Course Roadmap

(start of week 5!)

So far: Multi-task & transfer learning basics

Core meta-learning algorithms

Core unsupervised pre-training algorithms

Next two weeks:
(more advanced topics!)

Advanced meta-learning topics

- Task construction (today)
- Large-scale meta-optimization (Weds)

Bayesian meta-learning

Plan for Today

Brief Recap of Meta-Learning & Supervised Task Construction

Memorization in Meta-Learning
- When it arises
- Potential solutions

Meta-Learning without Tasks Provided
- Unsupervised Meta-Learning
- Semi-Supervised Meta-Learning

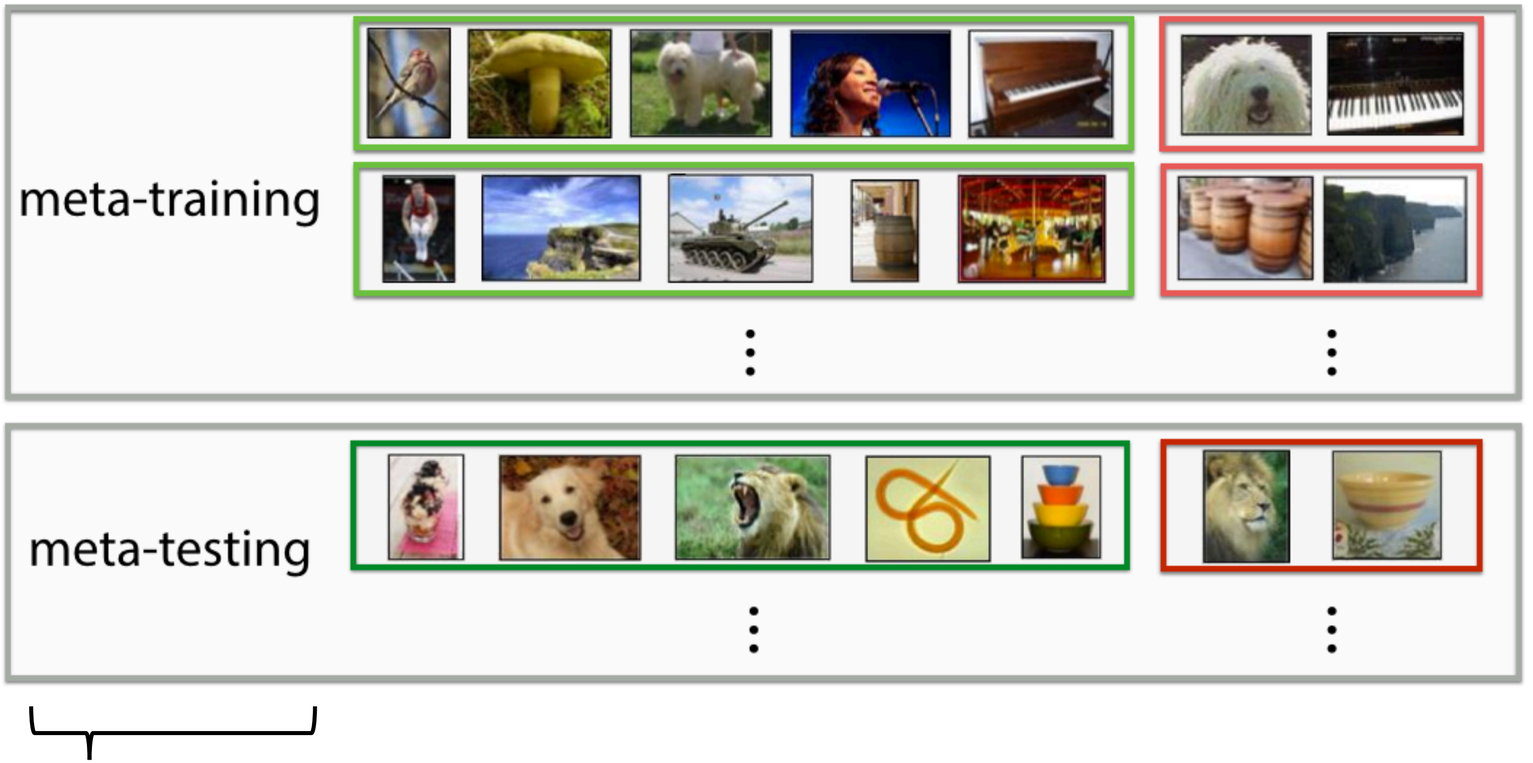
Goals for by the end of lecture:
- Understand when & how memorization in meta-learning may occur
- Understand techniques for constructing tasks automatically

Revisiting meta-learning terminology

task training set $\mathcal{D}_i^{\text{tr}}$ "support set"

task test dataset $\mathcal{D}_i^{\text{test}}$

"query set"

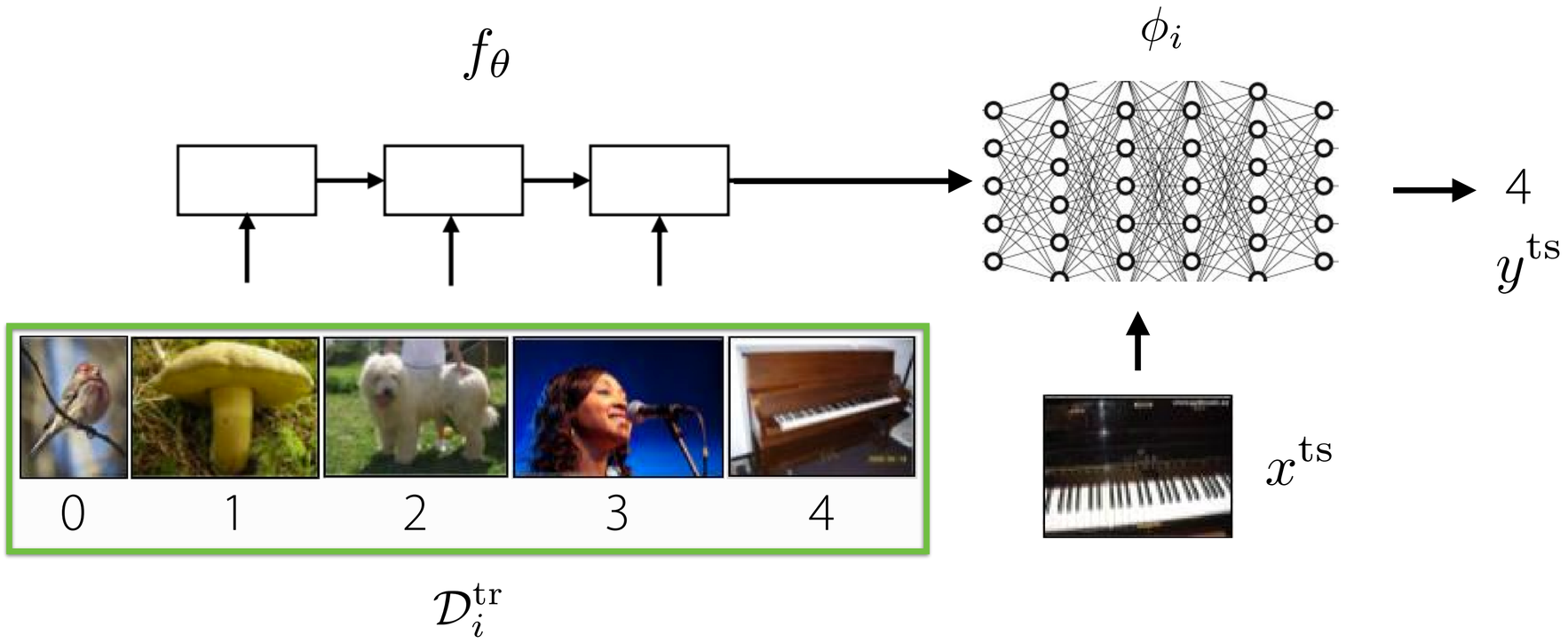


Homeworks 1 & 2 sometimes refer to these as **train** & **test**

(which is ambiguous & confusing!)

Recap: Black-Box Meta-Learning

Key idea: parametrize learner as a neural network



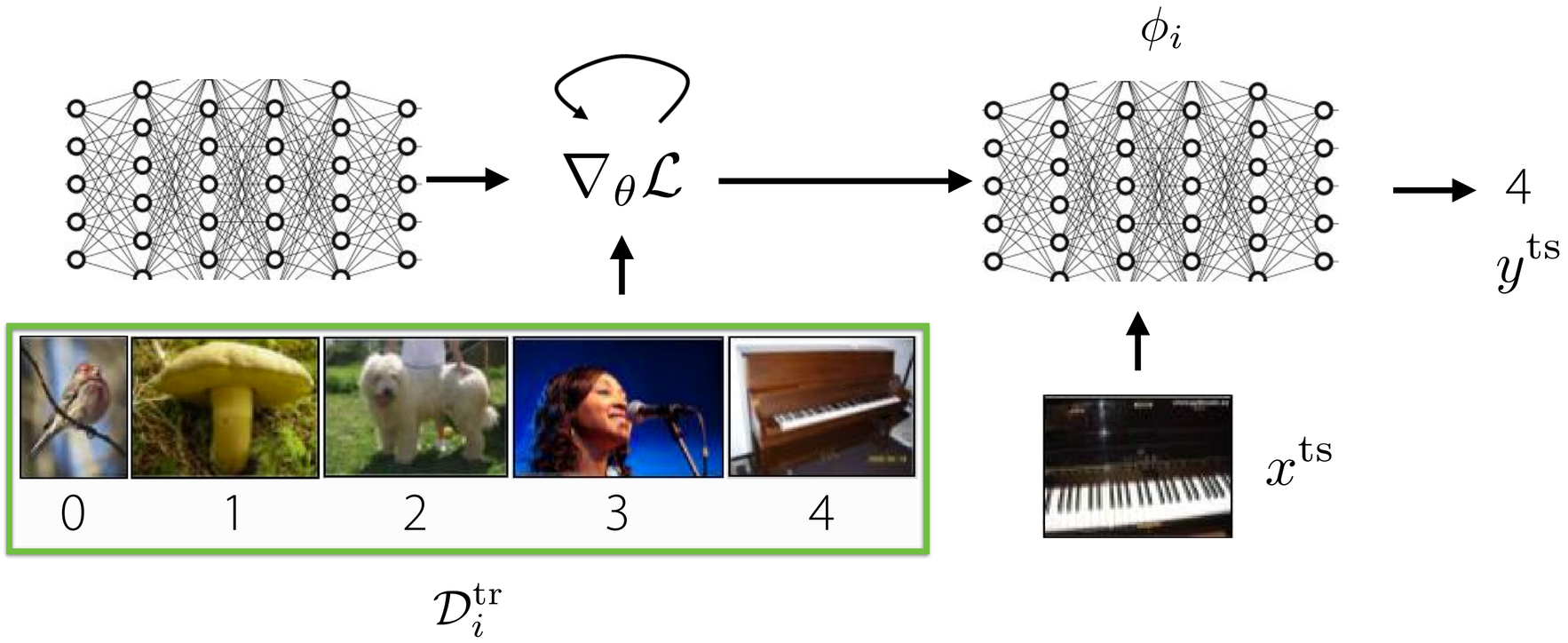
This network: inner loop, in-context learning

Training this network: outer loop

+ **expressive** - **challenging optimization** problem

Recap: Optimization-Based Meta-Learning

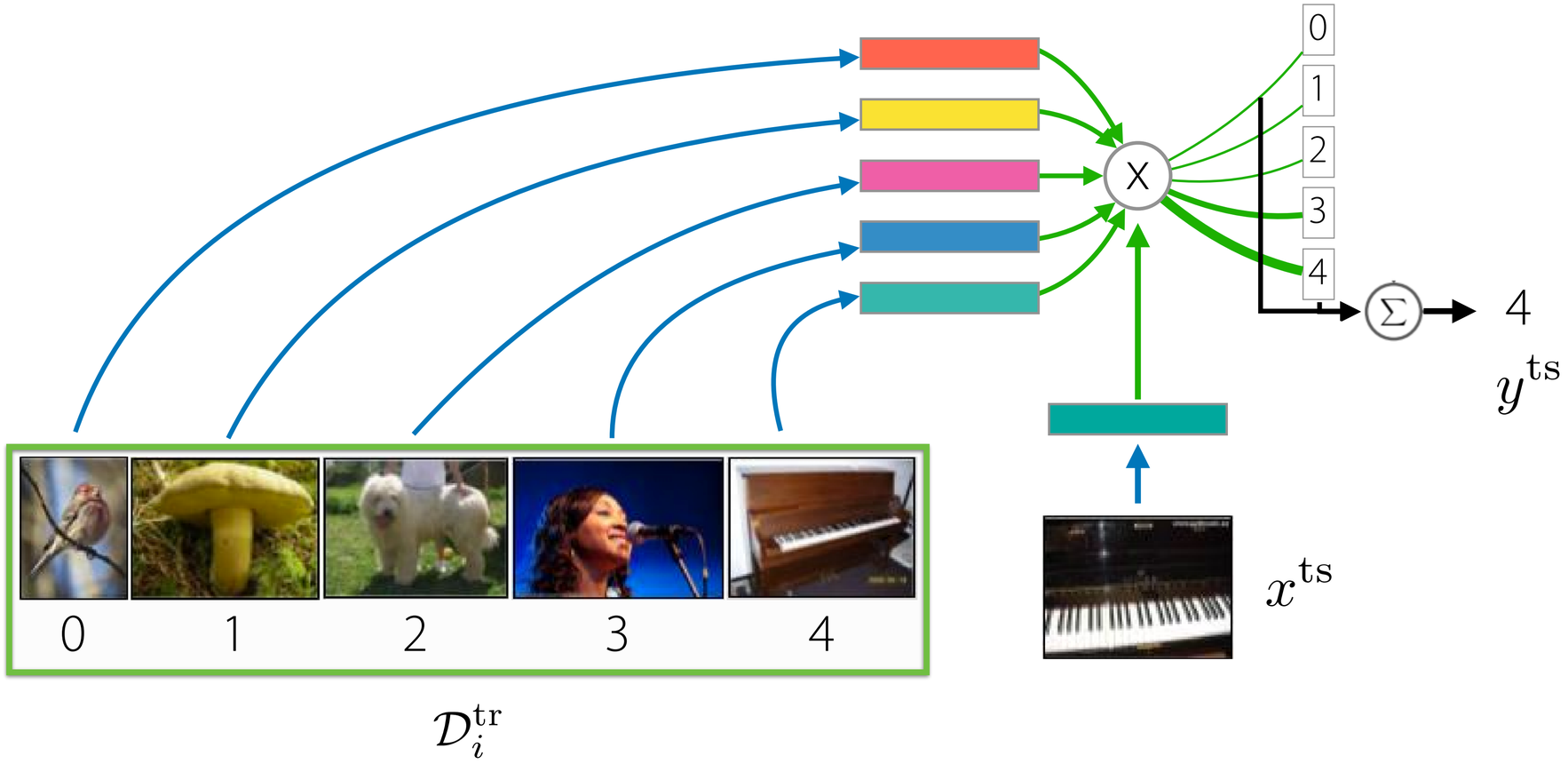
Key idea: embed optimization inside the inner learning process



+ **structure** of **optimization** embedded into meta-learner

- typically requires **second-order optimization**

Recap: Non-Parametric Meta-Learning



Key idea: *non-parametric learner* with *parametric* embedding / distance
(e.g. kNN to examples/prototypes)

+ easy to optimize, computationally fast

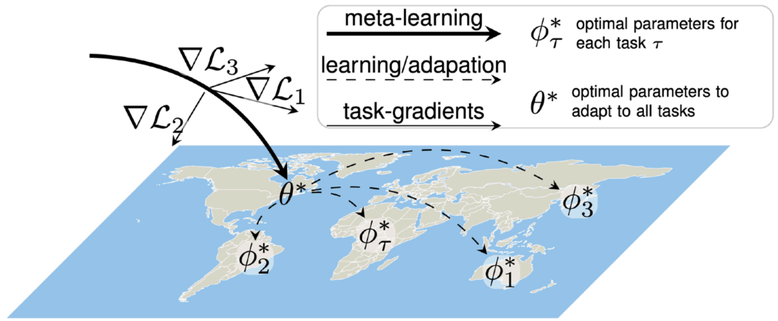
- largely restricted to classification

Supervised Task Construction

For N-way image classification

Use **labeled images** from prior classes

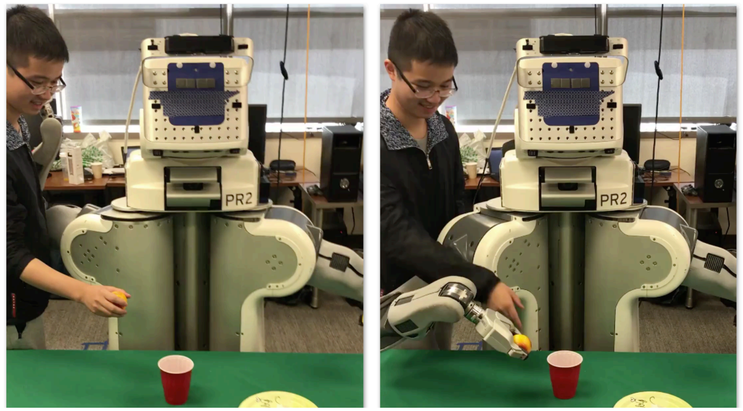
For adapting to regional differences



Rußwurm et al. Meta-Learning for Few-Shot Land Cover Classification. CVPR 2020 EarthVision Workshop

Use **labeled images** from prior regions

For few-shot imitation learning



Yu et al. One-Shot Imitation Learning from Observing Humans. RSS 2018

Use **demonstrations** for prior tasks

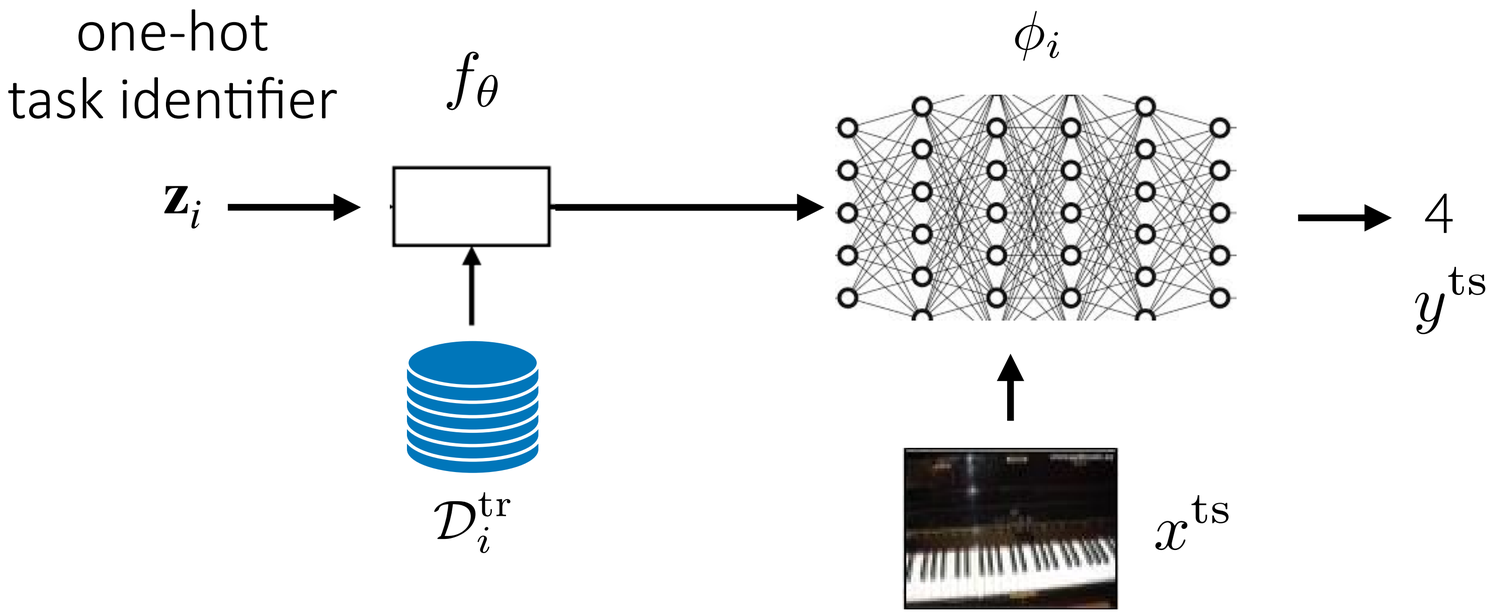
Plan for Today

Brief Recap of Meta-Learning & Task Construction

Memorization in Meta-Learning
- When it arises
- Potential solutions

Meta-Learning without Tasks Provided
- Unsupervised Meta-Learning
- Semi-Supervised Meta-Learning

Thought Exercise #1



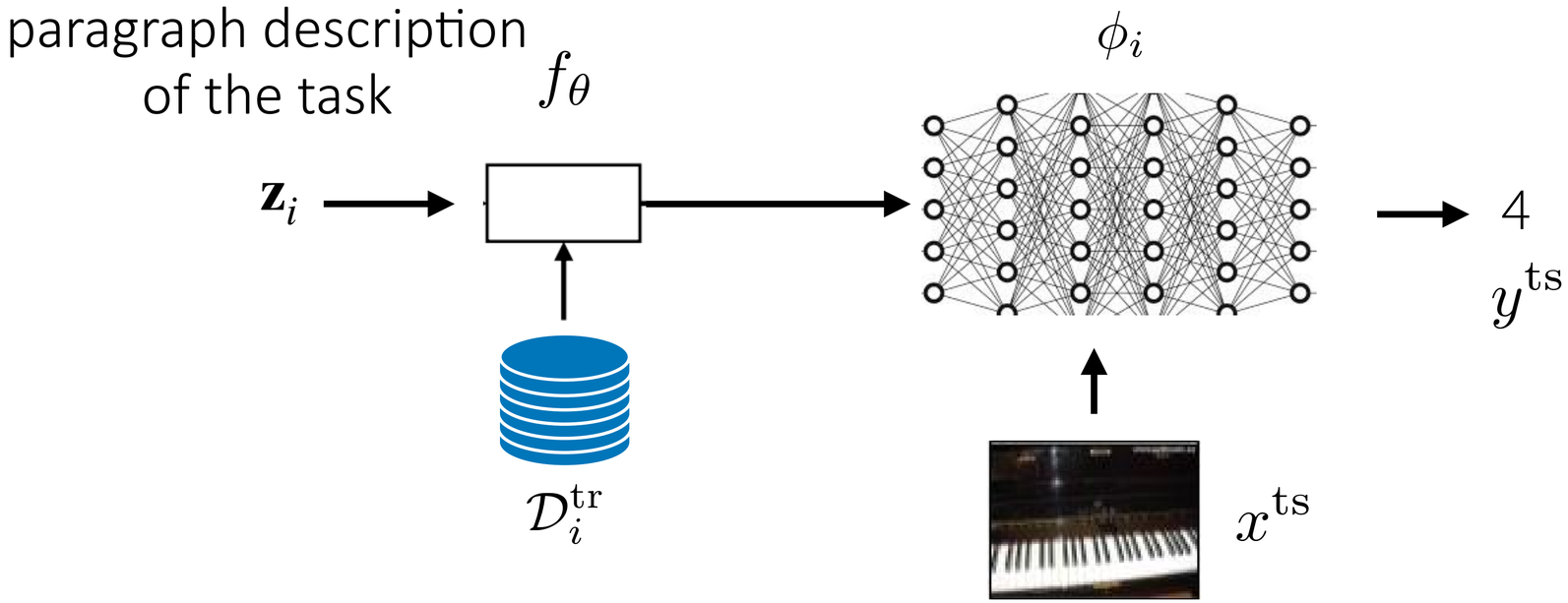
Question: What happens during meta-training if you pass in D_i^{tr} ***and*** the task identifier?

If it is difficult to learn from the data, the model will learn rely on $\mathbf{z}_i$.

Question: What happens at meta-test time if you pass in D_j^{tr} ***and*** the task identifier for a new task?

It won't generalize to the new task.

Thought Exercise #2



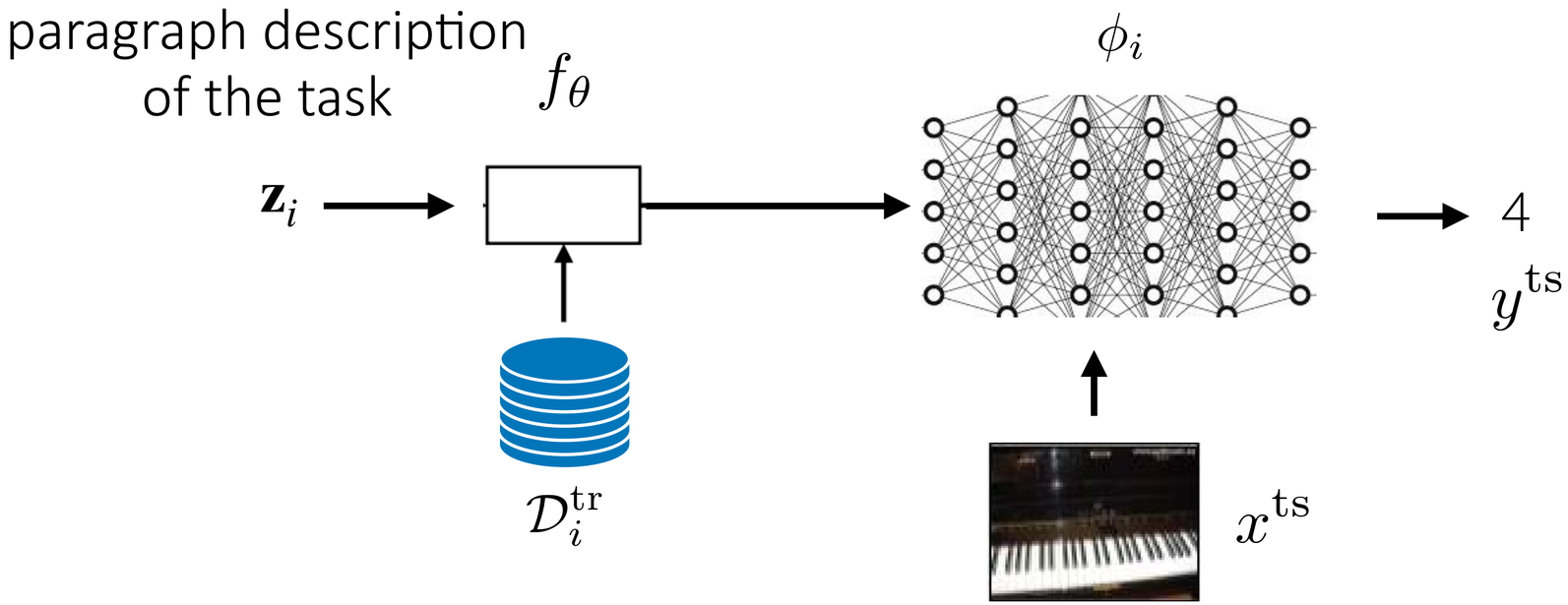
Question: What happens during meta-training if you pass in D_i^{tr} ***and*** the task identifier?

It depends on whether using the **description** or the **data** is simpler.

Question: What happens at meta-test time if you pass in D_j^{tr} ***and*** the task identifier for a new task?

It depends on what it learns to use during meta-training.

Thought Exercise #2



Question: What happe ***nd*** the task identifier?

It depends c ta is simpler.

Key problem: Model can minimize meta-training loss without looking at D_i^{tr}

Question: What happens at meta-test time if you pass in D_j^{tr} ***and*** the task identifier for a new task?

It depends on what it learns to use during meta-training.

How we construct tasks for meta-learning.

Randomly assign class labels to image classes for each task —> Tasks are *mutually exclusive*.

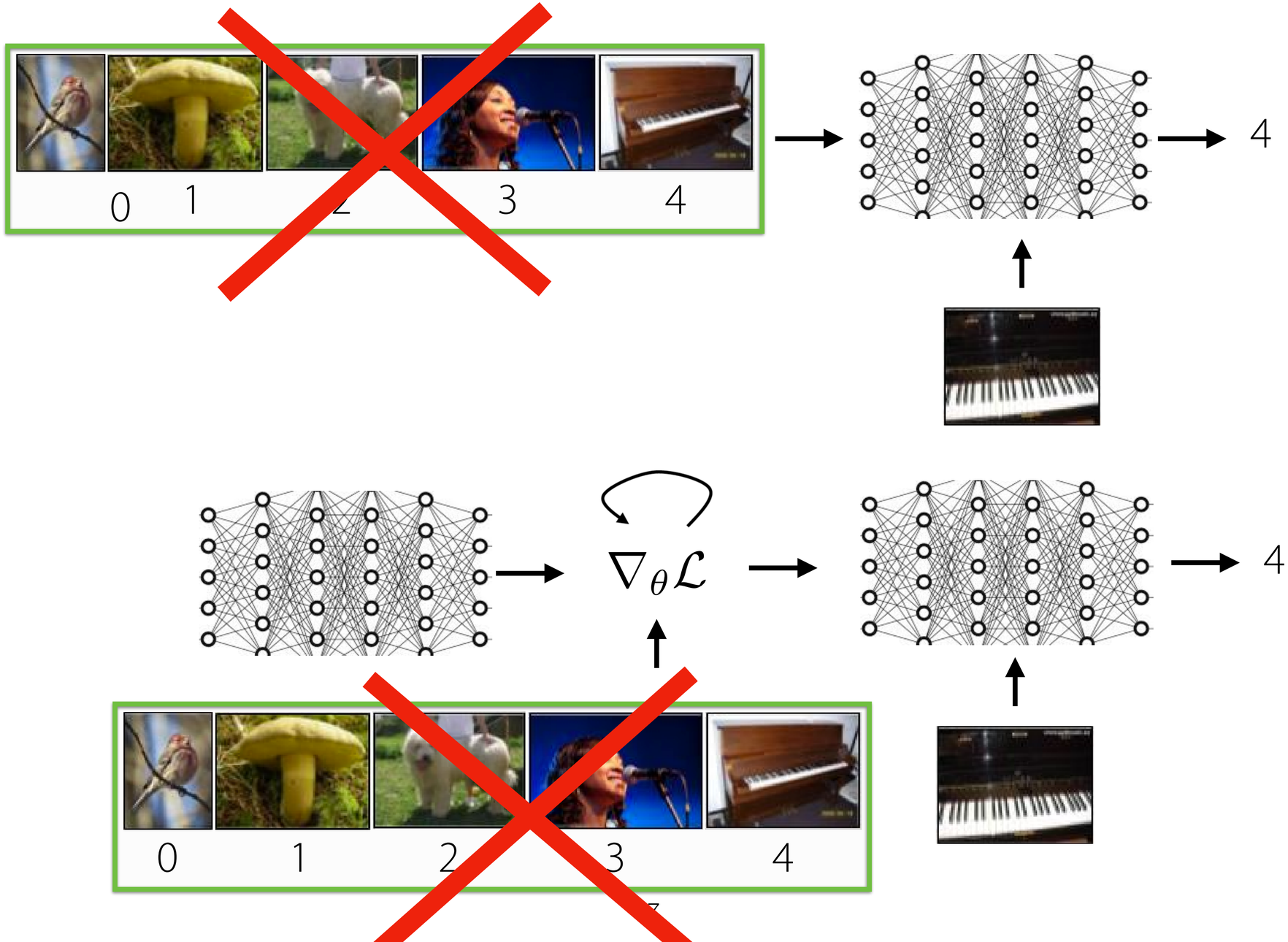
Algorithms **must** use training data to infer label ordering.

Thought Exercise #3: What if label assignment is consistent across tasks?

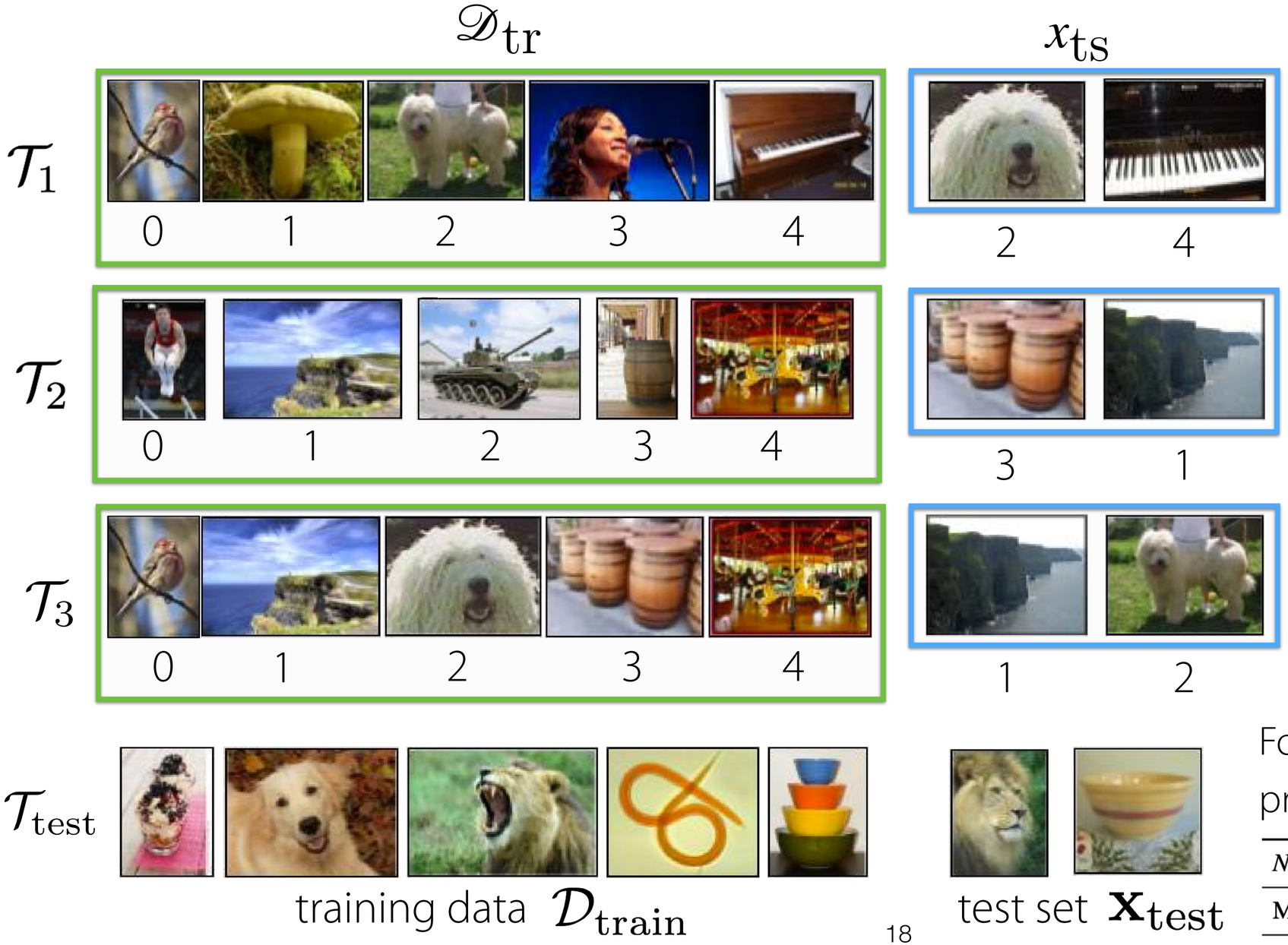
Tasks are **non-mutually exclusive**: a single function can solve all tasks.

The network can simply learn to classify inputs, irrespective of $\mathcal{D}_{\text{tr}}$

The network can simply learn to classify inputs, irrespective of $\mathscr{D}_{\text{tr}}$



What if label order is consistent?



For new image classes: can't make predictions w/o $\mathcal{D}_{\text{tr}}$

NME Omniglot	20-way 1-shot	20-way 5-shot
MAML	7.8 (0.2)%	50.7 (22.9)%

Is this a problem?

Help, it's not working when
I don't shuffle the labels.

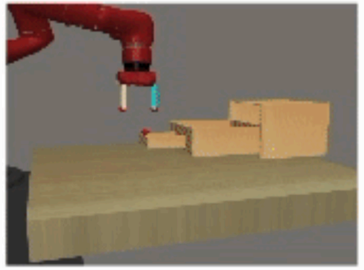
- **No**: for image classification, just shuffle labels*
- **No**, if we see the same image classes as training (no need to adapt at meta-test time)
- But, ***yes***, if we want to be able to adapt with data for new tasks.

Another example

meta-training

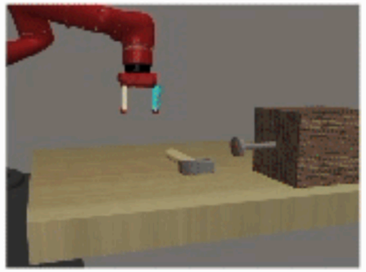
"close drawer"

$\mathcal{T}_1$



"hammer"

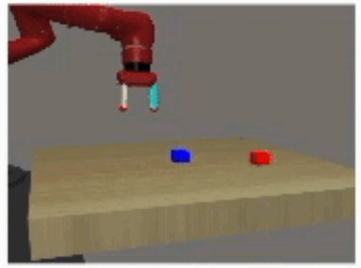
$\mathcal{T}_2$



$\cdots$

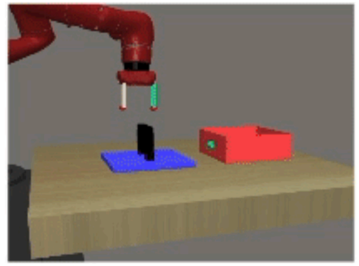
"stack"

$\mathcal{T}_{50}$



"close box"

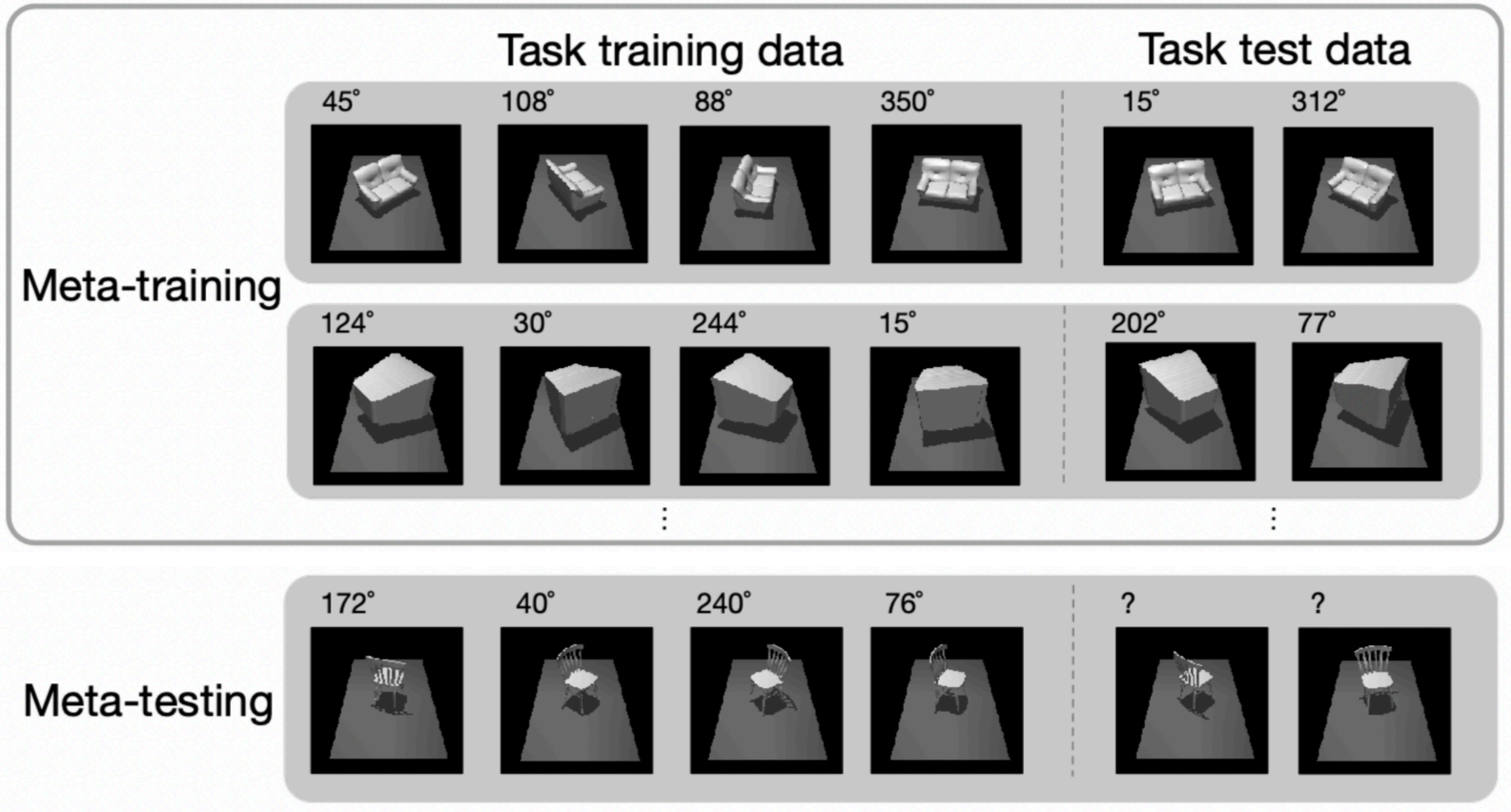
$\mathcal{T}_{\text{test}}$



If you tell the robot the task goal, the robot can **ignore** the trials.

T Yu, D Quillen, Z He, R Julian, K Hausman, C Finn, S Levine. ***Meta-World***. CoRL '19

Another example



Model can memorize the canonical orientations of the training objects.

Yin, Tucker, Yuan, Levine, Finn. **Meta-Learning without Memorization**. ICLR '19

Can we do something about it?

If tasks *mutually exclusive*: single function cannot solve all tasks

(i.e. due to label shuffling, hiding information)

If tasks are *non-mutually exclusive*: single function can solve all tasks

multiple solutions to the meta-learning problem

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

One solution: memorize canonical pose info in $\boldsymbol{\theta}$ & ignore $\mathcal{D}_i^{\mathrm{tr}}$

Another solution: carry no info about canonical pose in $\boldsymbol{\theta}$, acquire from $\mathcal{D}_i^{\mathrm{tr}}$

An entire spectrum of solutions based on how **information** flows.

Suggests a potential approach: control information flow.

Yin, Tucker, Yuan, Levine, Finn. **Meta-Learning without Memorization**. ICLR '19

If tasks are *non-mutually* ***exclusive***: single function can solve all tasks

multiple solutions to the meta-learning problem

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

One solution: memorize canonical pose info in θ & ignore $\mathcal{D}_i^{\text{tr}}$

Another solution: carry no info about canonical pose in θ, acquire from $\mathcal{D}_i^{\text{tr}}$

An entire spectrum of solutions based on how **information** flows.

Meta-regularization one option: $\max I(\hat{\mathbf{y}}_{\text{ts}}, \mathcal{D}_{\text{tr}} \,|\, \mathbf{x}_{\text{ts}})$

minimize meta-training loss + information in θ

$$\mathscr{L}(\theta, \mathscr{D}_{meta-train}) + \beta D_{KL}(q(\theta; \theta_\mu, \theta_\sigma) \| p(\theta))$$

Places precedence on using information from $\mathcal{D}_{\text{tr}}$ over storing info in θ.

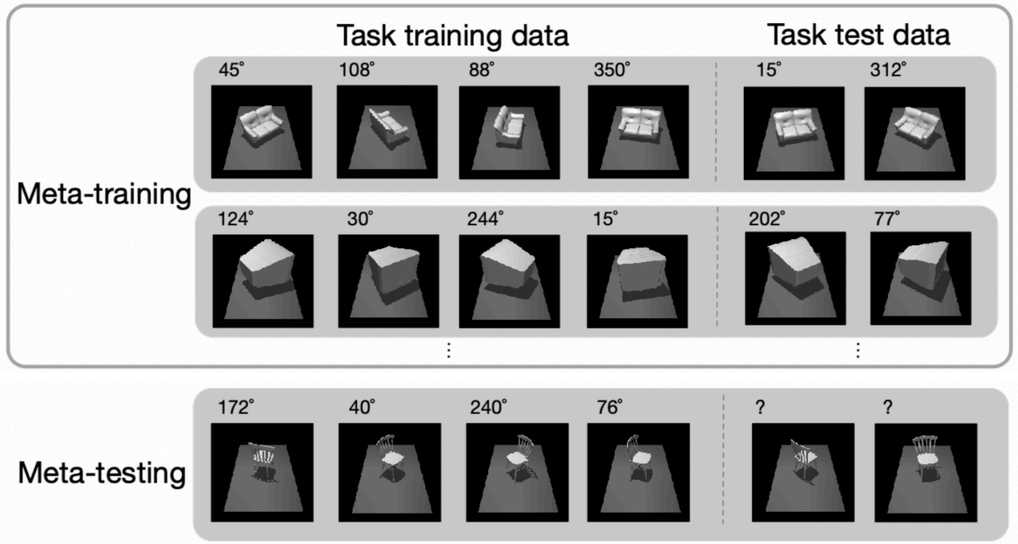
Can combine with your favorite meta-learning algorithm.

Yin, Tucker, Yuan, Levine, Finn. **Meta-Learning without Memorization**. ICLR '19

Omniglot without label shuffling: "non-mutually-exclusive" Omniglot

NME Omniglot	20-way 1-shot	20-way 5-shot
MAML	7.8 (0.2)%	50.7 (22.9)%
TAML	9.6 (2.3)%	67.9 (2.3)%
MR-MAML (W) (ours)	**83.3 (0.8)%**	**94.1 (0.1)%**

On **pose prediction** task:



Method	MAML	MR-MAML(W) (ours)	CNP	MR-CNP(W) (ours)
MSE	5.39 (1.31)	**2.26 (0.09)**	8.48 (0.12)	2.89 (0.18)

(and it's not just as simple as standard regularization)

CNP	CNP + Weight Decay	CNP + BbB	MR-CNP (W) (ours)
8.48 (0.12)	6.86 (0.27)	7.73 (0.82)	**2.89 (0.18)**

TAML: Jamal & Qi. **Task-Agnostic Meta-Learning for Few-Shot Learning**. CVPR '19

Yin, Tucker, Yuan, Levine, Finn. **Meta-Learning without Memorization**. ICLR '19

Does meta-regularization lead to better generalization?

Let $P(\theta)$ be an arbitrary distribution over θ that doesn't depend on the meta-training data.

(e.g. $P(\theta) = \mathcal{N}(\theta; \mathbf{0}, \mathbf{I})$)

For MAML, with probability at least $1 - \delta$,

$$\underbrace{er(\theta_\mu, \theta_\sigma)}_{\text{generalization error}} \leq \underbrace{\frac{1}{n}\sum_{i=1}^{n} \hat{er}(\theta_\mu, \theta_\sigma, \mathcal{D}_i, \mathcal{D}_i^*)}_{\text{error on the meta-training set}} + \left(\sqrt{\frac{1}{2(K-1)}} + \sqrt{\frac{1}{2(n-1)}}\right) \underbrace{\sqrt{D_{KL}(\mathcal{N}(\theta; \theta_\mu, \theta_\sigma) \| P) + \log \frac{n(K+1)}{\delta}}}_{\text{meta-regularization}},$$

$\forall \theta_\mu, \theta_\sigma$

With a Taylor expansion of the RHS + a particular value of β —> *recover the MR MAML objective*.

Proof: draws heavily on Amit & Meier '18

Yin, Tucker, Yuan, Levine, Finn. **Meta-Learning without Memorization**. ICLR '19

Summary of Memorization Problem

meta-learning | **standard supervised learning**

meta overfitting ⟷ **standard overfitting**

memorize training functions f_i | memorize training datapoints (x_i, y_i)

corresponding to tasks in your meta-training dataset | in your training dataset

meta regularization | **standard regularization**

control information flow ⟷ regularize hypothesis class

regularizes description length of meta-parameters ⟷ (though not always for DNNs)

Plan for Today

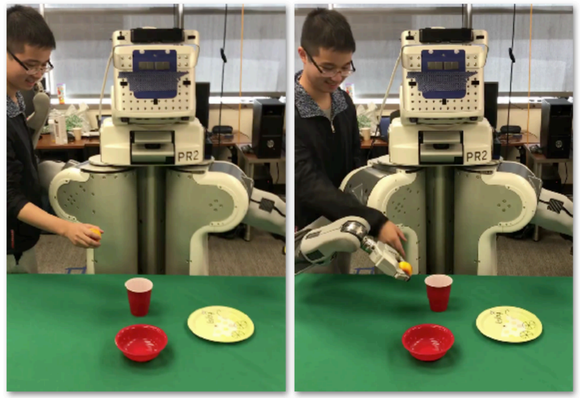
Brief Recap of Meta-Learning & Task Construction

Memorization in Meta-Learning
- When it arises
- Potential solutions

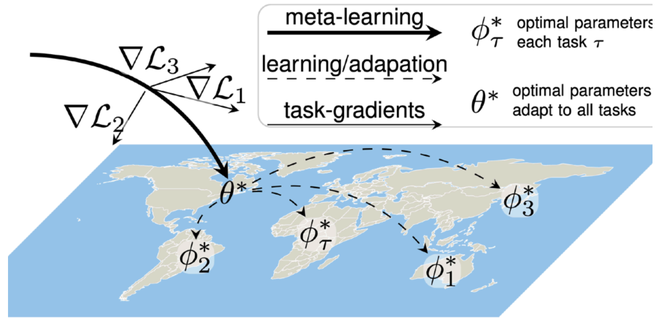
Meta-Learning without Tasks Provided
- Unsupervised Meta-Learning
- Semi-Supervised Meta-Learning

Where do tasks come from?

Requires tasks constructed from labeled data



Requires demos for many previous tasks



Rußwurm et al. Meta-Learning for Few-Shot Land Cover Classification. 2020

Requires labeled data from other regions

What if we only have unlabeled data? e.g., unlabeled images, unlabeled text

Last week: Pre-train representations & fine-tune

Today: Explicit meta-learning with unlabeled data.

A general recipe for unsupervised meta-learning

Given unlabeled dataset(s) → Propose tasks → Run meta-learning

Goal of unsupervised meta-learning methods:
Automatically construct tasks from unlabeled data

Question: What do you want the task set to look like?

1. diverse (more likely to cover test tasks)
2. structured (so that few-shot meta-learning is possible)

Next:

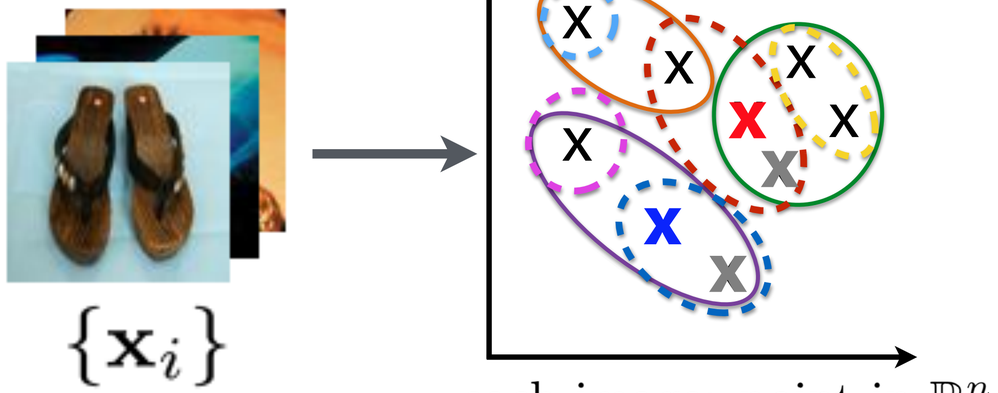
Task construction from unlabeled image data

Task construction from unlabeled text data

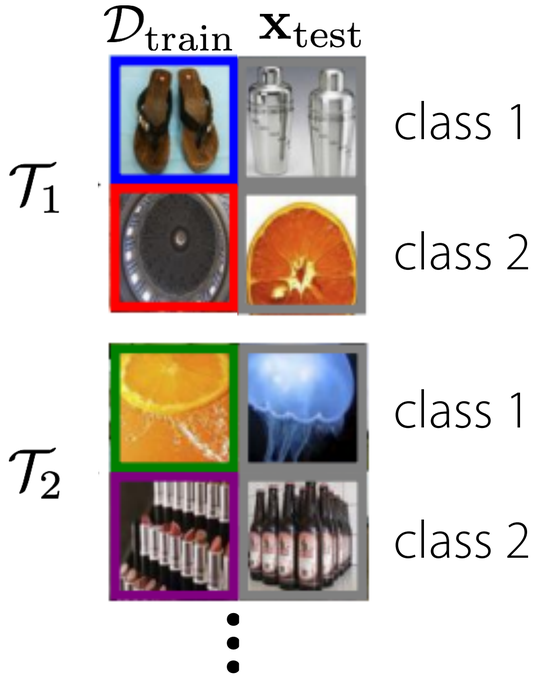
Can we meta-learn with only **unlabeled** images?

— — Task construction — —

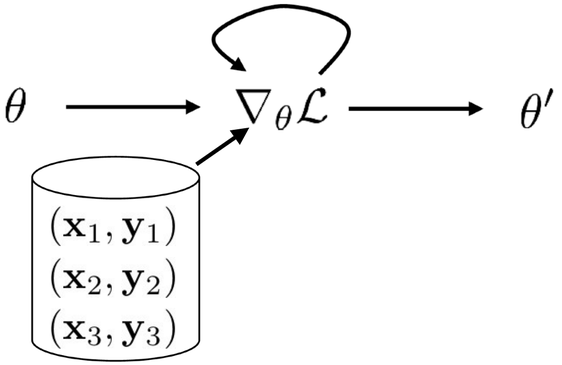
Unsupervised learning
(to get an embedding space)



Propose cluster discrimination tasks



Run meta-learning



Result: representation suitable for learning downstream tasks

Hsu, Levine, Finn. **Unsupervised Learning via Meta-Learning**. ICLR'19

Can we meta-learn with only **unlabeled** images?

Unsupervised learning
(to get an embedding space)

A few options:
BiGAN — Donahue et al. '17
DeepCluster — Caron et al. '18

→

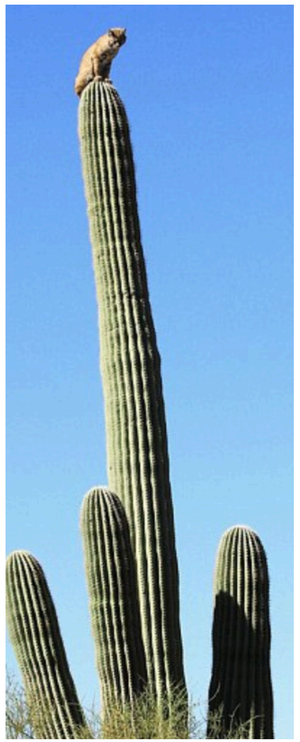
Propose cluster discrimination tasks

Clustering to Automatically Construct Tasks for Unsupervised Meta-Learning (CACTUs)

→

Run meta-learning

MAML — Finn et al. '17
ProtoNets — Snell et al. '17



CACTUs MAML

miniImageNet 5-way 5-shot

method	accuracy
MAML with labels	62.13%
BiGAN kNN	31.10%
BiGAN logistic	33.91%
BiGAN MLP + dropout	29.06%
BiGAN cluster matching	29.49%
BiGAN CACTUs MAML	51.28%
DeepCluster CACTUs MAML	**53.97%**

Same story for:

- **4 different embedding methods**
- **4 datasets** (Omniglot, CelebA, miniImageNet, MNIST)
- **2 meta-learning methods** (*)
- Test tasks with **larger datasets**

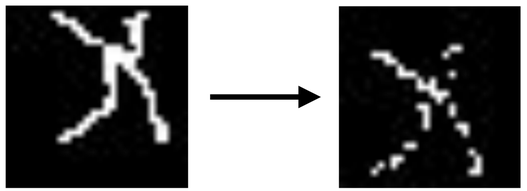
*ProtoNets underperforms in some cases.

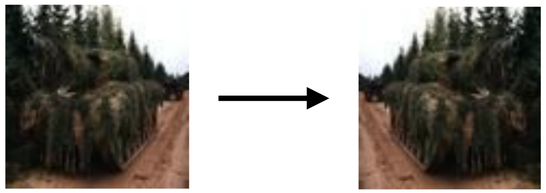
Hsu, Levine, Finn. **Unsupervised Learning via Meta-Learning**. ICLR'19

Can we use **domain knowledge** when constructing tasks?

e.g. **image's label** often **won't change** when you:

- drop out some pixels
- translate the image
- reflect the image





Task construction:

For each task $\mathcal{T}_i$:

i. Randomly sample N images & assign labels $1,\ldots,N$

—> Store in $\mathcal{D}_i^{\text{tr}}$

ii. For each datapoint in $\mathcal{D}_i^{\text{tr}}$, augment image using domain knowledge

—> Store in $\mathcal{D}_i^{\text{ts}}$

Khodadadeh, Bölöni, Shah. **Unsupervised Meta-Learning for Few-Shot Image Classification**. NeurIPS '19

Can we use **domain knowledge** when constructing tasks?

For each task $\mathcal{T}_i$:

i. Randomly sample N images & assign labels $1,\ldots,N$ —> Store in $\mathcal{D}_i^{\text{tr}}$

ii. For each datapoint in $\mathcal{D}_i^{\text{tr}}$, augment image using domain knowledge —> Store in $\mathcal{D}_i^{\text{ts}}$

How to augment in practice?

Omniglot: translation & random pixel dropout **MiniImagenet:** AutoAugment* (translation, rotation, shear)

		Omniglot				Mini-Imagenet			
Algorithm (N, K)	**Clustering**	**(5,1)**	**(5,5)**	**(20,1)**	**(20,5)**	**(5,1)**	**(5,5)**	**(5,20)**	**(5,50)**
Training from scratch	N/A	52.50	74.78	24.91	47.62	27.59	38.48	51.53	59.63
linear classifier	ACAI / DC	61.08	81.82	43.20	66.33	29.44	39.79	56.19	65.28
MLP with dropout	ACAI / DC	51.95	77.20	30.65	58.62	29.03	39.67	52.71	60.95
cluster matching	ACAI / DC	54.94	71.09	32.19	45.93	22.20	23.50	24.97	26.87
CACTUs-MAML	ACAI / DC	68.84	87.78	48.09	73.36	39.90	**53.97**	**63.84**	**69.64**
CACTUs-ProtoNets	ACAI / DC	68.12	83.58	47.75	66.27	39.18	53.36	61.54	63.55
UMTRA (ours)	N/A	**83.80**	**95.43**	**74.25**	**92.12**	**39.93**	50.73	61.11	67.15
MAML (Supervised)	N/A	94.46	98.83	84.60	96.29	46.81	62.13	71.03	75.54
ProtoNets (Supervised)	N/A	98.35	99.58	95.31	98.81	46.56	62.29	70.05	72.04

- outstanding Omniglot performance
  (where we have good domain knowledge!)
- MiniImageNet: slightly underperforms CACTUs

Khodadadeh, Bölöni, Shah. **Unsupervised Meta-Learning for Few-Shot Image Classification**. NeurIPS '19

* Cubuk et al. 2018

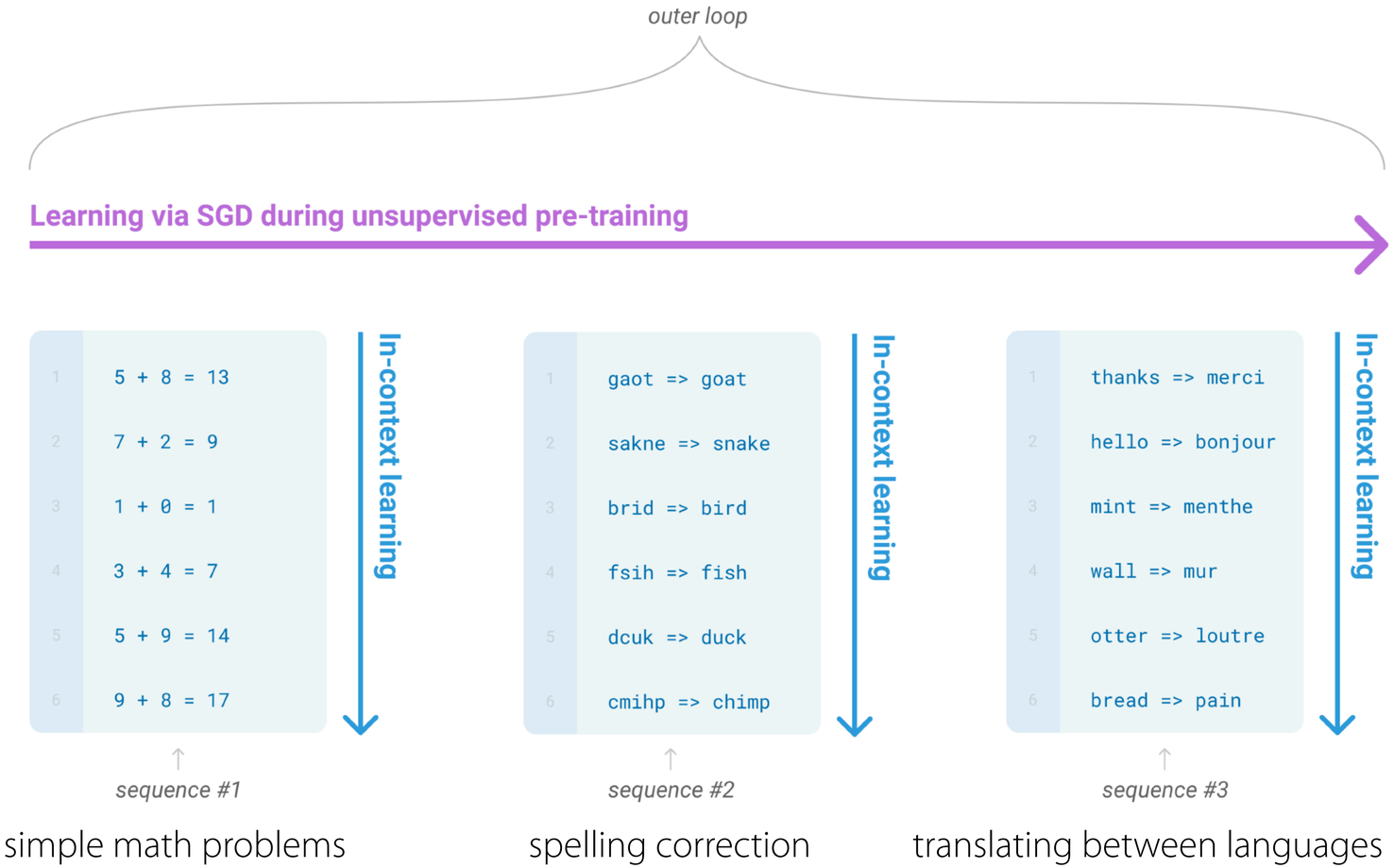
Can we meta-learn with only **unlabeled** text?

Option A: Formulate it as a language modeling problem.

Recall: GPT-3

$\mathcal{D}_i^{\text{tr}}$: sequence of characters

$\mathcal{D}_i^{\text{ts}}$: following sequence of characters



When might we not use this option?

- harder to combine w/ **optimization-based meta-learning**
- harder to apply to **classification** tasks (e.g. sentiment, political bias, etc)

Brown, Mann, Ryder, Subbiah et al. **Language Models are Few-Shot Learners**. arXiv '20

Can we meta-learn with only **unlabeled** text?

Option B: Construct tasks by masking out words

Task: Classify the masked word.

For each task $\mathcal{T}_i$:

i. Sample subset of N unique words & assign unique ID.

{Democratic, Capital} 1 2

ii. Sample $K + Q$ sentences with that word, *masking the word out*

iii. Construct $\mathcal{D}_i^{\text{tr}}$ and $\mathcal{D}_i^{\text{ts}}$ with masked sentences & corresponding word IDs

$\mathcal{D}_i^{\text{tr}}$

Support set

Sentence	Class
A member of the [m] Party, he was the first African American to be elected to the presidency.	1
The [m] Party is one of the two major contemporary political parties in the United States, along with its rival, the Republican Party.	1
Honolulu is the [m] and largest city of the U.S. state of Hawaii.	2
Washington, D.C., formally the District of Columbia and commonly referred to as Washington or D.C., is the [m] of the United States.	2

$\mathcal{D}_i^{\text{ts}}$

Query: New Delhi is an urban district of Delhi which serves as the [m] of India

Correct Prediction: 2

Bansal, Jha, Munkhdalai, McCallum. **Self-Supervised Meta-Learning for Few-Shot Natural Language Classification Tasks**. EMNLP '20

entirely unsupervised pre-training | supervised or semi-supervised pre-training

Task	N	k	BERT	SMLMT	MT-BERT$_{\text{softmax}}$	MT-BERT	LEOPARD	Hybrid-SMLMT
CoNLL	4	4	50.44 ± 08.57	46.81 ± 4.77	52.28 ± 4.06	55.63 ± 4.99	54.16 ± 6.32	**57.60** ± 7.11
		8	50.06 ± 11.30	61.72 ± 3.11	65.34 ± 7.12	58.32 ± 3.77	67.38 ± 4.33	**70.20** ± 3.00
		16	74.47 ± 03.10	75.82 ± 4.04	71.67 ± 3.03	71.29 ± 3.30	76.37 ± 3.08	**80.61** ± 2.77
		32	83.27 ± 02.14	84.01 ± 1.73	73.09 ± 2.42	79.94 ± 2.45	83.61 ± 2.40	**85.51** ± 1.73
MITR	8	4	49.37 ± 4.28	46.23 ± 3,90	45.52 ± 5.90	50.49 ± 4.40	49.84 ± 3.31	**52.29** ± 4.32
		8	49.38 ± 7.76	61.15 ± 1.91	58.19 ± 2.65	58.01 ± 3.54	62.99 ± 3.28	**65.21** ± 2.32
		16	69.24 ± 3.68	69.22 ± 2.78	66.09 ± 2.24	66.16 ± 3.46	70.44 ± 2.89	**73.37** ± 1.88
		32	78.81 ± 1.95	78.82 ± 1.30	69.35 ± 0.98	76.39 ± 1.17	78.37 ± 1.97	**79.96** ± 1.48
Airline	3	4	42.76 ± 13.50	42.83 ± 6.12	43.73 ± 7.86	46.29 ± 12.26	54.95 ± 11.81	**56.46** ± 10.67
		8	38.00 ± 17.06	51.48 ± 7.35	52.39 ± 3.97	49.81 ± 10.86	61.44 ± 03.90	**63.05** ± 8.25
		16	58.01 ± 08.23	58.42 ± 3.44	58.79 ± 2.97	57.25 ± 09.90	62.15 ± 05.56	**69.33** ± 2.24
		32	63.70 ± 4.40	65.33 ± 3.83	61.06 ± 3.89	62.49 ± 4.48	67.44 ± 01.22	**71.21** ± 3.28
Disaster	2	4	55.73 ± 10.29	**62.26** ± 9.16	52.87 ± 6.16	50.61 ± 8.33	51.45 ± 4.25	55.26 ± 8.32
		8	56.31 ± 09.57	**67.89** ± 6.83	56.08 ± 7.48	54.93 ± 7.88	55.96 ± 3.58	63.62 ± 6.84
		16	64.52 ± 08.93	**72.86** ± 1.70	65.83 ± 4.19	60.70 ± 6.05	61.32 ± 2.83	70.56 ± 2.23
		32	73.60 ± 01.78	**73.69** ± 2.32	67.13 ± 3.11	72.52 ± 2.28	63.77 ± 2.34	71.80 ± 1.85
Emotion	13	4	09.20 ± 3.22	09.84 ± 1.09	09.41 ± 2.10	09.84 ± 2.14	11.71 ± 2.16	**11.90** ± 1.74
		8	08.21 ± 2.12	11.02 ± 1.02	11.61 ± 2.34	11.21 ± 2.11	12.90 ± 1.63	**13.26** ± 1.01
		16	13.43 ± 2.51	12.05 ± 1.18	13.82 ± 2.02	12.75 ± 2.04	13.38 ± 2.20	**15.17** ± 0.89
		32	16.66 ± 1.24	14.28 ± 1.11	13.81 ± 1.62	**16.88** ± 1.80	14.81 ± 2.01	16.08 ± 1.16
Political Bias	2	4	54.57 ± 5.02	57.72 ± 5.72	54.32 ± 3.90	54.66 ± 3.74	60.49 ± 6.66	**61.17** ± 4.91
		8	56.15 ± 3.75	63.02 ± 4.62	57.36 ± 4.32	54.79 ± 4.19	61.74 ± 6.73	**64.10** ± 4.03
		16	60.96 ± 4.25	**66.35** ± 2.84	59.24 ± 4.25	60.30 ± 3.26	65.08 ± 2.14	66.11 ± 2.04
		32	65.04 ± 2.32	**67.73** ± 2.27	62.68 ± 3.21	64.99 ± 3.05	64.67 ± 3.41	67.30 ± 1.53

BERT - standard self-supervised learning + fine-tuning

SMLMT - proposed unsupervised meta-learning

MT-BERT - multi-task learning + fine-tuning (on supervised tasks)

LEOPARD - optimization-based meta-learner (only on supervised tasks)

Hybrid-SMLMT - meta-learning on proposed tasks + supervised tasks

More results & analysis in the paper!

Bansal, Jha, Munkhdalai, McCallum. **Self-Supervised Meta-Learning for Few-Shot Natural Language Classification Tasks**. EMNLP '20

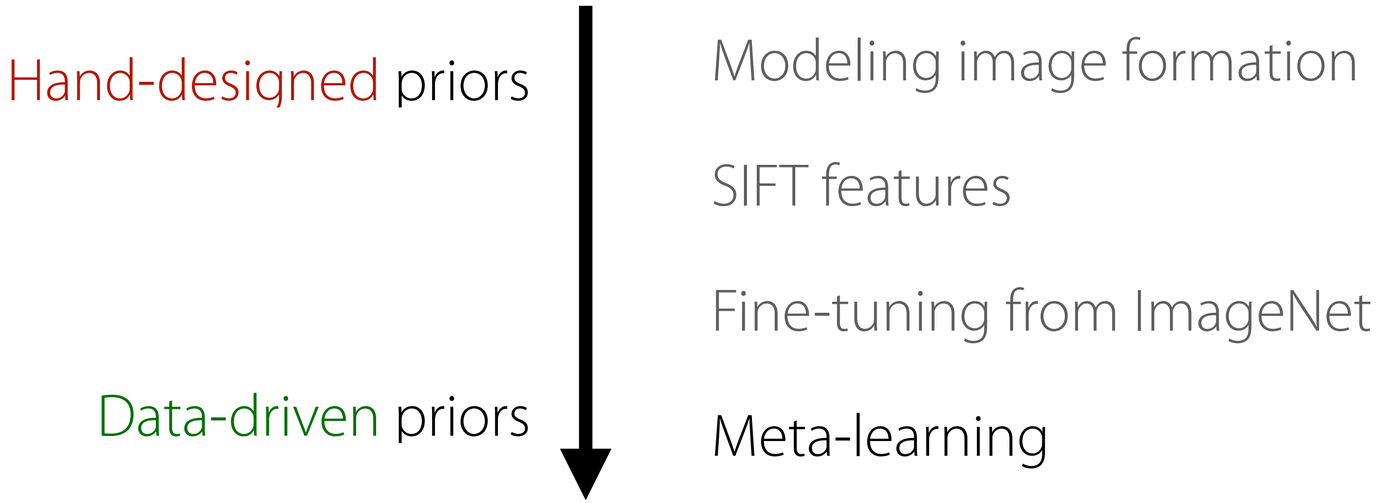
Summary of Unsupervised Meta-Training

Given unlabeled dataset(s) → Propose tasks → Run meta-learning

Existing task proposal techniques:

- Classify between clusters of images
- Classify augmented image vs. different image instance
- Generate text from a particular context
- Classify a masked word

Do meta-learning methods scale?



More data-driven approaches are supposed to be more scalable

Do meta-learning methods work at scale?

Plan for Today

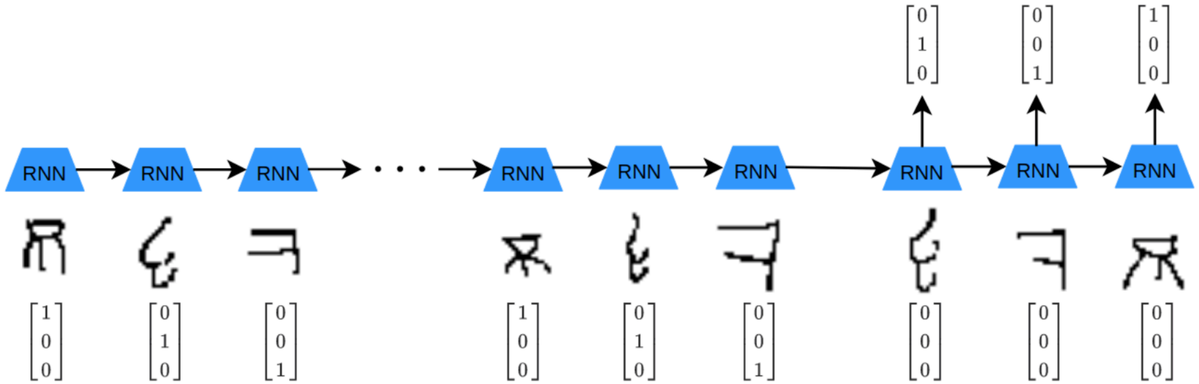
Why consider ***large-scale meta-optimization***?

Applications

Approaches
- Truncated backpropagation
- Gradient-free optimization

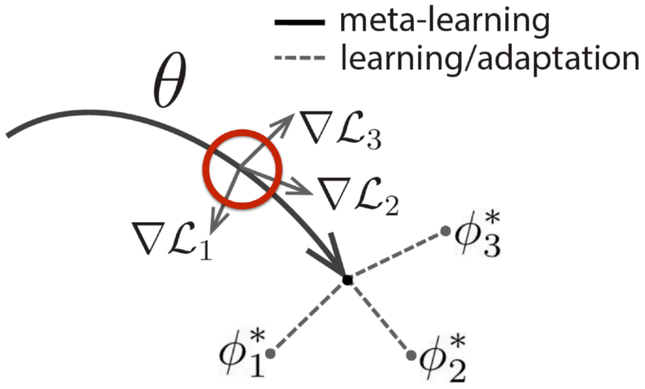
Goals for by the end of lecture:
- Know scenarios where existing meta-learning approaches fail due to scale
- Understand techniques for large-scale meta-optimization

Direct Backpropagation

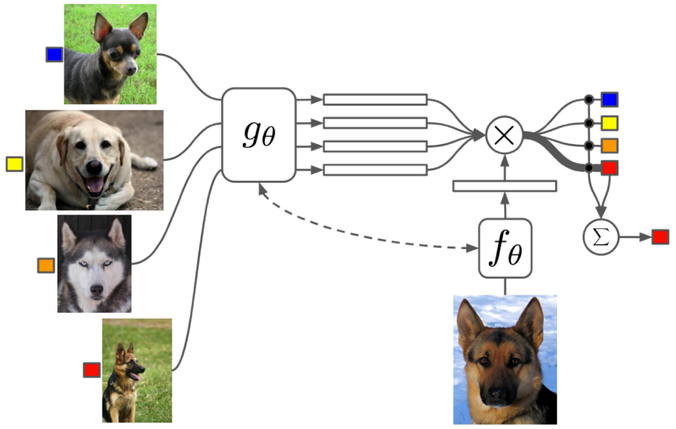


Black-box

General recipe: Build an inner computation graph, then backpropagate.



Optimization-based



Nonparametric

+ Automatically works with any differentiable computation graph

- **Memory cost** scales with **computation graph size!**

How Big are Computation Graphs?

model	backbone	miniImageNet 5-way 1-shot	miniImageNet 5-way 5-shot	tieredImageNet 5-way 1-shot	tieredImageNet 5-way 5-shot
Meta-Learning LSTM* [22]	64-64-64-64	43.44 ± 0.77	60.60 ± 0.71	-	-
Matching Networks* [33]	64-64-64-64	43.56 ± 0.84	55.31 ± 0.73	-	-
MAML [8]	32-32-32-32	48.70 ± 1.84	63.11 ± 0.92	51.67 ± 1.81	70.30 ± 1.75
Prototypical Networks*† [28]	64-64-64-64	49.42 ± 0.78	68.20 ± 0.66	53.31 ± 0.89	72.69 ± 0.74
Relation Networks* [29]	64-96-128-256	50.44 ± 0.82	65.32 ± 0.70	54.48 ± 0.93	71.32 ± 0.78
R2D2 [3]	96-192-384-512	51.2 ± 0.6	68.8 ± 0.1	-	-
Transductive Prop Nets [14]	64-64-64-64	55.51 ± 0.86	69.86 ± 0.65	59.91 ± 0.94	73.30 ± 0.75
SNAIL [18]	ResNet-12	55.71 ± 0.99	68.88 ± 0.92	-	-
Dynamic Few-shot [10]	64-64-128-128	56.20 ± 0.86	73.00 ± 0.64	-	-
AdaResNet [19]	ResNet-12	56.88 ± 0.62	71.94 ± 0.57	-	-
TADAM [20]	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	-	-
Activation to Parameter† [21]	WRN-28-10	59.60 ± 0.41	73.74 ± 0.19	-	-
LEO† [25]	WRN-28-10	61.76 ± 0.08	77.59 ± 0.12	**66.33 ± 0.05**	**81.44 ± 0.09**
MetaOptNet-RR (ours)	ResNet-12	61.41 ± 0.61	77.88 ± 0.46	**65.36 ± 0.71**	**81.34 ± 0.52**
MetaOptNet-SVM (ours)	ResNet-12	62.64 ± 0.61	78.63 ± 0.46	**65.99 ± 0.72**	**81.56 ± 0.53**
MetaOptNet-SVM-trainval (ours)†	ResNet-12	**64.09 ± 0.62**	**80.00 ± 0.45**	**65.81 ± 0.74**	**81.75 ± 0.53**

4-layer CNN
Parameters: <1e5

WRN-28-10
Parameters: <4e6

ResNet-12
Parameters: <1e7

From: "Meta-Learning with Differentiable Convex Optimization", Lee et al.

How Big are Computation Graphs?

```python
class NeuralNetwork(nn.Module):
    def __init__(self):
        super().__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

```python
epochs = 5
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
print("Done!")
```

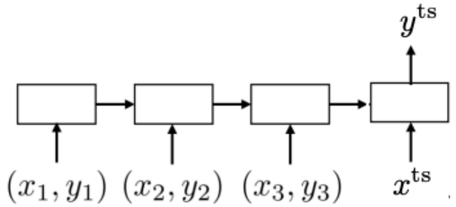
Toy 2-layer MLP from official PyTorch tutorial
Parameters: <7e6
Gradient steps: 5 epochs = ~4e3

Total floats: ~2e10 (>100GB!)

From: https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html

Black-box

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



Optimization-based

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

Non-parametric

$$\begin{aligned} y^{\text{ts}} &= f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) \\ &= \text{softmax}(-d\left(f_\theta(x^{\text{ts}}), \mathbf{c}_n\right)) \\ \text{where } \mathbf{c}_n &= \frac{1}{K} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} \mathbb{1}(y = n) f_\theta(x) \end{aligned}$$

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

Question: when might f_{LEARN} be too big to apply direct backpropagation?

Settings With Bigger Computation Graphs

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

Computation graph of f_{LEARN} is large when:

- It uses a big network and/or many gradient steps
- It includes second-order optimization (***meta-meta learning***?)

Meta-parameter θ can be any component of f_{LEARN}:

HW2

- Initial parameters
- Learning rate
- Optimizer
- Loss function
- Dataset
- Network architecture

$$\min_{\theta} \sum_{\text{task}_i} \left(\theta - \alpha \nabla_\theta L\left(\theta, D_i^{\text{tr}}\right), D_i^{\text{ts}}\right)$$

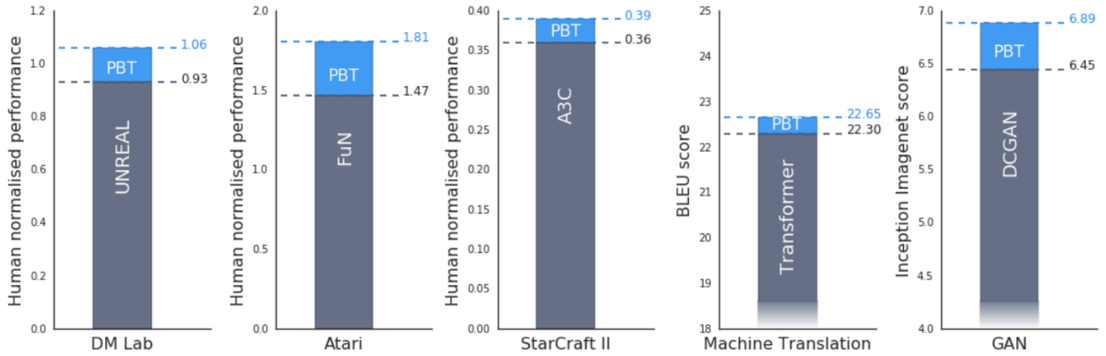
$$\min_{\theta,\alpha} \sum_{\text{task}_i} \left(\theta - \alpha \nabla_\theta L\left(\theta, D_i^{\text{tr}}\right), D_i^{\text{ts}}\right)$$

$$\min_{\theta,\psi} \sum_{\text{task}_i} \left(\theta - \alpha \nabla_\theta L_\psi\left(\theta, D_i^{\text{tr}}\right), D_i^{\text{ts}}\right)$$

$$\min_{\omega} \sum_{\theta \sim p(\theta_0)} \left(\theta - \alpha \nabla_\theta L\left(\theta, D_\omega\right), D_i^{\text{ts}}\right)$$

Application: Hyperparameter Optimization

Goal: Optimize ***hyperparameters*** for validation set performance



Benefits over random search in many domains

Method	Validation	Test	Time(s)
Grid Search	97.32	94.58	100k
Random Search	84.81	81.46	100k
Bayesian Opt.	72.13	69.29	100k
STN	70.30	67.68	25k
No HO	75.72	71.91	18.5k
Ours	69.22	66.40	18.5k
Ours, Many	**68.18**	**66.14**	18.5k

LSTM Hyperparameters

Inverse Approx.	Validation	Test
0	92.5 ±0.021	92.6 ±0.017
3 Neumann	**95.1** ±0.002	94.6 ±0.001
3 Unrolled Diff.	95.0 ±0.002	**94.7** ±0.001
I	94.6 ±0.002	94.1 ±0.002

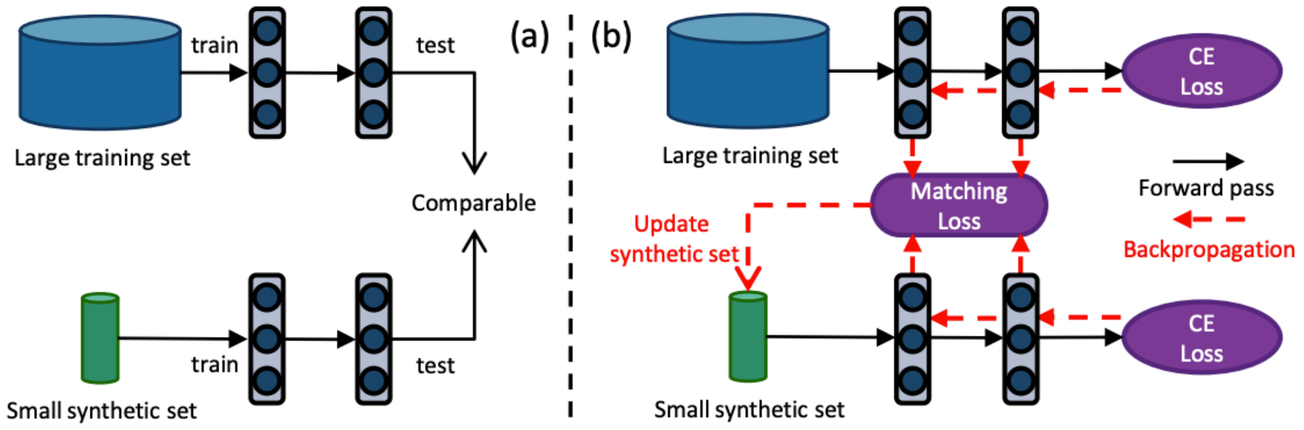
"Hyper"parameters of a data augmentation network

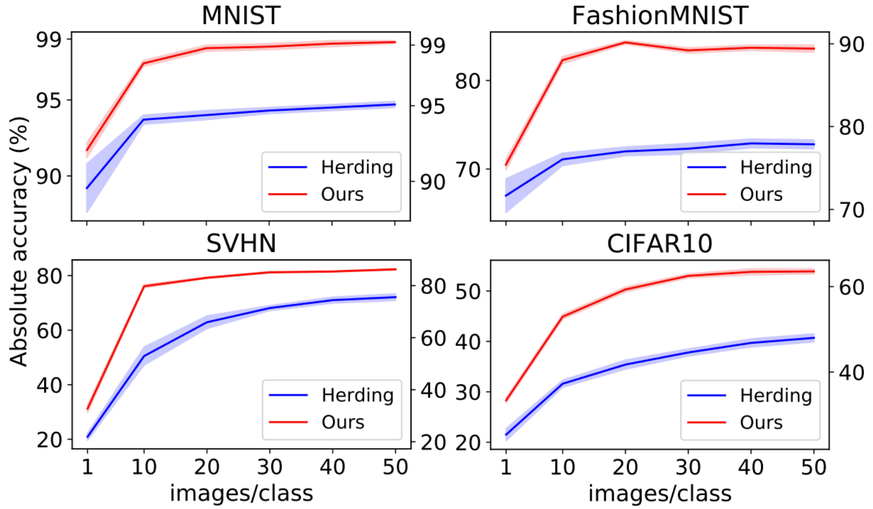
From: "Population Based Training of Neural Networks", Jaderberg et al. (2017)

From: "Optimizing Millions of Hyperparameters by Implicit Differentiation", Lorraine et al. (2019)

Application: Dataset Distillation

Goal: optimize a ***synthetic training set*** for validation set performance





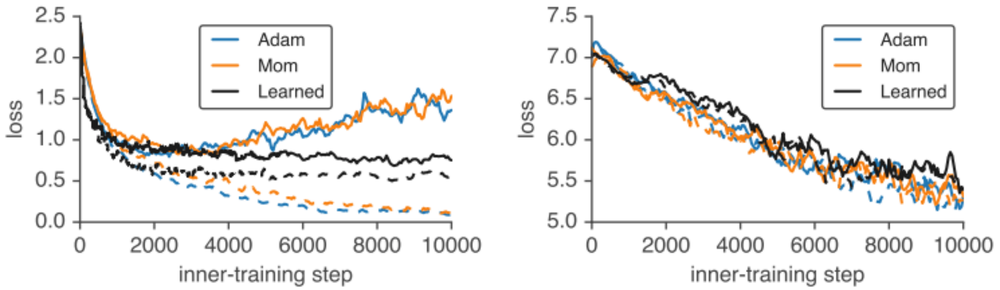
Method: Match training data gradients at each timestep

From: "Dataset Condensation with Gradient Matching", Zhao et al. (2020)

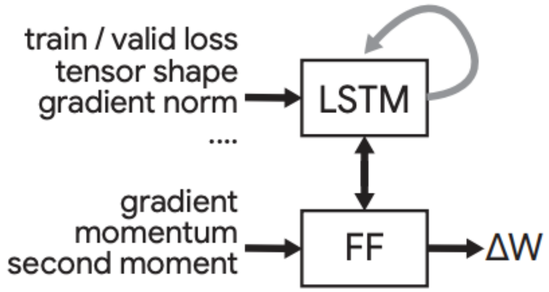
Application: Optimizer Learning

Goal: Optimize an ***optimizer*** for validation set performance

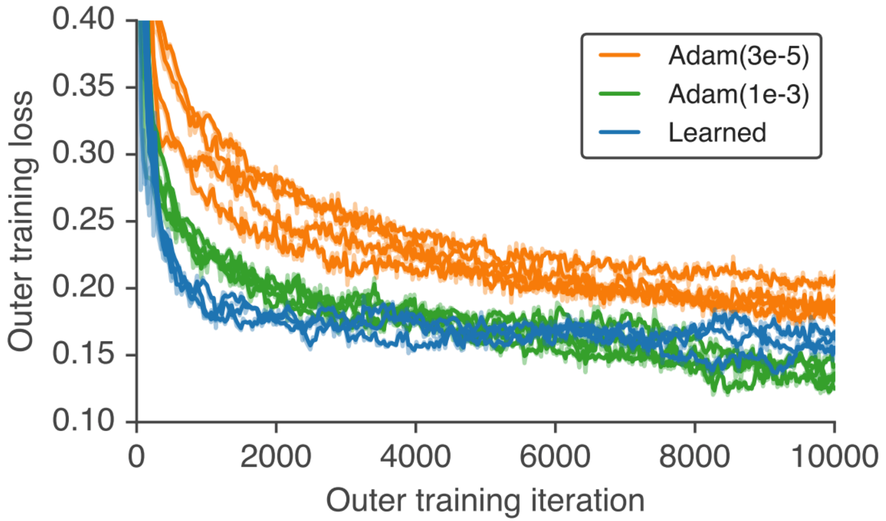
Simple architecture



Works at ResNet scale



More complex architecture
with per-tensor LSTM

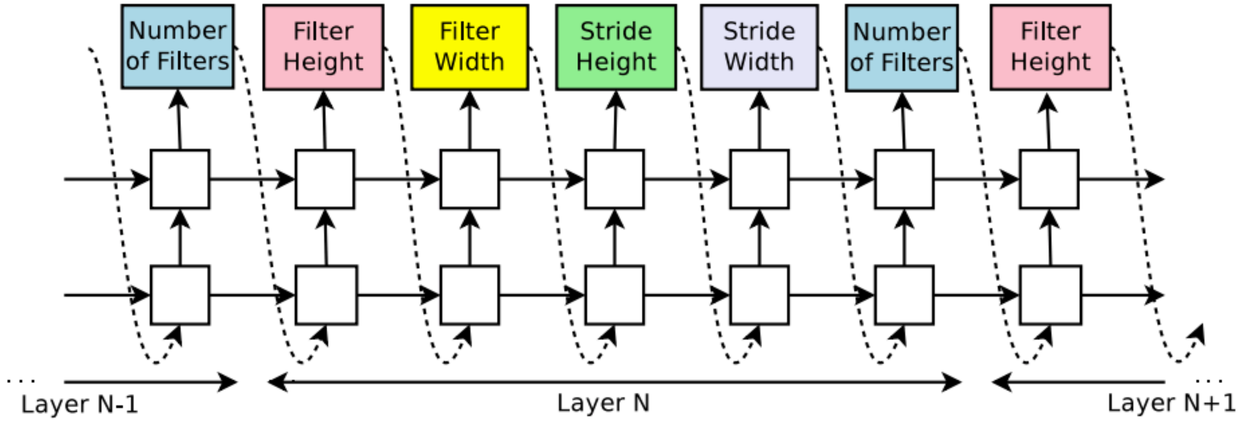


Can even train itself

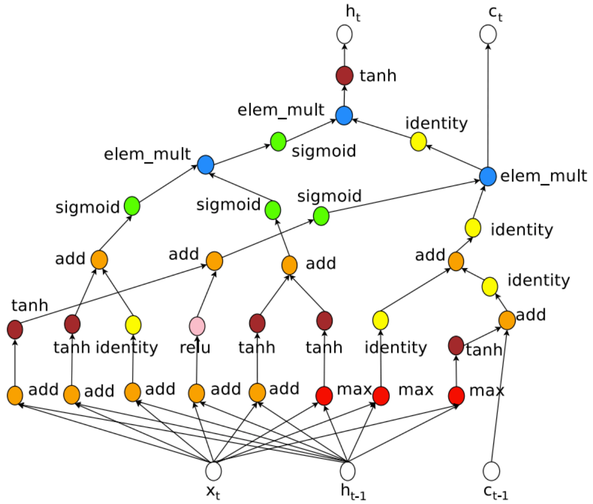
From: "Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves", Metz et al. (2020)

Application: Neural Architecture Search

Goal: Optimize an ***architecture*** for validation set performance



An RNN parameterizes a neural network



A generated cell for an RNN

Model	Depth	Parameters	Error rate (%)
Network in Network (Lin et al., 2013)	-	-	8.81
All-CNN (Springenberg et al., 2014)	-	-	7.25
Deeply Supervised Net (Lee et al., 2015)	-	-	7.97
Highway Network (Srivastava et al., 2015)	-	-	7.72
Scalable Bayesian Optimization (Snoek et al., 2015)	-	-	6.37
FractalNet (Larsson et al., 2016)	21	38.6M	5.22
with Dropout/Drop-path	21	38.6M	4.60
ResNet (He et al., 2016a)	110	1.7M	6.61
ResNet (reported by Huang et al. (2016c))	110	1.7M	6.41
ResNet with Stochastic Depth (Huang et al., 2016c)	110	1.7M	5.23
	1202	10.2M	4.91
Wide ResNet (Zagoruyko & Komodakis, 2016)	16	11.0M	4.81
	28	36.5M	4.17
ResNet (pre-activation) (He et al., 2016b)	164	1.7M	5.46
	1001	10.2M	4.62
DenseNet ($L = 40, k = 12$) Huang et al. (2016a)	40	1.0M	5.24
DenseNet($L = 100, k = 12$) Huang et al. (2016a)	100	7.0M	4.10
DenseNet ($L = 100, k = 24$) Huang et al. (2016a)	100	27.2M	3.74
DenseNet-BC ($L = 100, k = 40$) Huang et al. (2016b)	190	25.6M	3.46
Neural Architecture Search v1 no stride or pooling	15	4.2M	5.50
Neural Architecture Search v2 predicting strides	20	2.5M	6.01
Neural Architecture Search v3 max pooling	39	7.1M	4.47
Neural Architecture Search v3 max pooling + more filters	39	37.4M	3.65

Zoph and Le, "Neural Architecture Search with Reinforcement Learning" (2017)

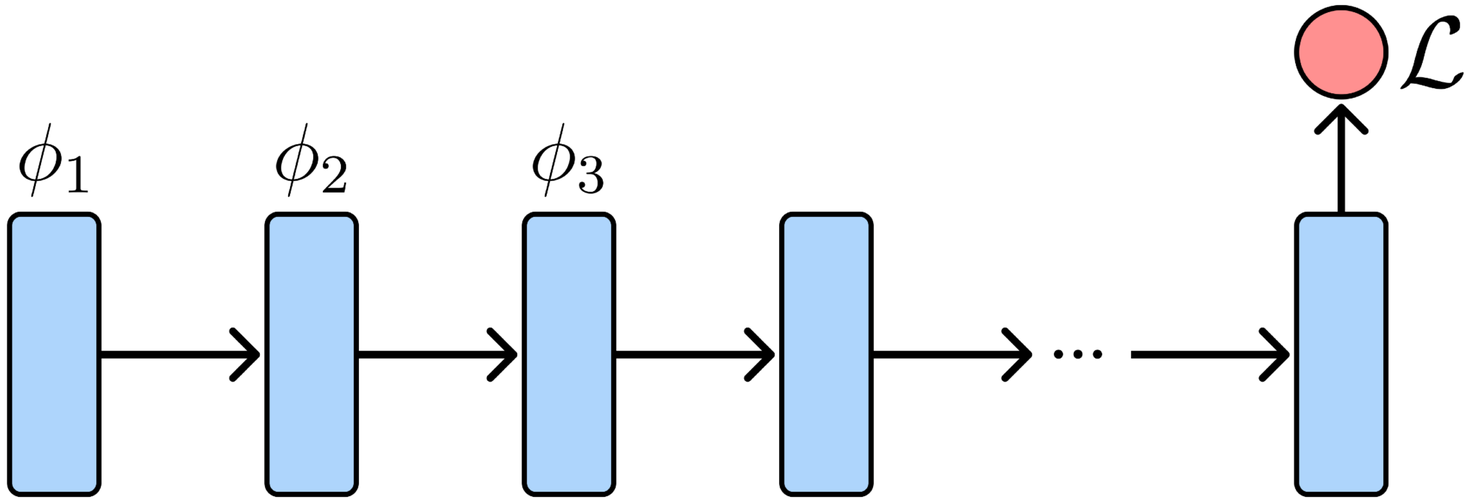
Plan for Today

Why consider large-scale meta-optimization?

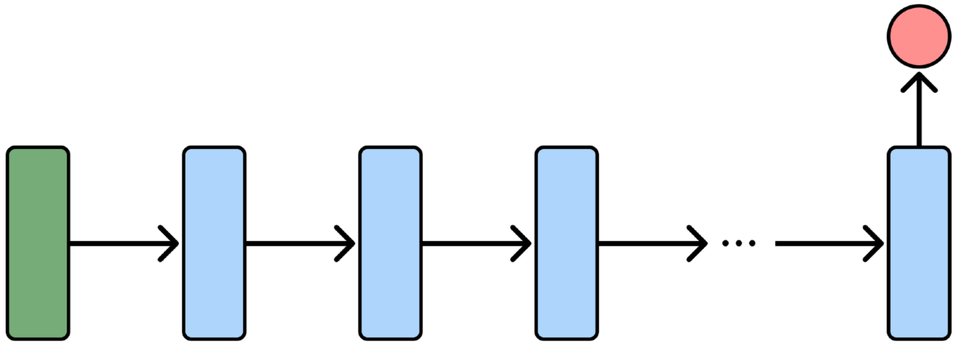
Applications

Approaches
- Truncated backpropagation
- Gradient-free optimization

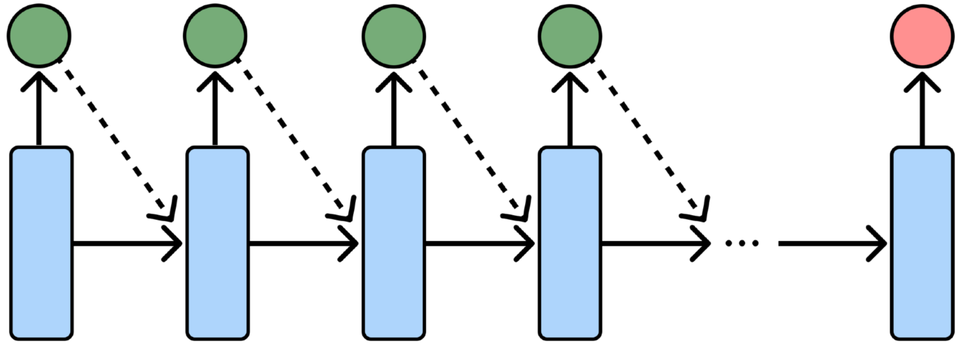
Unrolled Computation Graphs



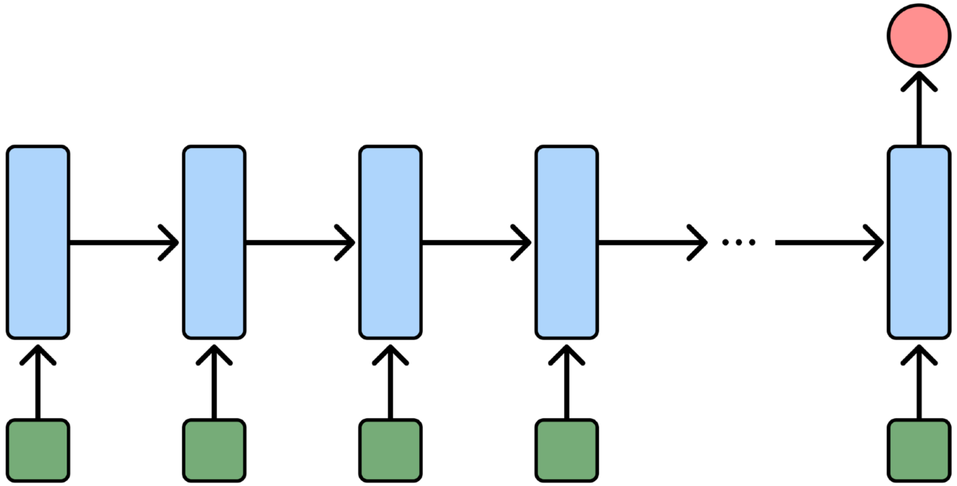
Unrolled Computation Graphs



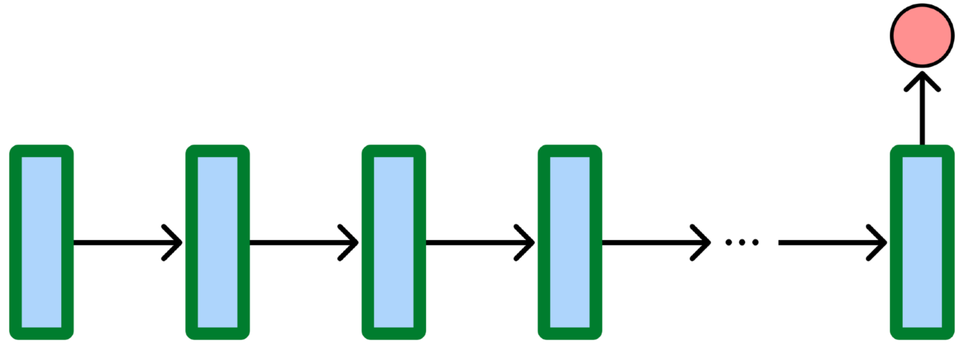
Initial Parameters



Learned loss / Regularizer, Optimizer



Synthetic dataset / augmentation



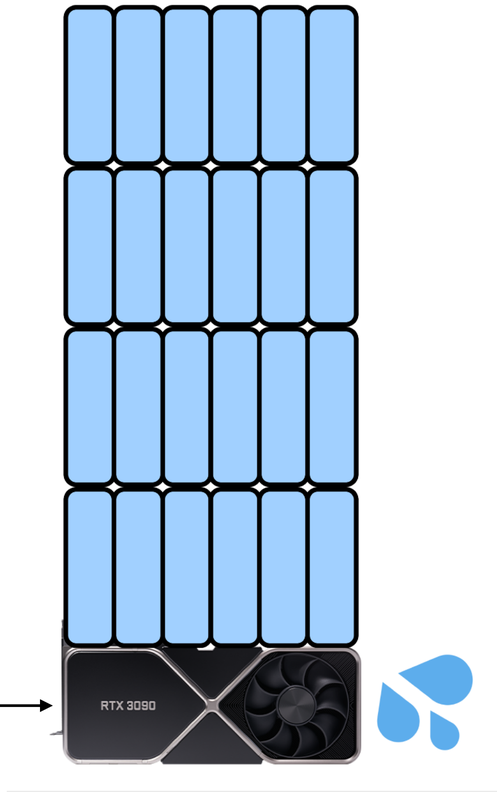
Architecture

Plan for Today

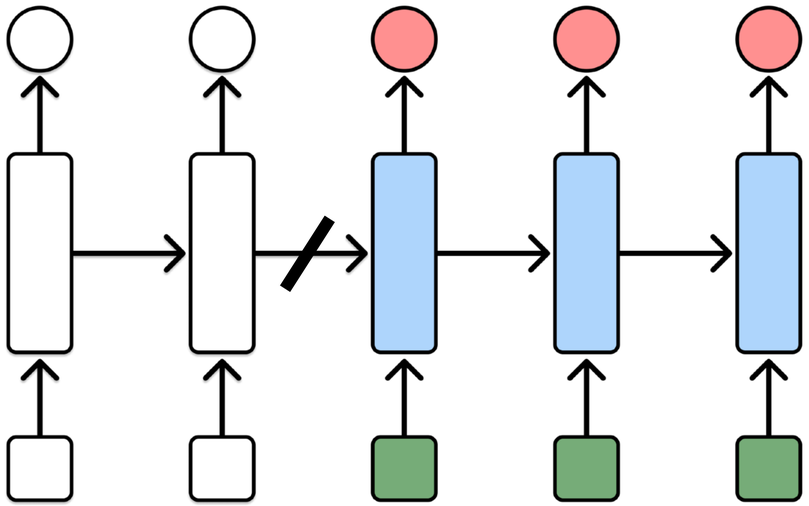
Why consider large-scale meta-optimization?

Applications

Approaches
- **Truncated backpropagation**
- Gradient-free optimization



Truncated Backpropagation



T=3

```
losses = []
for x, y in train_loader:
    y_pred, state = rnn(state)
    losses.append(loss_fn(y, y_pred))
    if len(losses) == T:
        torch.sum(losses).backward()
        opt.step()
        opt.zero_grad()
        state.detach_()
        losses = []
```

Split the full sequence into shorter slices, and backpropagate after processing each slice.

Question: what could happen if we use short T?

+ **Simple**: autograd handles everything
- **Biased** estimator
- Cannot take **long-range dependencies** into account
- Sequence length introduces a **tradeoff between correctness and memory cost**

Plan for Today

Why consider large-scale meta-optimization?

Applications

Approaches
- Truncated backpropagation
- **Gradient-free optimization**

Gradient-free Optimization

Backpropagation is costly for large computation graphs…

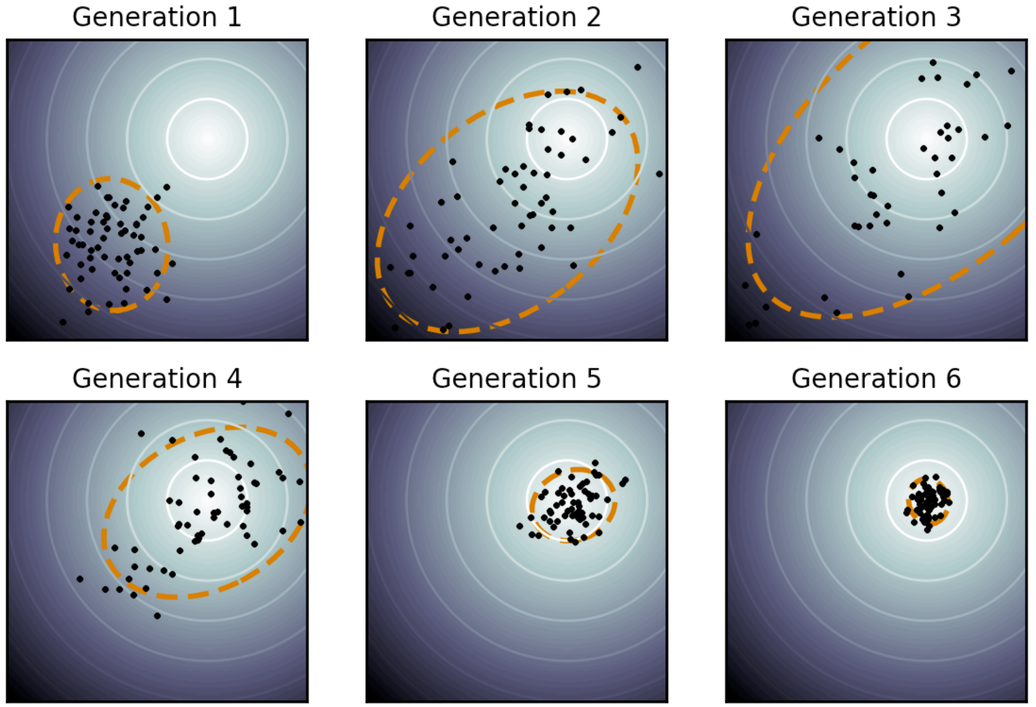
Optimization **does not necessarily require gradients!**

***Evolution Strategies*:** Estimates gradients using stochastic finite differences.

Evolution Strategies

Initialize parameters $(\mu, \sigma) \leftarrow (\mu_0, \sigma_0)$. Repeat:

1. Sample particles: $x^1, x^2, \ldots, x^N \sim \mathcal{N}(\mu, \sigma^2 I)$
2. Evaluate and get best: $\{e^1, \ldots, e^n\} \subset \{x^1, \ldots, x^N\}$
3. $\mu, \sigma^2 \leftarrow \text{Avg}(e^1, \ldots, e^n), \text{Var}(e^1, \ldots, e^n)$



From: Wikipedia CMA-ES page

Example: optimizing **learning rate**

Initialize lr and noise $(\alpha, \sigma) \leftarrow (\alpha_0, \sigma_0)$

1. Sample lr: $\alpha^1, \alpha^2, \ldots, \alpha^N \sim \mathcal{N}(\alpha, \sigma^2)$
2. Run SGD, get runs with best val accuracy: $\{e^1, \ldots, e^n\} \subset \{\alpha^1, \ldots, \alpha^N\}$
3. $\alpha, \sigma^2 \leftarrow \text{Avg}(e^1, \ldots, e^n), \text{Var}(e^1, \ldots, e^n)$

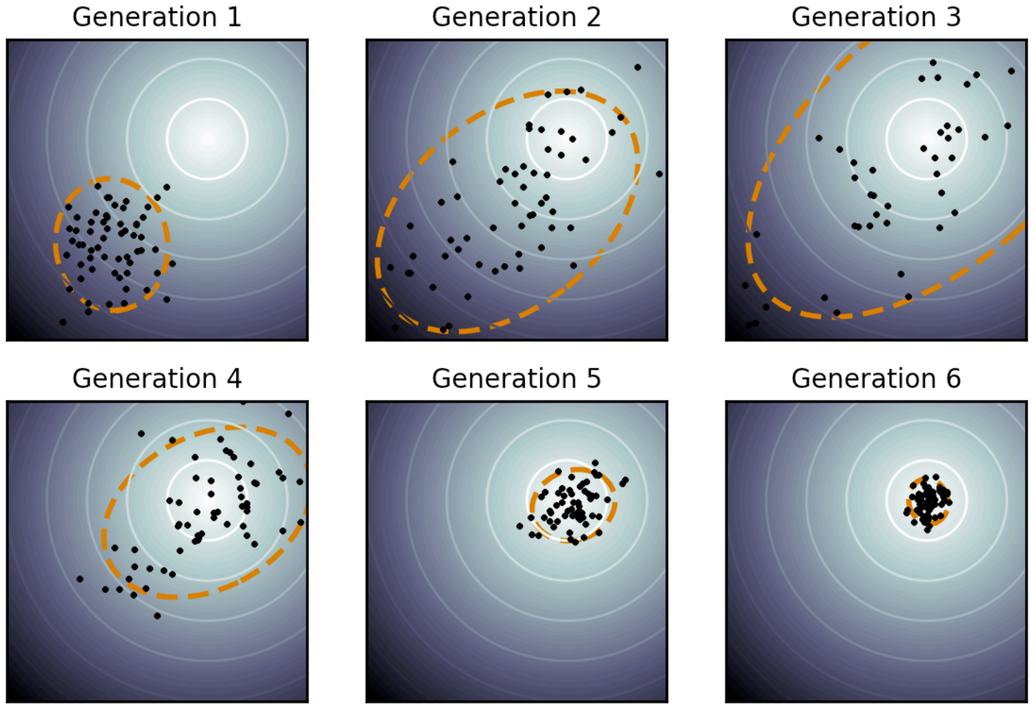
Question: what could happen if we try to optimize **initial parameters** with ES?

Unlikely to observe good initial parameters, because parameter space is high-dimensional.

Evolution Strategies

Initialize parameters $(\mu, \sigma) \leftarrow (\mu_0, \sigma_0)$. Repeat:

1. Sample particles: $x^1, x^2, \ldots, x^N \sim \mathcal{N}(\mu, \sigma^2 I)$
2. Evaluate and get best: $\{e^1, \ldots, e^n\} \subset \{x^1, \ldots, x^N\}$
3. $\mu, \sigma^2 \leftarrow \text{Avg}(e^1, \ldots, e^n), \text{Var}(e^1, \ldots, e^n)$

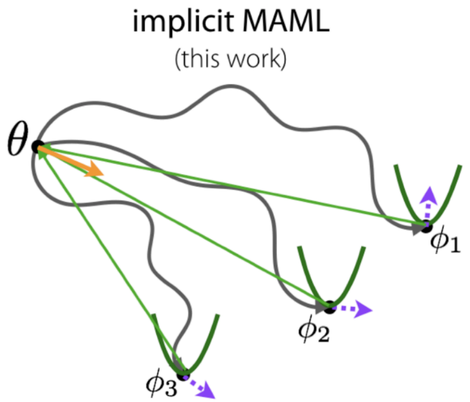


From: Wikipedia CMA-ES page

+ **Constant memory cost**
+ **Parallelizable** across particles
+ Inner steps can be non-differentiable
- Struggles with **high-dimensional covariates** and/or **complex loss surfaces**

Other Approaches

Implicit Differentiation



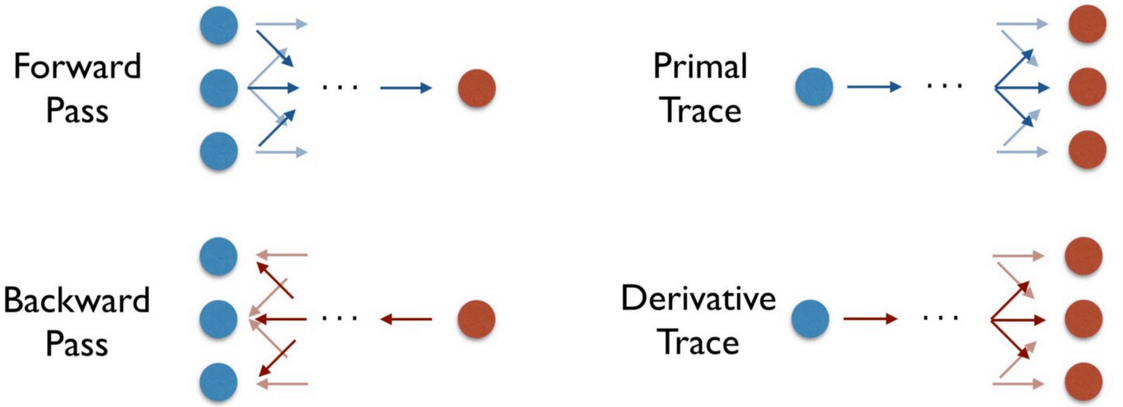
Computes full meta-gradient based only on the final result of the inner loop.

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \, \frac{1}{M} \sum_{i=1}^{M} \frac{d\mathcal{A}lg_i^\star(\boldsymbol{\theta})}{d\boldsymbol{\theta}} \, \nabla_\phi \mathcal{L}_i(\mathcal{A}lg_i^\star(\boldsymbol{\theta})).$$

$$\frac{d\mathcal{A}lg_i^\star(\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \left(\boldsymbol{I} + \frac{1}{\lambda} \nabla^2_{\boldsymbol{\phi}} \hat{\mathcal{L}}_i(\boldsymbol{\phi}_i) \right)^{-1}$$

From: "Meta-Learning with Implicit Gradients", Rajeswaran et al. (2019)

Forward-mode Differentiation



Uses the chain rule in the opposite direction from backprop, accumulating derivatives from start to finish.

From: "Forward Mode Automatic Differentiation & Dual Numbers", Lange

Plan for Today

Why consider large-scale meta-optimization?

Applications

Approaches
- Truncated backpropagation
- Gradient-free optimization

Goals for by the end of lecture:
- Know scenarios where existing meta-learning approaches fail due to scale
- Understand techniques for large-scale meta-optimization

This Week

A Bayesian perspective on meta-learning

Today: Approximate Bayesian inference via variational inference

Plan for Today

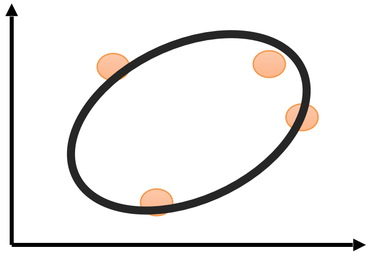
1. Latent variable models
2. Variational inference
3. Amortized variational inference
4. Example latent variables models

Goals

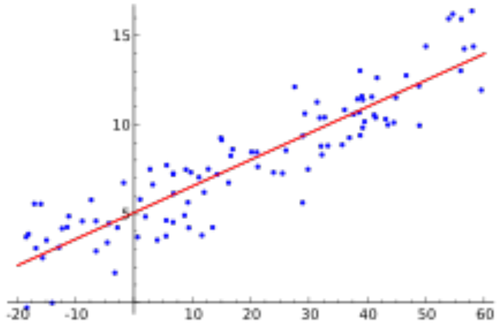
- Understand latent variable models in deep learning
- Understand how to use (amortized) variational inference

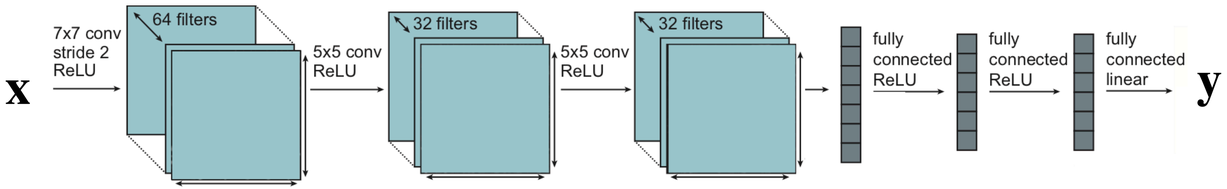
Probabilistic models

$p(x)$



$p(y|x)$





$f_\theta(\mathbf{y} \mid \mathbf{x})$

Most commonly:

- probability values of discrete **categorical distribution**
- mean and variance of a **Gaussian**

But it could be other distributions!

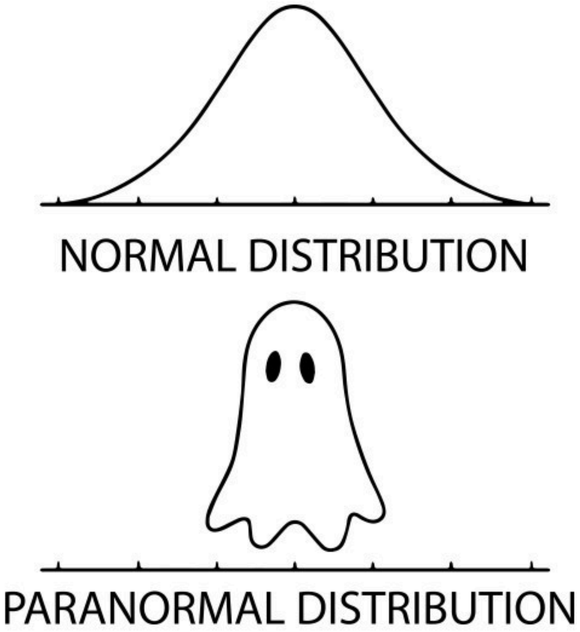
How do we train probabilistic models?

the model: $p_\theta(x)$

the data: $\mathcal{D} = \{x_1, x_2, x_3, \ldots, x_N\}$

maximum likelihood fit:

$$\theta \leftarrow \arg\max_\theta \frac{1}{N} \sum_i \log p_\theta(x_i)$$



Easy to evaluate & differentiate for **categorical** or **Gaussian** distributions.
i.e. cross-entropy, MSE losses

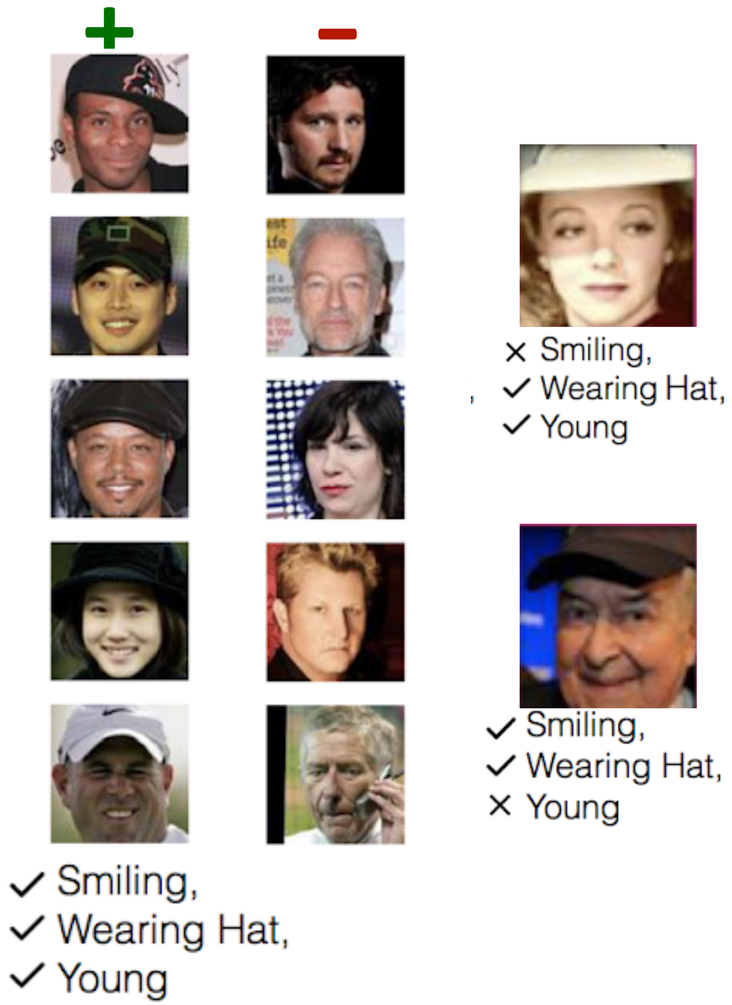
Goal: Can we model and train more complex distributions?

When might we want more complex distributions?

- **generative models** of images, text, video, or other data
- represent **uncertainty over labels** (e.g. ambiguity arising from limited data, partial observability)
- represent **uncertainty over** ***functions***

"HD Video: Riding a horse in the park at sunrise"

Meta-learning methods represent a **deterministic** $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$ (i.e. a point estimate)



Why/when is this a problem?

Few-shot learning problems may be ***ambiguous***.
(even with prior)

Can we learn to ***generate hypotheses***
about the underlying function?
i.e. sample from $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$

Important for:

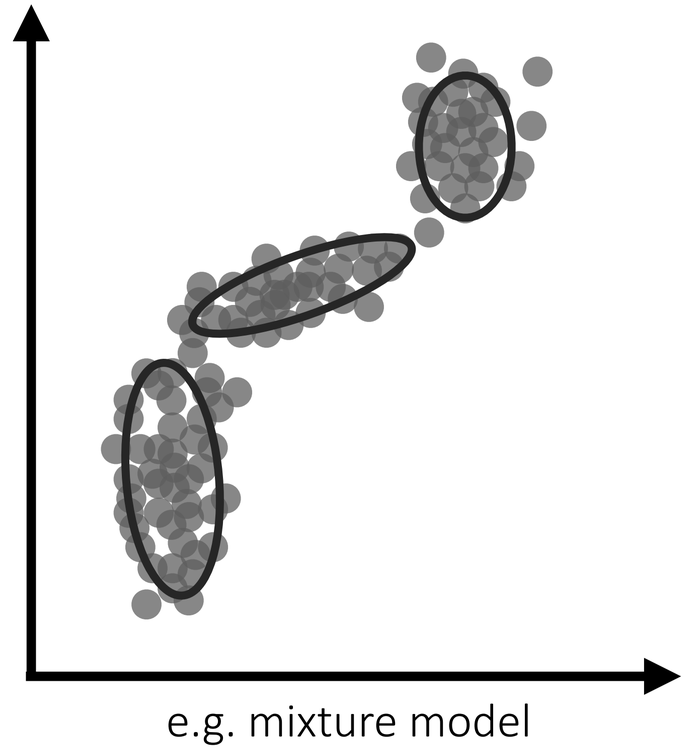
- **safety-critical** few-shot learning (e.g. medical imaging)
- learning to **actively learn**
- learning to **explore** in meta-RL

Active learning w/ meta-learning: Woodward & Finn ’16, Konyushkova et al. ’17, Bachman et al. ’17

Goal: Can we model and train complex distributions?

Latent variable models: examples

$p(x)$



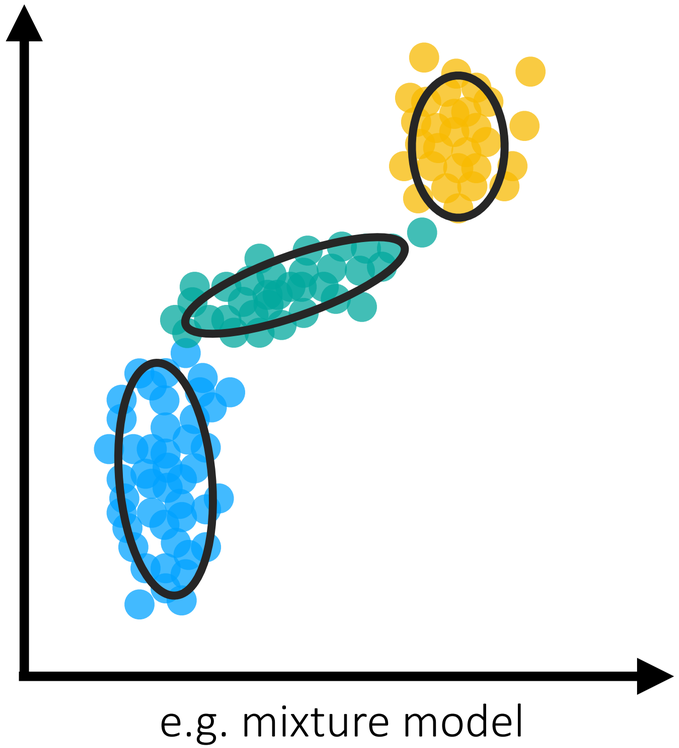
e.g. mixture model

Latent variable models: examples

$$p(x) = \sum_z p(x|z)p(z)$$

mixture element

e.g. Gaussian



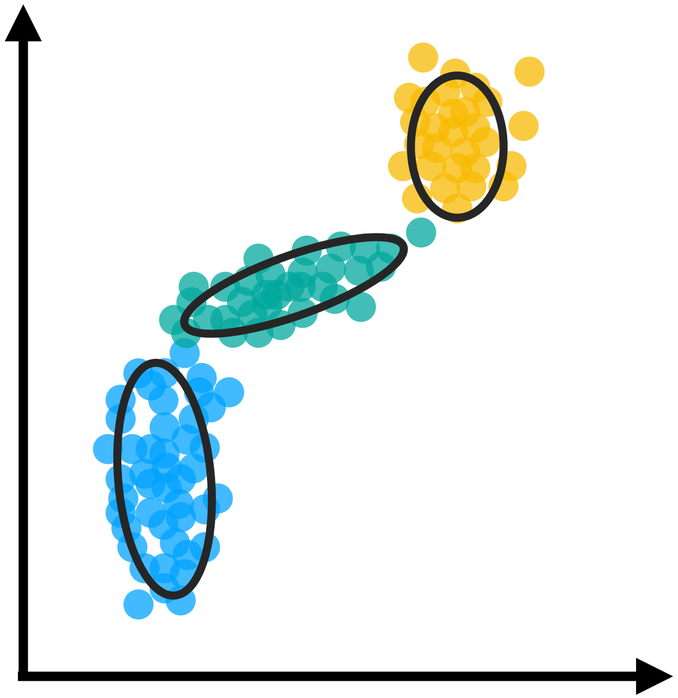
e.g. mixture model

Latent variable models: examples

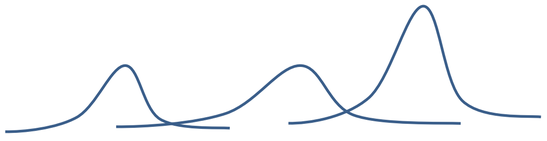
$$p(x) = \sum_z p(x|z)p(z)$$

mixture element

e.g. Gaussian

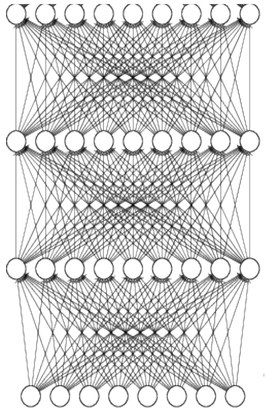


e.g. mixture model



$$w_1, \mu_1, \Sigma_1, \ldots, w_N, \mu_N, \sigma_N$$

length of paper



ImageNet Classification with Deep Convolutional Neural Networks

$$p(y|x) = \sum_z p(y|x,z)p(z|x)$$

e.g. mixture density network

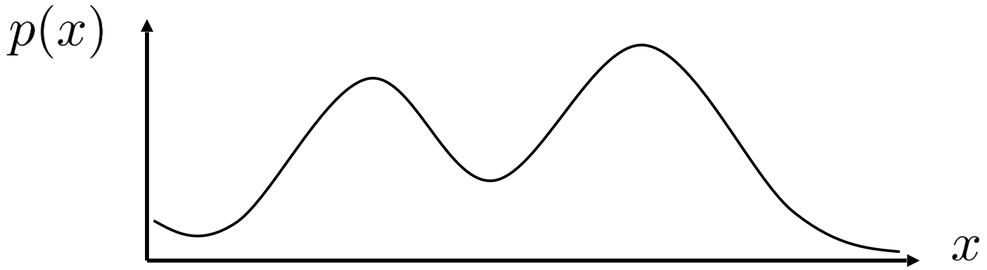
Latent variable models in general

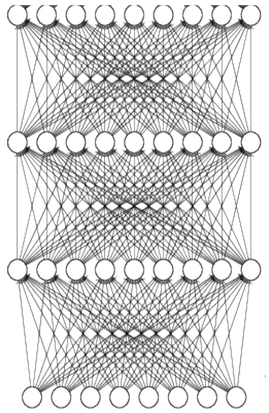
$$p(x) = \int p(x|z)p(z)dz$$

"easy" distribution
(e.g., conditional Gaussian)

"easy" distribution
(e.g., Gaussian)

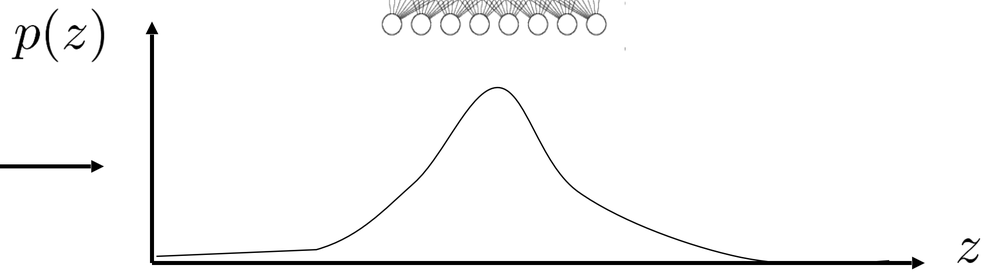
complicated distribution





$$p(x|z) = \mathcal{N}(\mu_{\text{nn}}(z), \sigma_{\text{nn}}(z))$$

"easy" distribution
(e.g., Gaussian)



💡 Key idea: represent complex distribution by composing two simple distributions

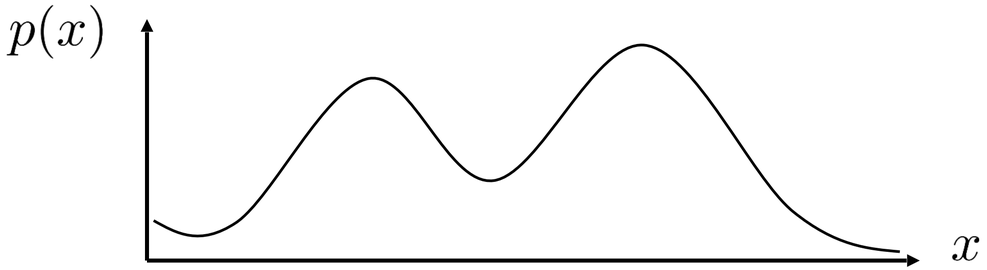
Latent variable models in general

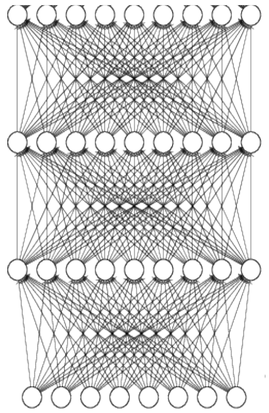
$$p(x) = \int p(x|z)p(z)dz$$

"easy" distribution
(e.g., conditional Gaussian)

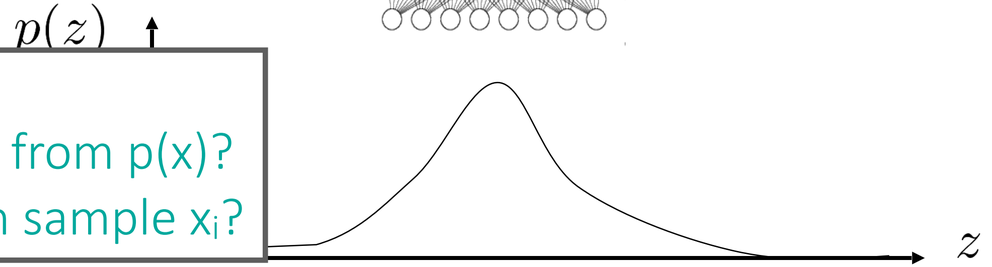
"easy" distribution
(e.g., Gaussian)

complicated distribution





$$p(x|z) = \mathcal{N}(\mu_{\text{nn}}(z), \sigma_{\text{nn}}(z))$$



Questions:
1. Once trained, how do you generate a sample from p(x)?
2. How do you evaluate the likelihood of a given sample x_i?

💡 Key idea: represent complex distribution by composing two simple distributions

How do we train latent variable models?

the model: $p_\theta(x)$

the data: $\mathcal{D} = \{x_1, x_2, x_3, \ldots, x_N\}$

maximum likelihood fit:

$$\theta \leftarrow \arg\max_\theta \frac{1}{N} \sum_i \log p_\theta(x_i)$$

$$p(x) = \int p(x|z)p(z)dz$$

$$\theta \leftarrow \arg\max_\theta \frac{1}{N} \sum_i \log \left(\int p_\theta(x_i|z)p(z)dz \right)$$

completely intractable

Flavors of Deep Latent Variable Models

Use latent variables:

- generative adversarial networks (GANs)
- variational autoencoders (VAEs)
- normalizing flow models
- diffusion models

All differ in how they are trained.

Do not use latent variables:

- autoregressive models

(recall generative pre-training lecture)

Variational Inference

A. Formulate **a lower bound** on the log likelihood objective.
B. Check **how tight** the bound is.
C. Variational inference -> ***Amortized*** variational inference
D. How to **optimize**

Estimating the log-likelihood

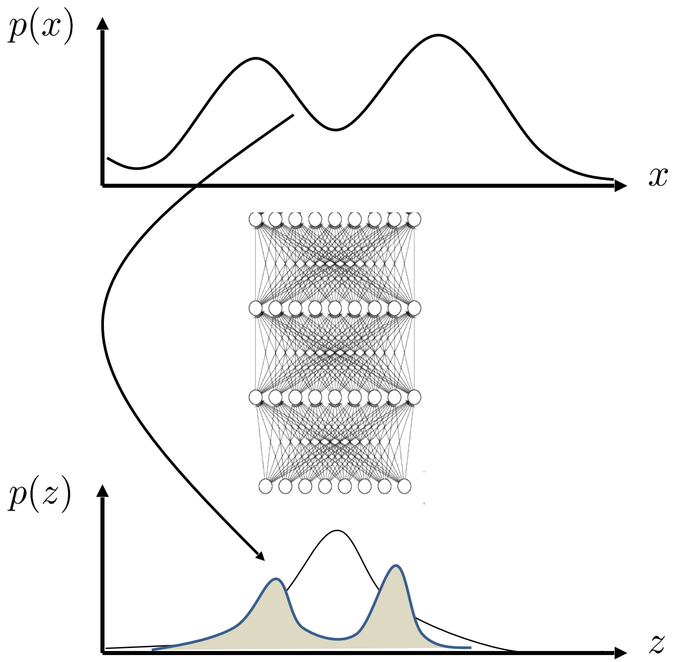
alternative: *expected* log-likelihood:

$$\theta \leftarrow \arg\max_{\theta} \frac{1}{N} \sum_{i} E_{z \sim p(z|x_i)} [\log p_\theta(x_i, z)]$$

but... how do we calculate $p(z|x_i)$?

intuition: "guess" most likely z given x_i, and pretend it's the right one

...but there are many possible values of z so use the distribution $p(z|x_i)$



The variational approximation

but... how do we calculate $p(z|x_i)$?

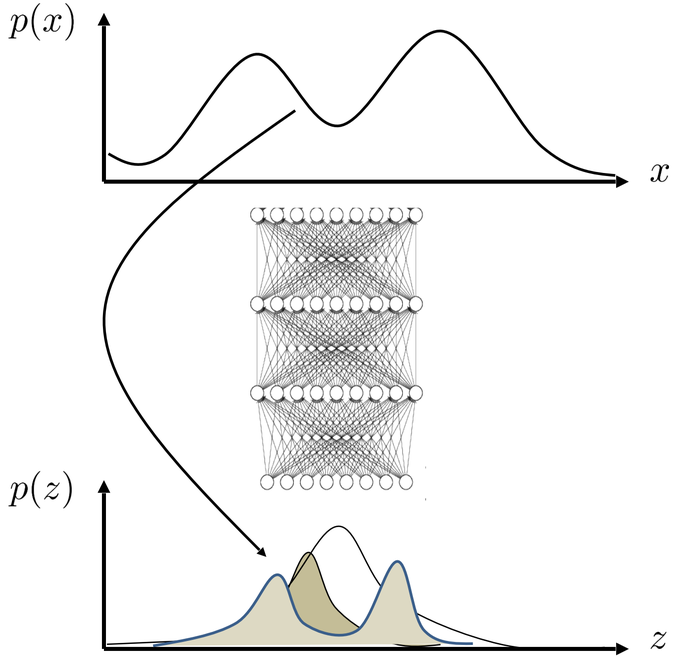
can bound $\log p(x_i)$!

$$\log p(x_i) = \log \int_z p(x_i|z)p(z)$$

$$= \log \int_z p(x_i|z)p(z)\frac{q_i(z)}{q_i(z)}$$

$$= \log E_{z\sim q_i(z)}\left[\frac{p(x_i|z)p(z)}{q_i(z)}\right]$$

what if we approximate with $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$



The variational approximation

Jensen's inequality

$$\log E[y] \geq E[\log y]$$

but... how do we calculate $p(z|x_i)$?

can bound $\log p(x_i)$!

$$\log p(x_i) = \log \int_z p(x_i|z)p(z)$$

$$= \log \int_z p(x_i|z)p(z)\frac{q_i(z)}{q_i(z)}$$

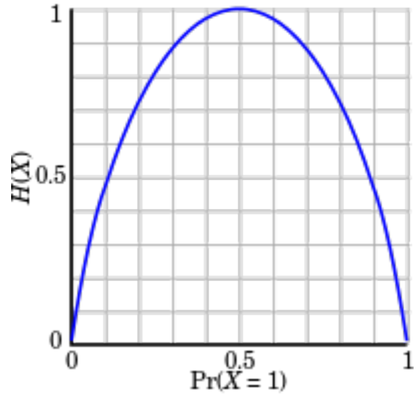
$$= \log E_{z\sim q_i(z)}\left[\frac{p(x_i|z)p(z)}{q_i(z)}\right]$$

$$\geq E_{z\sim q_i(z)}\left[\log \frac{p(x_i|z)p(z)}{q_i(z)}\right] = E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + H(q_i)$$

maximizing this maximizes $\log p(x_i)$

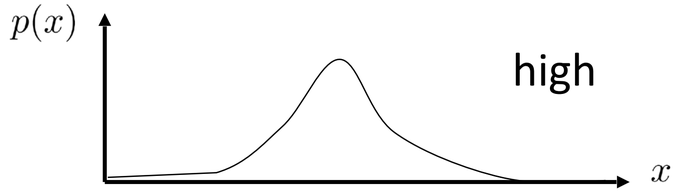
"evidence lower bound" (ELBO)

A brief aside…



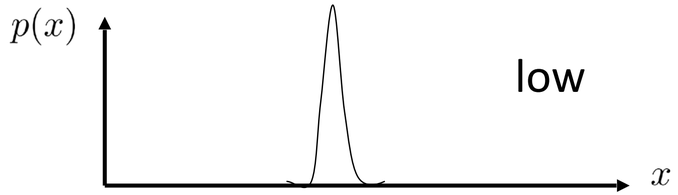
Entropy:

$$\mathcal{H}(p) = -E_{x\sim p(x)}[\log p(x)] = -\int_x p(x)\log p(x)dx$$



Intuition 1: how *random* is the random variable?

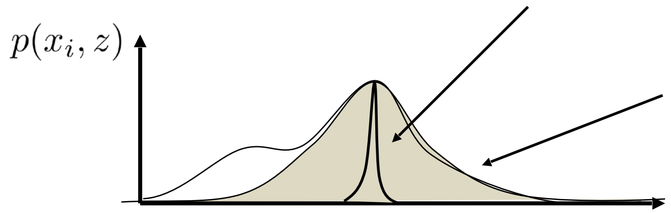
Intuition 2: how large is the log probability in expectation *under itself*



what do we expect this to do?

$$E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)$$

this maximizes the first part



this also maximizes the second part
(makes it as wide as possible)

A brief aside…

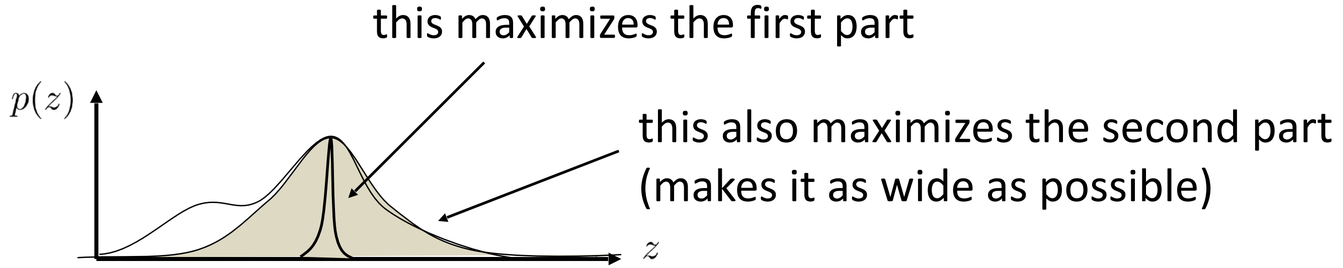
KL-Divergence:

$$D_{\mathrm{KL}}(q\|p) = E_{x\sim q(x)}\left[\log \frac{q(x)}{p(x)}\right] = E_{x\sim q(x)}[\log q(x)] - E_{x\sim q(x)}[\log p(x)] = -E_{x\sim q(x)}[\log p(x)] - \mathcal{H}(q)$$

Intuition 1: how *different* are two distributions? e.g. when q=p, KL divergence is 0

Intuition 2: how small is the expected log probability of one distribution under another, minus entropy?

why entropy?



How tight is the lower bound?

$\mathcal{L}_i(p, q_i)$ "evidence lower bound" (ELBO)

$$\log p(x_i) \geq \overbrace{E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}$$

what makes a good $q_i(z)$?

approximate in what sense?

why?

intuition: $q_i(z)$ should approximate $p(z|x_i)$

compare in terms of KL-divergence: $D_{\mathrm{KL}}(q_i(z)\|p(z|x))$

$$D_{\mathrm{KL}}(q_i(z)\|p(z|x_i)) = E_{z\sim q_i(z)}\left[\log \frac{q_i(z)}{p(z|x_i)}\right] = E_{z\sim q_i(z)}\left[\log \frac{q_i(z)p(x_i)}{p(x_i, z)}\right]$$

$$= -E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + E_{z\sim q_i(z)}[\log q_i(z)] + E_{z\sim q_i(z)}[\log p(x_i)]$$

$$= -E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] - \mathcal{H}(q_i) + \log p(x_i)$$

$$= -\mathcal{L}_i(p, q_i) + \log p(x_i)$$

$$\log p(x_i) = D_{\mathrm{KL}}(q_i(z)\|p(z|x_i)) + \mathcal{L}_i(p, q_i)$$

Note 1: If KL divergence is 0, then bound is tight.

$$\log p(x_i) \geq \mathcal{L}_i(p, q_i)$$

How tight is the lower bound?

$\mathcal{L}_i(p, q_i)$ "evidence lower bound" (ELBO)

$$\log p(x_i) \geq \overbrace{E_{z\sim q_i(z)}[\log p(x_i|z) + \log p(z)] + \mathcal{H}(q_i)}$$

what makes a good $q_i(z)$?

approximate in what sense?

why?

intuition: $q_i(z)$ should approximate $p(z|x_i)$

compare in terms of KL-divergence: $D_{\mathrm{KL}}(q_i(z)\|p(z|x))$

$$D_{\mathrm{KL}}(q_i(z)\|p(z|x_i)) = -\mathcal{L}_i(p, q_i) + \log p(x_i)$$

Note 2: Maximizing $L(p, q_i)$ w.r.t. q_i minimizes the KL divergence.

$$\log p(x_i) = D_{\mathrm{KL}}(q_i(z)\|p(z|x_i)) + \mathcal{L}_i(p, q_i)$$

Note 1: If KL divergence is 0, then bound is tight.

$$\log p(x_i) \geq \mathcal{L}_i(p, q_i)$$

Optimization objective: $\max_{\theta, q_i} \frac{1}{N} \sum_i \mathcal{L}_i(p_\theta, q_i)$

Optimizing the ELBO

$\mathcal{L}_i(p, q_i)$ "evidence lower bound" (ELBO)

$$\log p(x_i) \geq E_{z \sim q_i(z)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_i)$$

~~$\theta \leftarrow \arg\max_\theta \frac{1}{N} \sum_i \log p_\theta(x_i)$~~

$$\theta \leftarrow \arg\max_\theta \frac{1}{N} \sum_i \mathcal{L}_i(p, q_i)$$

for each x_i (or mini-batch):

calculate $\nabla_\theta \mathcal{L}_i(p, q_i)$:

sample $z \sim q_i(z)$

$\nabla_\theta \mathcal{L}_i(p, q_i) \approx \nabla_\theta \log p_\theta(x_i|z)$

$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_i(p, q_i)$

update q_i to maximize $\mathcal{L}_i(p, q_i)$ ← how?

let's say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on μ_i, σ_i

What's the problem?

for each x_i (or mini-batch):

calculate $\nabla_\theta \mathcal{L}_i(p, q_i)$:

sample $z \sim q_i(z)$

$\nabla_\theta \mathcal{L}_i(p, q_i) \approx \nabla_\theta \log p_\theta(x_i|z)$

$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_i(p, q_i)$

update q_i to maximize $\mathcal{L}_i(p, q_i)$

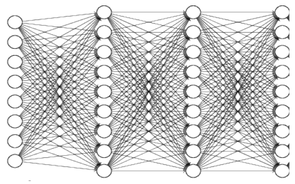
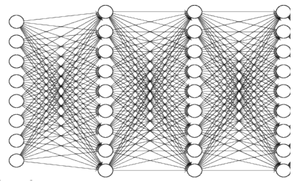
let's say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

gradient ascent on μ_i, σ_i

Question: How many parameters are there? $|\theta| + (|\mu_i| + |\sigma_i|) \times N$

intuition: $q_i(z)$ should approximate $p(z|x_i)$ what if we learn a *network* $q_i(z) = q(z|x_i) \approx p(z|x_i)$?

z  $p_\theta(x|z)$ x  $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$

Amortized Variational Inference

A. Formulate **a lower bound** on the log likelihood objective.
B. Check **how tight** the bound is.
C. Variational inference -> ***Amortized*** variational inference
D. How to **optimize**

What's the problem?

for each x_i (or mini-batch):

calculate $\nabla_\theta \mathcal{L}_i(p, q_i)$:

sample $z \sim q_i(z)$

$\nabla_\theta \mathcal{L}_i(p, q_i) \approx \nabla_\theta \log p_\theta(x_i|z)$

$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_i(p, q_i)$

update q_i to maximize $\mathcal{L}_i(p, q_i)$

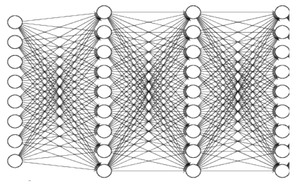
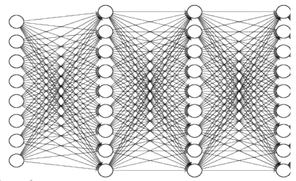
let's say $q_i(z) = \mathcal{N}(\mu_i, \sigma_i)$

use gradient $\nabla_{\mu_i} \mathcal{L}_i(p, q_i)$ and $\nabla_{\sigma_i} \mathcal{L}_i(p, q_i)$

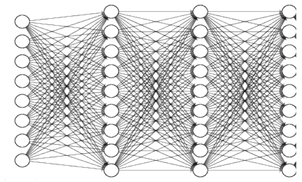
gradient ascent on μ_i, σ_i

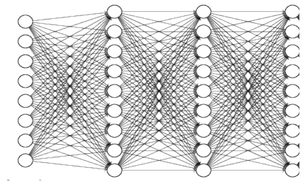
Question: How many parameters are there? $|\theta| + (|\mu_i| + |\sigma_i|) \times N$

intuition: $q_i(z)$ should approximate $p(z|x_i)$ what if we learn a *network* $q_i(z) = q(z|x_i) \approx p(z|x_i)$?

z  $p_\theta(x|z)$ x  $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$

Amortized variational inference

z  $p_\theta(x|z)$

x  $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$

$$\log p(x_i) \geq \overbrace{E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_\phi(z|x_i))}^{\mathcal{L}(p_\theta(x_i|z), q_\phi(z|x_i))}$$

for each x_i (or mini-batch):

  calculate $\nabla_\theta \mathcal{L}(p_\theta(x_i|z), q_\phi(z|x_i))$:

    sample $z \sim q_\phi(z|x_i)$

    $\nabla_\theta \mathcal{L} \approx \nabla_\theta \log p_\theta(x_i|z)$

  $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}$

  $\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{L}$

how do we calculate this?

Amortized variational inference

for each x_i (or mini-batch):

calculate $\nabla_\theta \mathcal{L}(p_\theta(x_i|z), q_\phi(z|x_i))$:

sample $z \sim q_\phi(z|x_i)$

$\nabla_\theta \mathcal{L} \approx \nabla_\theta \log p_\theta(x_i|z)$

$\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}$

$\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{L}$

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$

look up formula for entropy of a Gaussian

$$\mathcal{L}_i = \underbrace{E_{z \sim q_\phi(z|x_i)}[\log p_\theta(x_i|z) + \log p(z)]}_{J(\phi) = E_{z \sim q_\phi(z|x_i)}[r(x_i, z)]} + \mathcal{H}(q_\phi(z|x_i))$$

The reparameterization trick

$$J(\phi) = E_{z \sim q_\phi(z|x_i)}[r(x_i, z)]$$
$$= E_{\epsilon \sim \mathcal{N}(0,1)}[r(x_i, \mu_\phi(x_i) + \epsilon \sigma_\phi(x_i))]$$

$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$

$$z = \mu_\phi(x) + \epsilon \sigma_\phi(x)$$

$\epsilon \sim \mathcal{N}(0,1)$

independent of ϕ!

estimating $\nabla_\phi J(\phi)$:

sample $\epsilon_1, \ldots, \epsilon_M$ from $\mathcal{N}(0,1)$ (a single sample works well!)

$$\nabla_\phi J(\phi) \approx \frac{1}{M} \sum_j \nabla_\phi r(x_i, \mu_\phi(x_i) + \epsilon_j \sigma_\phi(x_i))$$

\+ Very simple to implement

\+ Low variance

\- Only continuous latent variables

Discrete latent variables:

- vector quantization & straight-through estimator (“VQ-VAE”)
- policy gradients / “REINFORCE”

Another way to look at everything…

$$\mathcal{L}_i = E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z) + \log p(z)] + \mathcal{H}(q_\phi(z|x_i))$$

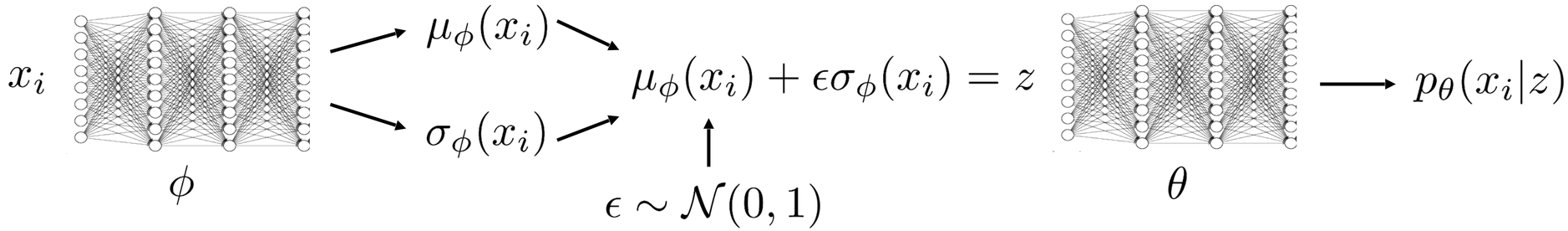
$$= E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] + \underbrace{E_{z\sim q_\phi(z|x_i)}[\log p(z)] + \mathcal{H}(q_\phi(z|x_i))}_{-D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))}$$

this has a convenient analytical form for Gaussians

$$= E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

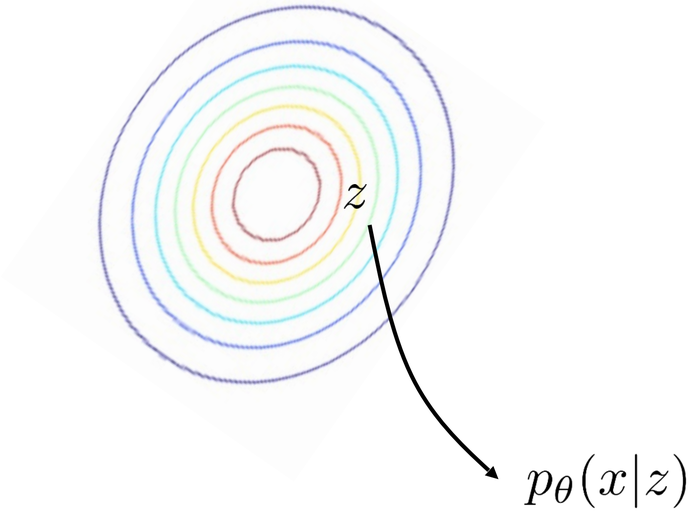
$$= E_{\epsilon\sim\mathcal{N}(0,1)}[\log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i))] - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

$$\approx \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i)) - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

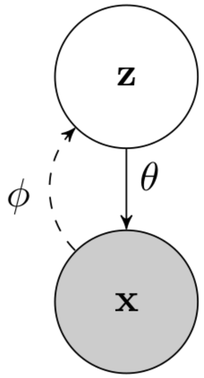


Example Models

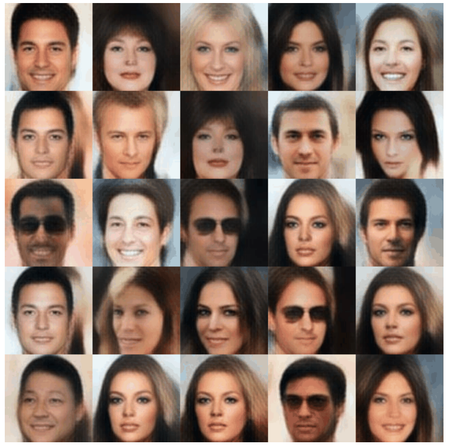
The variational autoencoder

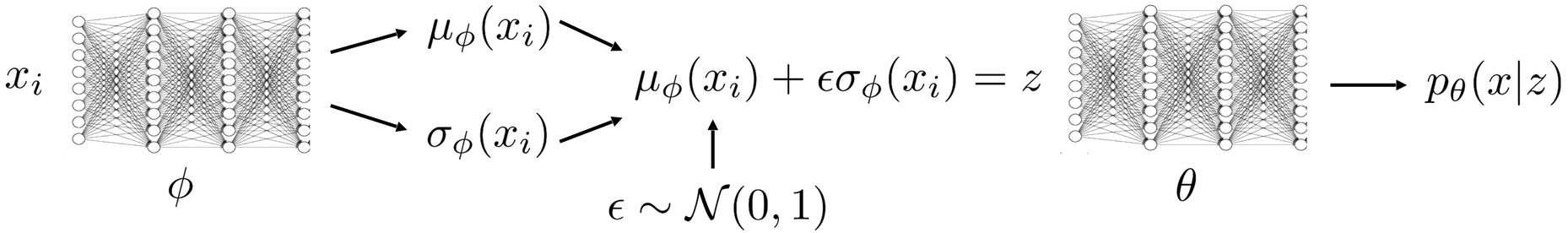


$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$



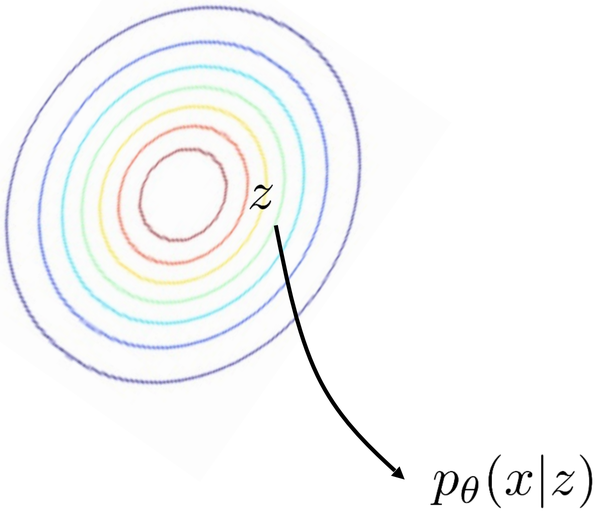
$$p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$$



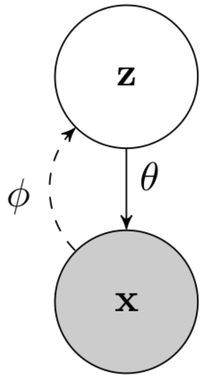


$$\max_{\theta,\phi} \frac{1}{N} \sum_i \log p_\theta(x_i|\mu_\phi(x_i) + \epsilon\sigma_\phi(x_i)) - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

Using the variational autoencoder



$$q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \sigma_\phi(x))$$



$$p_\theta(x|z) = \mathcal{N}(\mu_\theta(z), \sigma_\theta(z))$$

$$p(x) = \int p(x|z)p(z)dz$$

why does this work?

sampling:

$z \sim p(z)$

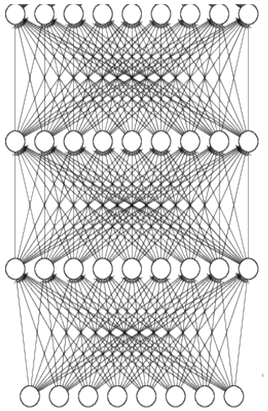
$x \sim p(x|z)$

$$\mathcal{L}_i = E_{z\sim q_\phi(z|x_i)}[\log p_\theta(x_i|z)] - D_{\mathrm{KL}}(q_\phi(z|x_i)\|p(z))$$

Conditional models

$$\mathcal{L}_i = E_{z \sim q_\phi(z|x_i,y_i)}[\log p_\theta(y_i|x_i,z) + \log p(z|x_i)] + \mathcal{H}(q_\phi(z|x_i,y_i))$$

just like before, only now generating y_i
and *everything* is conditioned on x_i



$p(y|x,z)$

class 109
(brain coral)

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

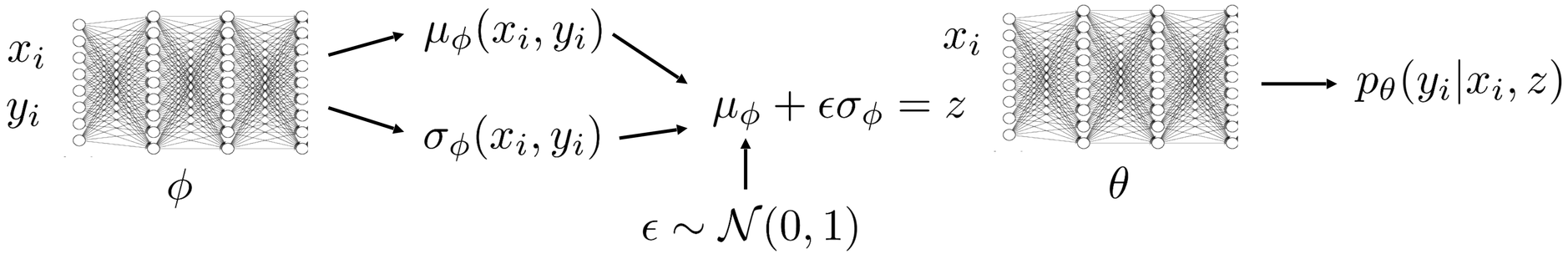
$p(z)$

can *optionally* depend on x

at test time:

$z \sim p(z|x_i)$

$y \sim p(y|x_i,z)$



Images from Razavi, van den Oord, Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. ‘19

Plan for Today

Why be Bayesian?

Bayesian meta-learning approaches

- black-box approaches
- optimization-based approaches

How to evaluate Bayesian meta-learners.

Goals for by the end of lecture:

- Understand the interpretation of meta-learning as Bayesian inference
- Understand techniques for representing uncertainty over parameters, predictions

Disclaimers

Bayesian meta-learning is an **active area of research**
(like most of the class content)

More **questions** than answers.

Recap: Properties of Meta-Learning Inner Loops

***Algorithmic properties* perspective**

Expressive power

the ability for f to represent a range of learning procedures

Why? scalability, applicability to a range of domains

Consistency

learned learning procedure will solve task with enough data

Why? reduce reliance on meta-training tasks, good OOD task performance

These properties are important for most applications!

Recap: Properties of Meta-Learning Inner Loops

***Algorithmic properties* perspective**

Expressive power

the ability for f to represent a range of learning procedures

Why? scalability, applicability to a range of domains

Consistency

learned learning procedure will solve task with enough data

Why? reduce reliance on meta-training tasks, good OOD task performance

Uncertainty awareness

ability to reason about ambiguity during learning

Why? active learning, calibrated uncertainty, RL principled Bayesian approaches

this lecture

Plan for Today

Why be Bayesian?

Bayesian meta-learning approaches

- black-box approaches
- optimization-based approaches

How to evaluate Bayesian meta-learners.

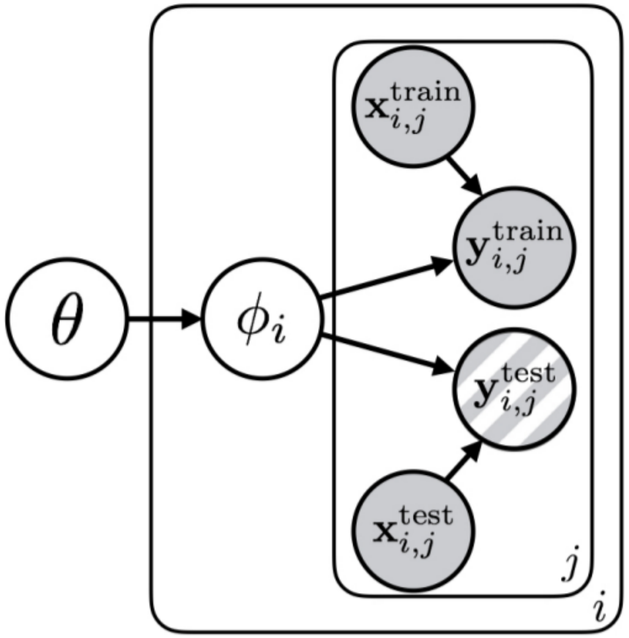
Multi-Task & Meta-Learning Principles

Training and testing must match.

Tasks must share "structure."

What does "structure" mean?

statistical dependence on shared latent information θ



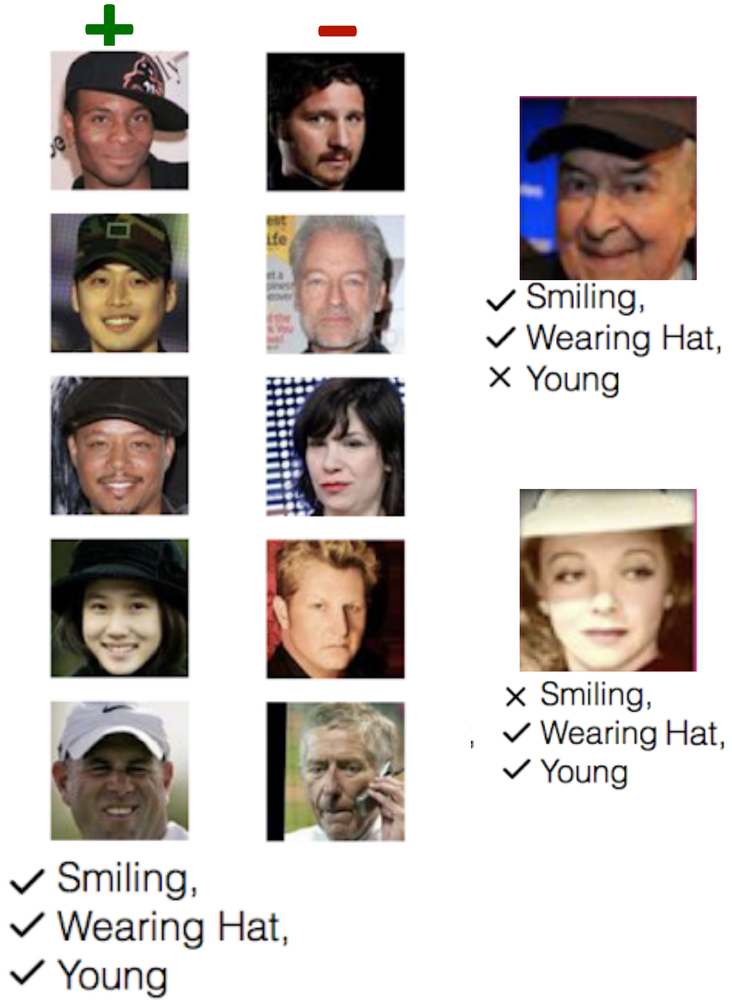
If you condition on that information,

- task parameters become independent
  i.e. $\phi_{i_1} \perp\!\!\!\perp \phi_{i_2} \mid \theta$
  and are not otherwise independent $\phi_{i_1} \not\perp\!\!\!\perp \phi_{i_2}$
- hence, you have a lower entropy
  i.e. $\mathscr{H}(p(\phi_i \mid \theta)) < \mathscr{H}(p(\phi_i))$

Thought exercise #1: If you can identify θ (i.e. with meta-learning), when should learning ϕ_i be faster than learning from scratch?

Thought exercise #2: what if $\mathscr{H}(p(\phi_i \mid \theta)) = 0 \quad \forall i$?

Recall parametric approaches: Use **deterministic** $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$ (i.e. a point estimate)



Why/when is this a problem?

Few-shot learning problems may be ***ambiguous***.
(even with prior)

Can we learn to ***generate hypotheses***
about the underlying function?
i.e. sample from $p(\phi_i|\mathcal{D}_i^{\text{tr}}, \theta)$

Important for:

- **safety-critical** few-shot learning (e.g. medical imaging)
- learning to **actively learn**
- learning to **explore** in meta-RL

Active learning w/ meta-learning: Woodward & Finn ’16, Konyushkova et al. ’17, Bachman et al. ’17

Plan for Today

Why be Bayesian?

Bayesian meta-learning approaches

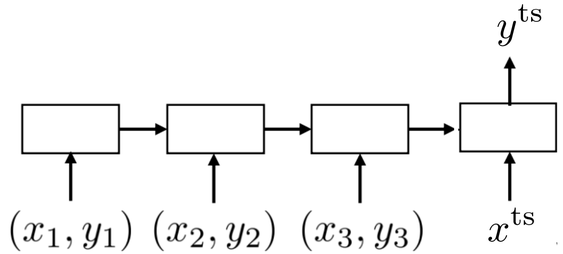
- black-box approaches
- optimization-based approaches

How to evaluate Bayesian meta-learners.

Meta-learning algorithms as computation graphs

Black-box

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



Optimization-based

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

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

Non-parametric

$$y^{\text{ts}} = f_{\text{PN}}(\mathcal{D}_i^{\text{tr}}, x^{\text{ts}}) = \text{softmax}(-d\left(f_\theta(x^{\text{ts}}), \mathbf{c}_n\right))$$

$$\text{where } \mathbf{c}_n = \frac{1}{K} \sum_{(x,y) \in \mathcal{D}_i^{\text{tr}}} \mathbb{1}(y = n) f_\theta(x)$$

Version 0: Let f output the parameters of a distribution over y^{ts}.

For example:

- probability values of discrete **categorical distribution**
- mean and variance of a **Gaussian**
- means, variances, and mixture weights of a **mixture of Gaussians**
- for multi-dimensional y^{ts}: parameters of a **sequence of distributions** (i.e. autoregressive model)

Then, optimize with maximum likelihood.

Version 0: Let f output the parameters of a distribution over y^{ts}.

For example:

- probability values of discrete **categorical distribution**
- mean and variance of a **Gaussian**
- means, variances, and mixture weights of a **mixture of Gaussians**
- for multi-dimensional y^{ts}: parameters of a **sequence of distributions** (i.e. autoregressive model)

Then, optimize with **maximum likelihood**.

Pros:
+ simple
+ can combine with variety of methods

Cons:
- can't reason about uncertainty over the underlying function [to determine how uncertainty across datapoints relate]
- limited class of distributions over y^{ts} can be expressed
- tends to produce poorly-calibrated uncertainty estimates

Thought exercise #4: Can you do the same maximum likelihood training for ϕ?

The Bayesian Deep Learning Toolbox

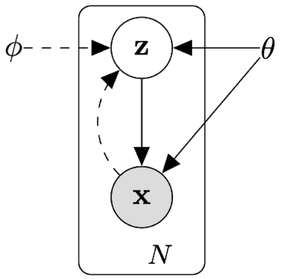
a broad one-slide overview

(CS 236 provides a thorough treatment)

Goal: represent distributions with neural networks

Latent variable models + variational inference (Kingma & Welling ‘13, Rezende et al. ‘14):

- approximate likelihood of latent variable model with variational lower bound

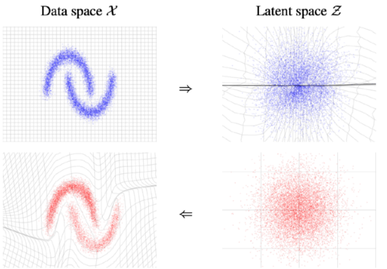


Bayesian ensembles (Lakshminarayanan et al. ‘17):

- particle-based representation: train separate models on bootstraps of the data

Bayesian neural networks (Blundell et al. ‘15):

- explicit distribution over the space of network parameters

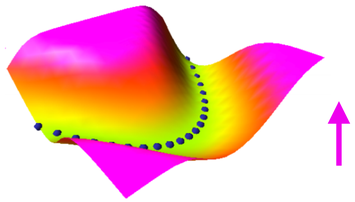


Normalizing Flows (Dinh et al. ‘16):

- invertible function from latent distribution to data distribution

Energy-based models & GANs (LeCun et al. ’06, Goodfellow et al. ‘14):

- estimate unnormalized density



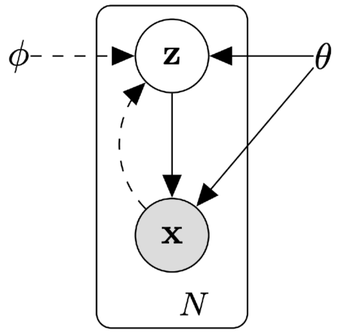
data

everything else

We’ll see how we can leverage the first two.

The others could be useful in developing new methods.

Recap: The Variational Lower Bound



Observed variable x, latent variable z

ELBO: $$\log p(x) \geq \mathbb{E}_{q(z|x)}\left[\log p(x,z)\right] + \mathscr{H}(q(z\,|\,x))$$

Can also be written as: $$= \mathbb{E}_{q(z|x)}\left[\log p(x\,|\,z)\right] - D_{KL}\left(q(z\,|\,x)\|p(z)\right)$$

p: model — $p(x\,|\,z)$ represented w/ neural net, $p(z)$ represented as $\mathcal{N}(\mathbf{0}, \mathbf{I})$

model parameters θ, variational parameters ϕ

$q(z\,|\,x)$: inference network, variational distribution

Problem: need to backprop through sampling

i.e. compute derivative of $\mathbb{E}_q$ w.r.t. q

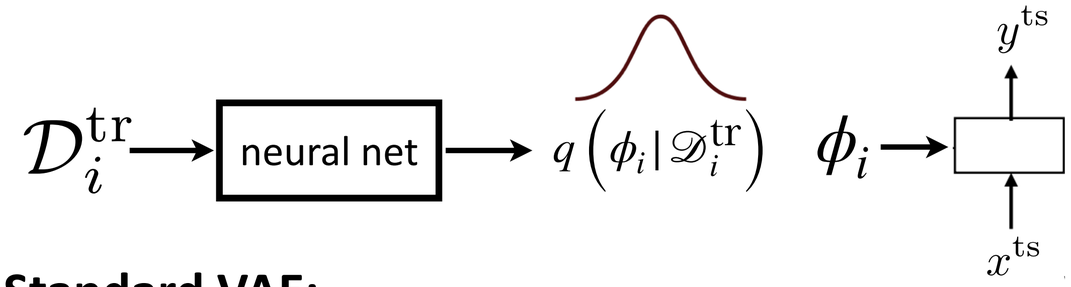
Reparametrization trick For Gaussian $q(z\,|\,x)$:

$q(z\,|\,x) = \mu_q + \sigma_q \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Can we use amortized variational inference for meta-learning?

Bayesian black-box meta-learning

with standard, deep variational inference



Standard VAE:

Observed variable x, latent variable z

ELBO: $\mathbb{E}_{q(z|x)}\left[\log p(x \mid z)\right] - D_{KL}\left(q(z \mid x) \| p(z)\right)$

p: model, represented by a neural net

q: inference network, variational distribution

Meta-learning:

Observed variable $\mathscr{D}$, latent variable ϕ

$$\max \mathbb{E}_{q(\phi)}\left[\log p(\mathscr{D} \mid \phi)\right] - D_{KL}\left(q(\phi) \| p(\phi)\right)$$

What should q condition on?

$$\max \mathbb{E}_{q\left(\phi \mid \mathscr{D}^{\text{tr}}\right)}\left[\log p(\mathscr{D} \mid \phi)\right] - D_{KL}\left(q\left(\phi \mid \mathscr{D}^{\text{tr}}\right) \| p(\phi)\right)$$

$$\max \mathbb{E}_{q\left(\phi \mid \mathscr{D}^{\text{tr}}\right)}\left[\log p\left(y^{\text{ts}} \mid x^{\text{ts}}, \phi\right)\right] - D_{KL}\left(q\left(\phi \mid \mathscr{D}^{\text{tr}}\right) \| p(\phi)\right)$$

What about the meta-parameters θ?

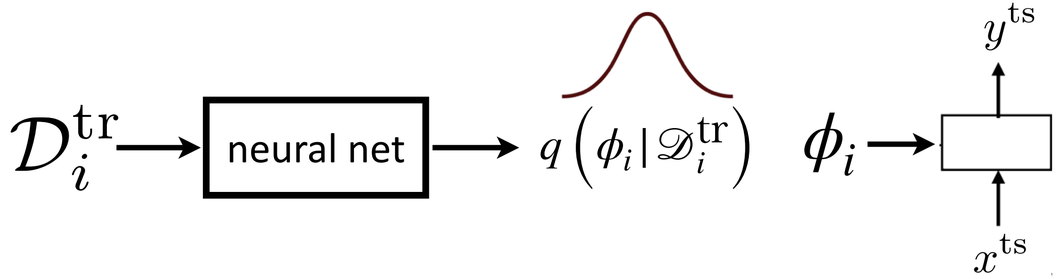
$$\max_{\theta} \mathbb{E}_{q\left(\phi \mid \mathscr{D}^{\text{tr}}, \theta\right)}\left[\log p\left(y^{\text{ts}} \mid x^{\text{ts}}, \phi\right)\right] - D_{KL}\left(q\left(\phi \mid \mathscr{D}^{\text{tr}}, \theta\right) \| p(\phi \mid \theta)\right)$$

Can also condition on θ here

Final objective (for completeness): $\max_{\theta} \mathbb{E}_{\mathscr{T}_i}\left[\mathbb{E}_{q\left(\phi_i \mid \mathscr{D}_i^{\text{tr}}, \theta\right)}\left[\log p\left(y_i^{\text{ts}} \mid x_i^{\text{ts}}, \phi_i\right)\right] - D_{KL}\left(q\left(\phi_i \mid \mathscr{D}_i^{\text{tr}}, \theta\right) \| p(\phi_i \mid \theta)\right)\right]$

Bayesian black-box meta-learning

with standard, deep variational inference



$$\max_{\theta} \mathbb{E}_{\mathcal{T}_i}\left[\mathbb{E}_{q\left(\phi_i \mid \mathcal{D}_i^{\text{tr}}, \theta\right)}\left[\log p\left(y_i^{\text{ts}} \mid x_i^{\text{ts}}, \phi_i\right)\right] - D_{KL}\left(q\left(\phi_i \mid \mathcal{D}_i^{\text{tr}}, \theta\right) \| p(\phi_i \mid \theta)\right)\right]$$

Pros:

+ can represent non-Gaussian distributions over y^{ts}
+ produces distribution over functions

Cons:

- Can only represent Gaussian distributions $p(\phi_i \mid \theta)$

Plan for Today

Why be Bayesian?

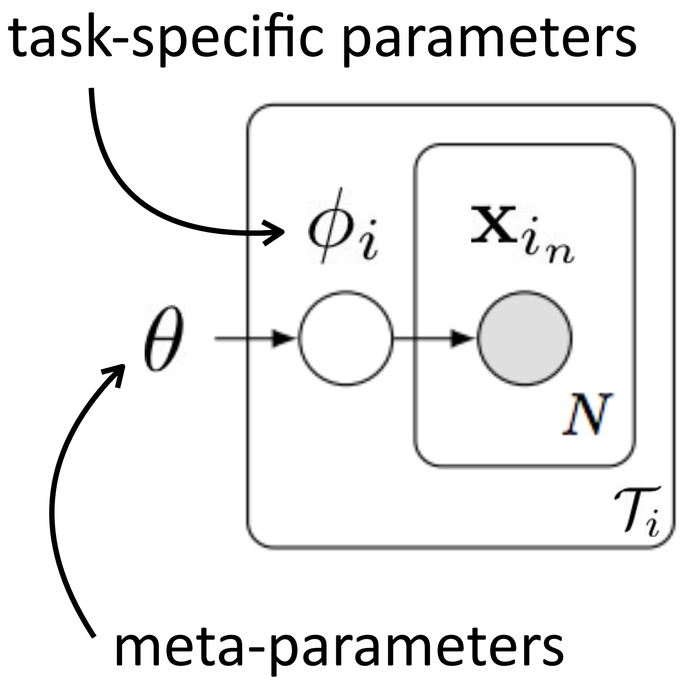
Bayesian meta-learning approaches

- black-box approaches
- **optimization-based approaches**

How to evaluate Bayesian meta-learners.

What about Bayesian **optimization-based** meta-learning?

Recasting Gradient-Based Meta-Learning as Hierarchical Bayes (Grant et al. ’18)



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

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

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

MAP estimate

How to compute MAP estimate?

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

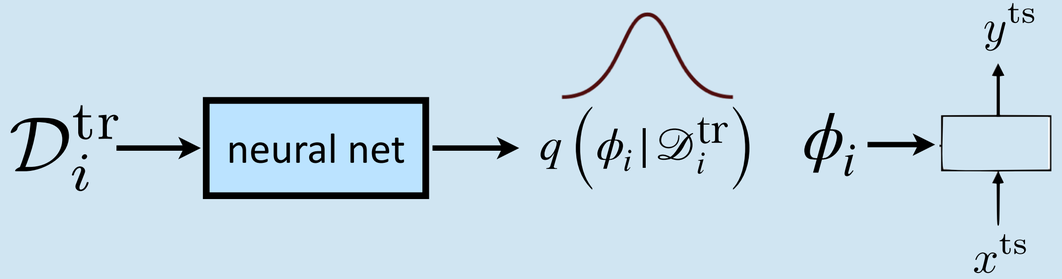
(exact in linear case, approximate in nonlinear case)

Provides a Bayesian interpretation of MAML.

But, we can’t **sample** from $p\left(\phi_i|\theta, \mathcal{D}_i^{\text{tr}}\right)$!

What about Bayesian **optimization-based** meta-learning?

Recall: Bayesian black-box meta-learning
with standard, deep variational inference



$$\max_{\theta} \mathbb{E}_{\mathcal{T}_i}\left[\mathbb{E}_{q\left(\phi_i \mid \mathcal{D}_i^{\text{tr}}, \theta\right)}\left[\log p\left(y_i^{\text{ts}} \mid x_i^{\text{ts}}, \phi_i\right)\right] - D_{KL}\left(q\left(\phi_i \mid \mathcal{D}_i^{\text{tr}}, \theta\right) \| p(\phi_i \mid \theta)\right)\right]$$

q: an arbitrary function

q can include a gradient operator!

Amortized Bayesian Meta-Learning
(Ravi & Beatson '19)

q corresponds to SGD on the mean & variance of neural network weights $(\mu_\phi, \sigma_\phi^2)$, w.r.t. $\mathcal{D}_i^{\text{tr}}$

Pro: Running gradient descent at test time. Con: $p(\phi_i \mid \theta)$ modeled as a Gaussian.

Can we model **non-Gaussian** posterior?

What about Bayesian **optimization-based** meta-learning?

Can we use **ensembles**?

Kim et al. Bayesian MAML ’18

An ensemble of mammals

Ensemble of MAMLs (EMAML)

Train M independent MAML models.

Won’t work well if ensemble members are too similar.

Note: Can also use ensembles w/ black-box, non-parametric methods!

A more diverse ensemble of mammals

Stein Variational Gradient (BMAML)

Use stein variational gradient (SVGD) to push particles away from one another

$$\phi(\theta_t) = \frac{1}{M}\sum_{j=1}^{M}\left[k(\theta_t^j, \theta_t)\nabla_{\theta_t^j}\log p(\theta_t^j) + \nabla_{\theta_t^j} k(\theta_t^j, \theta_t)\right]$$

Optimize for distribution of M particles to produce high likelihood.

$$\mathcal{L}_{\text{BFA}}(\Theta_\tau(\Theta_0); \mathcal{D}_\tau^{\text{val}}) = \log\left[\frac{1}{M}\sum_{m=1}^{M} p(\mathcal{D}_\tau^{\text{val}}|\theta_\tau^m)\right]$$

Pros: Simple, tends to work well, non-Gaussian distributions.

Con: Need to maintain M model instances. (or do gradient-based inference on **last layer only**)

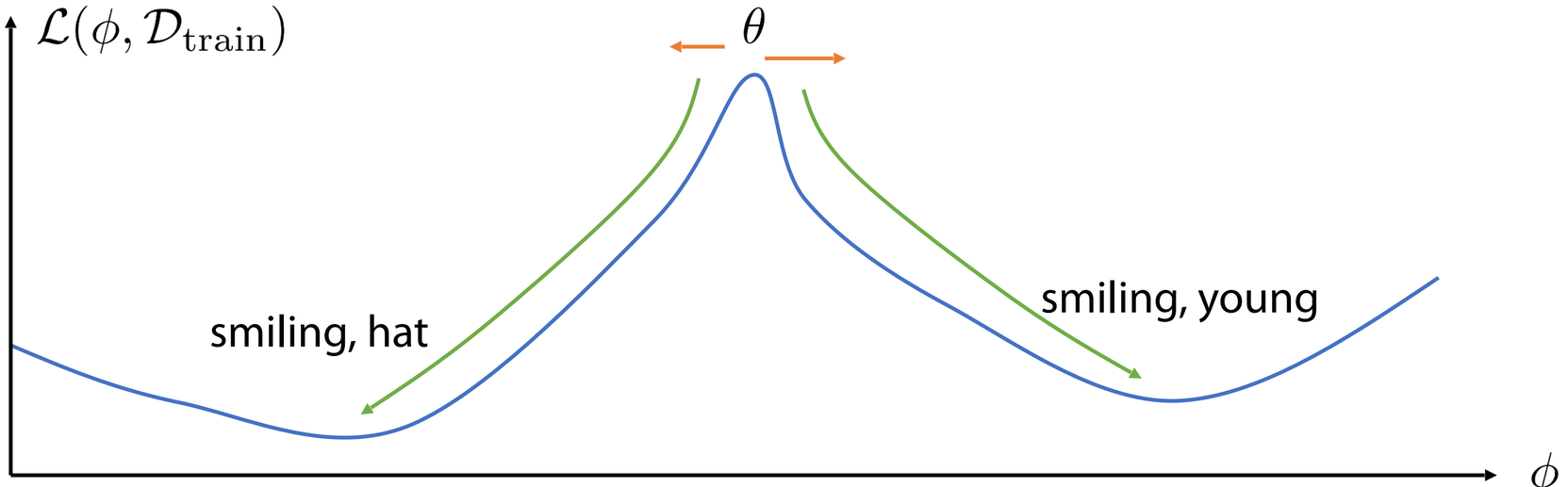
Can we model **non-Gaussian** posterior over **all parameters**?

What about Bayesian **optimization-based** meta-learning?

Sample parameter vectors with a procedure like **Hamiltonian Monte Carlo**?

Finn*, Xu*, Levine. Probabilistic MAML '18

Intuition: Learn a prior where a random kick can put us in different modes



$$\phi \leftarrow \theta + \epsilon$$

$$\phi \leftarrow \phi + \alpha \nabla_{\phi} \mathcal{L}(\phi, \mathcal{D}_{\text{train}})$$

What about Bayesian **optimization-based** meta-learning?

Sample parameter vectors with a procedure like **Hamiltonian Monte Carlo**?

Finn*, Xu*, Levine. Probabilistic MAML '18

$$\theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta) \qquad \phi_i \sim p(\phi_i|\theta)$$

(not single parameter vector anymore)

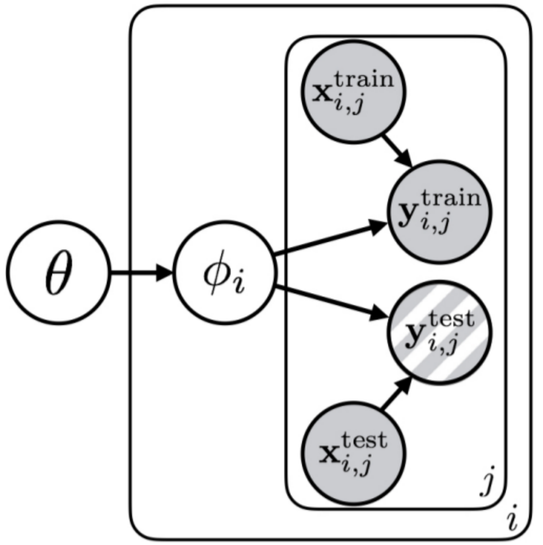
Goal: sample $\phi_i \sim p(\phi_i|x_i^{\text{train}}, y_i^{\text{train}}, x_i^{\text{test}})$

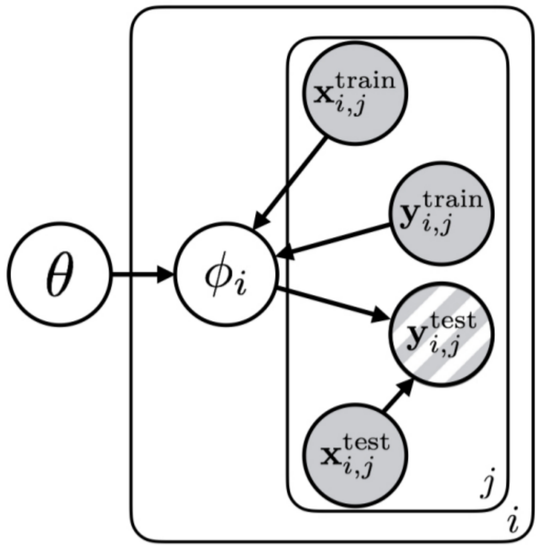
$$p(\phi_i|x_i^{\text{train}}, y_i^{\text{train}}) \propto \int p(\theta)p(\phi_i|\theta)p(y_i^{\text{train}}|x_i^{\text{train}}, \phi_i)d\theta$$

$\Rightarrow$ this is completely intractable!

what if we knew $p(\phi_i|\theta, x_i^{\text{train}}, y_i^{\text{train}})$?

$\Rightarrow$ now sampling is easy! just use ancestral sampling!





key idea: $p(\phi_i|\theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \delta(\hat{\phi}_i)$ ← approximate with MAP

this is **extremely** crude

but **extremely** convenient!

$$\hat{\phi}_i \approx \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}}|x_i^{\text{train}}, \theta)$$

(Santos '92, Grant et al. ICLR '18)

Training can be done with **amortized variational inference**.

What about Bayesian **optimization-based** meta-learning?

Sample parameter vectors with a procedure like **Hamiltonian Monte Carlo**?

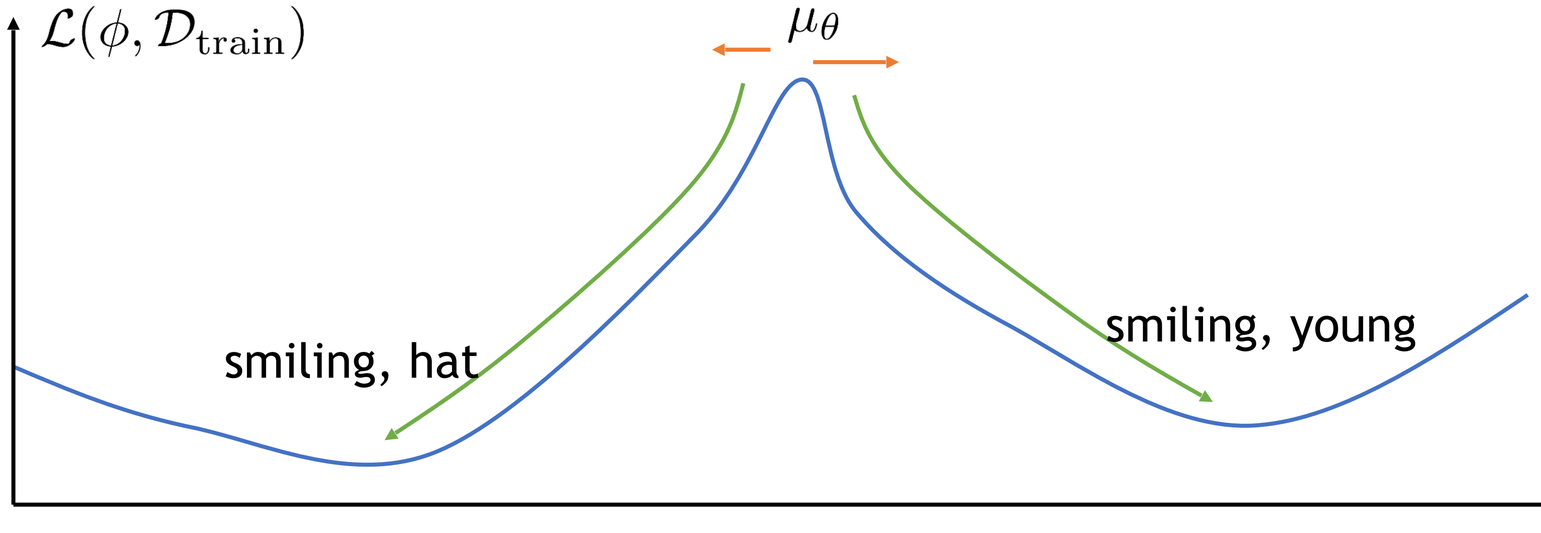
Finn*, Xu*, Levine. Probabilistic MAML '18

$$\theta \sim p(\theta) = \mathcal{N}(\mu_\theta, \Sigma_\theta)$$

key idea: $p(\phi_i|\theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \delta(\hat{\phi}_i) \qquad \hat{\phi}_i \approx \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}}|x_i^{\text{train}}, \theta)$

What does ancestral sampling look like?

1. $\theta \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$
2. $\phi_i \sim p(\phi_i|\theta, x_i^{\text{train}}, y_i^{\text{train}}) \approx \hat{\phi}_i = \theta + \alpha \nabla_\theta \log p(y_i^{\text{train}}|x_i^{\text{train}}, \theta)$



Pros: Non-Gaussian posterior, simple at test time, only one model instance.

Con: More complex training procedure.

Methods Summary

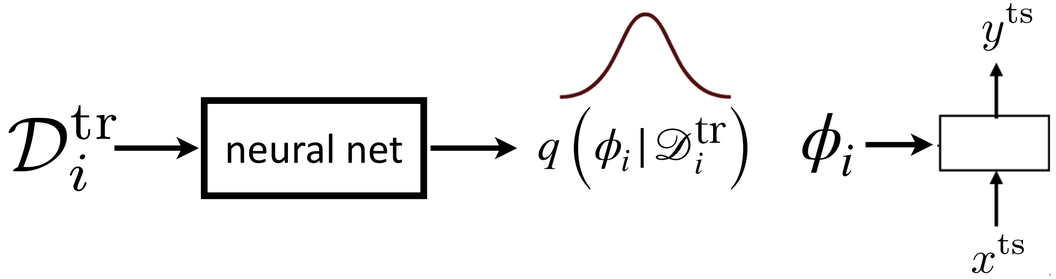
Version 0: f outputs a distribution over y^{ts}.

Pros: simple, can combine with variety of methods

Cons: can't reason about uncertainty over the underlying function,

limited class of distributions over y^{ts} can be expressed

Black box approaches: Use latent variable models + amortized variational inference



Pros: can represent non-Gaussian distributions over y^{ts}

Cons: Can only represent Gaussian distributions $p(\phi_i \mid \theta)$

(okay when ϕ_i is latent vector)

Optimization-based approaches:

Amortized inference

Pro: Simple.

Con: $p(\phi_i \mid \theta)$ modeled as a Gaussian.

Ensembles

Pros: Simple, tends to work well, non-Gaussian distributions.

Con: maintain M model instances. (or do inference on **last layer only**)

Hybrid inference

Pros: Non-Gaussian posterior, simple at test time, only one model instance.

Con: More complex training procedure.

Plan for Today

Why be Bayesian?

Bayesian meta-learning approaches
- black-box approaches
- optimization-based approaches

How to evaluate Bayesian meta-learners.

How to evaluate a Bayesian meta-learner?

Use the standard benchmarks?
(i.e. MiniImagenet accuracy)

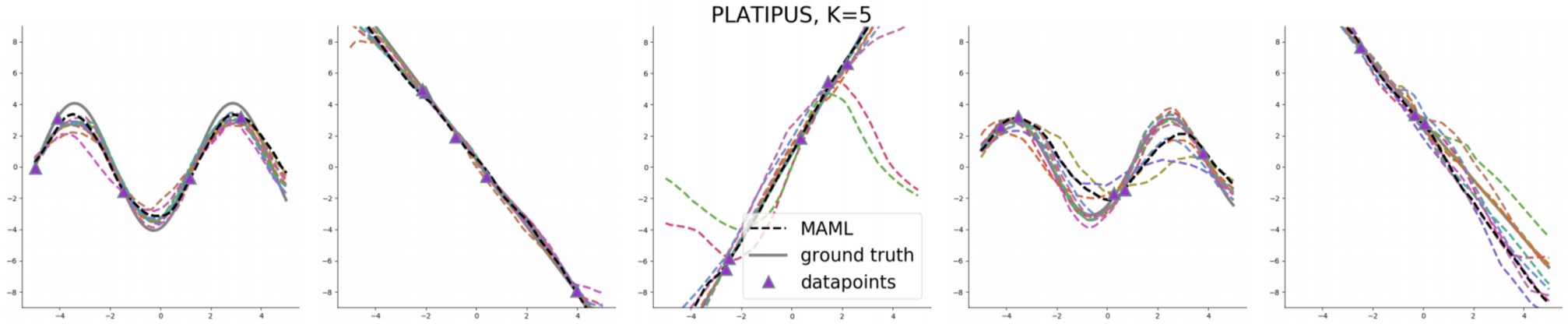
+ standardized
+ real images
+ good check that the approach didn’t break anything
- metrics like accuracy don't evaluate uncertainty
- tasks may not exhibit ambiguity
- uncertainty may not be useful on this dataset!

What are better problems & metrics?
It depends on the problem you care about!

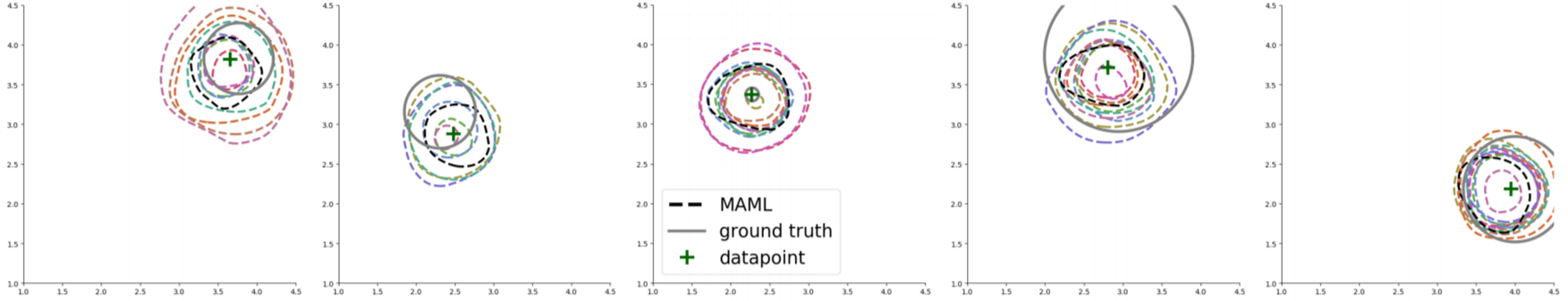
Qualitative Evaluation on Toy Problems with Ambiguity

(Finn*, Xu*, Levine, NeurIPS ’18)

Ambiguous regression:

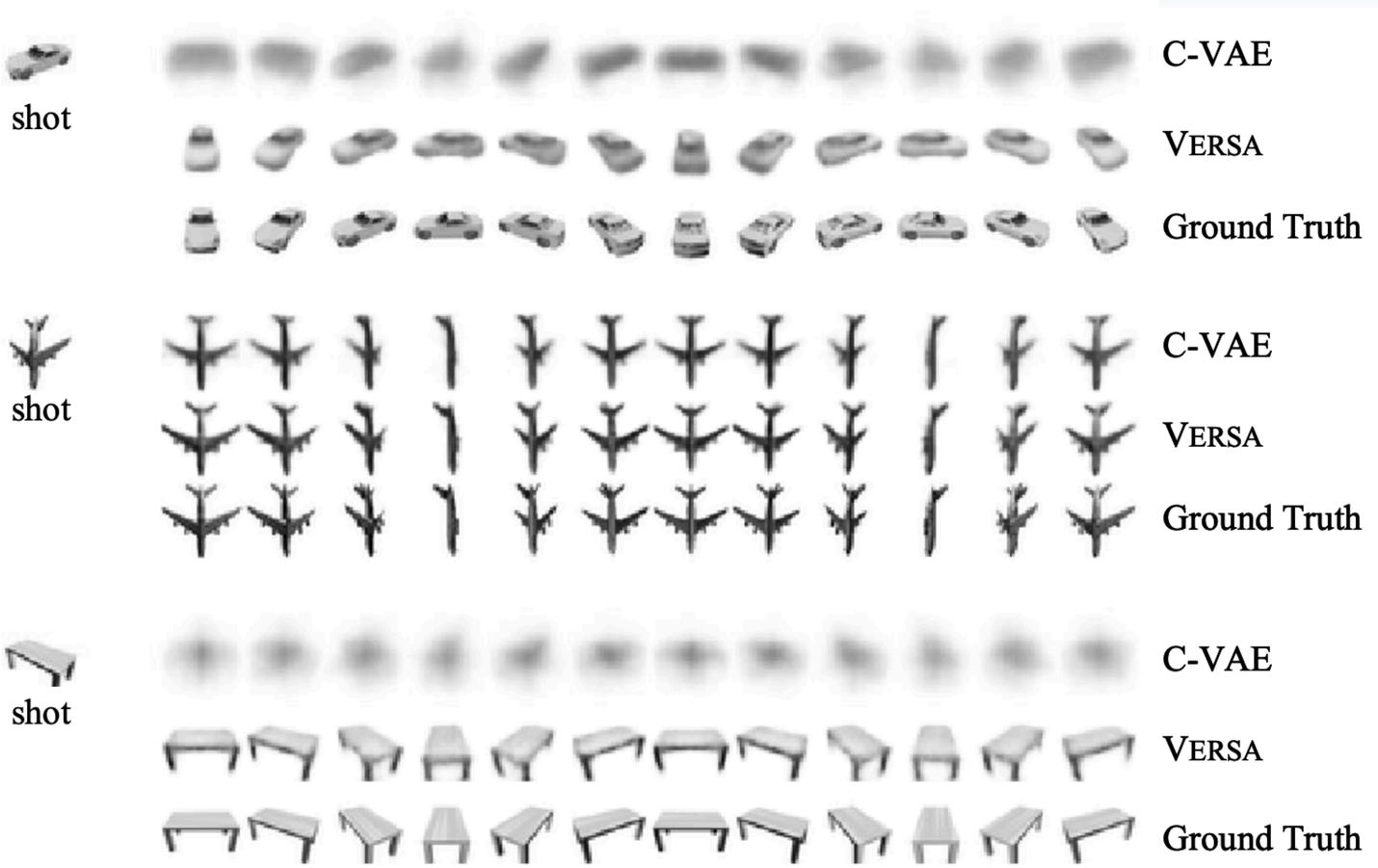


Ambiguous classification:



Evaluation on Ambiguous Generation Tasks

(Gordon et al., ICLR ’19)

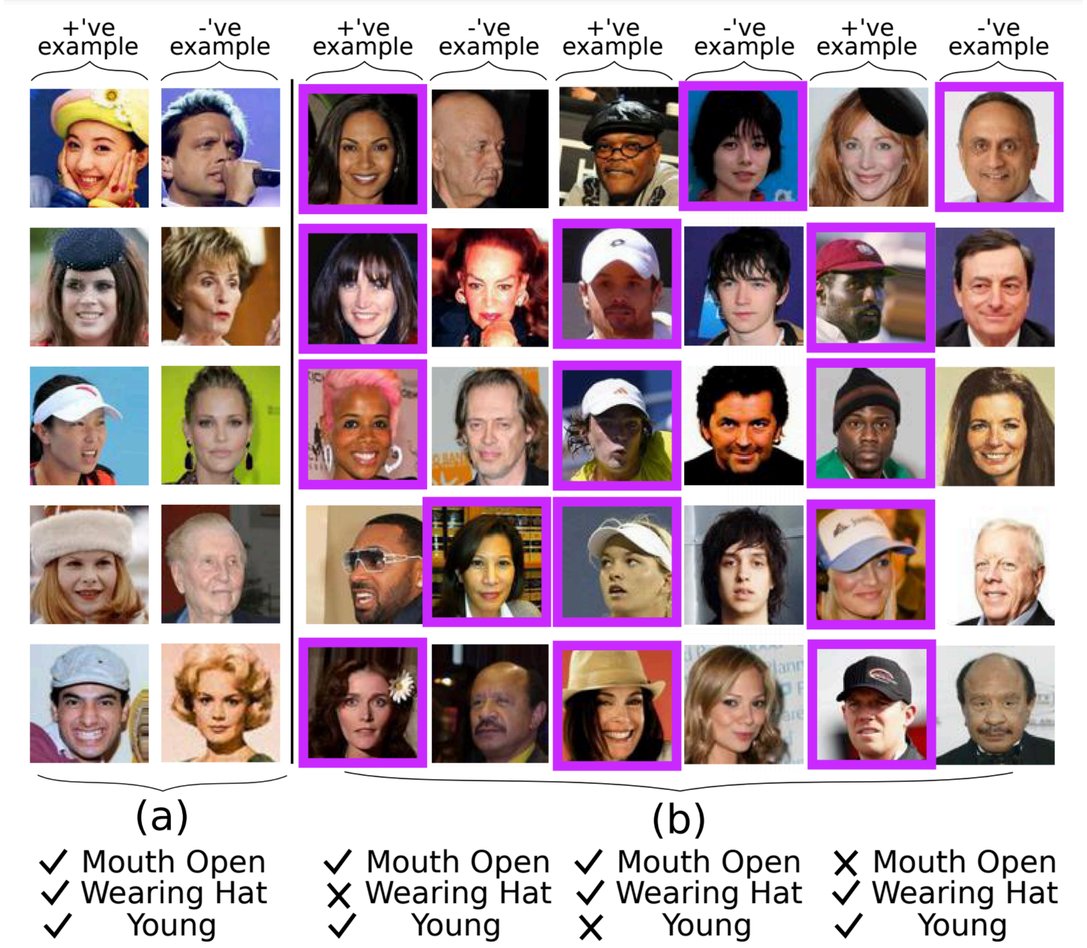


Model	MSE	SSIM
C-VAE 1-shot	0.0269	0.5705
VERSA 1-shot	0.0108	0.7893
VERSA 5-shot	0.0069	0.8483

Table 2: View reconstruction test results.

Accuracy, Mode Coverage, & Likelihood on Ambiguous Tasks

(Finn*, Xu*, Levine, NeurIPS ’18)



Ambiguous celebA (5-shot)			
	Accuracy	Coverage (max=3)	Average NLL
MAML	**89.00 ± 1.78%**	1.00 ± 0.0	0.73 ± 0.06
MAML + noise	84.3 ± 1.60 %	1.89 ± 0.04	0.68 ± 0.05
PLATIPUS (ours) (KL weight = 0.05)	**88.34 ± 1.06 %**	1.59 ± 0.03	0.67± 0.05
PLATIPUS (ours) (KL weight = 0.15)	**87.8 ± 1.03 %**	**1.94 ± 0.04**	**0.56 ± 0.04**

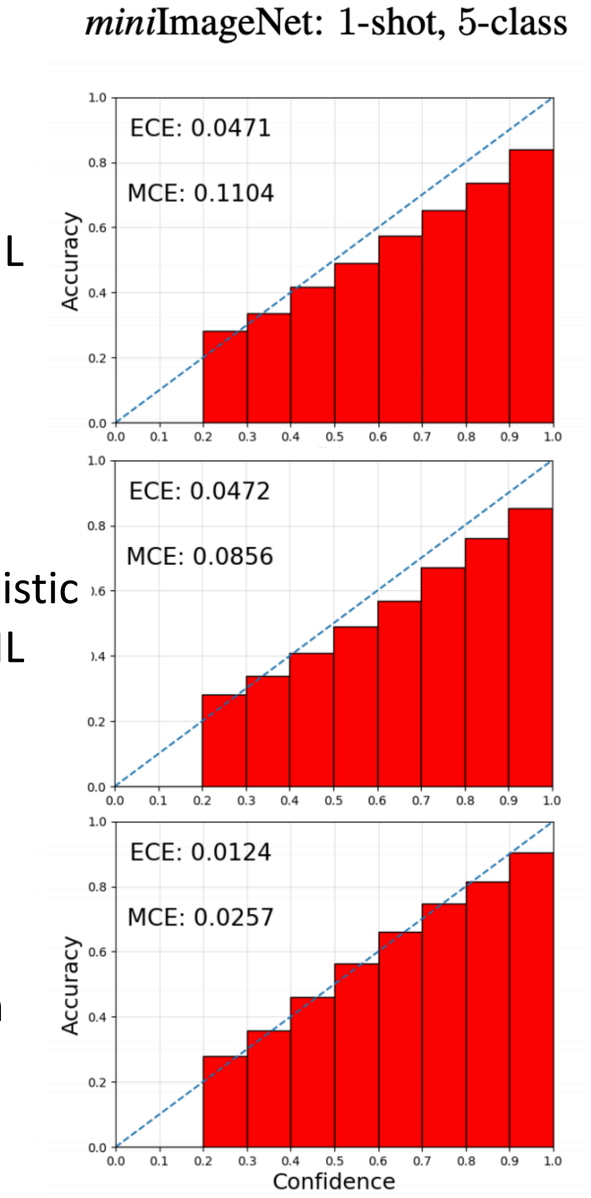
Reliability Diagrams & Accuracy

(Ravi & Beatson, ICLR ’19)

MAML

Probabilistic MAML

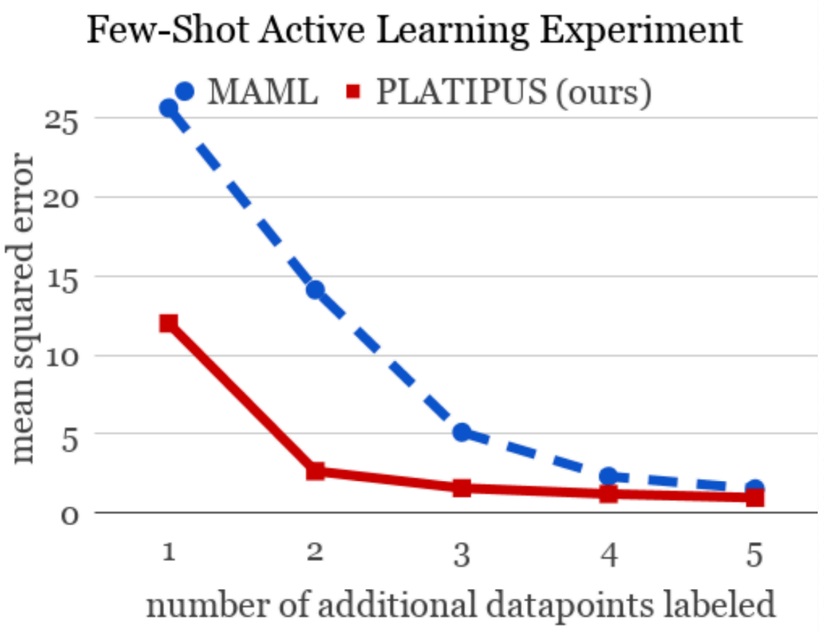
Ravi & Beatson



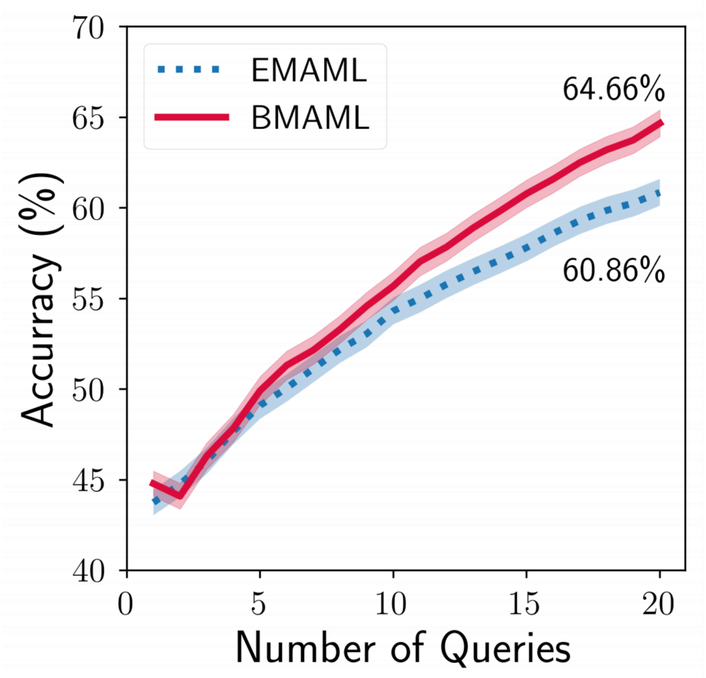
*mini*ImageNet	1-shot, 5-class
MAML (ours)	47.0 ± 0.59
Prob. MAML (ours)	47.8 ± 0.61
Our Model	45.0 ± 0.60

Active Learning Evaluation

Finn*, Xu*, Levine, NeurIPS ’18
Sinusoid Regression



Kim et al. NeurIPS ’18
MiniImageNet



Both experiments:

- Sequentially choose datapoint with **maximum predictive entropy** to be labeled
- Choose datapoint at random for non-Bayesian methods