.cs.nctu.edu.tw/members/detail/lin>