Meta-Learning Recipe, Black-Box Adaptation, Optimization-Based Approaches

CS 330

Course Reminders

HW1 due Weds 10/9

First paper presentations & discussions on Wednesday!

Plan for Today

- Recap **probabilistic formulation** of meta-learning
- **General recipe** of meta-learning algorithms
- **Black-box adaptation** approaches
- **Optimization-based** meta-learning

} Topic of Homework 1!

} Part of Homework 2

Recap from Last Time

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

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

$$\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_1^{\text{tr}}, \mathcal{D}_1^{\text{ts}}), \ldots, (\mathcal{D}_n^{\text{tr}}, \mathcal{D}_n^{\text{ts}})\}$$

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

$$\mathcal{D}_i^{\text{ts}} = \{(x_1^i, y_1^i), \ldots, (x_l^i, y_l^i)\}$$

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

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

$$\Updownarrow$$

$$\phi^\star = f_{\theta^\star}(\mathcal{D}^{\text{tr}})$$

meta-learning: $\theta^\star = \max_\theta \sum_{i=1}^{n} \log p(\phi_i|\mathcal{D}_i^{\text{ts}})$

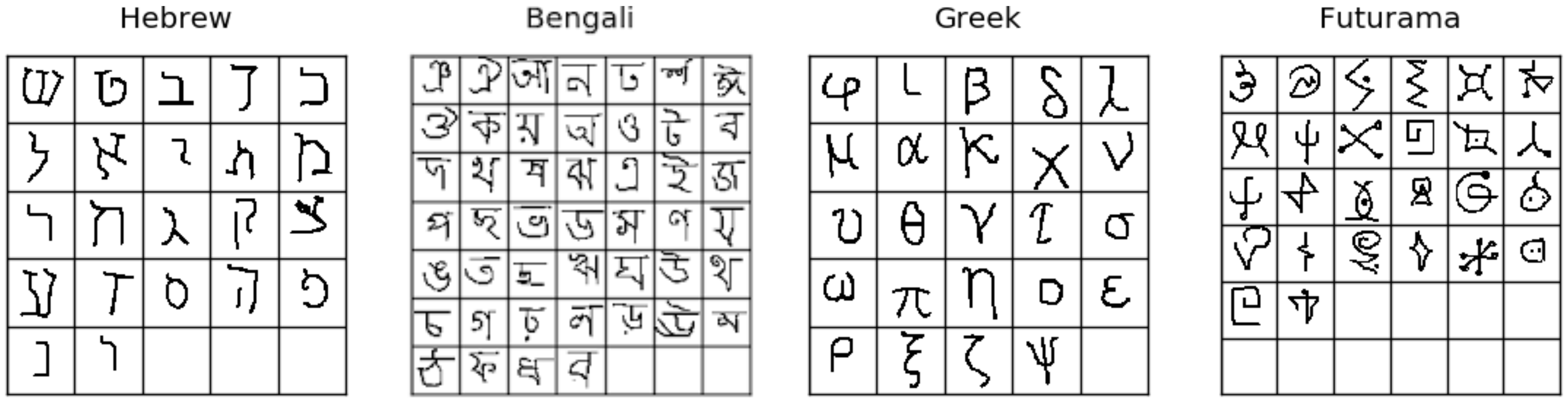
where $\phi_i = f_\theta(\mathcal{D}_i^{\text{tr}})$

General recipe

How to *evaluate* a meta-learning algorithm

the Omniglot dataset Lake et al. Science 2015

1623 characters from **50 different alphabets**



…

many classes, few examples

the "transpose" of MNIST

statistics more reflective
of **the real world**

20 instances of each character

Proposes both **few-shot discriminative** & **few-shot generative** problems

Initial few-shot learning approaches w/ **Bayesian models, non-parametrics**

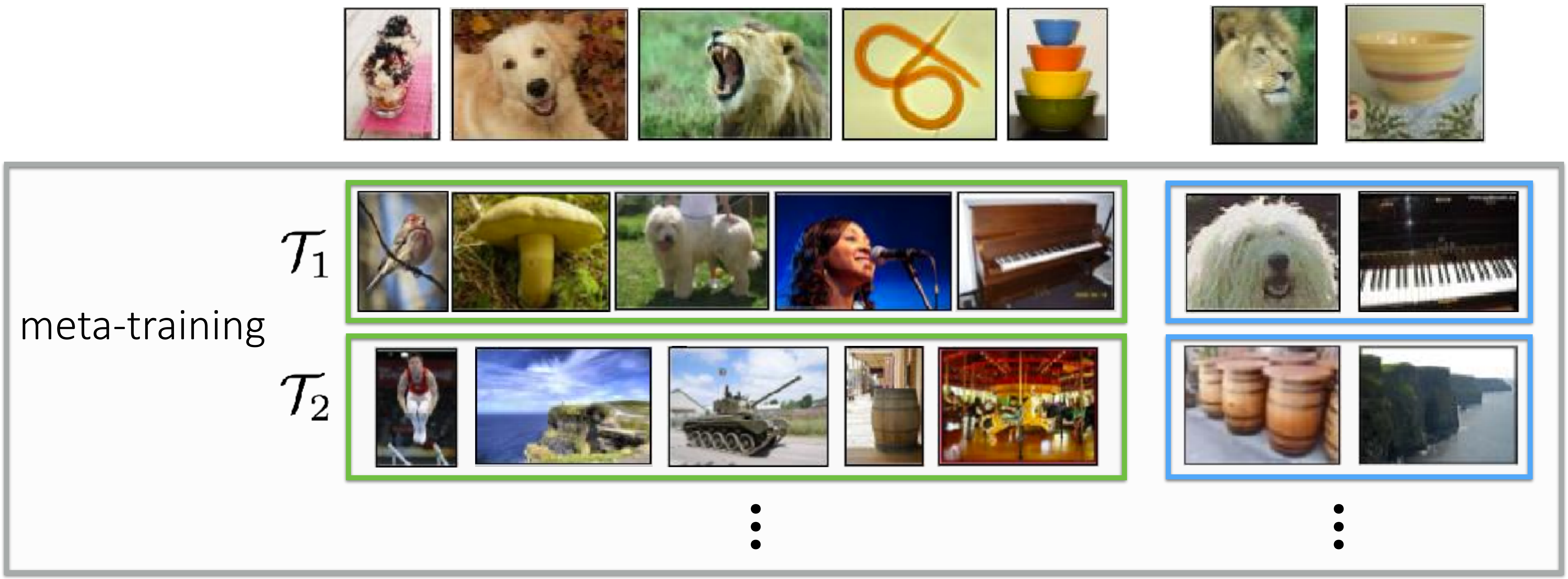
Fei-Fei et al. '03 Lake et al. '11 Salakhutdinov et al. '12 Lake et al. '13

Other datasets used for **few-shot image recognition**: MiniImagenet, CIFAR, CUB, CelebA, others

General recipe

How to *evaluate* a meta-learning algorithm

5-way, **1**-shot image classification (MiniImagenet)



Given 1 example of 5 classes:

Classify new examples

held-out classes

meta-training

$\mathcal{T}_1$

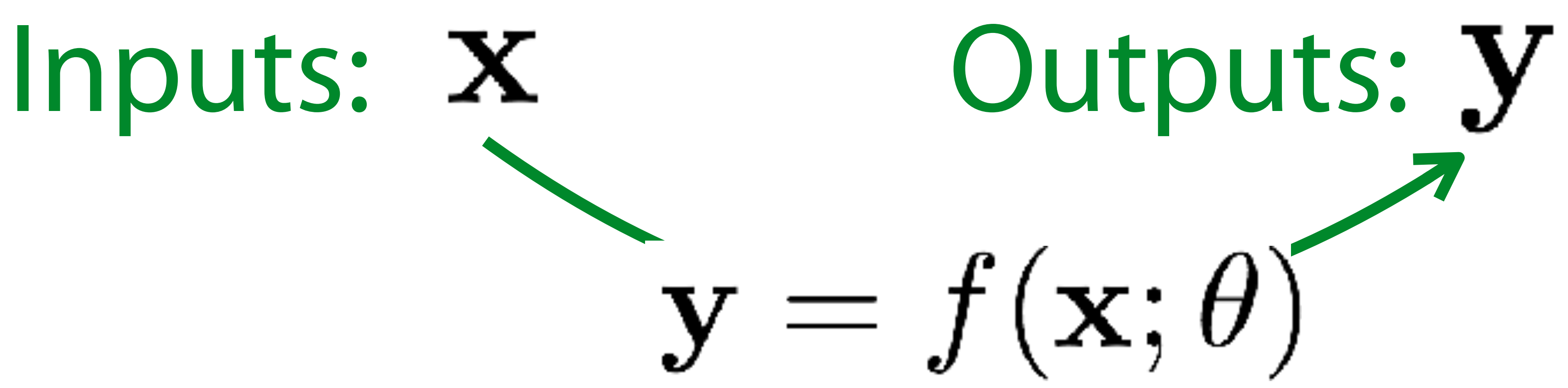
$\mathcal{T}_2$

training classes

Can replace image classification with: **regression**, **language generation**, **skill learning**, **any ML problem**

The Meta-Learning Problem: The Mechanistic View

Supervised Learning:



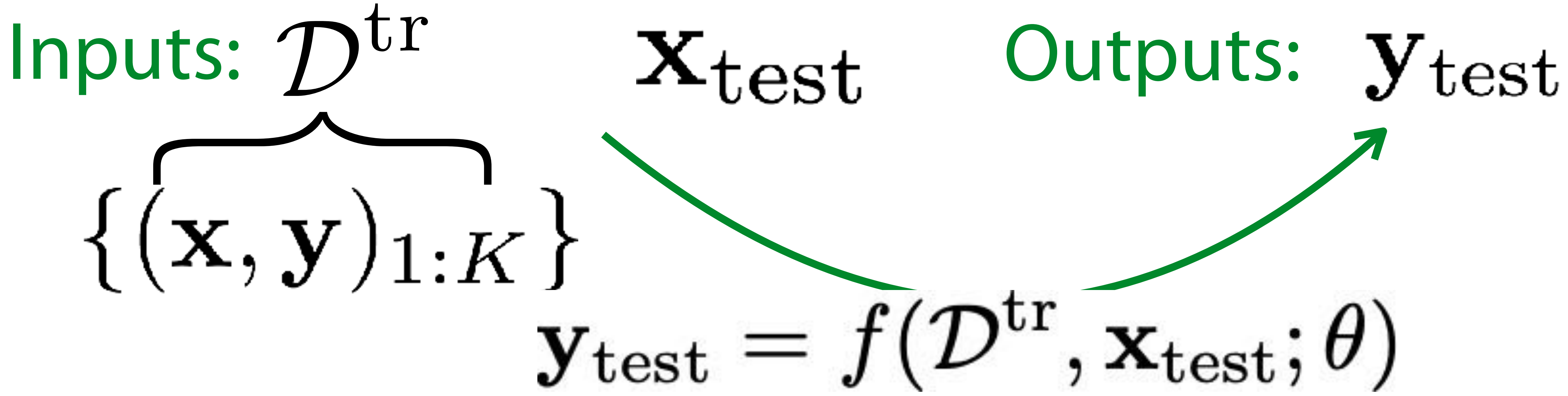
Inputs: $\mathbf{x}$

Outputs: $\mathbf{y}$

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Data: $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})_i\}$

Meta-Supervised Learning:



Inputs: $\underbrace{\mathcal{D}^{\mathrm{tr}}}_{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}$ $\mathbf{x}_{\mathrm{test}}$

Outputs: $\mathbf{y}_{\mathrm{test}}$

$$\mathbf{y}_{\mathrm{test}} = f(\mathcal{D}^{\mathrm{tr}}, \mathbf{x}_{\mathrm{test}}; \theta)$$

Data: $\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_i\}$

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

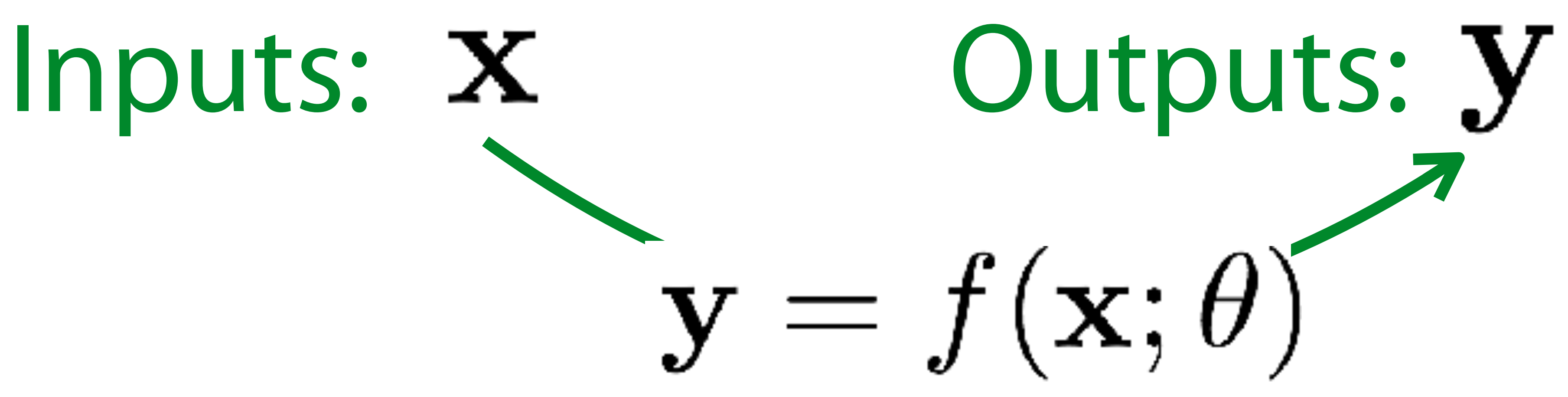
Why is this view useful?

Reduces the problem to the design & optimization of *f*.

Finn. Learning to Learn with Gradients. PhD thesis 2018

The Meta-Learning Problem: The Probabilistic View

Supervised Learning:



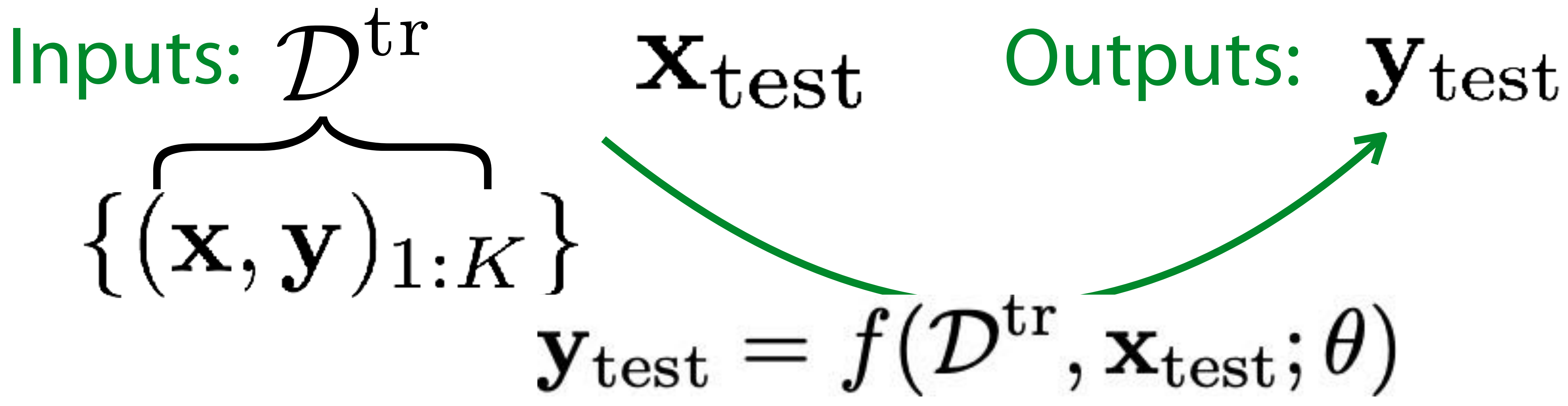
Inputs: $\mathbf{x}$ Outputs: $\mathbf{y}$

$$\mathbf{y} = f(\mathbf{x}; \theta)$$

Data: $\mathcal{D} = \{(\mathbf{x}, \mathbf{y})_i\}$

As inference: $p(\theta | \mathcal{D})$

Meta-Supervised Learning:



Inputs: $\mathcal{D}^{\mathrm{tr}}$ $\mathbf{x}_{\mathrm{test}}$ Outputs: $\mathbf{y}_{\mathrm{test}}$

$$\underbrace{\{(\mathbf{x}, \mathbf{y})_{1:K}\}}_{\mathcal{D}^{\mathrm{tr}}}$$

$$\mathbf{y}_{\mathrm{test}} = f(\mathcal{D}^{\mathrm{tr}}, \mathbf{x}_{\mathrm{test}}; \theta)$$

Data: $\mathcal{D}_{\text{meta-train}} = \{\mathcal{D}_i\}$

$$\mathcal{D}_i : \{(\mathbf{x}, \mathbf{y})_j\}$$

As inference: $p(\phi_i | \mathcal{D}_i^{\mathrm{tr}}, \theta)$ $\max_\theta \sum_i \log p(\phi_i | \mathcal{D}_i^{\mathrm{ts}})$

Finn. Learning to Learn with Gradients. PhD thesis 2018

General recipe

How to *design* a meta-learning algorithm

1. Choose a form of $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$

2. Choose how to optimize θ w.r.t. max-likelihood objective using $\mathcal{D}_{\text{meta-train}}$

Can we treat $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$ as an **inference** problem?

Neural networks are good at inference.

Plan for Today

- Recap **probabilistic formulation** of meta-learning
- **General recipe** of meta-learning algorithms
- **Black-box adaptation approaches**
- **Optimization-based** meta-learning

} Topic of Homework 1!

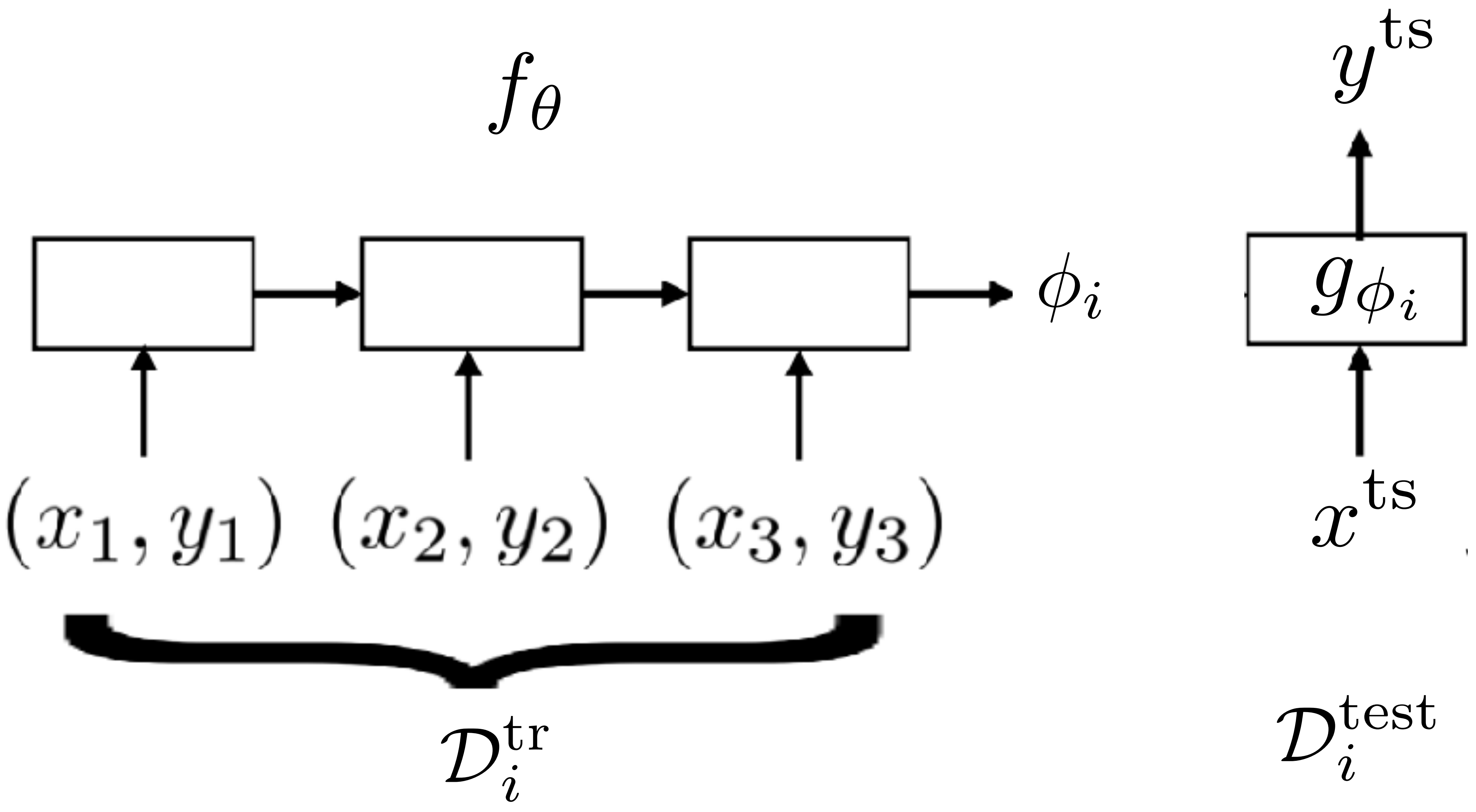
} Part of Homework 2

Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$

For now: Use **deterministic** (point estimate) $\phi_i = f_\theta(\mathcal{D}_i^{\mathrm{tr}})$

(Bayes will come back later)


f_θ
ϕ_i
(x_1, y_1) (x_2, y_2) (x_3, y_3)
$\mathcal{D}_i^{\mathrm{tr}}$
y^{ts}
g_{ϕ_i}
x^{ts}
$\mathcal{D}_i^{\mathrm{test}}$

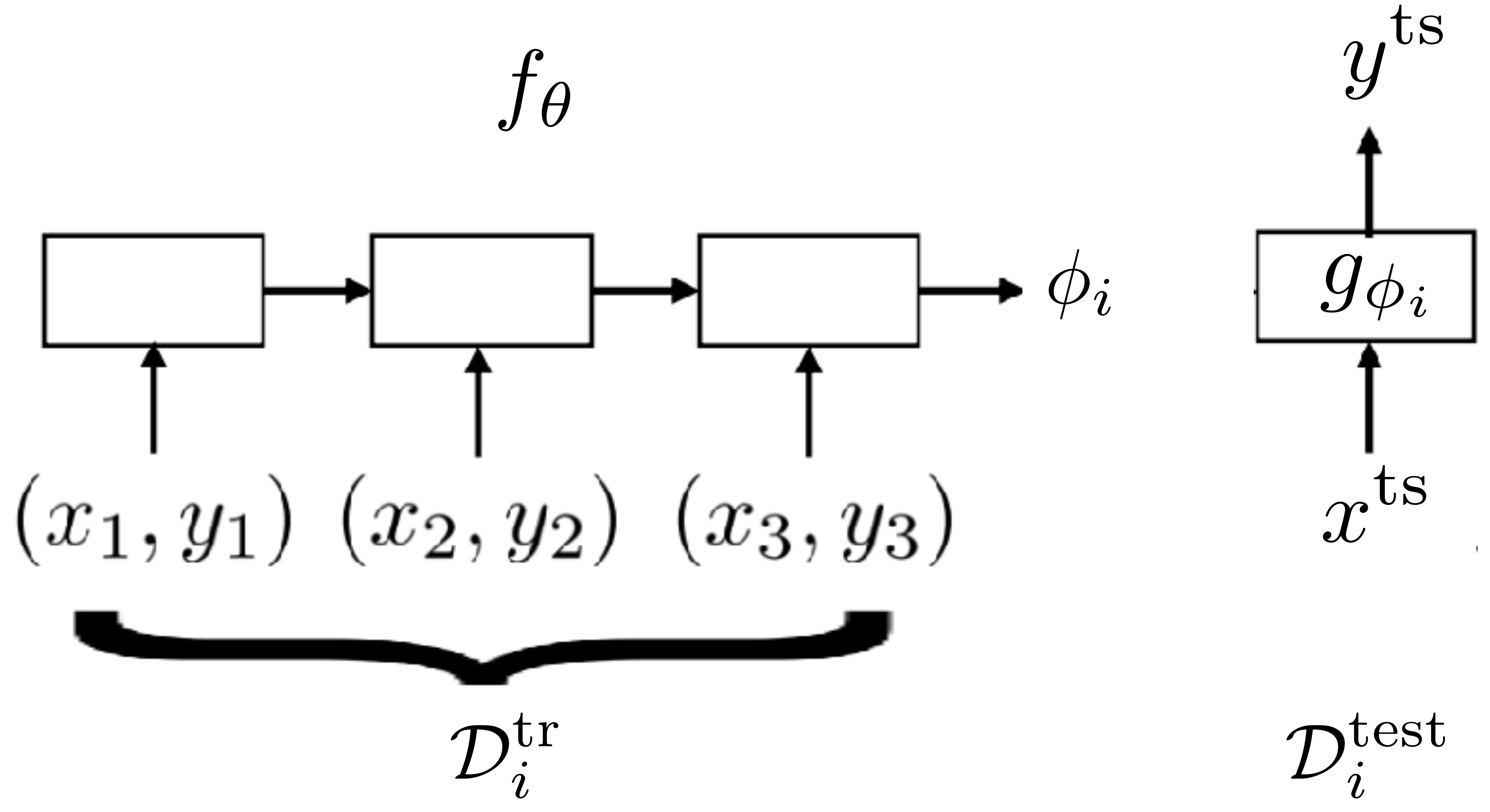
Train with standard supervised learning!

$$\max_\theta \sum_{\mathcal{T}_i} \underbrace{\sum_{(x,y)\sim\mathcal{D}_i^{\mathrm{test}}} \log g_{\phi_i}(y|x)}_{\mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})}$$

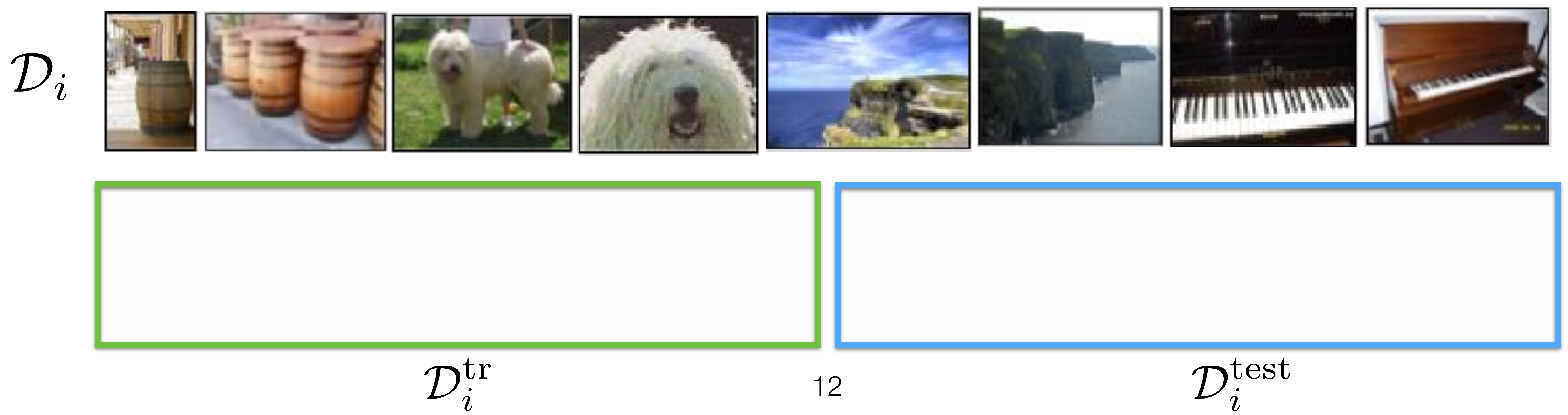
$$\max_\theta \sum_{\mathcal{T}_i} \mathcal{L}(f_\theta(\mathcal{D}_i^{\mathrm{tr}}), \mathcal{D}_i^{\mathrm{test}})$$

Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$

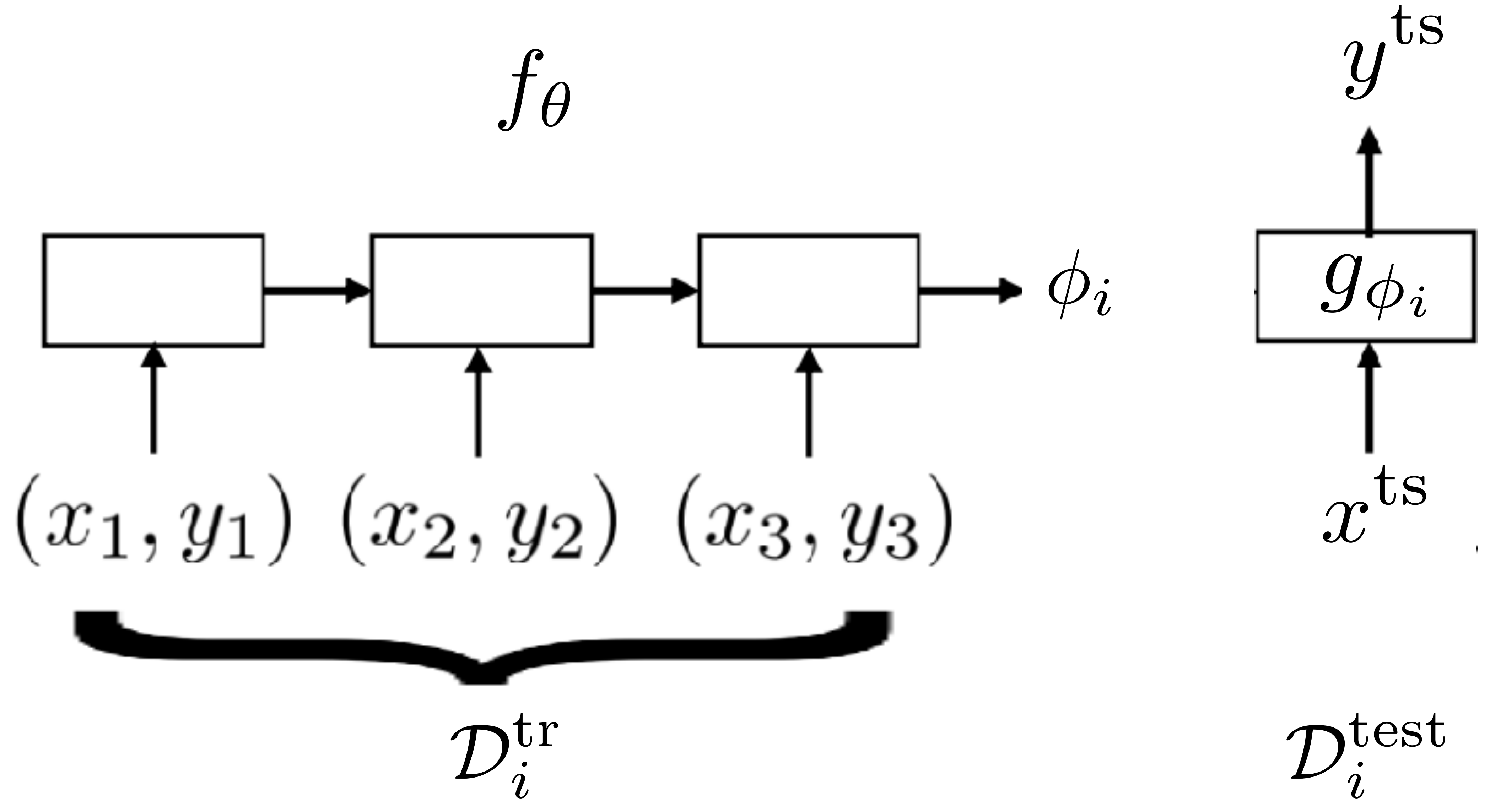


1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$

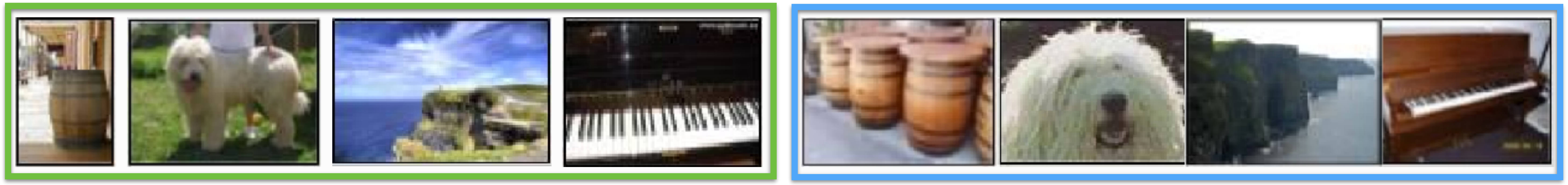


Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i|\mathcal{D}_i^{\mathrm{tr}}, \theta)$



1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\mathrm{tr}}, \mathcal{D}_i^{\mathrm{test}}$ from $\mathcal{D}_i$
3. Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\mathrm{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\mathrm{test}})$



$\mathcal{D}_i^{\mathrm{tr}}$ $\mathcal{D}_i^{\mathrm{test}}$

Black-Box Adaptation

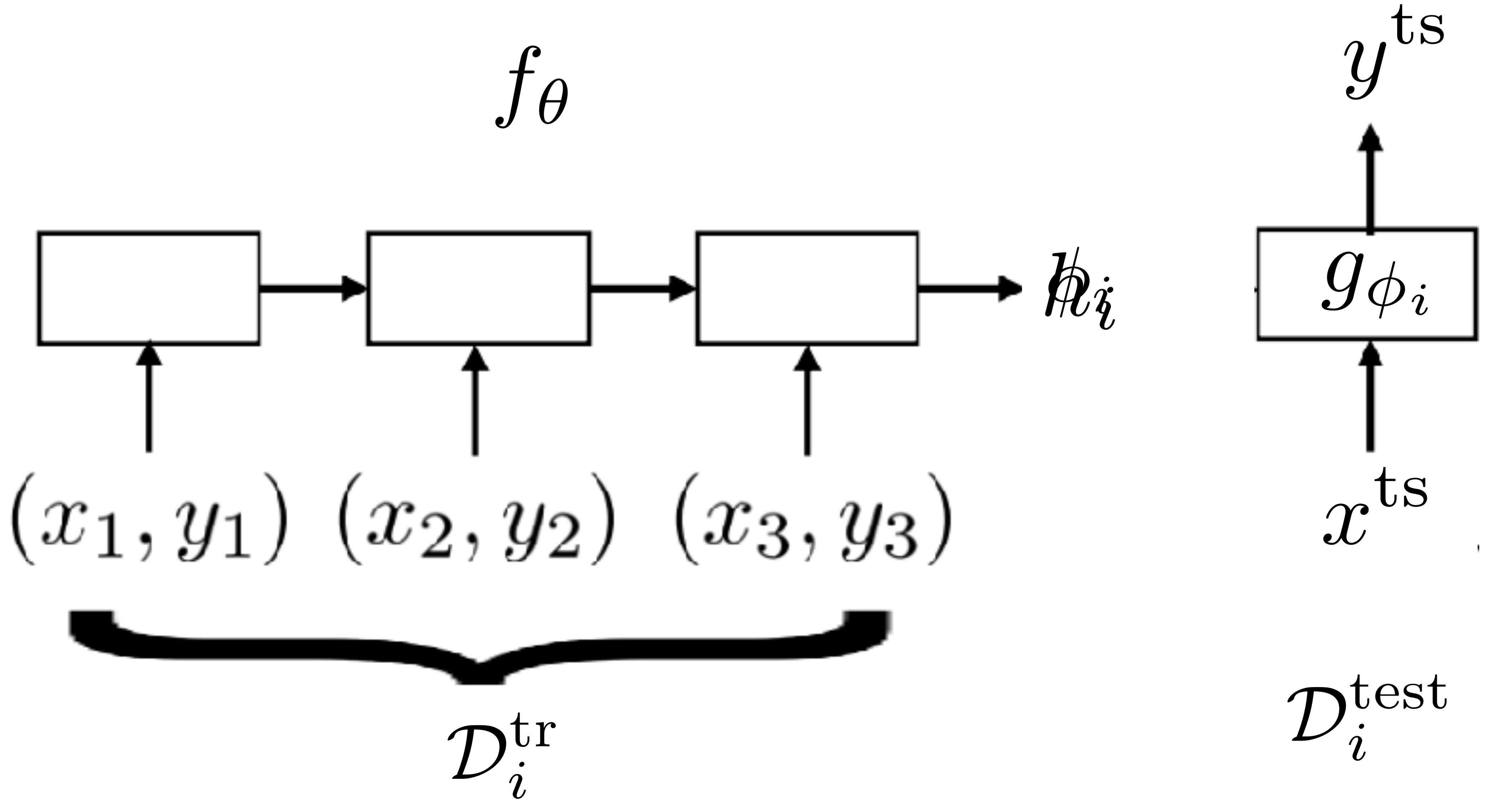
Key idea: Train a neural network to represent $p(\phi_i | \mathcal{D}_i^{\text{tr}}, \theta)$

Challenge

Outputting all neural net parameters does not seem scalable?

Idea: Do not need to output **all** parameters of neural net, only sufficient statistics

(Santoro et al. MANN, Mishra et al. SNAIL)

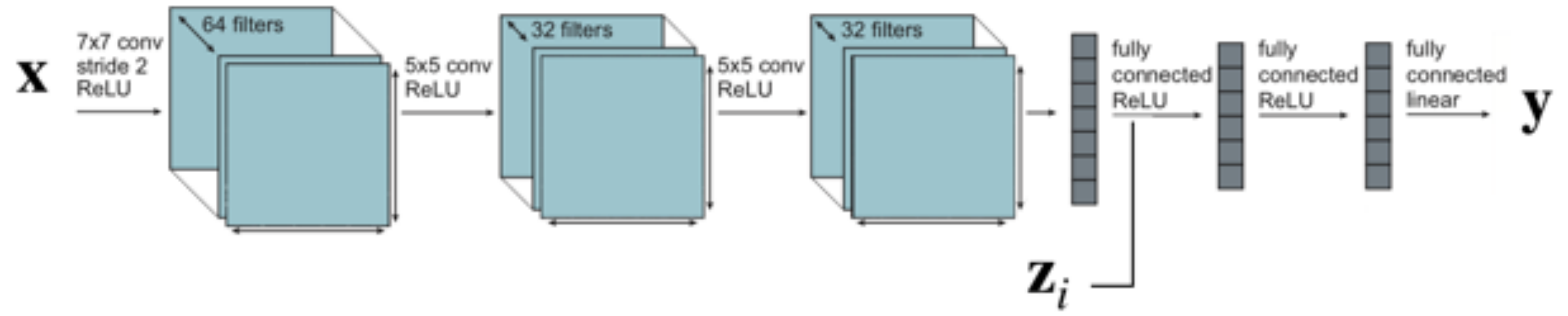


low-dimensional vector h_i

represents contextual task information

$$\phi_i = \{h_i, \theta_g\}$$

recall:

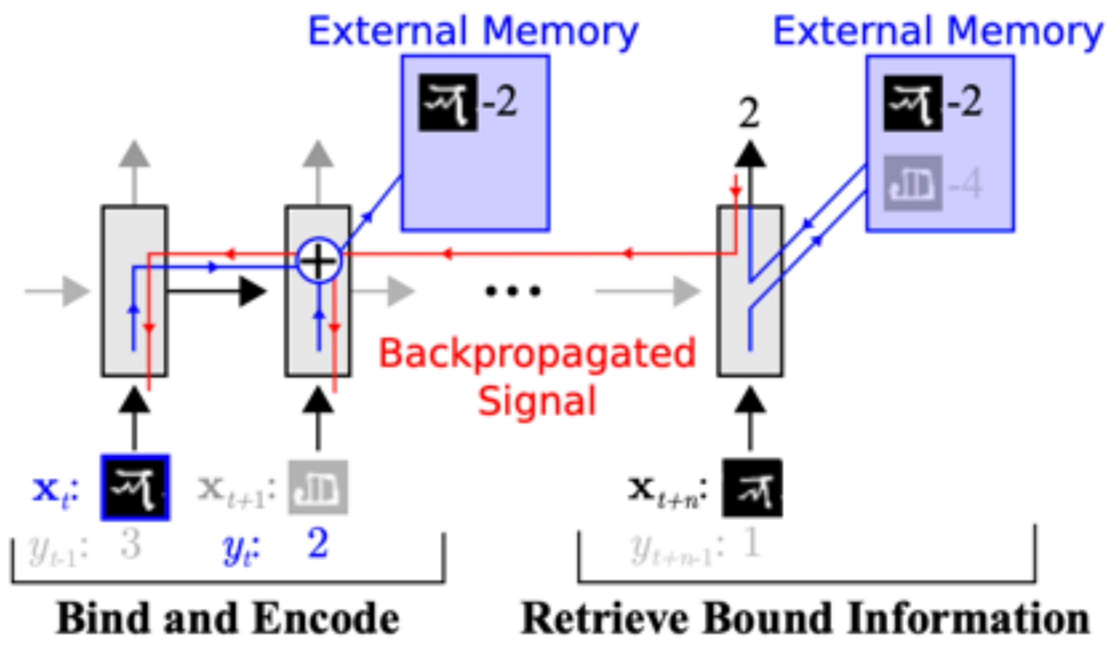


general form: $y^{\text{ts}} = f_\theta(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$

What architecture should we use for f_θ**?**

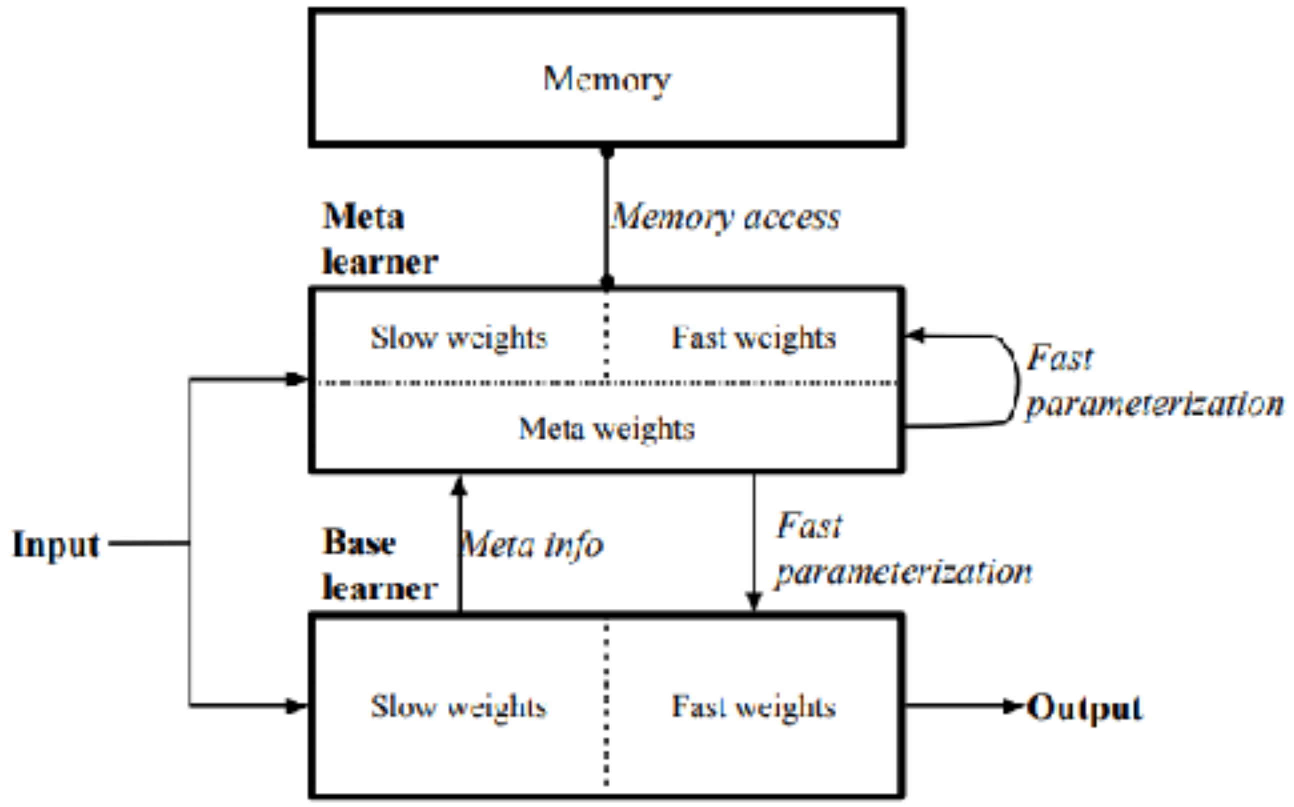
Black-Box Adaptation

LSTMs or Neural turing machine (NTM)



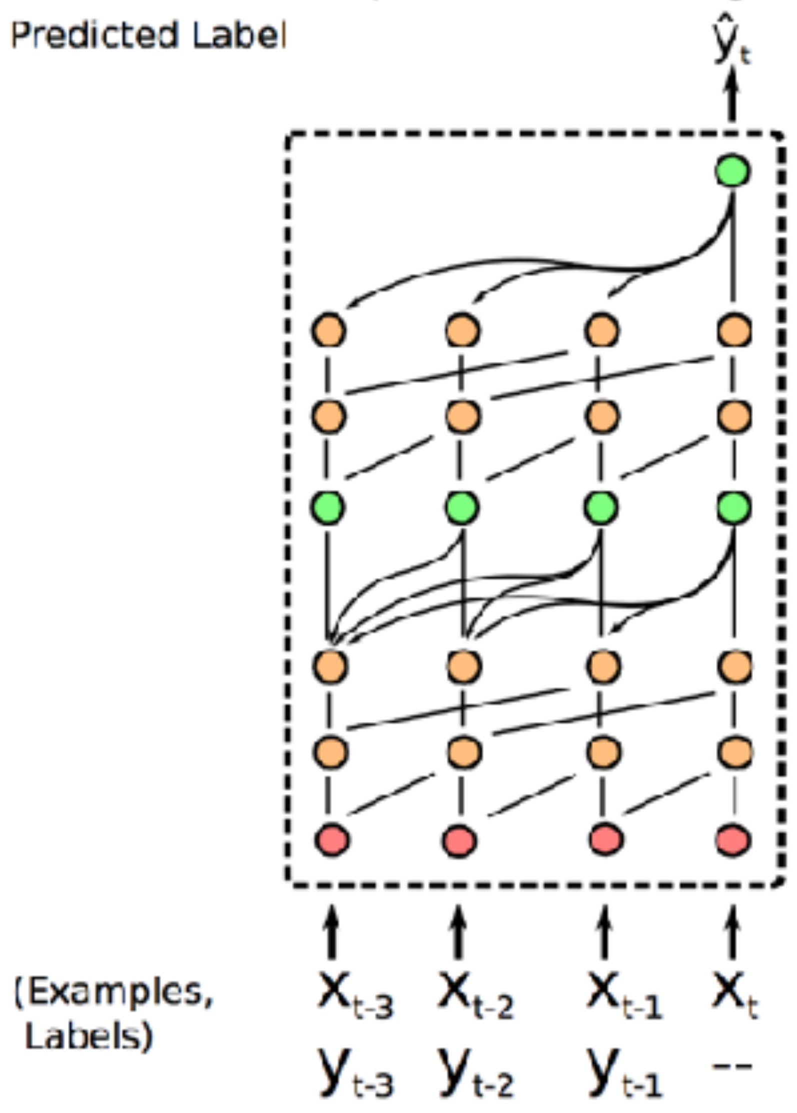
Meta-Learning with Memory-Augmented Neural Networks
Santoro, Bartunov, Botvinick, Wierstra, Lillicrap. ICML '16

Other external memory mechanisms

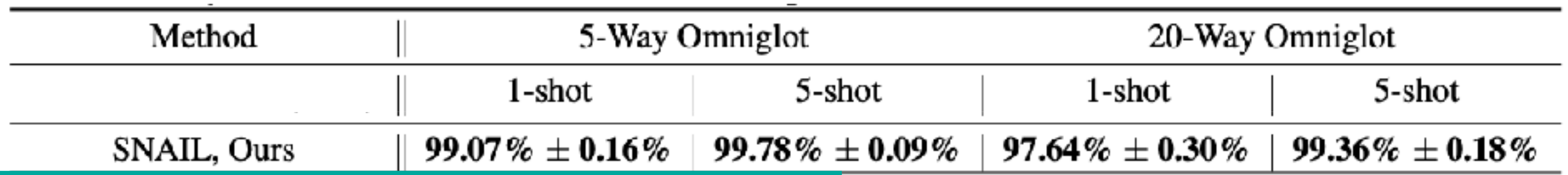


Meta Networks
Munkhdalai, Yu. ICML '17

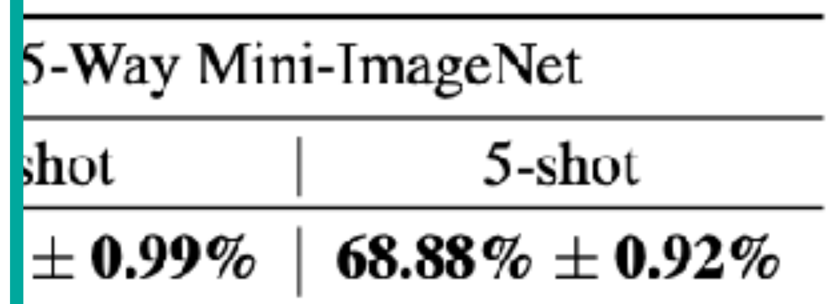
Convolutions & attention


Predicted Label
$\hat{y}_t$
(Examples, Labels)
x_{t-3} x_{t-2} x_{t-1} x_t
y_{t-3} y_{t-2} y_{t-1} --

A Simple Neural Attentive Meta-Learner
Mishra, Rohaninejad, Chen, Abbeel. ICLR '18

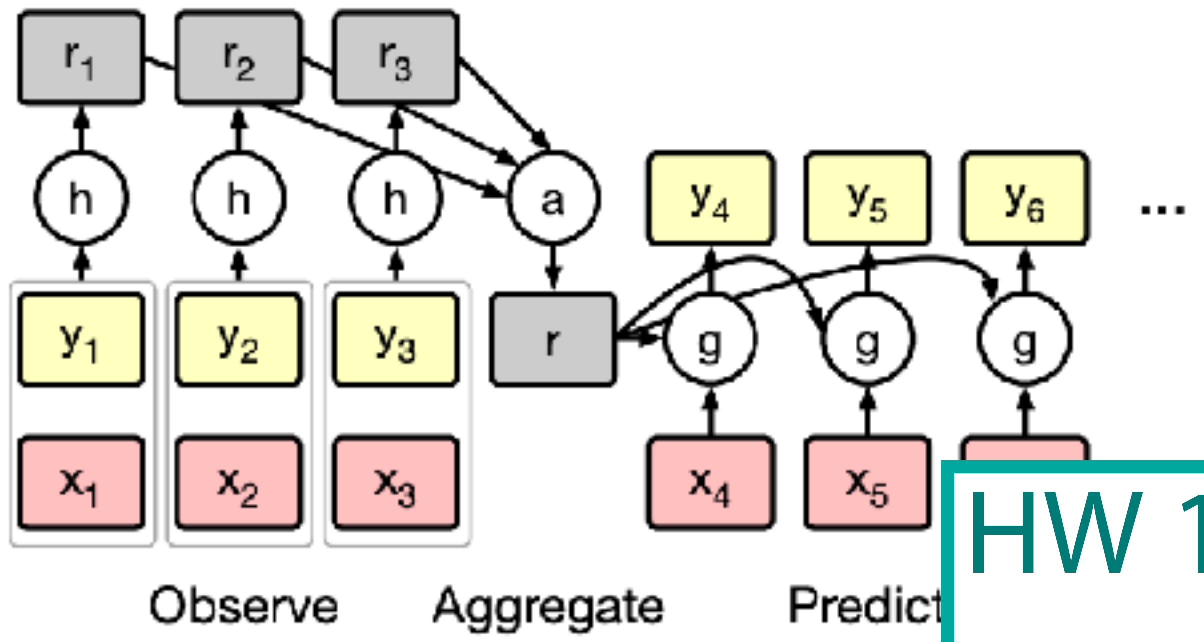


Method	5-Way Omniglot		20-Way Omniglot	
	1-shot	5-shot	1-shot	5-shot
SNAIL, Ours	**99.07% ± 0.16%**	**99.78% ± 0.09%**	**97.64% ± 0.30%**	**99.36% ± 0.18%**



5-Way Mini-ImageNet	
shot	5-shot
± 0.99%	**68.88% ± 0.92%**

Feedforward + average

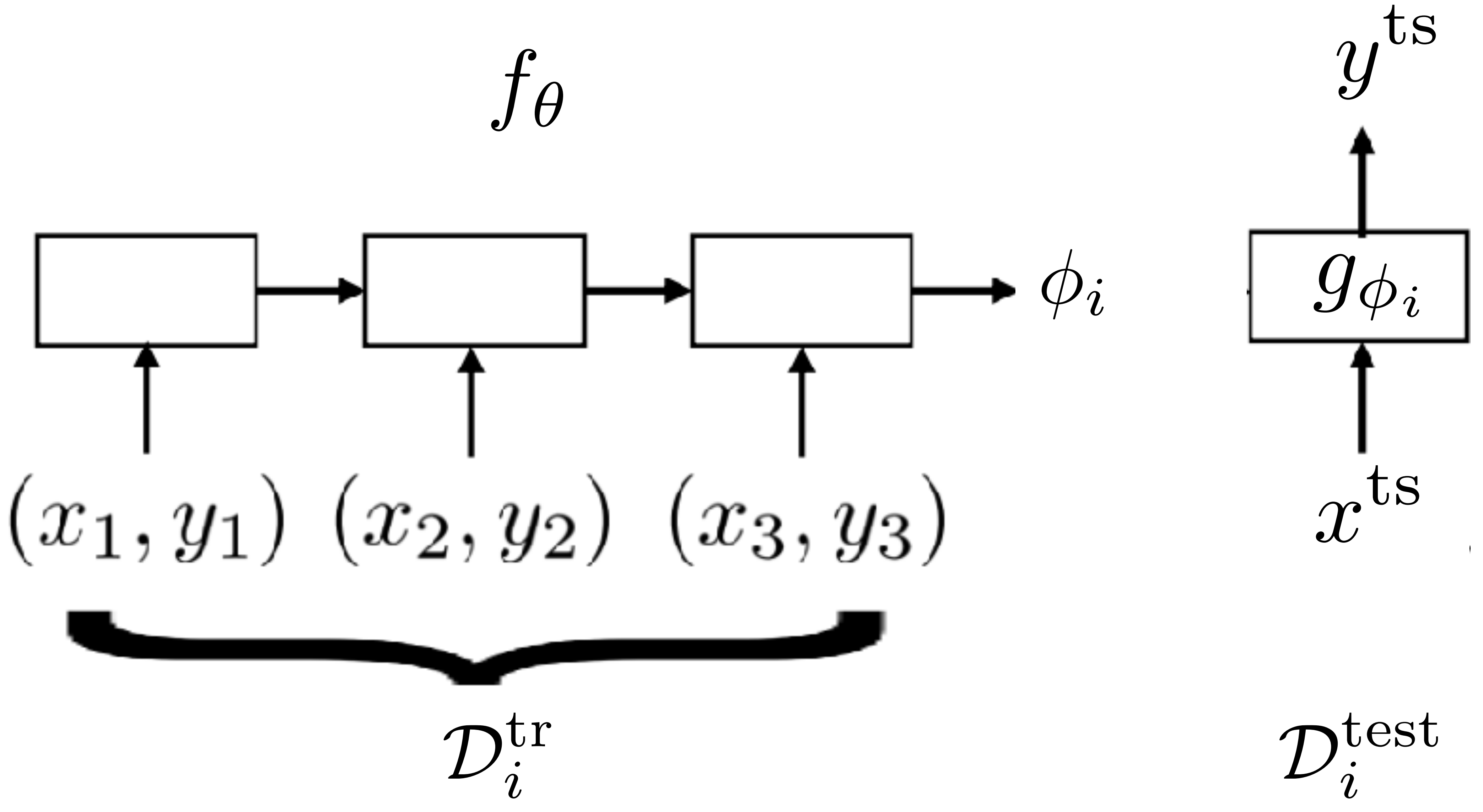


Conditional Neural Processes. Garnelo, Rose
Ramalho, Saxton, Shanahan, Teh, Rezende, E

HW 1:
- implement data processing
- implement simple black-box meta-learner
- train few-shot Omniglot classifier

Black-Box Adaptation

Key idea: Train a neural network to represent $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$


f_θ
ϕ_i
(x_1, y_1) (x_2, y_2) (x_3, y_3)
$\mathcal{D}_i^{\text{tr}}$
y^{ts}
g_{ϕ_i}
x^{ts}
$\mathcal{D}_i^{\text{test}}$

+ **expressive**

+ easy to combine with **variety of learning problems** (e.g. SL, RL)

- **complex model** w/ **complex task**: **challenging optimization** problem

- often **data-inefficient**

How else can we represent $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$?

Is there a way to infer **all parameters** in a scalable way?

What if we treat it as an **optimization** procedure?

Plan for Today

- Recap **probabilistic formulation** of meta-learning
- **General recipe** of meta-learning algorithms
- **Black-box** adaptation approaches
- **Optimization-based meta-learning**

} Topic of Homework 1!

} Part of Homework 2

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

$$\max_{\phi_i} \log p(\mathcal{D}_i^{\mathrm{tr}} | \phi_i) + \log p(\phi_i | \theta)$$

Meta-parameters θ serve as a prior.   What form of prior?

One successful form of prior knowledge: **initialization** for **fine-tuning**

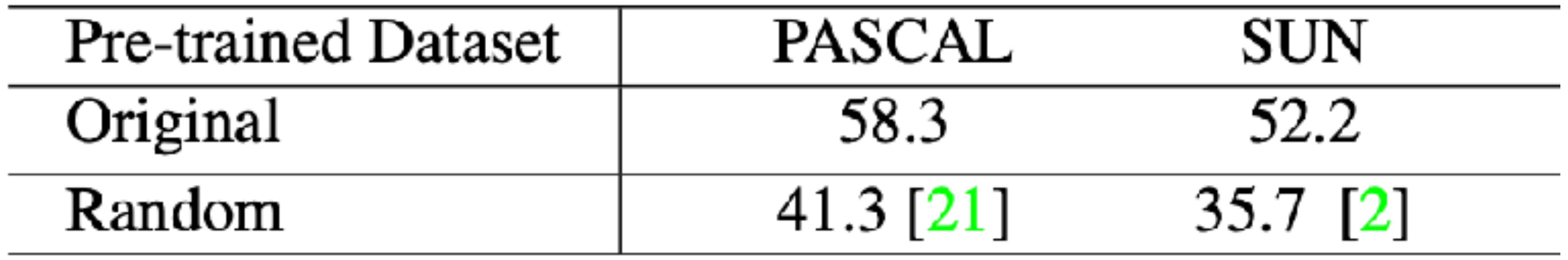
Optimization-Based Inference

pre-trained parameters

Fine-tuning $$\phi \leftarrow \theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}^{\mathrm{tr}})$$

training data for new task

(typically for many gradient steps)



Pre-trained Dataset	PASCAL	SUN
Original	58.3	52.2
Random	41.3 [21]	35.7 [2]

What makes ImageNet good for transfer learning? Huh, Agrawal, Efros. '16

Where do you get the pre-trained parameters?

- ImageNet classification
- Models trained on large language corpora (BERT, LMs)
- Other unsupervised learning techniques
- Whatever large, diverse dataset you might have

Pre-trained models often available online.

Some common practices

- Fine-tune with a smaller learning rate
- Lower learning rate for lower layers
- Freeze earlier layers, gradually unfreeze
- Reinitialize last layer
- Search over hyperparameters via cross-val
- Architecture choices matter (e.g. ResNets)

Optimization-Based Inference

pre-trained parameters

Fine-tuning $$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\mathrm{tr}})$$

training data
for new task

(typically for many gradient steps)

Universal Langauge Model Fine-Tuning for Text Classification. Howard, Ruder. ‘18

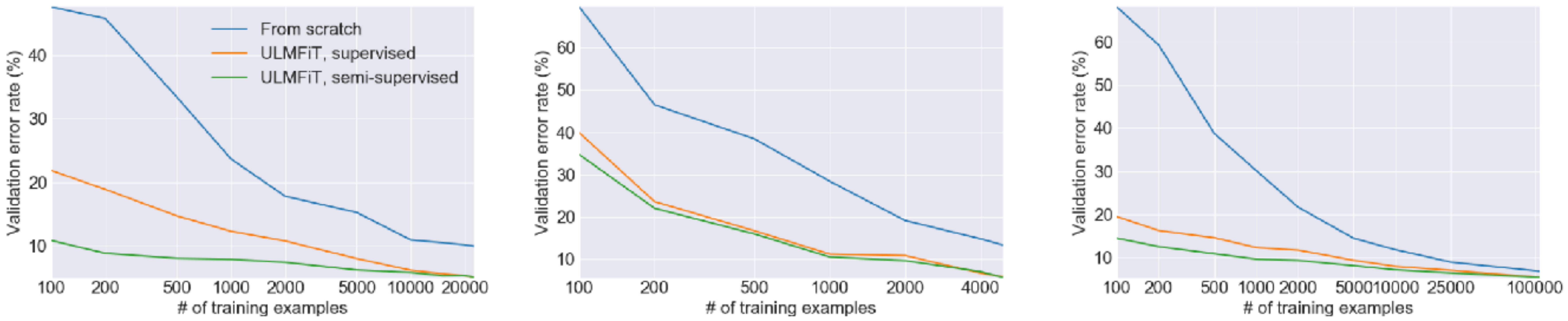


Figure 3: Validation error rates for supervised and semi-supervised ULMFiT vs. training from scratch with different numbers of training examples on IMDb, TREC-6, and AG (from left to right).

Fine-tuning less effective with **very small datasets**.

Optimization-Based Inference

pre-trained parameters

Fine-tuning

[test-time]

$$\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\mathrm{tr}})$$

training data for new task

Meta-learning $$\min_\theta \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\mathrm{tr}}), \mathcal{D}_i^{\mathrm{ts}})$$

Key idea: Over many tasks, learn parameter vector θ that transfers via fine-tuning

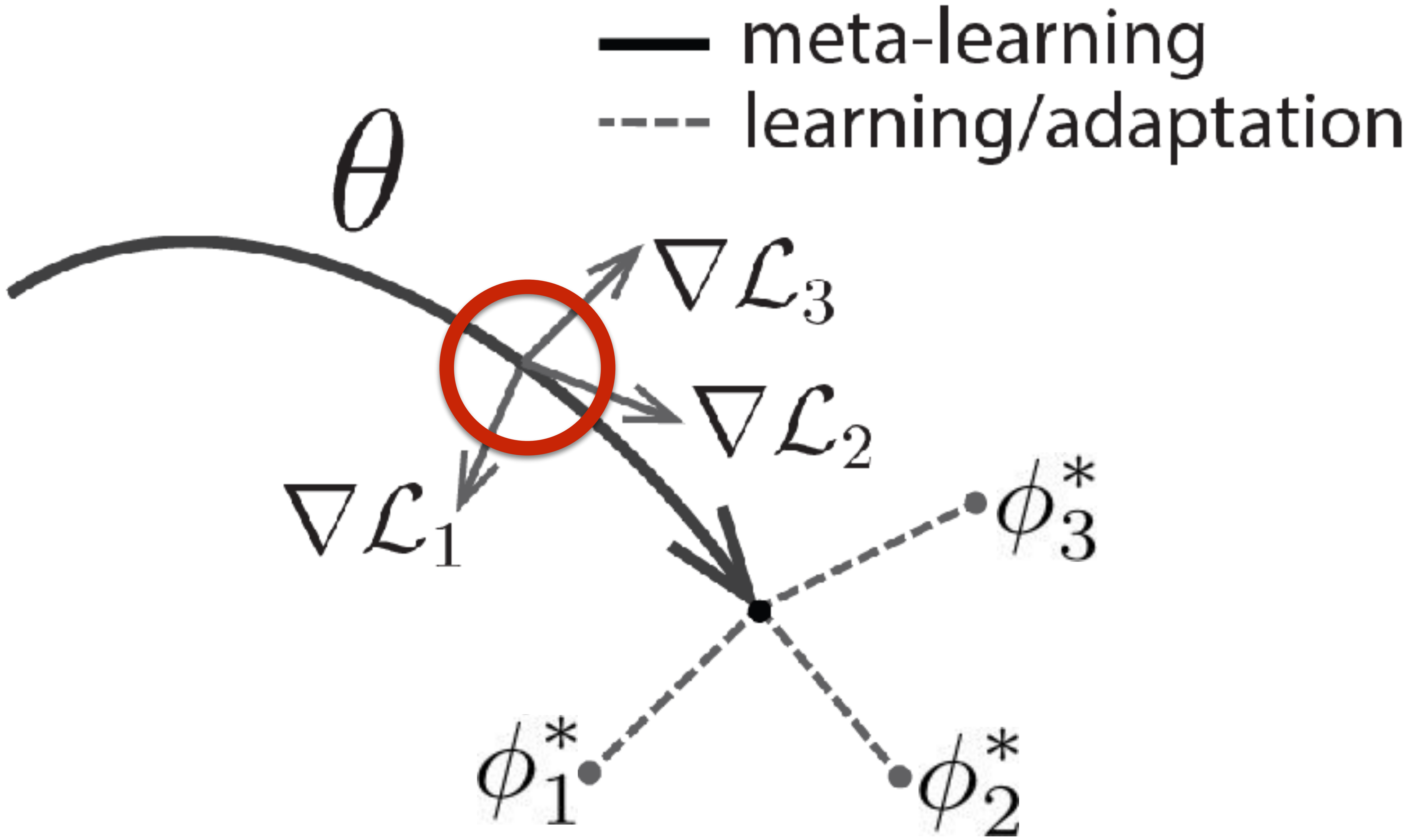
Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017

Optimization-Based Inference

$$\min_{\theta} \sum_{\text{task } i} \mathcal{L}(\theta - \alpha \nabla_{\theta} \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}}), \mathcal{D}_i^{\text{ts}})$$

θ parameter vector being meta-learned

ϕ_i^* optimal parameter vector for task i



Model-**A**gnostic **M**eta-**L**earning

Finn, Abbeel, Levine. Model-Agnostic Meta-Learning. ICML 2017

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

General Algorithm:

~~Amortized approach~~ Optimization-based approach

1. Sample task $\mathcal{T}_i$ *(or mini batch of tasks)*
2. Sample disjoint datasets $\mathcal{D}_i^{\text{tr}}, \mathcal{D}_i^{\text{test}}$ from $\mathcal{D}_i$
3. ~~Compute $\phi_i \leftarrow f_\theta(\mathcal{D}_i^{\text{tr}})$~~ Optimize $\phi_i \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$
4. Update θ using $\nabla_\theta \mathcal{L}(\phi_i, \mathcal{D}_i^{\text{test}})$

—> brings up **second-order** derivatives

Do we need to compute the full Hessian? 😱

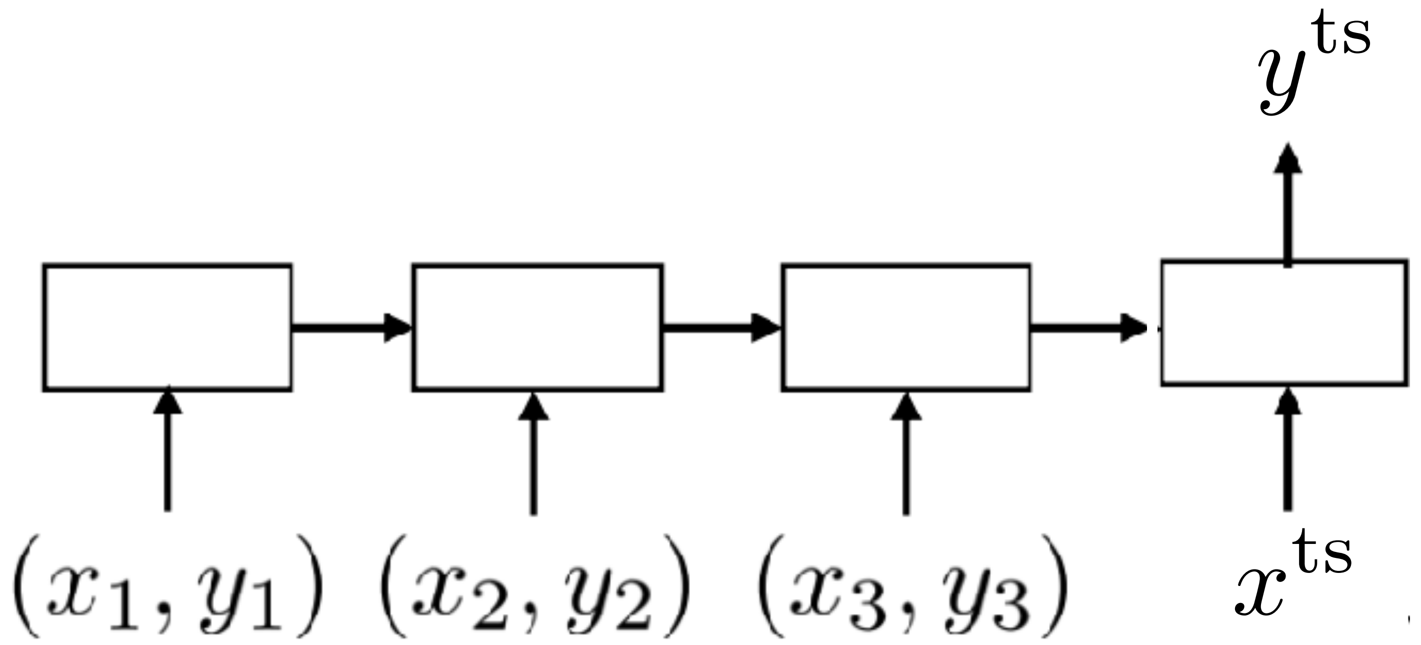
-> whiteboard

Do we get higher-order derivatives with more inner gradient steps?

Optimization vs. Black-Box Adaptation

Black-box adaptation

general form: $y^{\text{ts}} = f_\theta(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$



Model-agnostic meta-learning

$$\begin{aligned} y^{\text{ts}} &= f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= f_{\phi_i}(x^{\text{ts}}) \end{aligned}$$

$$\text{where } \phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})$$

MAML can be viewed as **computation graph**, with embedded gradient operator

Note: Can mix & match components of computation graph

Learn initialization but replace gradient update with learned network

$$\text{where } \phi_i = \theta - \alpha \cancel{\nabla_\theta \mathcal{L}(\theta, \mathcal{D}_i^{\text{tr}})}$$
$$f(\theta, \mathcal{D}_i^{\text{tr}}, \nabla_\theta \mathcal{L})$$

Ravi & Larochelle ICLR ’17
(actually precedes MAML)

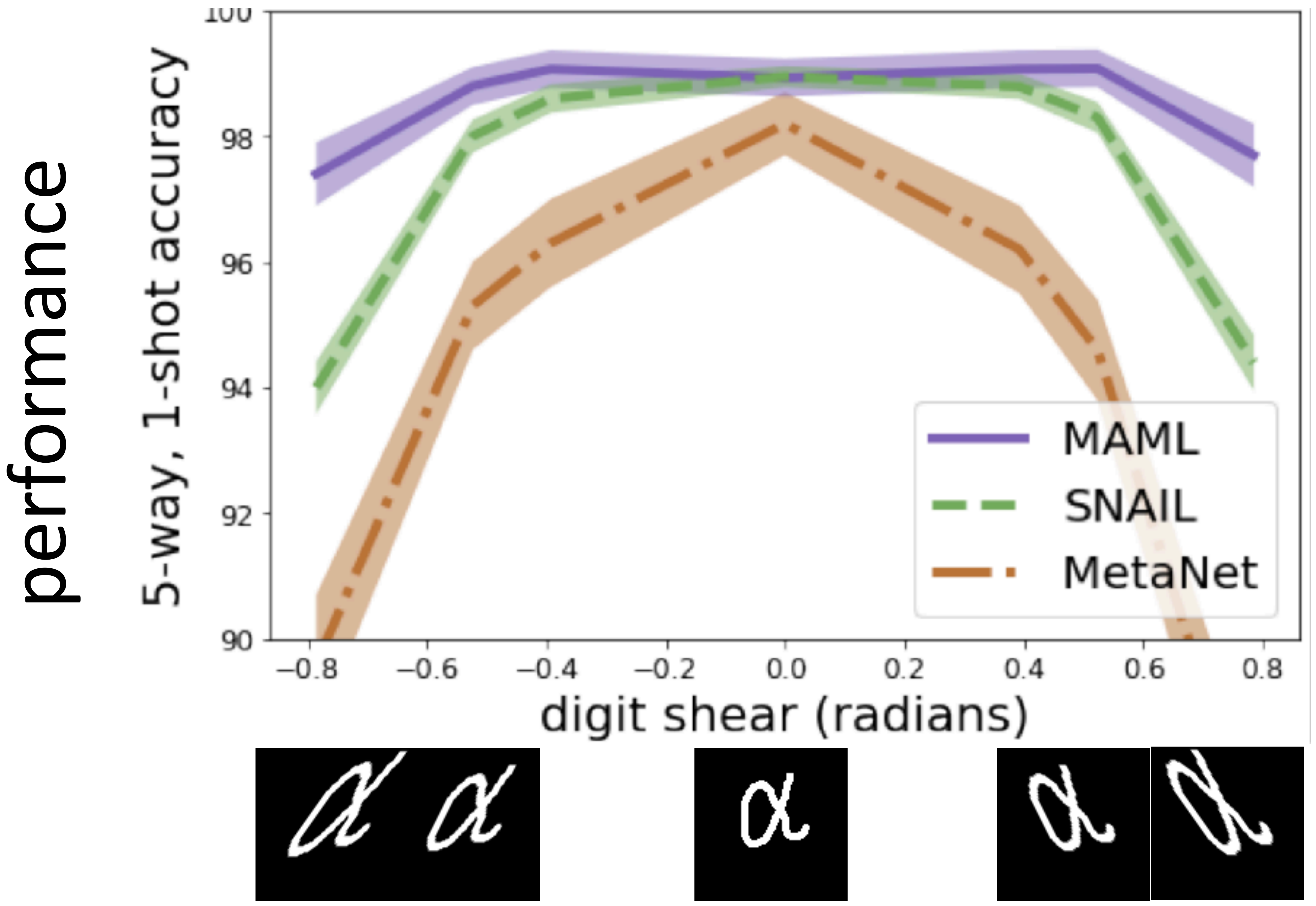
This **computation graph view** of meta-learning will come back again!

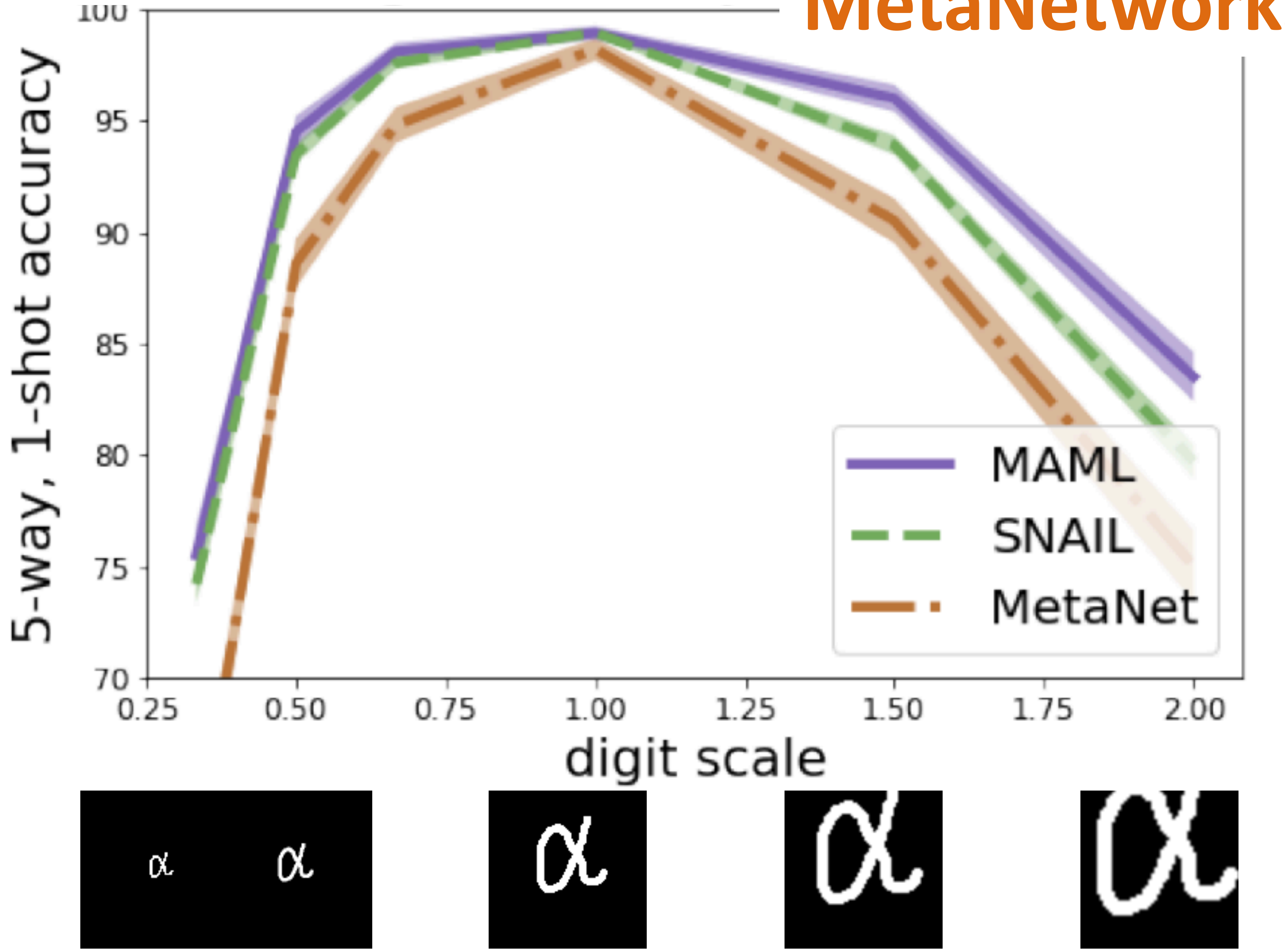
Optimization vs. Black-Box Adaptation

How well can learning procedures generalize to similar, but extrapolated tasks?

MAML SNAIL, MetaNetworks

Omniglot image classification





Does this structure come at a cost?

Finn & Levine ICLR ’18

Black-box adaptation

$$y^{\text{ts}} = f_\theta(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Optimization-based (MAML)

$$y^{\text{ts}} = f_{\text{MAML}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}})$$

Does this structure come at a cost?

For a sufficiently deep f,

MAML function can approximate any function of $\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}$

Finn & Levine, ICLR 2018

Assumptions:

- nonzero α
- loss function gradient does not lose information about the label
- datapoints in $\mathcal{D}_i^{\text{tr}}$ are unique

Why is this interesting?

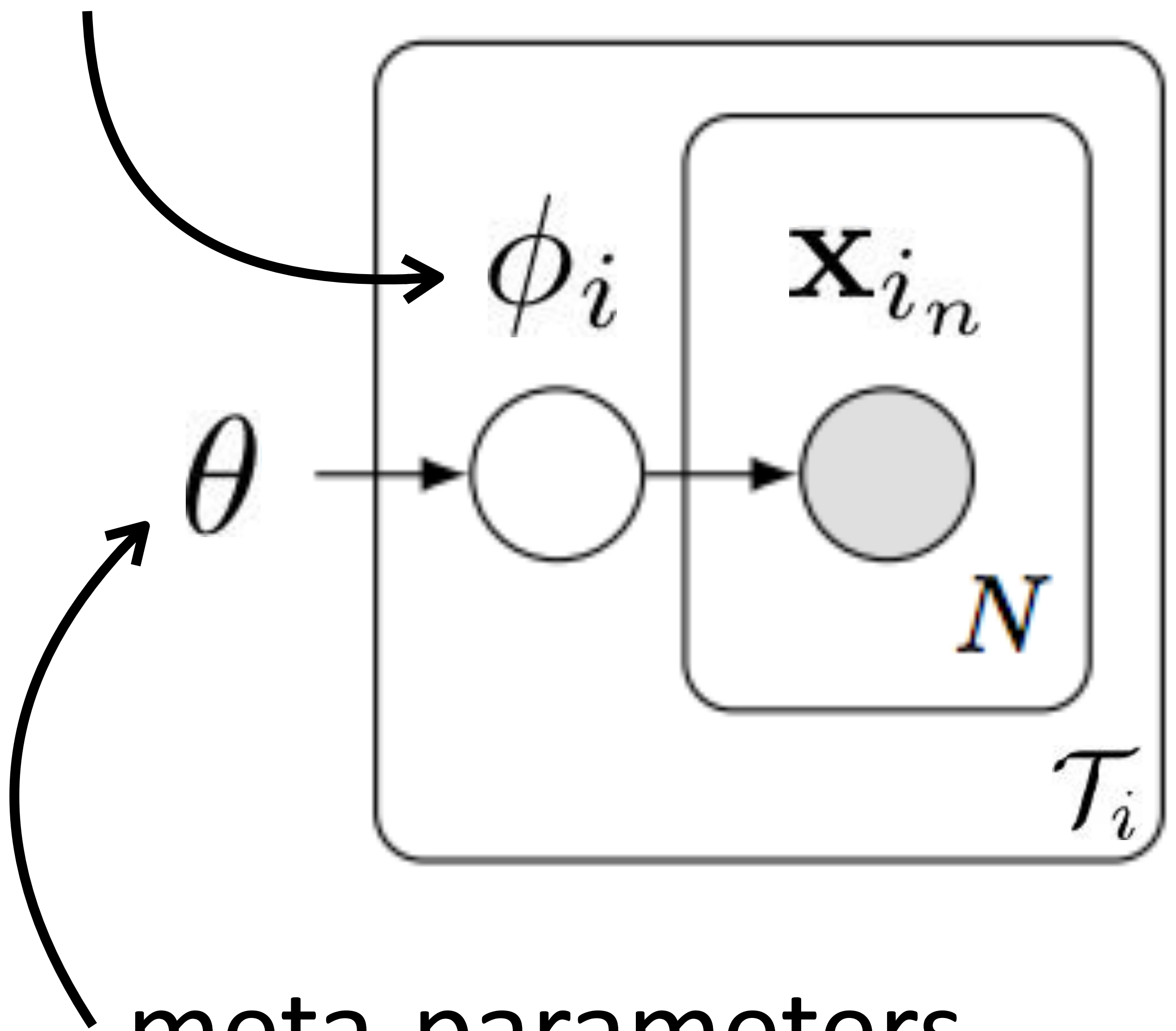
MAML has benefit of inductive bias without losing expressive power.

Probabilistic Interpretation of Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Meta-parameters θ serve as a prior. One form of prior knowledge: **initialization** for **fine-tuning**

task-specific parameters



meta-parameters

$$\max_\theta \log \prod_i p(\mathcal{D}_i|\theta)$$

$$= \log \prod_i \int p(\mathcal{D}_i|\phi_i)p(\phi_i|\theta)d\phi_i \quad \text{(empirical Bayes)}$$

$$\approx \log \prod_i p(\mathcal{D}_i|\hat{\phi}_i)p(\hat{\phi}_i|\theta)$$

↖MAP estimate

How to compute MAP estimate?

Gradient descent with **early stopping** = **MAP inference** under **Gaussian prior** with mean at initial parameters [Santos '96]

(exact in linear case, approximate in nonlinear case)

MAML approximates hierarchical Bayesian inference. Grant et al. ICLR '18

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Meta-parameters θ serve as a prior. One form of prior knowledge: **initialization** for **fine-tuning**

Gradient-descent + early stopping (MAML): implicit Gaussian prior $\phi \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{\text{tr}})$

Other forms of priors?

Gradient-descent with **explicit Gaussian prior** $\phi \leftarrow \min_{\phi'} \mathcal{L}(\phi', \mathcal{D}^{\text{tr}}) + \frac{\lambda}{2} ||\theta - \phi'||^2$

Rajeswaran et al. implicit MAML '19

Bayesian linear regression *on learned features* Harrison et al. ALPaCA '18

Closed-form *or* **convex optimization** *on learned features*

ridge regression, logistic regression
Bertinetto et al. R2-D2 '19

support vector machine
Lee et al. MetaOptNet '19

Current **SOTA** on few-shot image classification

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Challenges

How to choose architecture that is effective for inner gradient-step?

Idea: Progressive neural architecture search + MAML

(Kim et al. Auto-Meta)

- finds highly non-standard architecture (deep & narrow)
- different from architectures that work well for standard supervised learning

MiniImagenet, 5-way 5-shot

MAML, basic architecture: **63.11%**

MAML + AutoMeta: **74.65%**

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Challenges

Bi-level optimization can exhibit instabilities.

Idea: Automatically learn inner vector learning rate, tune outer learning rate

(Li et al. Meta-SGD, Behl et al. AlphaMAML)

Idea: Optimize only a subset of the parameters in the inner loop

(Zhou et al. DEML, Zintgraf et al. CAVIA)

Idea: Decouple inner learning rate, BN statistics per-step (Antoniou et al. MAML++)

Idea: Introduce context variables for increased expressive power.

(Finn et al. bias transformation, Zintgraf et al. CAVIA)

Takeaway: a range of simple tricks that can help optimization significantly

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Challenges

Backpropagating through many inner gradient steps is compute- & memory-intensive.

Idea: [Crudely] approximate $\frac{d\phi_i}{d\theta}$ as identity

(Finn et al. first-order MAML '17, Nichol et al. Reptile '18)

Takeaway: works for simple few-shot problems, but (anecdotally) not for more complex meta-learning problems.

Can we compute the meta-gradient *without differentiating through the optimization path*?

-> whiteboard

Idea: Derive meta-gradient using the implicit function theorem

(Rajeswaran, Finn, Kakade, Levine. Implicit MAML '19)

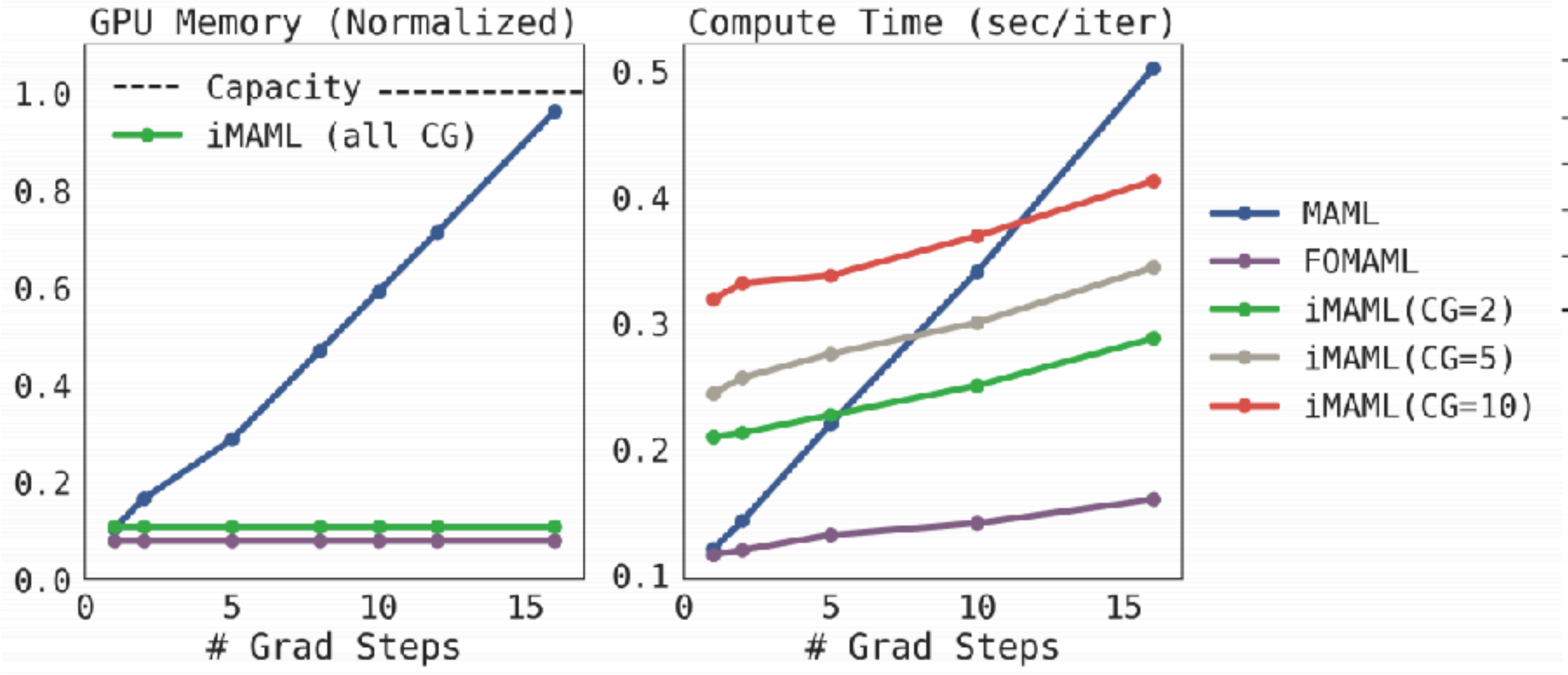
Optimization-Based Inference

Can we compute the meta-gradient *without differentiating through the optimization path*?

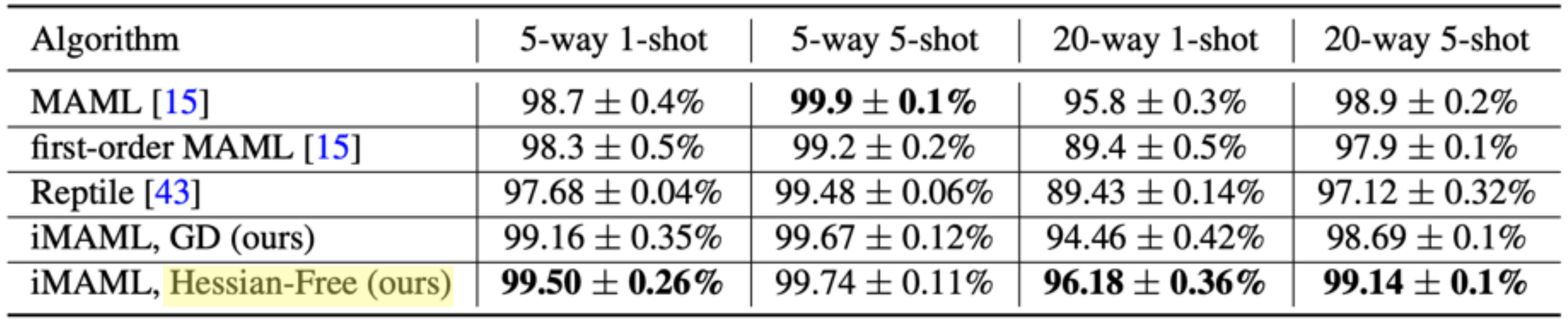
Idea: Derive meta-gradient using the implicit function theorem

(Rajeswaran, Finn, Kakade, Levine. Implicit MAML)

Memory and computation trade-offs



Allows for second-order optimizers in inner loop



Algorithm	5-way 1-shot	5-way 5-shot	20-way 1-shot	20-way 5-shot
MAML [15]	98.7 ± 0.4%	**99.9 ± 0.1%**	95.8 ± 0.3%	98.9 ± 0.2%
first-order MAML [15]	98.3 ± 0.5%	99.2 ± 0.2%	89.4 ± 0.5%	97.9 ± 0.1%
Reptile [43]	97.68 ± 0.04%	99.48 ± 0.06%	89.43 ± 0.14%	97.12 ± 0.32%
iMAML, GD (ours)	99.16 ± 0.35%	99.67 ± 0.12%	94.46 ± 0.42%	98.69 ± 0.1%
iMAML, Hessian-Free (ours)	**99.50 ± 0.26%**	99.74 ± 0.11%	**96.18 ± 0.36%**	**99.14 ± 0.1%**

A very recent development (NeurIPS ’19)

(thus, all the typical caveats with recent work)

Optimization-Based Inference

Key idea: Acquire ϕ_i through optimization.

Takeaways: Construct *bi-level optimization* problem.

+ positive inductive bias at the start of meta-learning

+ consistent procedure, tends to extrapolate better

+ maximally expressive with sufficiently deep network

+ model-agnostic (easy to combine with your favorite architecture)

- typically requires second-order optimization

- usually compute and/or memory intensive

Next time:

Wednesday: Applications of meta-learning, multi-task learning to: imitation learning, generative models, drug discovery, machine translation

student presentations & discussions

Monday: Non-parametric few-shot learners, comparison of approaches

lecture