UNIVERSITY OF CAMBRIDGE

INVENIA LABS

Microsoft

TaskNorm:
Rethinking Batch Normalization for Meta-Learning





John Bronskill
University of Cambridge

Jonathan Gordon
University of Cambridge

James Requeima
University of Cambridge, Invenia Labs

Sebastian Nowozin
Microsoft Research

Richard E. Turner
University of Cambridge, Microsoft Research

Department of Engineering

Paper: [*]Bronskill, J. [*]Gordon, J. Requeima, J., Nowozin, S. and Turner, R.E. “TaskNorm: Rethinking Batch Normalization for Meta-Learning.” Proceedings of the 37th International Conference on Machine Learning, PMLR 108 (2020). [*]*Equal contribution*.

Code: https://github.com/cambridge-mlg/cnaps

TaskNorm: Batch Normalization for Meta-learning with Images

- We demonstrate the significant effect of batch normalization (BN) on meta-learning image classification accuracy and training efficiency.
- We identify issues with transductive BN schemes used in well known meta-learning algorithms.
- We introduce TaskNorm, a normalization algorithm that is tailored for the meta-learning setting and improves both image classification accuracy and training efficiency.

Meta-Learning

Meta-Learning

- ***Early Machine Learning***: Learn <u>classifier</u> based on engineered features

Meta-Learning

- ***Early Machine Learning***: Learn classifier based on engineered features
- ***Deep learning***: Jointly learn features and classifier

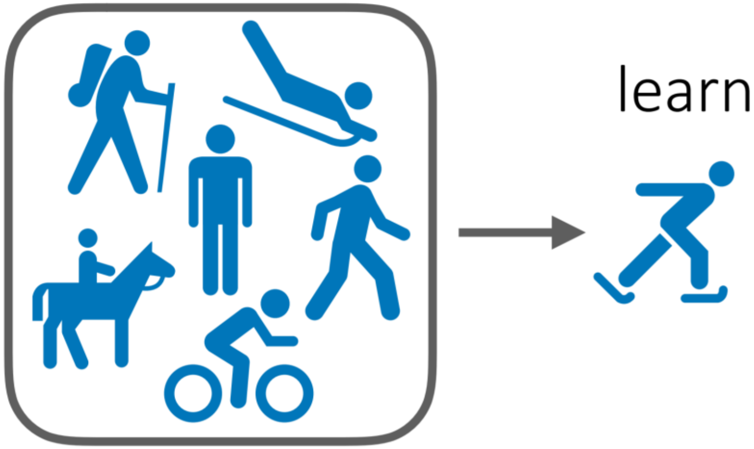
Meta-Learning

- ***Early Machine Learning***: Learn classifier based on engineered features
- ***Deep learning***: Jointly learn classifier and model
- ***Meta-Learning***: Jointly learn features, classifier, and algorithm[1]

[1] Hospedales, Timothy, et al. "Meta-learning in neural networks: A survey." arXiv preprint arXiv:2004.05439 (2020).

Meta-Learning

- ***Early Machine Learning***: Learn <u>model</u> based on engineered features
- ***Deep learning***: Jointly learn <u>features</u> and <u>model</u>
- ***Meta-Learning***: Jointly learn <u>features</u>, <u>model</u>, and <u>algorithm</u>[1]

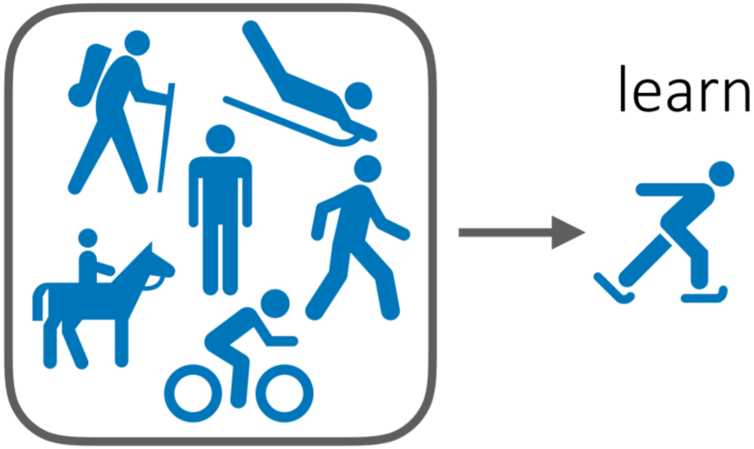


Given a task distribution, learn a new task efficiently.[2]

[1] Hospedales, Timothy, et al. "Meta-learning in neural networks: A survey." arXiv preprint arXiv:2004.05439 (2020).
[2] Sergey Levine & Chelsea Finn - Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning: https://metalearning-cvpr2019.github.io/assets/CVPR_2019_Metalearning_Tutorial_Chelsea_Finn.pdf

Meta-Learning

- ***Early Machine Learning***: Learn model based on engineered features
- ***Deep learning***: Jointly learn features and model
- ***Meta-Learning***: Jointly learn features, model, and algorithm[1]



Given a task distribution, learn a new task efficiently.[2]

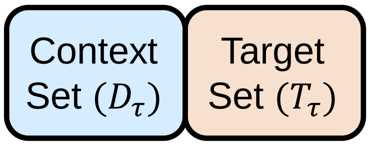
- Focus on utilizing meta-learning in the few-shot classification scenario

[1] Hospedales, Timothy, et al. "Meta-learning in neural networks: A survey." arXiv preprint arXiv:2004.05439 (2020).
[2] Sergey Levine & Chelsea Finn - Meta-Learning: from Few-Shot Learning to Rapid Reinforcement Learning: https://metalearning-cvpr2019.github.io/assets/CVPR_2019_Metalearning_Tutorial_Chelsea_Finn.pdf

Few-Shot Meta-Training / Meta-Testing

Few-Shot Meta-Training / Meta-Testing

Task τ

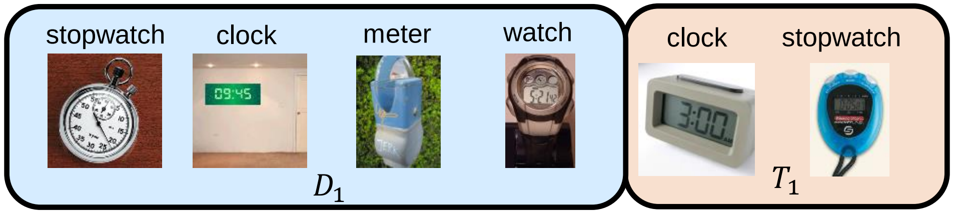


Few-Shot Meta-Training / Meta-Testing

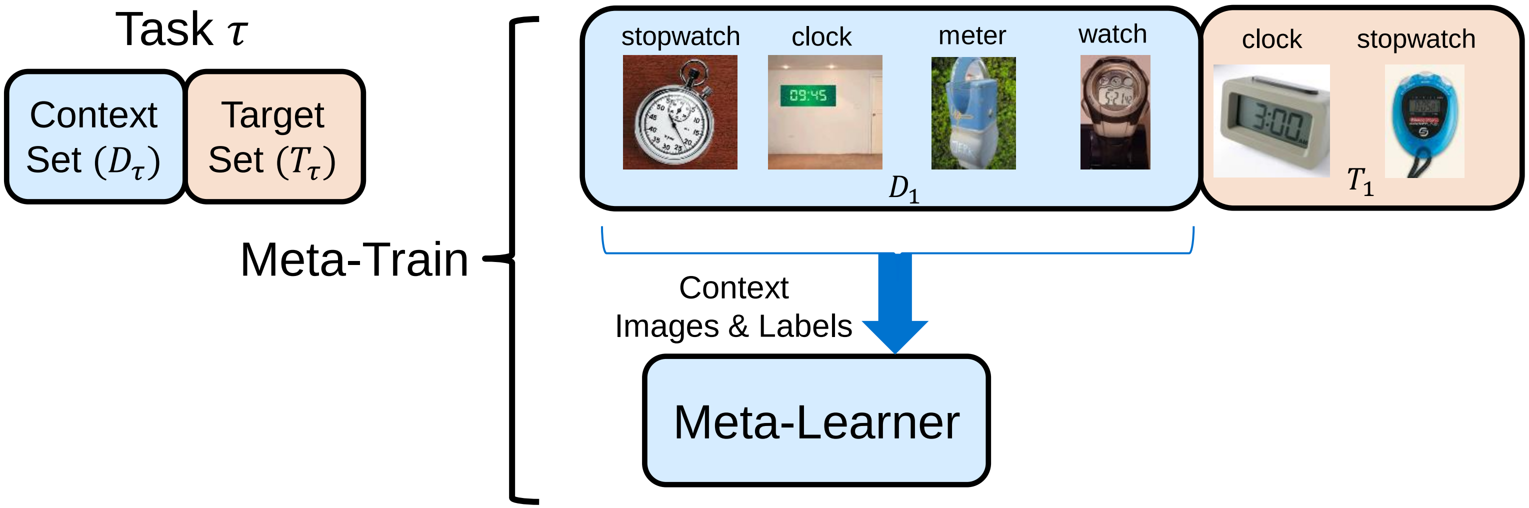
Task τ

Context Set (D_τ)	Target Set (T_τ)

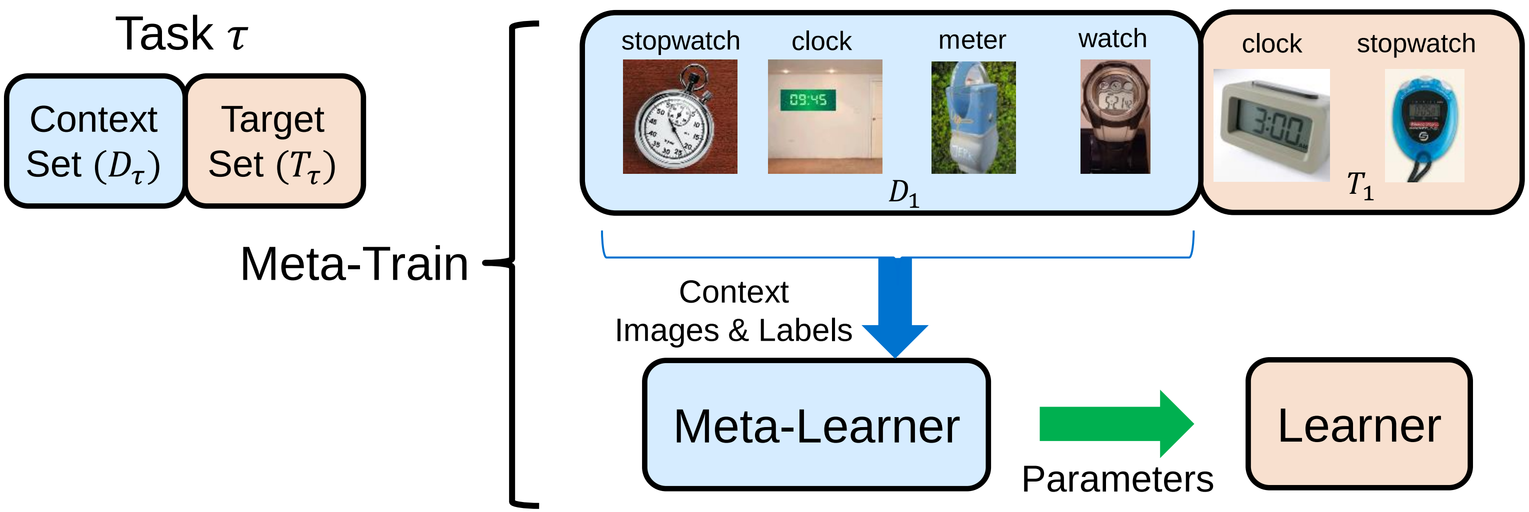
Meta-Train



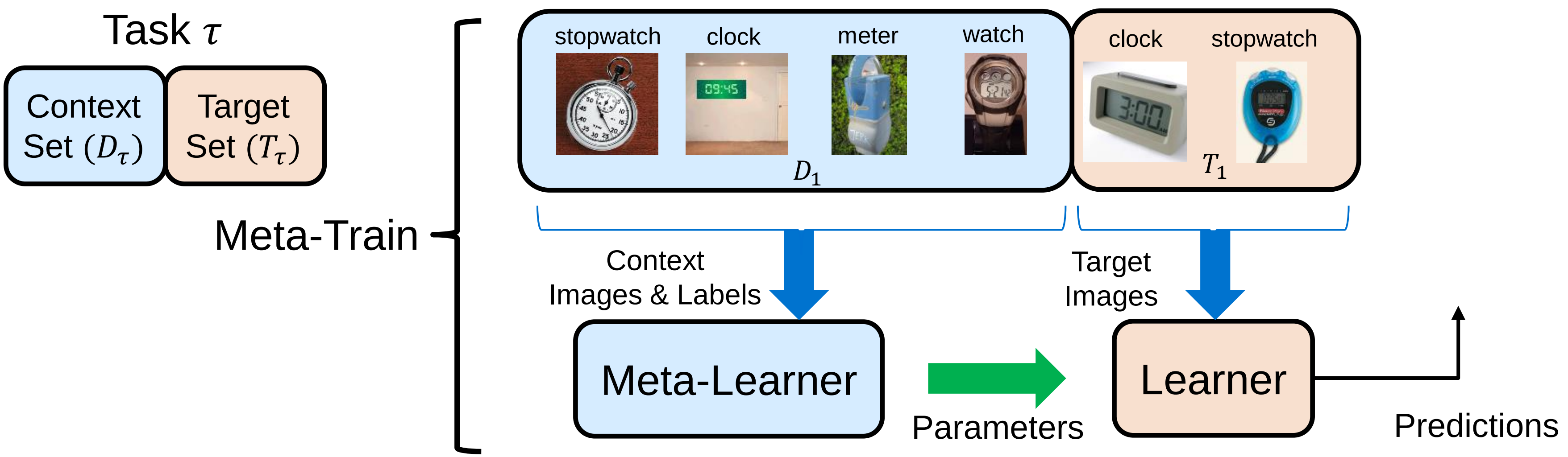
Few-Shot Meta-Training / Meta-Testing



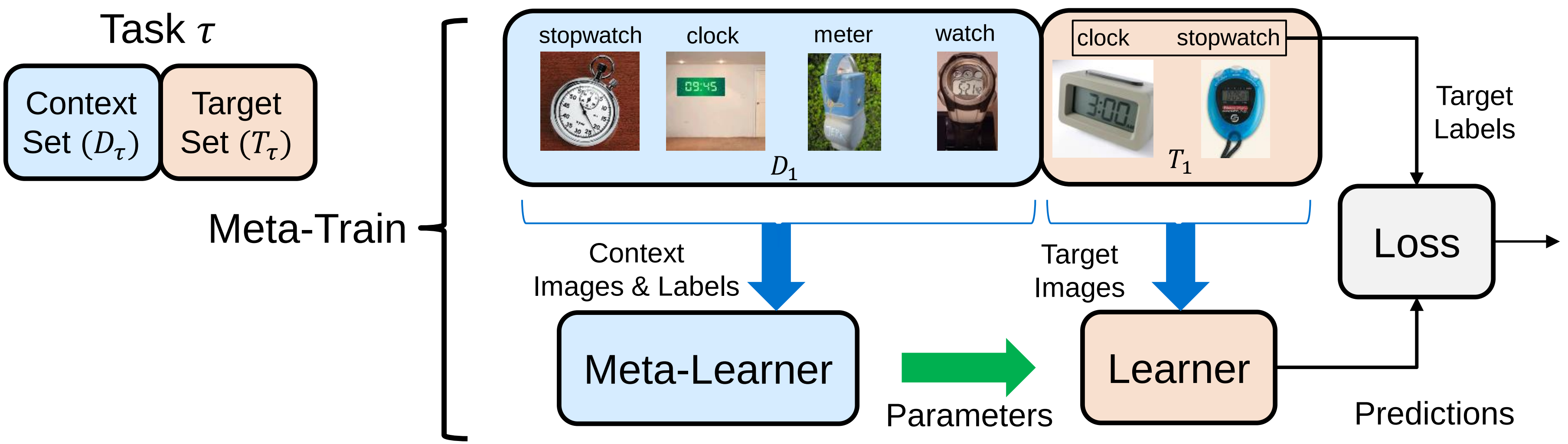
Few-Shot Meta-Training / Meta-Testing



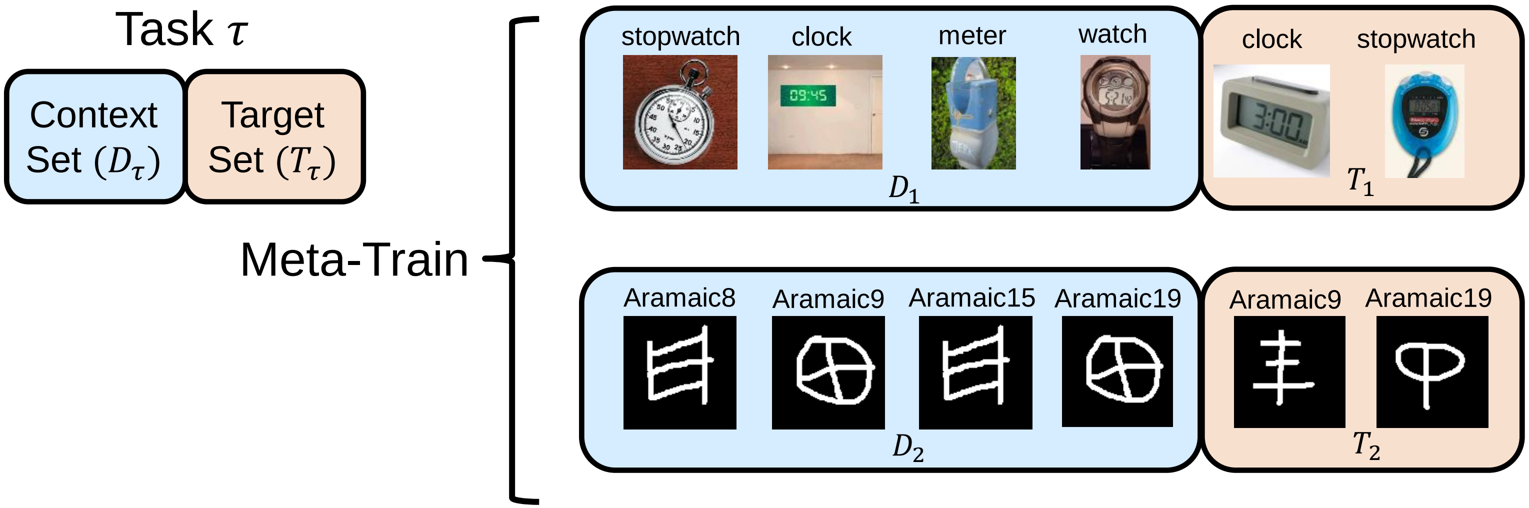
Few-Shot Meta-Training / Meta-Testing



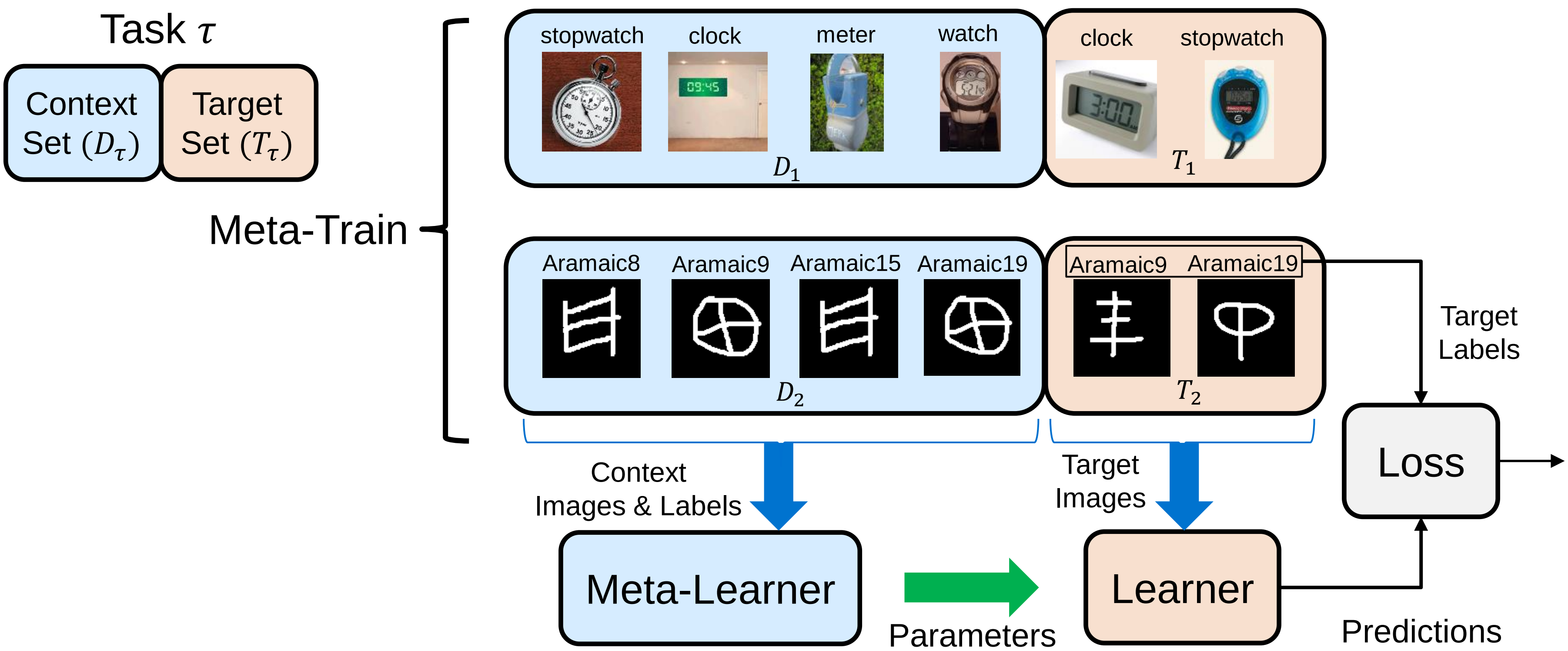
Few-Shot Meta-Training / Meta-Testing



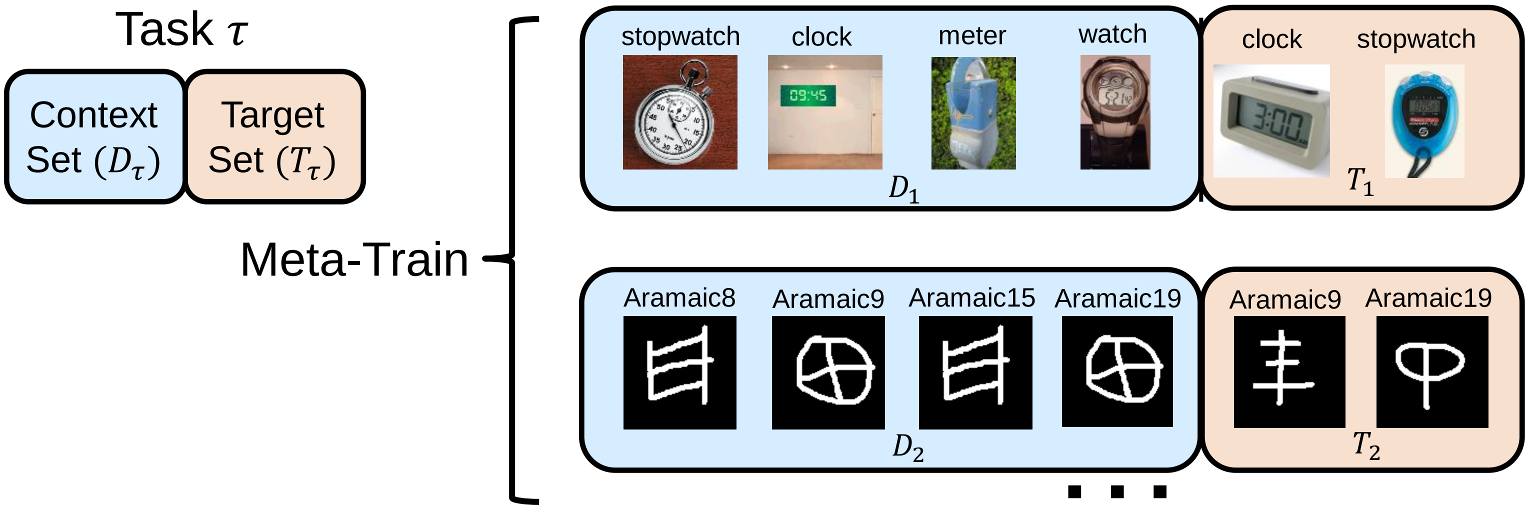
Meta-Training / Meta-Testing



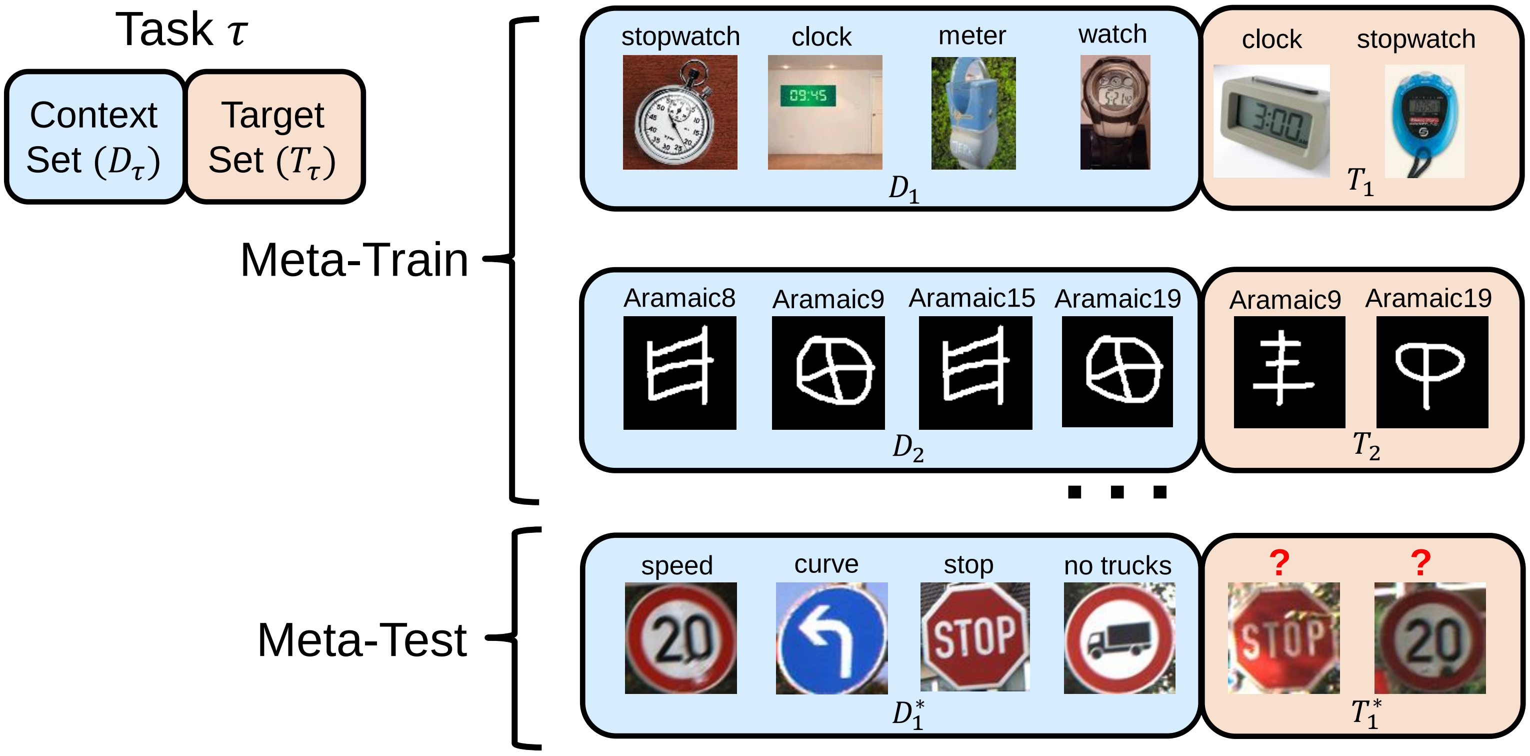
Meta-Training / Meta-Testing



Meta-Training / Meta-Testing

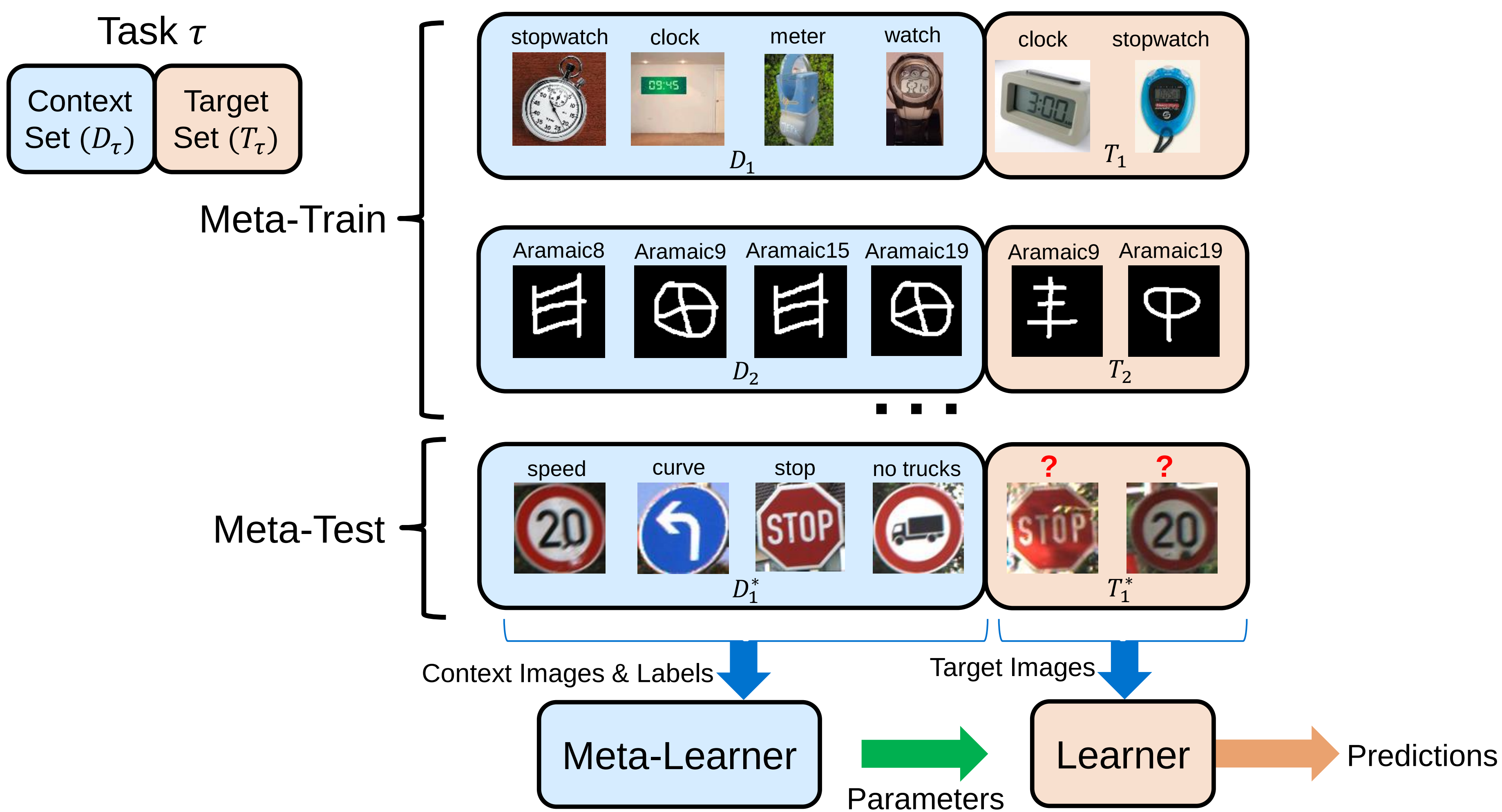


Meta-Training / Meta-Testing



Hugo Larochelle – Generalizing From Few Examples With Meta-Learning:
https://www.dropbox.com/s/sm68skkkbxbob0i/metalearning.pdf?dl=0

Meta-Training / Meta-Testing



Batch Normalization

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

Batch Normalization

- *Goal*: Normalize each training batch so that it has:
  - zero mean
  - unit variance

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

Batch Normalization

- *Goal*: Normalize each training batch so that it has:
  - zero mean
  - unit variance
- *Accelerates* Neural Network training by:
  - Allowing the use of higher learning rates.
  - Decreasing the sensitivity to network initialization.

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

“Conventional” Batch Normalization Algorithm

Training:

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

“Conventional” Batch Normalization Algorithm

Training:

0 $B = \{x_1, x_2, \dots, x_m\}$ # a mini-batch

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

"Conventional" Batch Normalization Algorithm

Training:

0 $B = \{x_1, x_2, \ldots, x_m\}$ # a mini-batch

1 $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ # compute batch mean

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

"Conventional" Batch Normalization Algorithm

Training:

⓪ $B = \{x_1, x_2, \ldots, x_m\}$ # a mini-batch

① $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ # compute batch mean

② $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ # compute batch variance

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

“Conventional” Batch Normalization Algorithm

Training:

⓪ $B = \{x_1, x_2, \ldots, x_m\}$ # a mini-batch

① $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ # compute batch mean

② $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ # compute batch variance

③ $x_i' = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$

normalize
γ, β are learned
ϵ is a small constant
to avoid division by 0

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

"Conventional" Batch Normalization Algorithm

Training:

⓪ $B = \{x_1, x_2, \ldots, x_m\}$ # a mini-batch

① $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ # compute batch mean

② $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ # compute batch variance

③ $x_i' = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$

normalize
γ, β are learned
ϵ is a small constant
to avoid division by 0

④ Accumulate moving averages of μ_B, σ_B^2 over all batches as μ_r, σ_r^2

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

“Conventional” Batch Normalization Algorithm

Training:

⓪ $B = \{x_1, x_2, \ldots, x_m\}$ # a mini-batch

① $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ # compute batch mean

② $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2$ # compute batch variance

③ $x_i' = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$ # normalize
γ, β are learned
ϵ is a small constant
to avoid division by 0

Inference:

use moving averages to normalize

$$x_i' = \gamma \frac{x_i - \mu_r}{\sqrt{\sigma_r^2 + \epsilon}} + \beta$$

④ Accumulate moving averages of μ_B, σ_B^2 over all batches as μ_r, σ_r^2

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

"Conventional" Batch Normalization Algorithm

Training:

⓪ $B = \{x_1, x_2, \ldots, x_m\}$ # a mini-batch

① $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$ # compute batch mean

② $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ # compute batch variance

③ $x_i' = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$ # normalize
γ, β are learned
ϵ is a small constant
to avoid division by 0

Inference:

use moving averages to normalize

$$x_i' = \gamma \frac{x_i - \mu_r}{\sqrt{\sigma_r^2 + \epsilon}} + \beta$$

We call the mean and variance of a batch its *moments*.

④ Accumulate moving averages of μ_B, σ_B^2 over all batches as μ_r, σ_r^2

Ioffe, Sergey, and Christian Szegedy. "Batch normalization: Accelerating deep network training by Reducing internal covariate shift." *arXiv preprint arXiv:1502.03167* (2015).

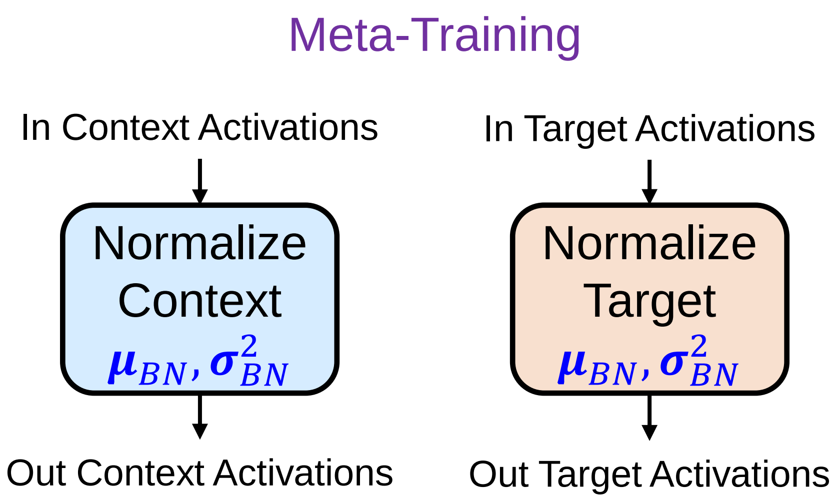
How should batch normalization for meta-learning work?

How should batch normalization for meta-learning work?

- First idea: Use ***conventional batch normalization*** (CBN):

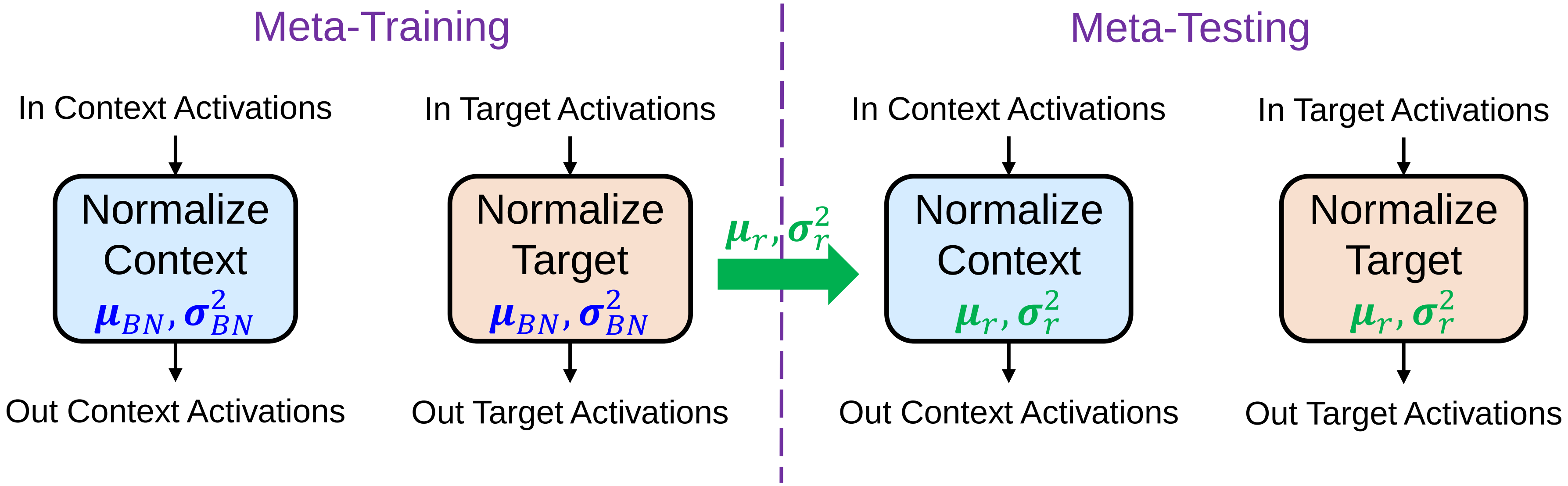
How should batch normalization for meta-learning work?

- First idea: Use ***conventional batch normalization*** (CBN):
  - Meta-Training: Normalize with computed moments ($\boldsymbol{\mu}_{BN}, \boldsymbol{\sigma}^2_{BN}$).



How should batch normalization for meta-learning work?

- First idea: Use ***conventional batch normalization*** (CBN):
  - Meta-Training: Normalize with computed moments ($\boldsymbol{\mu}_{BN}, \boldsymbol{\sigma}^2_{BN}$).
  - Meta-Testing: Normalize with running averages of moments (μ_r, σ_r^2) that were computed during meta-training.



Classification Accuracy (%) of Model Agnostic Meta-Learning (MAML) on Omniglot and *mini*ImagNet datasets

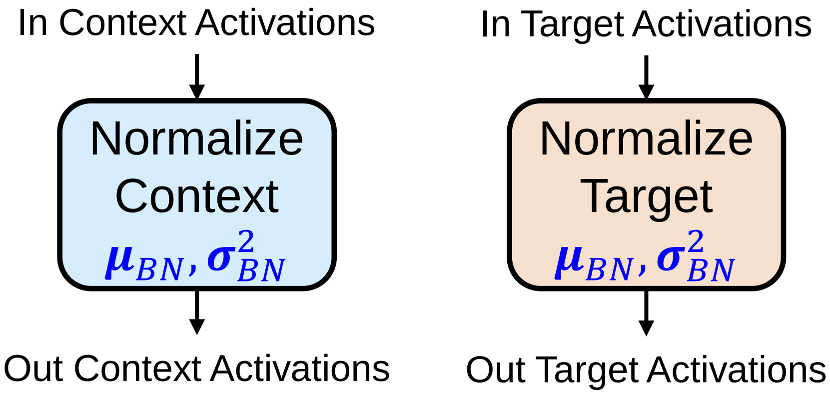
Configuration	CBN
Omniglot 5-way, 1-shot	20.1±0.0
Omniglot 5-way, 5-shot	20.0±0.0
Omniglot 20-way, 1-shot	5.0±0.0
Omniglot 20-way, 5-shot	5.0±0.0
*mini*ImageNet 5-way, 1-shot	20.1±0.0
*mini*ImageNet 5-way, 5-shot	20.2±0.0

These results are terrible. The classification accuracy is no better than chance.

MAML uses Transductive Batch Normalization (TBN)

MAML uses Transductive Batch Normalization (TBN)

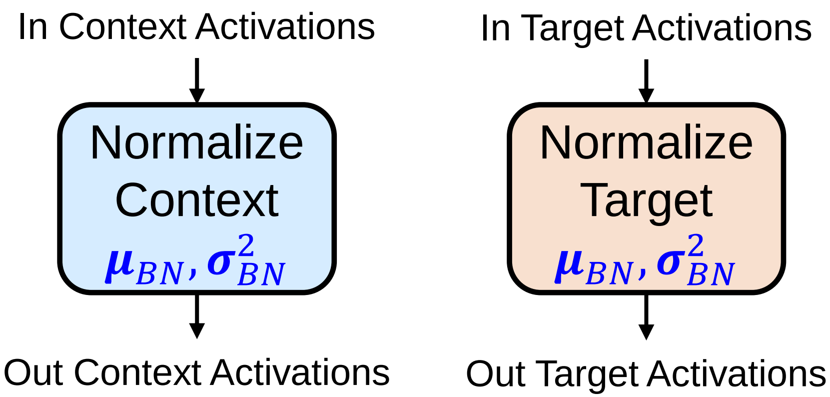
Meta-Training and Meta-Testing



- TBN ignores the running moments ($\boldsymbol{\mu}_r, \boldsymbol{\sigma}^2_r$).
- Uses computed moments ($\boldsymbol{\mu}_{BN}, \boldsymbol{\sigma}^2_{BN}$) to normalize during *both* meta-training and meta-testing.

MAML uses Transductive Batch Normalization (TBN)

Meta-Training and Meta-Testing



- TBN ignores the running moments ($\boldsymbol{\mu}_r, \boldsymbol{\sigma}^2_r$).
- Uses computed moments ($\boldsymbol{\mu}_{BN}, \boldsymbol{\sigma}^2_{BN}$) to normalize during *both* meta-training and meta-testing.

Configuration	CBN	TBN
Omniglot 5-way, 1-shot	20.1±0.0	98.4±0.7
Omniglot 5-way, 5-shot	20.0±0.0	99.2±0.2
Omniglot 20-way, 1-shot	5.0±0.0	90.9±0.5
Omniglot 20-way, 5-shot	5.0±0.0	96.6±0.2
*mini*ImageNet 5-way, 1-shot	20.1±0.0	45.5±1.8
*mini*ImageNet 5-way, 5-shot	20.2±0.0	59.7±0.9

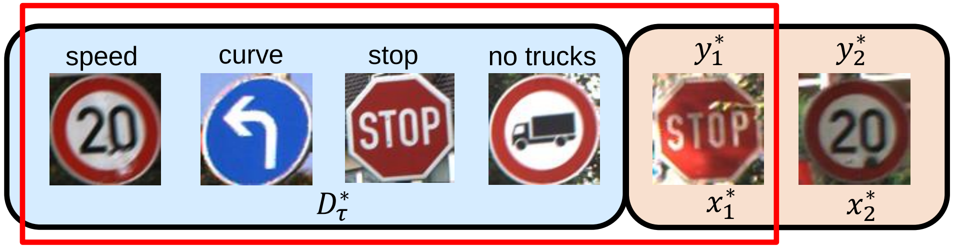
The TBN accuracies are what we would expect for MAML.

Transductive vs Non-Transductive

Transductive vs Non-Transductive

Non-Transductive

$p(y_1^* | x_1^*, D_1^*)$

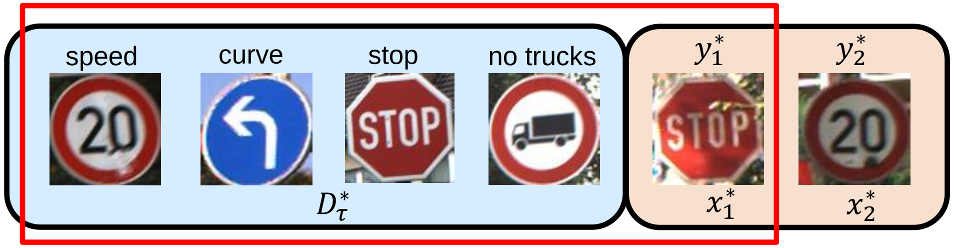


At meta-test time, the prediction for a label y_i^* for an input x_i^* is conditioned **only on** $\boldsymbol{x_i^*}$ and the context set D_τ^*.

Transductive vs Non-Transductive

Non-Transductive

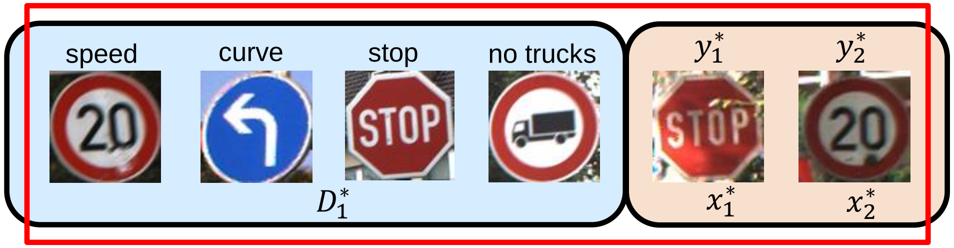
$p(y_1^* | x_1^*, D_1^*)$



At meta-test time, the prediction for a label y_i^* for an input x_i^* is conditioned **only on $\boldsymbol{x_i^*}$** and the context set D_τ^*.

Transductive

$p(y_1^* | x_1^*, x_2^*, D_1^*)$



At meta-test time, the prediction for a label y_i^* for an input x_i^* is conditioned on **all $\boldsymbol{x}^*$** in the target set and the context set D_τ^*.

Transductive Batch Normalization Issues

Note: Under normal circumstances, at meta-test time, we have no control over the makeup of the target set in terms of the relative proportions of the true labels as these are unknown. There are two key issues with TBN:

Transductive Batch Normalization Issues

Note: Under normal circumstances, at meta-test time, we have no control over the makeup of the target set in terms of the relative proportions of the true labels as these are unknown. There are two key issues with TBN:

1. Transductive learning is sensitive to the distribution of the target set learned during meta-training and will fail if required to make good predictions:

    - One example at a time (e.g. online learning).
    - When the target set contains a class balance different from meta-training.
    - Respecting some privacy constraints.

Transductive Batch Normalization Issues

Note: Under normal circumstances, at meta-test time, we have no control over the makeup of the target set in terms of the relative proportions of the true labels as these are unknown. There are two key issues with TBN:

1. Transductive learning is sensitive to the distribution of the target set learned during meta-training and will fail if required to make good predictions:
   - One example at a time (e.g. online learning).
   - When the target set contains a class balance different from meta-training.
   - Respecting some privacy constraints.
2. Transductive learners have more information available to them at prediction time, which may lead to unfair comparisons.

Transductive Batch Normalization Issues (con't)

Configuration	CBN	TBN	TBN (1 example at a time)	TBN (1 class at a time)
Omniglot 5-way, 1-shot	20.1±0.0	98.4±0.7	21.6±1.3	21.6±1.3
Omniglot 5-way, 5-shot	20.0±0.0	99.2±0.2	22.0±0.5	23.2±0.5
Omniglot 20-way, 1-shot	5.0±0.0	90.9±0.5	3.7±0.2	3.7±0.2
Omniglot 20-way, 5-shot	5.0±0.0	96.6±0.2	5.5±0.2	14.5±0.3
*mini*ImageNet 5-way, 1-shot	20.1±0.0	45.5±1.8	26.9±1.5	26.9±1.5
*mini*ImageNet 5-way, 5-shot	20.2±0.0	59.7±0.9	30.3±0.7	27.2±0.6

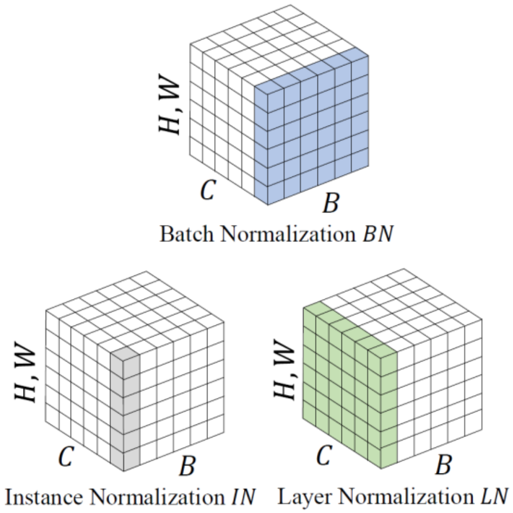
- TBN accuracy degrades significantly when predictions are made one example at a time (streaming) or one class at a time (class imbalance).

Need to Rethink Normalization for Meta-Learning

- For MAML, CBN doesn't work and TBN has potentially unwanted side effects

Need to Rethink Normalization for Meta-Learning

- For MAML, CBN doesn't work and TBN has potentially unwanted side effects.
- There are other non-transductive learners including Instance Normalization[1] (IN), Layer Normalization[2] (LN), and Group Normalization[3], but they don't work well in the few-shot classification setting.



Configuration	CBN	TBN	LN	IN
Omniglot 5-way, 1-shot	20.1±0.0	98.4±0.7	83.0±1.3	87.4±1.2
Omniglot 5-way, 5-shot	20.0±0.0	99.2±0.2	91.0±0.8	93.9±0.5
Omniglot 20-way, 1-shot	5.0±0.0	90.9±0.5	78.1±0.7	80.4±0.7
Omniglot 20-way, 5-shot	5.0±0.0	96.6±0.2	92.3±0.2	92.9±0.2
*mini*ImageNet 5-way, 1-shot	20.1±0.0	45.5±1.8	41.2±1.6	40.7±1.7
*mini*ImageNet 5-way, 5-shot	20.2±0.0	59.7±0.9	52.8±0.9	54.3±0.9

[1] Ulyanov et al. "Instance normalization: The missing ingredient for fast stylization." arXiv:1607.08022 (2016).
[2] Ba et al. "Layer normalization." arXiv preprint arXiv:1607.06450 (2016).
[3] Wu et al. "Group normalization." Proceedings of the European Conference on Computer Vision (ECCV). 2018.

Desiderata for Meta-Learning Normalization

Desiderata for Meta-Learning Normalization

1. Improves speed and stability of training without harming test performance (accuracy or log-likelihood).

Desiderata for Meta-Learning Normalization

1. Improves speed and stability of training without harming test performance (accuracy or log-likelihood).

2. Works well across a range of context set sizes.

Desiderata for Meta-Learning Normalization

1. Improves speed and stability of training without harming test performance (accuracy or log-likelihood).

2. Works well across a range of context set sizes.

3. Is non-transductive, thus supporting inference at meta-test time in a variety of circumstances.

A Few Principles

A Few Principles

- Data is i.i.d. only within a task τ, but not across tasks.
  - Hence, normalization statistics $\boldsymbol{\mu}, \boldsymbol{\sigma}$ should be local at the task level.

A Few Principles

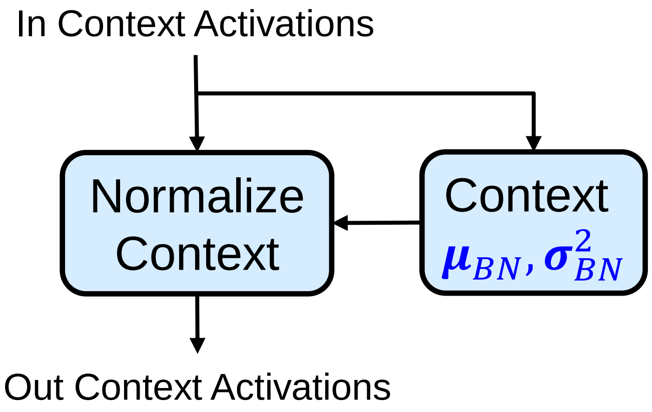
- Data is i.i.d. only within a task τ, but not across tasks.
  - Hence, normalization statistics $\boldsymbol{\mu}, \boldsymbol{\sigma}$ should be local at the task level.
- To avoid being transductive, the target set T^{τ} normalization should only have access to:

  1. The context set D^{τ}

  2. The single example being predicted $\boldsymbol{x}_i^{\tau *}$

MetaBN

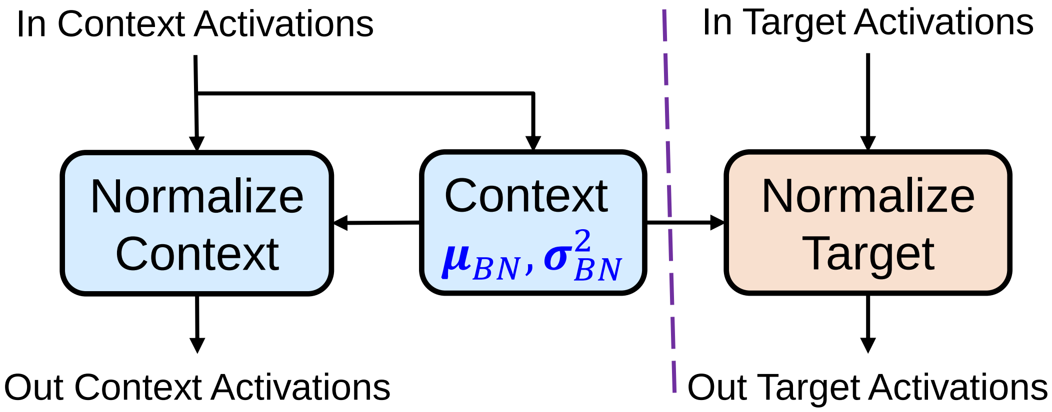
- Simple idea inspired by the previous principles:
  - Use the batch statistics from the context set to normalize both the context set and the target set.



MetaBN

➢ Simple idea inspired by the previous principles:

- Use the batch statistics from the context set to normalize both the context set <u>and</u> the target set.

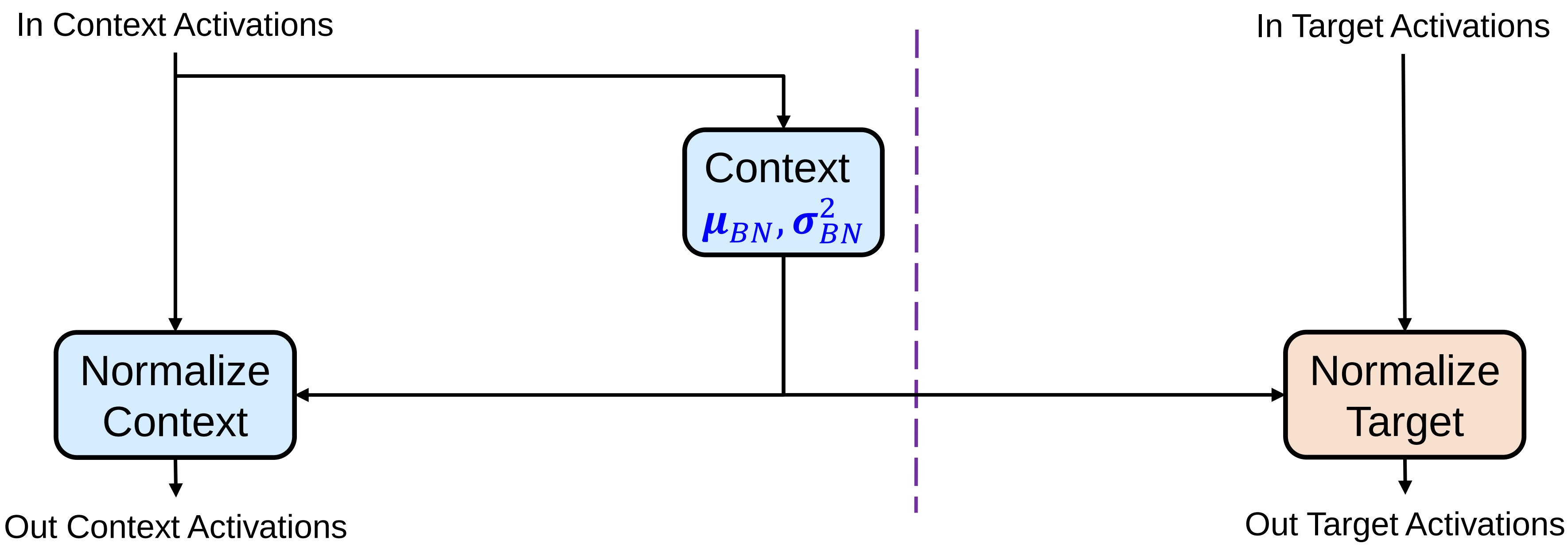


MetaBN

- MetaBN works well, but:
  - Classification accuracy suffers when the context set is small (poor estimate of true statistics)
  - Doesn't leverage information from the target example under test.

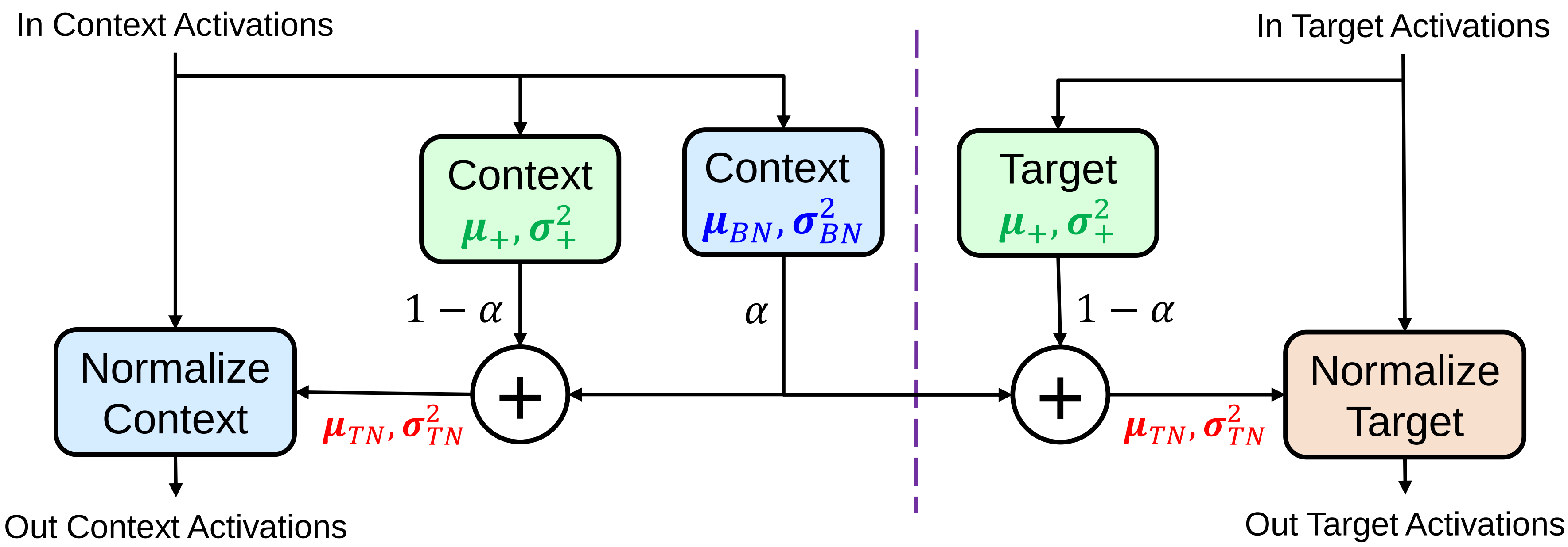
TaskNorm

TaskNorm



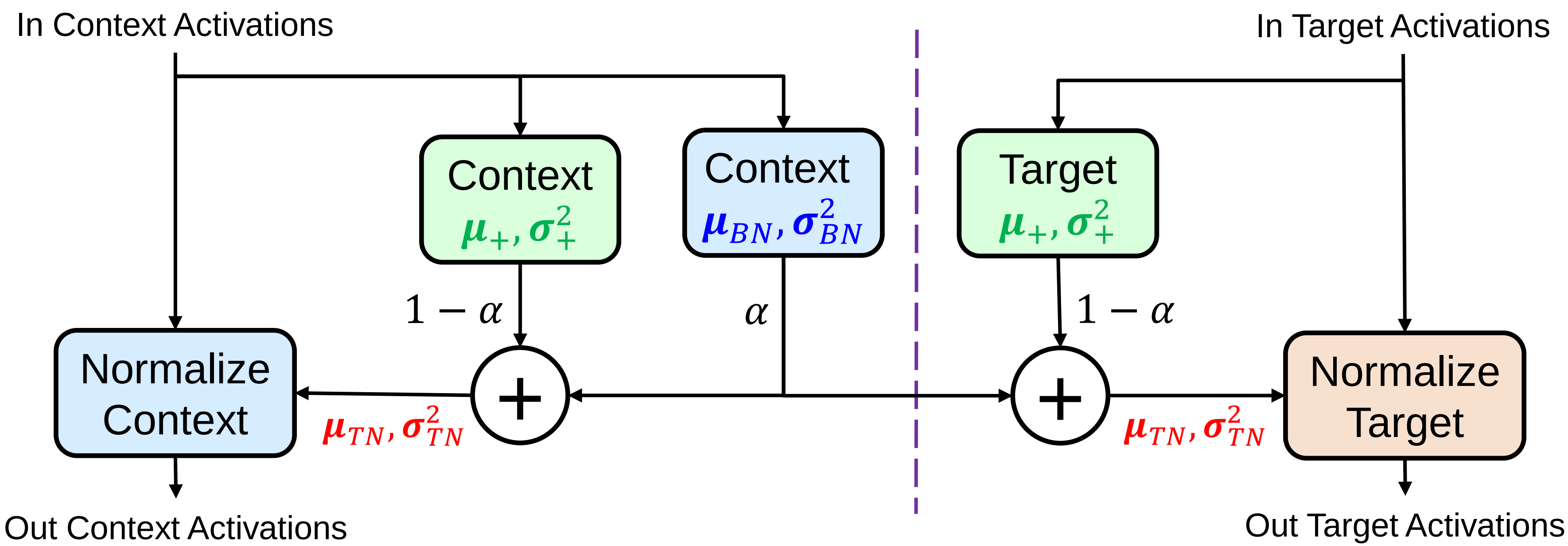
In Context Activations
In Target Activations
Context
$\boldsymbol{\mu}_{BN}, \boldsymbol{\sigma}^2_{BN}$
Normalize Context
Normalize Target
Out Context Activations
Out Target Activations

TaskNorm



In Context Activations
In Target Activations
Context
μ_+, σ_+^2
Context
μ_{BN}, σ_{BN}^2
Target
μ_+, σ_+^2
$1-\alpha$
α
$1-\alpha$
Normalize Context
μ_{TN}, σ_{TN}^2
μ_{TN}, σ_{TN}^2
Normalize Target
Out Context Activations
Out Target Activations

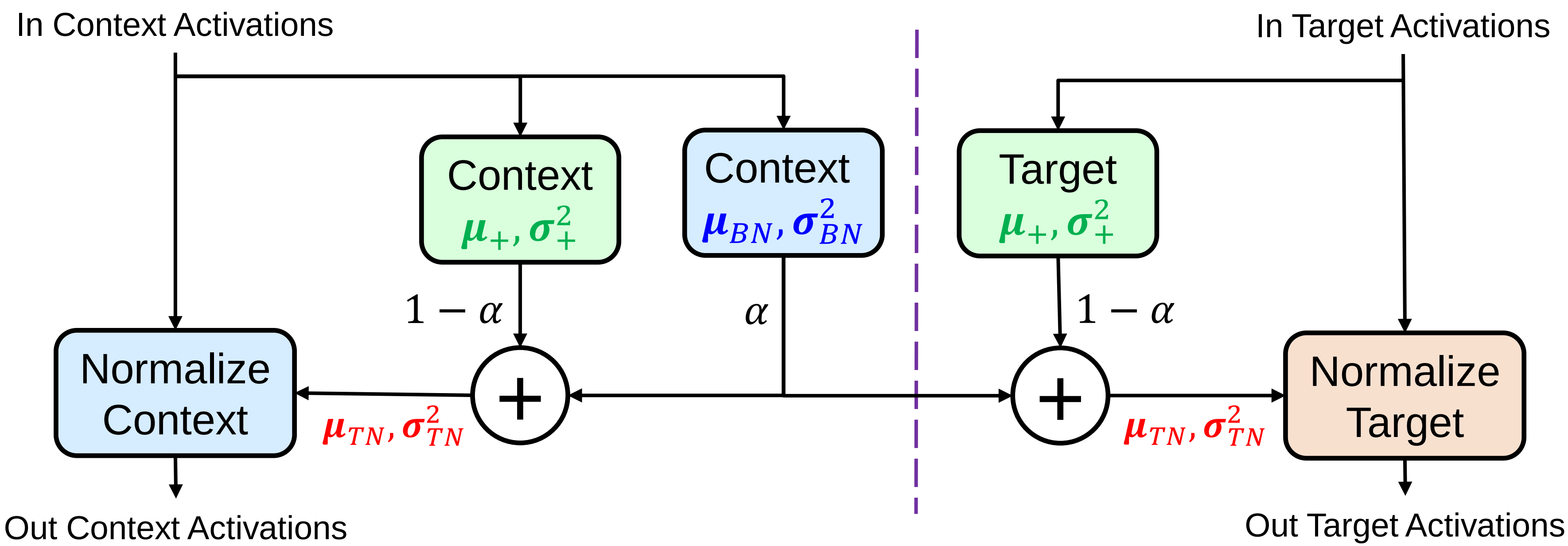
TaskNorm



$$\mu_{TN} = \alpha\mu_{BN} + (1-\alpha)\mu_+$$

$$\sigma_{TN}^2 = \alpha(\sigma_{BN}^2 + (\mu_{BN} - \mu_{TN})^2) + (1-\alpha)(\sigma_+^2 + (\mu_+ - \mu_{TN})^2)$$

TaskNorm



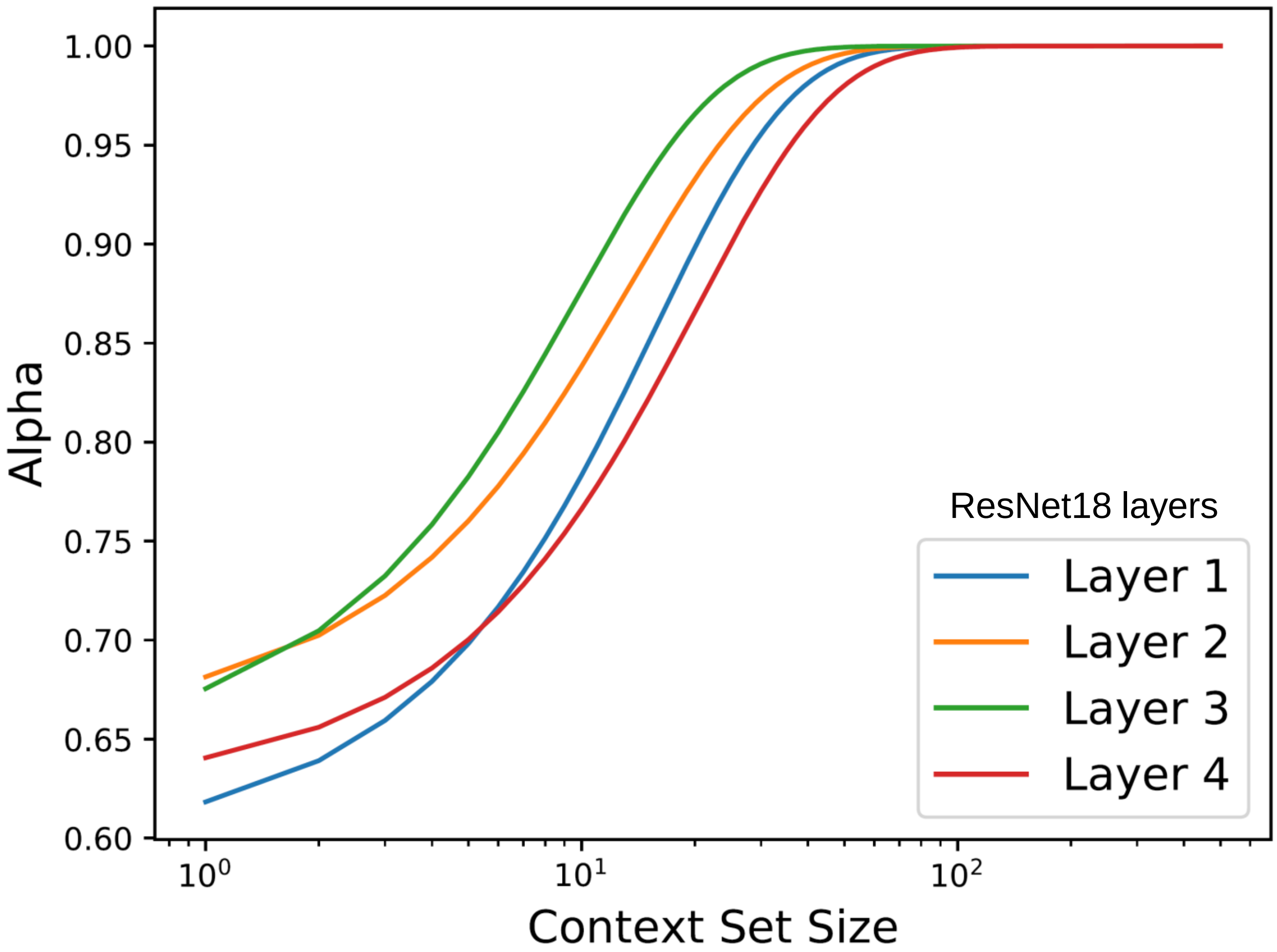
$$\mu_{TN} = \alpha\mu_{BN} + (1-\alpha)\mu_+$$

$$\sigma_{TN}^2 = \alpha(\sigma_{BN}^2 + (\mu_{BN} - \mu_{TN})^2) + (1-\alpha)(\sigma_+^2 + (\mu_+ - \mu_{TN})^2)$$

$$\alpha = SIGMOID(SCALE|D^\tau| + OFFSET), 0 \leq \alpha \leq 1$$

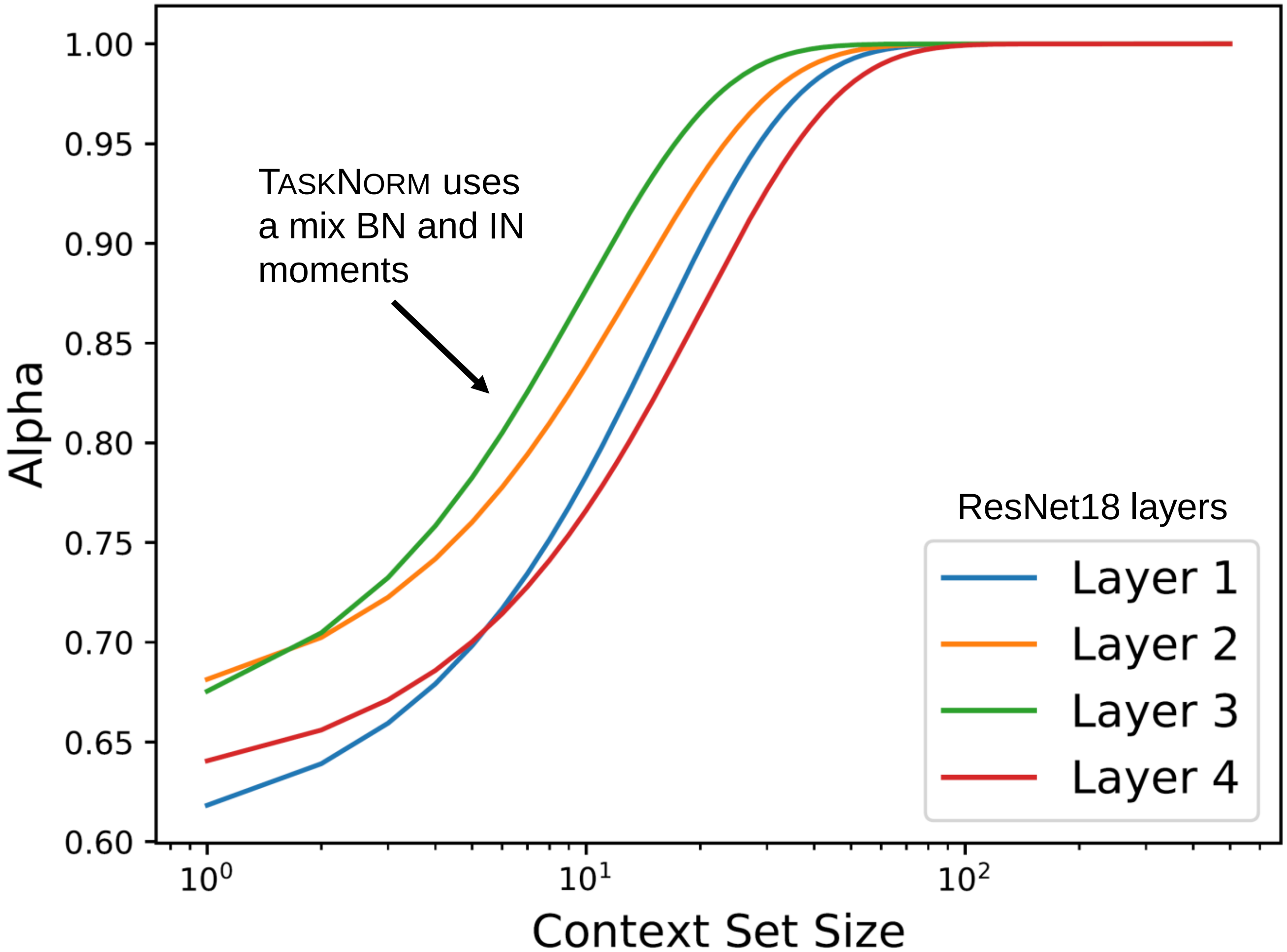
$SCALE, OFFSET$ are learned during training

Learned Alpha (α) vs Context Set Size (D^{τ})



Each curve is the learned value of α in the first TaskNorm in each of the four ResNet 18 layers.

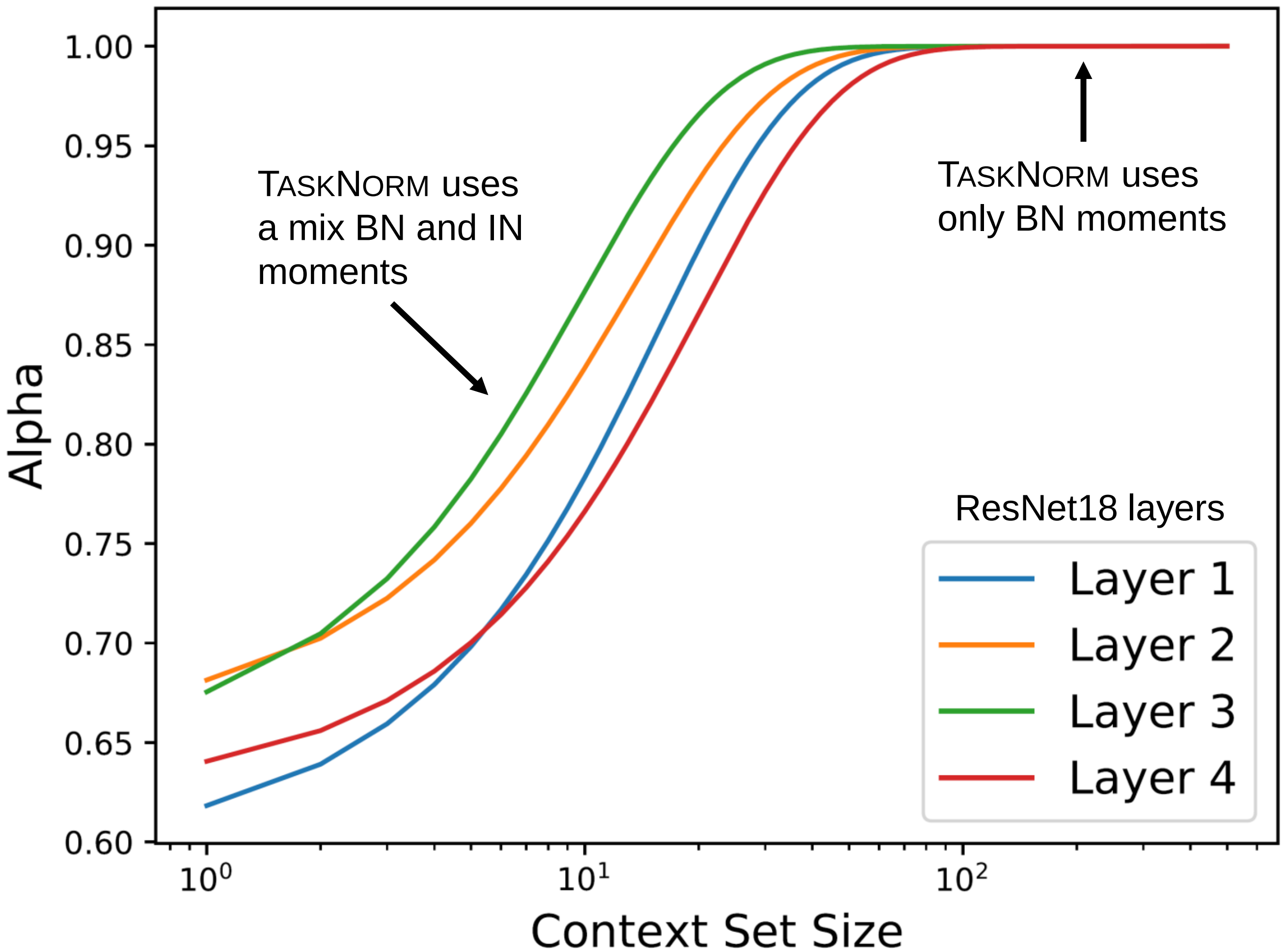
Learned Alpha (α) vs Context Set Size (D^{τ})



Each curve is the learned value of α in the first TaskNorm in each of the four ResNet 18 layers.

When the context set size is small (< 30), TaskNorm learns to use a blend of BN and IN moments.

Learned Alpha (α) vs Context Set Size (D^τ)

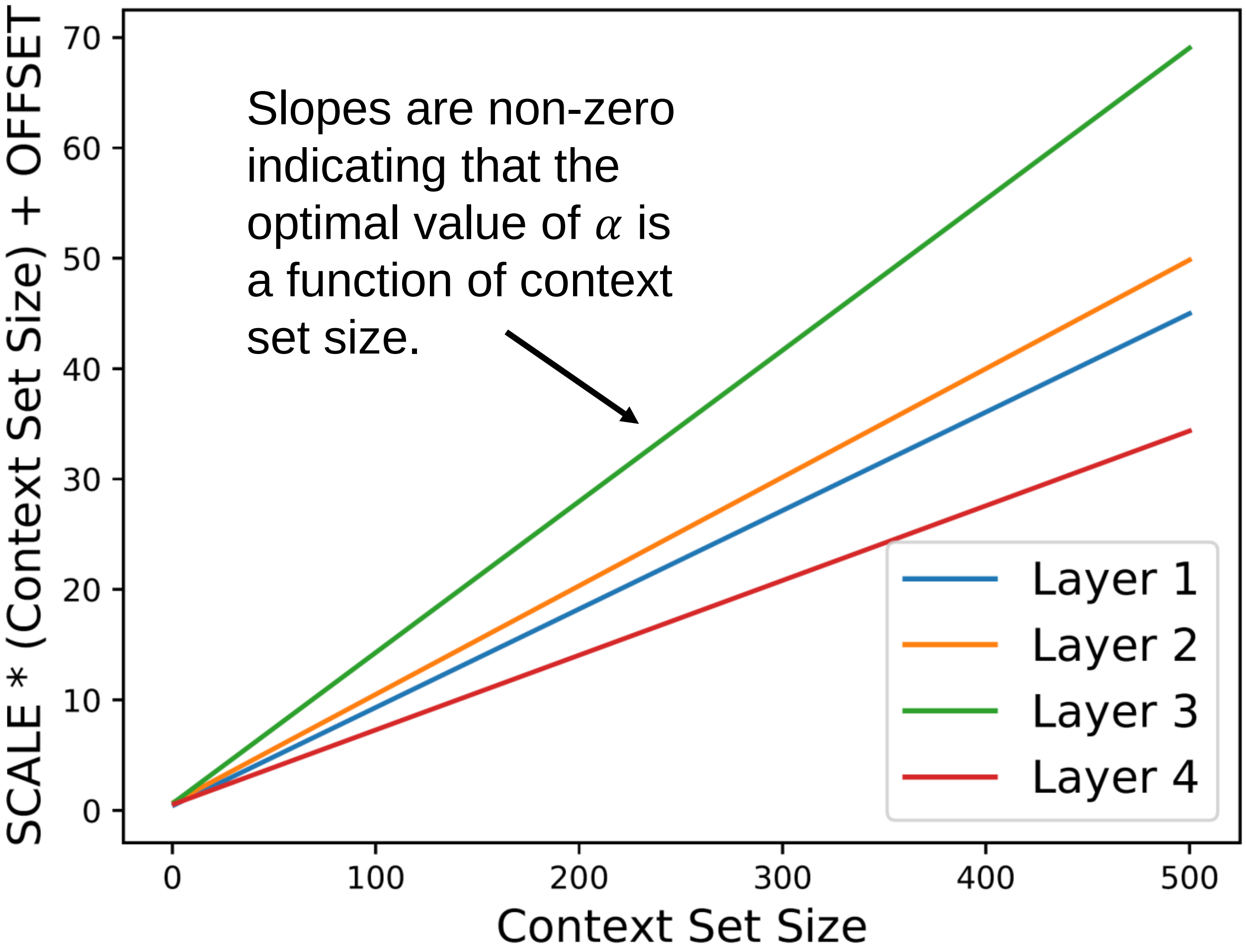


Each curve is the learned value of α in the first TaskNorm in each of the four ResNet 18 layers.

When the context set size is small (< 30), TaskNorm learns to use a blend of BN and IN moments.

When the context set size is large (> 30), TaskNorm learns to use only the BN moments.

SCALE * (Context Set Size) + OFFSET vs Context Set Size



TaskNorm Fixes the Transductive Issue in MAML

Configuration	CBN	TBN	TBN (1 example at a time)	TBN (1 class at a time)	TaskNorm
Omniglot 5-way, 1-shot	20.1±0.0	98.4±0.7	21.6±1.3	21.6±1.3	94.4±0.8
Omniglot 5-way, 5-shot	20.0±0.0	99.2±0.2	22.0±0.5	23.2±0.5	98.6±0.2
Omniglot 20-way, 1-shot	5.0±0.0	90.9±0.5	3.7±0.2	3.7±0.2	90.0±0.5
Omniglot 20-way, 5-shot	5.0±0.0	96.6±0.2	5.5±0.2	14.5±0.3	96.3±0.2
*mini*ImageNet 5-way, 1-shot	20.1±0.0	45.5±1.8	26.9±1.5	26.9±1.5	42.4±1.7
*mini*ImageNet 5-way, 5-shot	20.2±0.0	59.7±0.9	30.3±0.7	27.2±0.6	58.7±0.9

TaskNorm accuracy approaches that of TBN.

TaskNorm Fixes the Transductive Issue in MAML

Configuration	CBN	TBN	TBN (1 example at a time)	TBN (1 class at a time)	TaskNorm	TaskNorm (1 example at a time)	TaskNorm (1 class at a time)
Omniglot 5-way, 1-shot	20.1±0.0	98.4±0.7	21.6±1.3	21.6±1.3	94.4±0.8	94.4±0.8	94.4±0.8
Omniglot 5-way, 5-shot	20.0±0.0	99.2±0.2	22.0±0.5	23.2±0.5	98.6±0.2	98.6±0.2	98.6±0.2
Omniglot 20-way, 1-shot	5.0±0.0	90.9±0.5	3.7±0.2	3.7±0.2	90.0±0.5	90.0±0.5	90.0±0.5
Omniglot 20-way, 5-shot	5.0±0.0	96.6±0.2	5.5±0.2	14.5±0.3	96.3±0.2	96.3±0.2	96.3±0.2
*mini*ImageNet 5-way, 1-shot	20.1±0.0	45.5±1.8	26.9±1.5	26.9±1.5	42.4±1.7	42.4±1.7	42.4±1.7
*mini*ImageNet 5-way, 5-shot	20.2±0.0	59.7±0.9	30.3±0.7	27.2±0.6	58.7±0.9	58.7±0.9	58.7±0.9

TaskNorm accuracy approaches that of TBN.

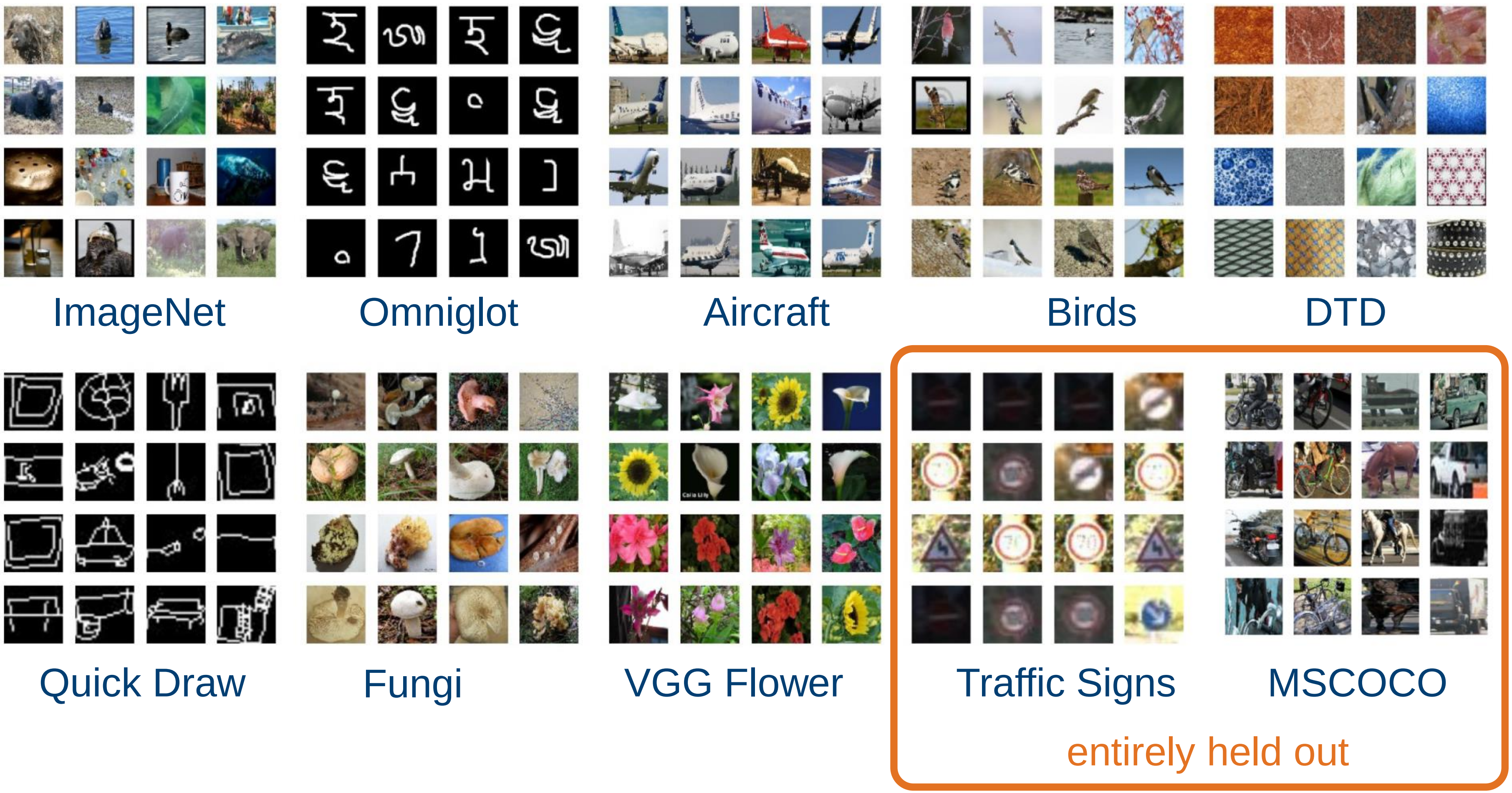
TaskNorm accuracy does not change when tested differently.

Meta-Dataset[1] Multi-task, Few-shot Benchmark

UNIVERSITY OF CAMBRIDGE

[1] Triantafillou, Eleni, et al. "Meta-dataset: A dataset of datasets for learning to learn from few examples.“ *arXiv preprint arXiv:1903.03096* (2019).

Meta-Dataset[1] Multi-task, Few-shot Benchmark



UNIVERSITY OF CAMBRIDGE

[1] Triantafillou, Eleni, et al. "Meta-dataset: A dataset of datasets for learning to learn from few examples.“ *arXiv preprint arXiv:1903.03096* (2019).

Meta-Dataset Classification Accuracy Using ProtoNets[1]

	Dataset	TBN	CBN	BRN	LN	IN	RN	MetaBN	TaskNorm-r	TaskNorm-L	TaskNorm-I
Held out classes	ILSVRC	**44.7±1.0**	43.6±1.0	43.0±1.0	33.9±0.9	32.5±0.9	**45.1±1.0**	**44.2±1.0**	42.7±1.0	**45.1±1.1**	**44.9±1.0**
	Omniglot	**90.7±0.6**	77.5±1.1	89.1±0.7	**90.8±0.6**	83.4±0.8	**90.8±0.6**	**90.4±0.6**	88.6±0.7	**90.2±0.6**	**90.6±0.6**
	Aircraft	83.3±0.6	77.0±0.7	**84.4±0.5**	73.9±0.7	75.0±0.6	80.9±0.6	82.3±0.6	79.6±0.6	81.2±0.6	**84.7±0.5**
	Birds	69.6±0.9	67.5±0.9	69.0±0.9	54.1±1.0	50.2±1.0	68.6±0.9	68.6±0.8	64.2±0.9	68.8±0.9	**71.0±0.9**
	Textures	61.2±0.7	57.7±0.7	58.0±0.7	55.8±0.7	45.3±0.7	64.1±0.7	60.5±0.7	60.8±0.7	63.4±0.8	**65.9±0.7**
	Quick Draw	75.0±0.8	62.1±1.0	74.3±0.8	72.5±0.8	70.8±0.8	75.4±0.7	74.2±0.7	73.2±0.8	75.4±0.7	**77.5±0.7**
	Fungi	46.4±1.0	43.6±1.0	46.5±1.0	33.2±1.1	29.8±1.0	46.7±1.0	46.5±1.0	42.3±1.1	46.5±1.0	**49.6±1.1**
	VGG Flower	83.1±0.6	82.3±0.6	84.5±0.6	78.3±0.8	69.4±0.8	84.4±0.7	**86.0±0.6**	81.1±0.7	82.9±0.7	83.2±0.6
Held out datasets	Traffic Signs	64.0±0.8	59.5±0.8	65.7±0.8	**69.1±0.7**	60.7±0.8	66.0±0.8	63.2±0.8	64.9±0.8	67.0±0.7	65.8±0.7
	MSCOCO	38.2±1.0	36.6±1.0	**38.4±1.0**	30.1±0.9	27.7±0.9	37.3±1.0	**38.6±1.1**	35.4±1.0	**39.2±1.0**	**38.5±1.0**
	MNIST	93.4±0.4	86.5±0.6	91.9±0.4	**94.0±0.4**	87.4±0.5	**93.9±0.4**	**93.9±0.4**	92.5±0.4	91.9±0.4	93.3±0.4
	CIFAR10	64.7±0.8	57.3±0.8	60.1±0.8	51.5±0.8	50.5±0.8	62.3±0.8	63.0±0.8	61.4±0.8	**66.9±0.8**	**67.6±0.8**
	CIFAR100	48.0±1.1	43.1±1.0	43.9±1.0	34.0±0.9	32.1±1.0	47.2±1.1	47.0±1.0	45.2±1.0	**51.3±1.1**	**50.0±1.0**
	Average Rank	4.04	8.19	5.31	7.46	9.58	3.65	3.96	6.73	3.58	2.50

TaskNorm achieves the highest overall rank of all methods including Transductive BatchNorm (TBN)

TaskNorm with *Instance Normalization* is best on 10 of 13 datasets

TBN = Transductive Batch Norm
CBN = Conventional Batch Norm
BRN = Batch Renormalization
LN = Layer Normalization
IN = Instance Normalization

RN = Reptile Norm
MetaBN = Meta Batch Norm
TaskNorm-L = TaskNorm with LN
TaskNorm-I = TaskNorm with IN
TaskNorm-I = TaskNorm with running moments

[1] Snell, Jake, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning." Advances in neural information processing systems. 2017.

Meta-Dataset Classification Accuracy Using CNAPs[1]

	Dataset	TBN	Baseline	CBN	BRN	LN	IN	RN	MetaBN	TaskNorm-r	TaskNorm-L	TaskNorm-I
Held out classes	ILSVRC	**50.2±1.0**	**51.3±1.0**	24.8±0.7	19.2±0.7	45.5±1.1	46.7±1.0	49.7±1.1	**51.3±1.1**	49.3±1.0	**51.2±1.1**	**50.6±1.1**
	Omniglot	**91.4±0.5**	88.0±0.7	47.9±1.4	60.0±1.6	87.4±0.8	79.7±1.0	**91.0±0.6**	**90.9±0.6**	87.8±0.7	90.6±0.6	**90.7±0.6**
	Aircraft	81.6±0.6	76.8±0.8	29.5±0.9	56.3±0.8	76.5±0.8	74.7±0.7	82.4±0.6	**83.9±0.6**	81.1±0.7	81.9±0.6	**83.8±0.6**
	Birds	**74.5±0.8**	71.4±0.9	42.1±1.0	32.6±0.8	67.3±0.9	64.9±1.0	72.4±0.8	73.2±0.9	72.8±0.9	72.4±0.8	**74.6±0.8**
	Textures	59.7±0.7	**62.5±0.7**	37.5±0.7	50.5±0.6	60.1±0.6	59.7±0.7	58.6±0.7	58.9±0.8	**63.2±0.8**	57.2±0.7	62.1±0.7
	Quick Draw	70.8±0.8	71.9±0.8	44.5±1.0	56.7±1.0	71.6±0.8	68.2±0.9	**74.3±0.8**	**74.1±0.7**	71.6±0.8	**74.3±0.8**	**74.8±0.7**
	Fungi	46.0±1.0	46.0±1.1	21.1±0.8	26.1±0.9	39.6±1.0	37.8±1.0	**49.0±1.0**	**47.9±1.0**	42.0±1.1	47.1±1.1	**48.7±1.0**
	VGG Flower	86.6±0.5	**89.2±0.5**	79.0±0.7	75.7±0.7	84.4±0.6	82.6±0.6	86.9±0.6	85.9±0.6	87.7±0.6	87.3±0.5	**89.6±0.6**
Held out datasets	Traffic Signs	**66.6±0.9**	60.1±0.9	38.3±0.9	38.8±1.2	57.3±0.8	62.5±0.8	**66.6±0.8**	58.9±0.9	62.7±0.8	62.0±0.8	**67.0±0.7**
	MSCOCO	41.3±1.0	**42.0±1.0**	14.2±0.7	19.1±0.8	32.9±1.0	40.8±1.0	**42.1±1.0**	41.6±1.1	40.1±1.0	41.6±1.0	**43.4±1.0**
	MNIST	92.1±0.4	88.6±0.5	65.9±0.8	82.5±0.6	86.8±0.5	89.8±0.5	91.3±0.4	92.1±0.4	**93.2±0.3**	90.5±0.4	**92.3±0.4**
	CIFAR10	**70.1±0.8**	60.0±0.8	26.1±0.7	29.1±0.6	55.8±0.8	65.9±0.8	**69.7±0.7**	**69.6±0.8**	66.9±0.8	**70.3±0.8**	**69.3±0.8**
	CIFAR100	55.6±1.0	48.1±1.0	16.7±0.8	16.7±0.7	37.9±1.0	52.9±1.0	55.0±1.0	54.2±1.1	53.0±1.1	**59.5±1.0**	54.6±1.1
	Average Rank	3.92	5.58	10.69	10.31	7.96	7.54	3.77	4.04	5.38	4.42	2.38

TaskNorm achieves the highest overall rank of all methods including Transductive BatchNorm (TBN)

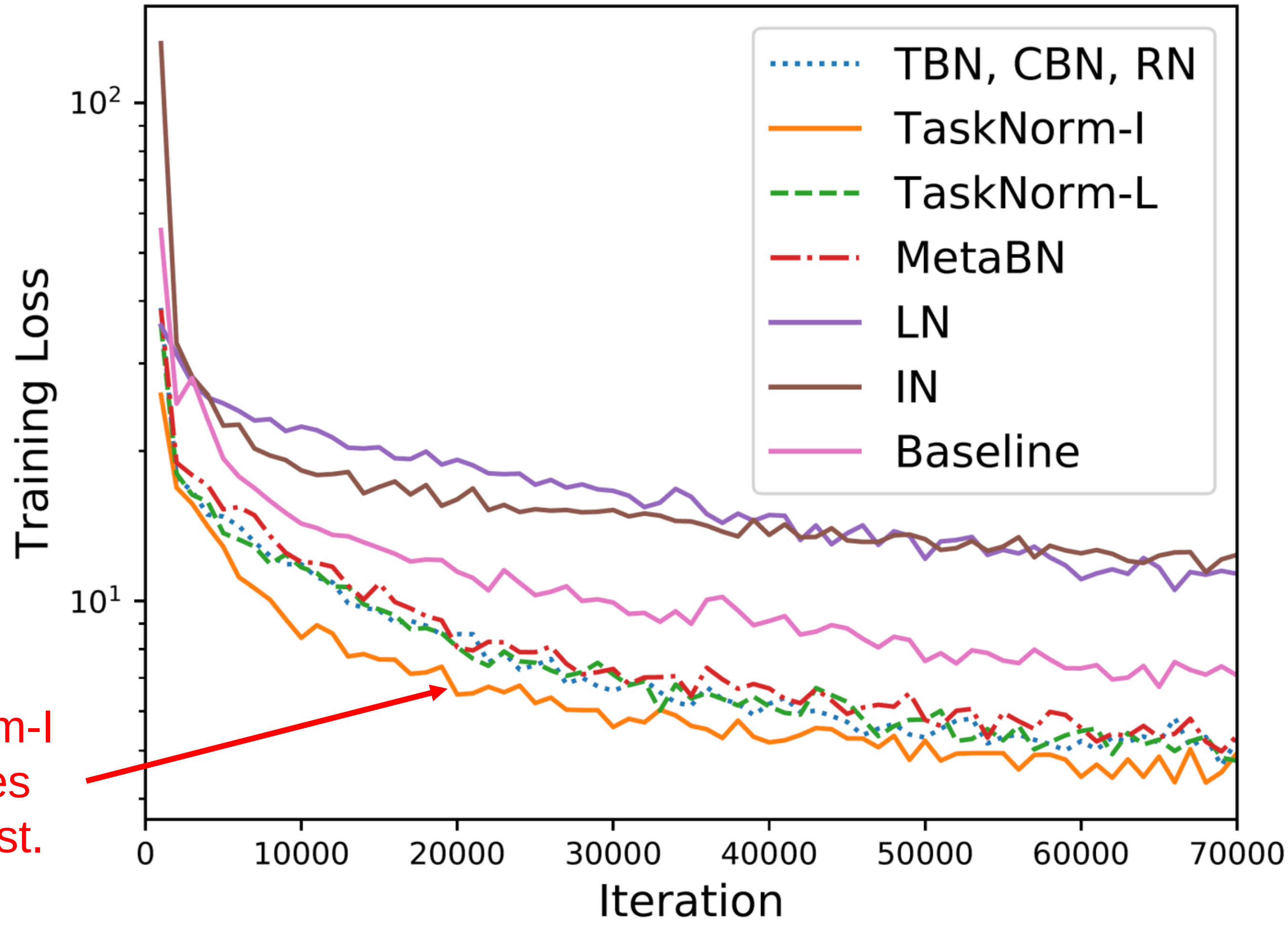
TASKNORM with *Instance Normalization* is best on 11 of 13 datasets

TBN = Transductive Batch Norm
CBN = Conventional Batch Norm
BRN = Batch Renormalization
LN = Layer Normalization
IN = Instance Normalization

RN = Reptile Norm
MetaBN = Meta Batch Norm
TaskNorm-L = TaskNorm with LN
TaskNorm-I = TaskNorm with IN
TaskNorm-I = TaskNorm with running moments
Baseline = No Normalization

[1] Requeima, James, et al. "Fast and flexible multi-task classification using conditional neural adaptive processes." Advances in Neural Information Processing Systems. 2019.

Meta-Dataset Training Curves



TaskNorm-I converges the fastest.

Thanks for watching!

- Paper: https://arxiv.org/pdf/2003.03284.pdf
- Code: https://github.com/cambridge-mlg/cnaps