A. Additional Normalization Layers

Here we discuss various additional NLs that are relevant to meta-learning.

A.1. Batch Renormalization (BRN)

Batch renormalization (BRN; Ioffe, 2017) is intended to mitigate the issue of non-identically distributed and/or small batches while retaining the training efficiency and stability of CBN. In BRN, the CBN algorithm is augmented with an affine transform with batch-derived parameters which correct for the batch statistics being different from the overall population. The normalized activations of a BRN layer are computed as follows:

$$\boldsymbol{a}'_n = \boldsymbol{\gamma}\left(r\left(\frac{\boldsymbol{a}_n - \boldsymbol{\mu}_{BN}}{\boldsymbol{\sigma}_{BN} + \epsilon}\right) + d\right) + \boldsymbol{\beta},$$

where

$$r = \texttt{stop\_grad}\left(\texttt{clip}_{[1/r_{max}, r_{max}]}\left(\frac{\boldsymbol{\sigma}_{BN}}{\boldsymbol{\sigma}_r}\right)\right),$$
$$d = \texttt{stop\_grad}\left(\texttt{clip}_{[-d_{max}, d_{max}]}\left(\frac{\boldsymbol{\mu}_{BN} - \boldsymbol{\mu}_r}{\boldsymbol{\sigma}_r}\right)\right).$$

Here $\texttt{stop\_grad}(\cdot)$ denotes a gradient blocking operation, and $\texttt{clip}_{[a,b]}$ denotes an operation returning a value in the range $[a, b]$. Like CBN, BRN is not well suited to the meta-learning scenario as it does not map directly to the hierarchical form of meta-learning models. In Section 5, we show that using BRN can improve predictive performance compared to CBN, but still performs significantly worse than competitive approaches. Table 1 shows that batch renormalization performs poorly when using MAML.

A.2. Group Normalization (GN)

A key insight of Wu & He (2018) is that CBN performance suffers with small batch sizes. The goal of Group Normalization (GN; Wu & He, 2018) is thus to address the problem of normalization of small batch sizes, which, among other matters, is crucial for training large models in a data-parallel fashion. This is achieved by dividing the image channels into a number of groups G and subsequently computing the moments for each group. GN is equivalent to LN when there is only a single group ($G = 1$) and equivalent to IN when the number of groups is equal to the number of channels in the layer ($G = C$).

A.3. Other NLs

There exist additional NLs including Weight Normalization (Salimans & Kingma, 2016), Cosine Normalization (Luo et al., 2018), Filter Response Normalization (Singh & Krishnan, 2019), among many others.

Weight normalization reparameterizes weight vectors in a neural network to improve the conditioning for optimization. Weight normalization is non-transductive, but we don't consider this approach further in this work as we focus on NLs that modify activations as opposed to weights.

Filter Response Normalization (FRN) is another non-transductive NL that performs well for all batch sizes. However we did not include it in our evaluation as FRN also encompasses the activation function as an essential part of normalization making it difficult to be a drop in replacement for CBN in pre-trained networks as is the case for some of our experiments.

Cosine normalization replaces the dot-product calculation in neural networks with cosine similarity for improved performance. We did not consider this method further in our work as it is not a simple drop-in replacement for CBN in pre-existing networks such as the ResNet-18 we use in our experiments.

B. Experimental Details

In this section, we provide the experimental details required to reproduce our experiments. The experiments using MAML (Finn et al., 2017) were implemented in TensorFlow (Abadi et al., 2015), the Prototypical Networks experiments were implemented in Pytorch (Paszke et al., 2019), and the experiments using CNAPS (Requeima et al., 2019a) were implemented using a combination of TensorFlow (Abadi et al., 2015) and Pytorch. All experiments were executed on NVIDIA Tesla P100-16GB GPUs.

B.1. MAML Experiments

We evaluate MAML using a range of normalization layers on:

1. Omniglot (Lake et al., 2011): a few-shot learning dataset consisting of 1623 handwritten characters (each with 20 instances) derived from 50 alphabets.
2. miniImageNet (Vinyals et al., 2016): a dataset of 60,000 color images that is sub-divided into 100 classes, each with 600 instances.

For all the MAML experiments, we used the codebase provided by the MAML authors (Finn, 2017) with only small modifications to enable additional normalization techniques. Note that we used the first-order approximation version of MAML for all experiments. MAML was invoked with the command lines as specified in the `main.py` file in the MAML codebase. No hyper-parameter tuning was performed and we took the results from a single run. All models

were trained for 60,000 iterations and then tested. No early stopping was used. We did not select the model based on validation accuracy or other criteria. The MAML code employs ten gradient steps at test time and computes classification accuracy after each step. We report the maximum accuracy across those ten steps. To generate the plot in Figure 4a, we use the same command line as Omniglot-5-1, but vary the update batch size from one to ten.

B.2. CNAPS Experiments

We evaluate CNAPS using a range of normalization layers on a demanding few-shot classification challenge called Meta-Dataset (Triantafillou et al., 2020). Meta-Dataset is composed of ten (eight train, two test) image classification datasets. We augment Meta-Dataset with three additional held-out datasets: MNIST (LeCun et al., 2010), CIFAR10 (Krizhevsky & Hinton, 2009), and CIFAR100 (Krizhevsky & Hinton, 2009). The challenge constructs few-shot learning tasks by drawing from the following distribution. First, one of the datasets is sampled uniformly; second, the "way" and "shot" are sampled randomly according to a fixed procedure; third, the classes and context / target instances are sampled. Where a hierarchical structure exists in the data (ILSVRC or OMNIGLOT), task-sampling respects the hierarchy. In the meta-test phase, the identity of the original dataset is not revealed and the tasks must be treated independently (i.e. no information can be transferred between them). Notably, the meta-training set comprises a disjoint and dissimilar set of classes from those used for meta-test. Full details are available in Triantafillou et al. (2020).

For all the CNAPS experiments, we use the code provided by the the CNAPS authors (Requeima et al., 2019b) with only small modifications to enable additional normalization techniques. We follow an identical dataset configuration and training process as prescribed in Requeima et al. (2019b). To generate results in Table 2, we used the following CNAPS options: FiLM feature adaptation, a learning rate of 0.001, and TBN, CBN, BRN, and RN used 70,000 training iterations, IN used 200,000 iterations, LN used 110,000 iterations, and TASKNORM used 60,000 iterations. The CNAPS code generates two models: fully trained and best validation. We report the better of the two. We performed no hyper-parameter tuning and report the test results from the first run. Note that CBN, TBN, and RN share the same trained model. They differ only in how meta-testing is done.

B.3. Prototypical Networks Experiments

We evaluate the Prototypical Networks (Snell et al., 2017) algorithm with a range of NLs using the same Omniglot, miniImageNet, and Meta-Dataset benchmarks.

For Omniglot, we used the codebase created by the Prototypical Networks authors (Snell, 2017). For miniImageNet, we used the a different codebase ((Chen, 2018)) as the first codebase did not support miniImageNet. Only small modifications were made to the two codebases to enable additional NLs. For Omniglot and miniImageNet, we set hyper-parameters as prescribed in (Snell et al., 2017). Early stopping was employed and the model that produced the best validation was used for testing.

For Meta-Dataset, we use the code provided by the the CNAPS authors (Requeima et al., 2019b) with only small modifications to enable additional normalization techniques and a new classifier adaptation layer to generate the linear classifier weights per equation (8) in (Snell et al., 2017). We follow an identical dataset configuration and training process as prescribed in Requeima et al. (2019b). To generate results in Table 3, we used the following CNAPS options: no feature adaptation, a learning rate of 0.001, 60,000 training iterations for all NLs, and the pretrained feature extractor weights were not frozen and allowed to update during meta-training.

C. Additional Classification Results

Table C.1 shows the classification accuracy results for the ProtoNets algorithm on the Omniglot and miniImageNet datasets. Figure C.1a and Figure C.1b show the training curves for the ProtoNets algorithm on Omniglot and Meta-Dataset, respectively.

D. Additional Transduction Tests

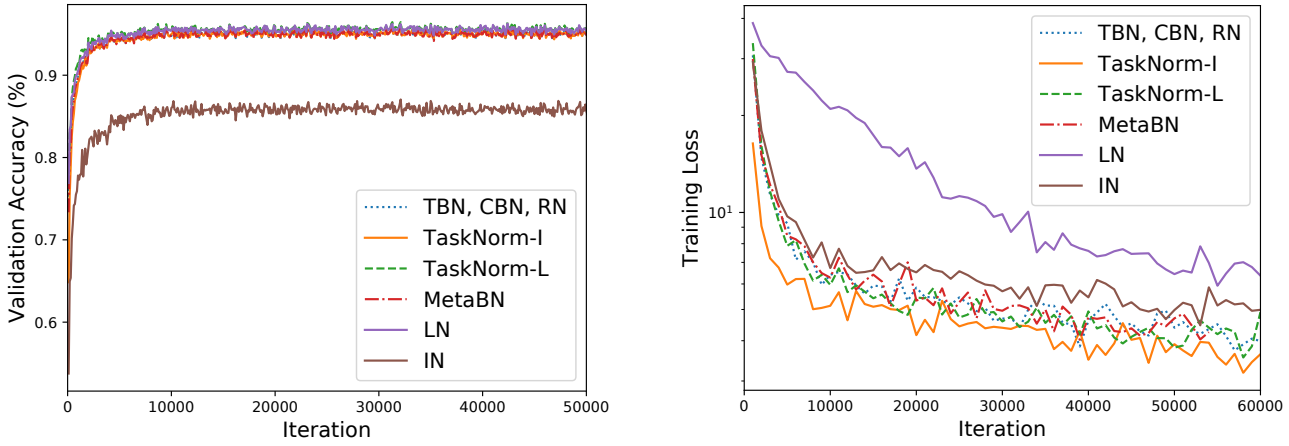
A non-transductive meta-learning system makes predictions for a single test set label conditioned only on a single input and the context set. A transductive meta-learning system also conditions on additional samples from the test set.

Table D.2 demonstrates failure modes for transductive learning. In addition to reporting the classification accuracy results when the target set is evaluated all at once (first column of results for each NL), we report the classification accuracy when meta-testing is performed one target-set *example* at a time (second column of results for each NL), and one target-set *class* at a time (third column of results for each NL). Table D.2 demonstrates that classification accuracy drops dramatically for TBN when testing is performed one example or one class at a time.

Importantly, in the case of TASKNORM-I (or any non-transductive NL – i.e. all of NLs evaluated in this work apart from TBN), the evaluation results are identical whether they are meta-tested on the entire target set at once, one example at a time, or one class at a time. This shows that transductive learning is sensitive to the distribution over the target set used during meta-training, demonstrating that transductive learning is less generally applicable than non-transductive

Table C.1. Accuracy results for different few-shot settings on Omniglot and miniImageNet using the Prototypical Networks algorithm. All figures are percentages and the ± sign indicates the 95% confidence interval. Bold indicates the highest scores. The numbers after the configuration name indicate the way and shots, respectively. The vertical lines in the TBN column indicate that this method is transductive.

Configuration	TBN	CBN	BRN	LN	IN	RN	MetaBN	TaskNorm-L	TaskNorm-I
Omniglot-5-1	98.4±0.2	**98.5±0.2**	**98.5±0.2**	**98.7±0.2**	93.7±0.4	98.0±0.2	98.4±0.2	**98.6±0.2**	98.4±0.2
Omniglot-5-5	**99.6±0.1**	**99.6±0.1**	**99.6±0.1**	**99.7±0.1**	98.8±0.1	**99.6±0.1**	**99.6±0.1**	**99.6±0.1**	**99.6±0.1**
Omniglot-20-1	94.5±0.2	94.5±0.2	94.6±0.2	**94.9±0.2**	83.5±0.3	94.1±0.2	94.5±0.2	**95.0±0.2**	93.4±0.2
Omniglot-20-5	**98.6±0.1**	**98.6±0.1**	**98.6±0.1**	**98.7±0.1**	96.3±0.1	**98.6±0.1**	**98.6±0.1**	**98.7±0.1**	**98.6±0.1**
miniImageNet-5-1	45.9±0.6	**47.8±0.6**	46.3±0.6	**47.5±0.6**	30.4±0.5	39.7±0.5	42.6±0.6	**47.5±0.6**	43.2±0.6
miniImageNet-5-5	65.5±0.5	**66.7±0.5**	64.7±0.5	**66.3±0.5**	48.8±0.5	63.1±0.5	64.6±0.5	65.3±0.5	63.9±0.5
Average Rank	4.58	3.25	4.33	2.75	9.00	6.67	5.25	3.08	6.08



(a) (b)

Figure C.1. (a) Plot of validation accuracy versus training iteration using ProtoNets for Omniglot 20-way, 1-shot corresponding to the results in Table C.1. (b) Training Loss versus iteration corresponding to the results using the ProtoNets algorithm on META-DATASET in Table 3. Note that TBN, CBN, and RN all share the same meta-training step.

learning. In particular, transductive learners may fail to make good predictions if target sets contains a different class balance than what was observed during meta-training, or if they are required to make predictions for one example at a time (e.g. in streaming applications).

E. Ablation Study: Choosing the best parameterization for α

There are a number of possibilities for the parameterization of the TASKNORM blending parameter α. We consider four different configurations for each NL:

1. α is learned separately for each channel (i.e. channel specific) as an independent parameter.
2. α is learned shared across all channels as an independent parameter.
3. α is learned separately for each channel (i.e. channel specific) as a function of context set size (i.e. $\alpha = \text{SIGMOID}(\text{SCALE}|D^\tau| + \text{OFFSET})$).
4. α is learned shared across all channels as a function of context set size (i.e. $\alpha = \text{SIGMOID}(\text{SCALE}|D^\tau| + \text{OFFSET})$).

Accuracy Table E.3 and Table E.4 show classification accuracy for the various parameterizations for MAML and the CNAPS algorithms, respectively using the TASKNORM-I NL.

When using the MAML algorithm, there are only two options to evaluate as the context size is fixed for each configuration of dataset, shot, and way and thus we need only evaluate the independent options (1 and 2 above). Table E.3 indicates that the classification accuracy for the channel specific and shared parameterizations are nearly identical, but the shared parameterization is better in the Omniglot-5-1 benchmark and hence has the best ranking overall.

Table D.2. Few-shot classification results for TBN and TASKNORM-I on META-DATASET using the CNAPS algorithm. For each NL, the first column of results "All" reports accuracy when meta-testing is performed on the entire target set at once. The second column of results "Example" reports accuracy when meta-testing is performed one example at a time. The third column of results "Class" reports accuracy when meta-testing is performed one class at a time. All figures are percentages and the ± sign indicates the 95% confidence interval over tasks. Meta-training is performed on datasets above the dashed line, while datasets below the dashed line are entirely held out.

	TBN			TASKNORM-I		
Dataset	All	Example	Class	All	Example	Class
ILSVRC	50.2±1.0	9.5±0.3	11.8±0.4	50.4±1.1	50.4±1.1	50.4±1.1
Omniglot	91.4±0.5	7.5±0.4	9.6±0.4	91.3±0.6	91.3±0.6	91.3±0.6
Aircraft	81.6±0.6	11.8±0.4	14.4±0.4	83.8±0.6	83.8±0.6	83.8±0.6
Birds	74.5±0.8	7.6±0.4	8.4±0.4	74.4±0.9	74.4±0.9	74.4±0.9
Textures	59.7±0.7	17.0±0.2	18.1±0.4	61.1±0.7	61.1±0.7	61.1±0.7
Quick Draw	70.8±0.8	5.6±0.4	8.8±0.4	74.7±0.7	74.7±0.7	74.7±0.7
Fungi	46.0±1.0	5.0±0.3	6.5±0.4	50.6±1.1	50.6±1.1	50.6±1.1
VGG Flower	86.6±0.5	11.2±0.4	12.6±0.4	87.8±0.5	87.8±0.5	87.8±0.5
Traffic Signs	66.6±0.9	6.0±0.3	8.1±0.4	64.8±0.8	64.8±0.8	64.8±0.8
MSCOCO	41.3±1.0	6.1±0.3	7.9±0.4	42.2±1.0	42.2±1.0	42.2±1.0
MNIST	92.1±0.4	14.4±0.3	19.3±0.4	91.3±0.4	91.3±0.4	91.3±0.4
CIFAR10	70.1±0.8	14.4±0.3	16.4±0.4	70.0±0.8	70.0±0.8	70.0±0.8
CIFAR100	55.6±1.0	5.6±0.3	7.7±0.4	54.6±1.0	54.6±1.0	54.6±1.0

Table E.3. Few-shot classification results for two α parameterizations on Omniglot and miniImageNet using the MAML algorithm. All figures are percentages and the ± sign indicates the 95% confidence interval over tasks. Bold indicates the highest scores.

	Independent	
Configuration	Channel Specific	Shared
Omniglot-5-1	90.7±1.0	**94.4±0.8**
Omniglot-5-5	98.3±0.2	**98.6±0.2**
Omniglot-20-1	**90.6±0.5**	**90.0±0.5**
Omniglot-20-5	**96.4±0.2**	**96.3±0.2**
miniImageNet-5-1	**42.6±1.8**	**42.4±1.7**
miniImageNet-5-5	**58.8±0.9**	**58.7±0.9**
Average Rank	1.67	1.33

When using the CNAPS algorithm on the Meta-Dataset benchmark, the best parameterization option in terms of classification accuracy is α shared across channels as a function of context size. One justification for having α be a function of context size can be seen in Figure 3b. Here we plot the line SCALE$|D^\tau|$ + OFFSET on a linear scale for a representative set of NLs in the ResNet-18 used in the CNAPS algorithm. The algorithm has learned that the SCALE parameter is non-zero and the OFFSET is almost zero in all cases. If a constant α would lead to better accuracy, we would see the opposite (i.e the SCALE parameter would be at or near zero and the OFFSET parameter being some non-zero value). From Table E.4 we can also see that accuracy is better when the parameterization is a shared α opposed to having a channel-specific α.

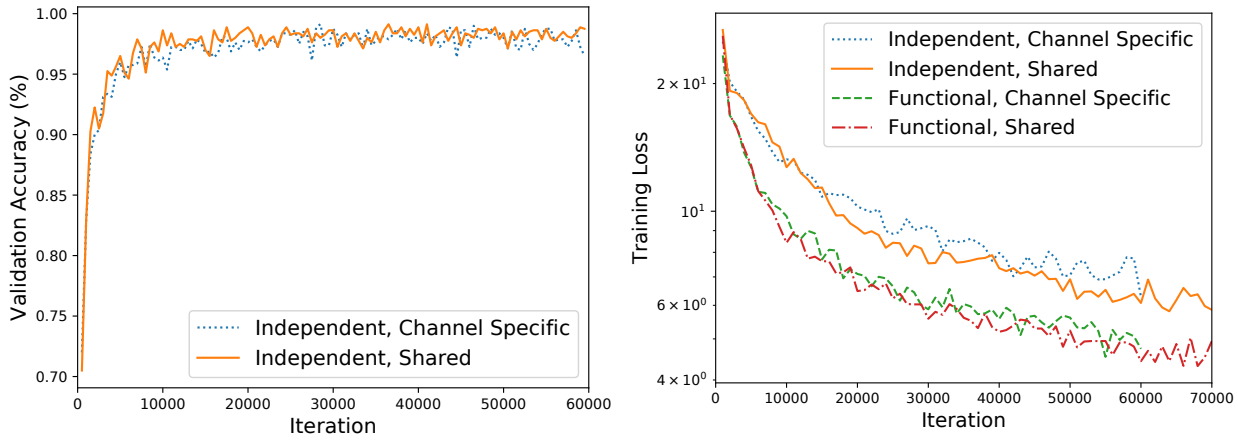
Training Speed Figure E.2a and Figure E.2b show the learning curves for the various parameterization options using the MAML and the CNAPS algorithms, respectively with a TASKNORM-I NL.

For the MAML algorithm the training efficiency of the shared and channel specific parameterizations are almost identical. For the CNAPS algorithm, Figure E.2b indicates the training efficiency of the independent parameterization is considerably worse than the functional one. The two functional representations for the CNAPS algorithm have almost identical training curves. Based on Figure E.2a and Figure E.2b, we conclude that the training speed of the functional parameterization is superior to that of the independent parameterization and that there is little or no difference in the training speeds between the functional, shared parameterization and the functional, channel specific parameterization.

In summary, the best parameterization for α when it is learned shared across channels as a function of context set size (option 4, above). We use this parameterization in all of the CNAPS experiments in the main paper. For the MAML experiments, the functional parameterization is meaningless given that all the test configurations have a fixed context size. In that case, we used the independent, shared across channels parameterization for α for the experiments in the main paper.

Table E.4. Few-shot classification results for various α parameterizations on META-DATASET using the CNAPS algorithm. All figures are percentages and the $\pm$ sign indicates the 95% confidence interval over tasks. Bold indicates the highest scores. Meta-training performed on datasets above the dashed line, while datasets below the dashed line are entirely held out.

	Independent		Functional	
Dataset	Channel Specific	Shared	Channel Specific	Shared
ILSVRC	45.3±1.0	**49.6±1.1**	**49.8±1.1**	**50.6±1.1**
Omniglot	**90.8±0.6**	**90.9±0.6**	**90.1±0.6**	**90.7±0.6**
Aircraft	82.3±0.7	**84.6±0.6**	**84.4±0.6**	**83.8±0.6**
Birds	70.1±0.9	73.2±0.9	73.1±0.9	**74.6±0.8**
Textures	54.8±0.7	58.5±0.7	61.0±0.8	**62.1±0.7**
Quick Draw	73.0±0.8	**73.9±0.7**	**74.2±0.7**	**74.8±0.7**
Fungi	43.8±1.0	**47.6±1.0**	**48.0±1.0**	**48.7±1.0**
VGG Flower	85.9±0.6	86.3±0.5	86.5±0.7	**89.6±0.6**
Traffic Signs	62.6±0.8	62.6±0.8	60.1±0.8	**67.0±0.7**
MSCOCO	38.3±1.1	40.9±1.0	40.2±1.0	**43.4±1.0**
MNIST	**92.6±0.4**	91.7±0.4	91.1±0.4	**92.3±0.4**
CIFAR10	65.7±0.9	67.7±0.8	67.3±0.9	**69.3±0.8**
CIFAR100	48.1±1.2	52.1±1.1	**53.3±1.0**	**54.6±1.1**
Average Rank	3.5	2.5	2.5	1.5



(a) (b)

Figure E.2. (a) Plots of validation accuracy versus training iteration corresponding to the parameterization experiments using the MAML algorithm in Table E.3. (b) Plot of training loss versus iteration corresponding to the parameterization experiments using the CNAPS algorithm in Table E.4.