A NeurIPS 2020 Reviews and Changelog

An earlier version of this work was submitted to NeurIPS 2020 and rejected with borderline reviews. Here, we first summarize the reviewer concerns and describe in detail the steps we took to address these concerns. Then, we include the raw reviewer comments for completeness.

A.1 Review summary

Generally, the reviewers agreed that this work "addresses a critical problem" with a "simple and interesting solution". The reviewers also mostly agreed that the results demonstrate the effectiveness of the proposed solution. The concerns primarily revolved around the framing and novelty of the contribution and the assumptions made in the problem statement.

R1 said that the "novelty of the proposed method is not significant", mirroring concerns from **R3** that "the method for instantiation is relatively simple" and "more possibility [for methods] should be investigated". The first key step we took to address these comments was to clarify that the focus and primary novel contribution of this paper is to introduce the ARM problem formulation and not any particular method for solving ARM. Indeed, we agree with **R1** and **R3** that some standard meta learning approaches, including the simple CML approach, can be readily adapted to the ARM setting. This is by design, and framing the distribution shift problem such that it can be addressed via meta learning is, to the best of our knowledge, a novel contribution. It is our hope that being able to extend existing tools will accelerate progress on this important problem, and we have edited the paper throughout to reflect this central goal.

Second, we devised and evaluated additional methods beyond ARM-CML, including ARM-BN and ARM-LL. Beyond further demonstrating the generality of our proposed framework and providing extensive evaluations of different approaches, this contribution also addresses other concerns such as **R2**'s question about "[why is Algorithm 1] a meta-learning algorithm?". This question is answered by seeing how a variety of different meta-learning approaches, beyond CML, fit into the general procedure prescribed by Algorithm 1.

As for the problem assumptions, **R1** and **R4** commented that "the assumption that distributions are grouped... is also strong" and "training the models requires explicit labels specifying the group", respectively. **R1** further commented that the approach "assumes that a batch of data... is available during test time, which is not realistic". We addressed this by first editing the exposition, mainly in subsection 3.2, in order to make it clear that not only are these assumptions well established in the literature on distribution shift, but also that we must consider assumptions stronger than ERM in order to learn models that generalize in the face of shift. These assumptions are used in prior works because they are not as onerous as the reviewers may suggest. Constructing training groups can be readily accomplished via meta-data, which is present for most datasets, and assuming multiple data points at test time is feasible for most applications.

Nevertheless, we also ran additional experiments in subsection 4.4 that loosened both of these assumptions, by learning groups using a VAE and evaluating ARM methods under a streaming setting. We empirically demonstrated that ARM is still competitive even when the training and test assumptions do not hold, while outperforming prior methods when the assumptions do hold.

Finally, there were some comments regarding the experimental setup. **R2** said "I think that the authors should focus on the feature extraction and the domain generalization method". Though we discuss domain generalization in our updated paper, we note that there are a few significant differences between our work and domain generalization, e.g., we consider multiple test time shifts and measure worst case performance. Nevertheless, we think that exploring domain generalization testbeds using ARM can be an interesting direction for future work. **R4** had concerns about parameter counts in the MNIST and FEMNIST experiments and evaluation in a prior experiment on CelebA. In this version of the paper, we fixed the parameter count issue and replaced the CelebA experiment with CIFAR-10-C and Tiny ImageNet-C, which do not have the same evaluation concerns.

Before we present the raw comments, we would like to point out that we revised the work according to *all* the comments made by the meta-reviewer: "better highlighting the novelty to the formulation and clearly stating the relevance of existing meta learning approaches", "justifying the strong assumptions being made in reference to prior work", "improving the evaluation", "incorporating the gradient based meta learner experiments", and "clarifying points made in the author response".

A.2 Raw reviewer comments

A.2.1 Meta-reviewer

The paper studies the important problem of robust adaptation to test-time distributions shift. Unfortunately the reviewer and AC consistently agree that for the proposal of Adaptive Risk Minimization to be impactful and convincingly adopted, several concerns need to be addressed, including better highlighting the novelty to the formulation and clearly stating the relevance of existing meta learning approaches, justifying the strong assumptions being made in reference to prior work, improving the evaluation, incorporating the gradient based meta learner experiments and clarifying points made in the author response.

A.2.2 Reviewer 1

1. Summary and contributions: Briefly summarize the paper and its contributions.

They have addressed the problem of adaptive risk minimization to handle dataset shift occurring at test time. A meta-learning approach is proposed to tackle the challenge and the experimental results demonstrate the validity of the proposed method to some extent.

2. Strengths: Describe the strengths of the work. Typical criteria include: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the NeurIPS community.

The distribution gap between training and test is a serious problem that practitioners often face with. Improving the (expected) worst-case performance under the distribution shift is an interesting topic for the community.

Formulations in the paper are straightforward to follow and look sound theoretically. The proposed algorithm is simple and easy to reproduce.

The experimental results are comprehensive and the validity and efficacy of the proposed approach are demonstrated.

3. Weaknesses: Explain the limitations of this work along the same axes as above.

While the concept of adaptive risk minimization is general, the key assumptions made by the paper narrows the applicability of the proposed method. First, the paper assumes that a batch of test data (10-50 samples) is available during test time, which is not realistic. Most users test an AI model sequentially and the feedback by the AI should be returned immediately, instead of batching and delaying outputs. In some sense, seeing many future test samples for adapting the model during test time can be unfair when compared to other machine learning methods that follow strict training/test splits. For that reason, I do not think the experimental results are completely fair.

Second, the assumption that distributions are grouped and the group membership is given during training is also strong. In most cases, we do not know which cluster the data points are sampled from. Predefining the data-generating clusters and mapping the distribution shift problem to the group choice problem oversimplifies the fundamental generalization problem of machine learning.

The novelty of the proposed method is not significant. The adaptation power directly comes from the meta-learning methods and I do not see a technical challenge that is found and solved uniquely in the paper.

4. Correctness: Are the claims and method correct? Is the empirical methodology correct?

The claims, formulations, and the method are correct and well defined. Empirical results are correct if we can ignore that the proposed method uses more information during test time.

5. Clarity: Is the paper well written?

The paper is well written.

6. Relation to prior work: Is it clearly discussed how this work differs from previous contributions?

Recent works in the field are well addressed and the comparison is adequate.

7. Reproducibility: Are there enough details to reproduce the major results of this work?

Yes

9. Please provide an "overall score" for this submission.

5: Marginally below the acceptance threshold.

10. Please provide a "confidence score" for your assessment of this submission.

4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.

11. Have the authors adequately addressed the broader impact of their work, including potential negative ethical and societal implications of their work?

Yes

A.2.3 Reviewer 2

1. Summary and contributions: Briefly summarize the paper and its contributions.

This work proposes a learning-to-adapt algorithm that adapts to distribution shift at test time using a batch of unlabeled test data points. The algorithm learns an adaptation strategy that can generalizes across domains. The batch of unlabeled test data points can be a batch of the inputs of the test data. Actually, it is not only a domain adaptation method but also a domain generalization method.

2. Strengths: Describe the strengths of the work. Typical criteria include: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the NeurIPS community.

The out-of-distribution generalization problem is important. This work provides a simple and interesting solution to the domain adaptation. What's more, the proposed algorithm can also deal with the domain generalization problems.

3. Weaknesses: Explain the limitations of this work along the same axes as above.

In fact, this paper considers a varying-weight model g(x, \theta(x)), where x is a mini-batch of inputs. Each batch is considered as a unit or group. Thus, the proposed algorithm learns adaptation to batches rather than domains. Is it essentially different to feature extraction?

4. Correctness: Are the claims and method correct? Is the empirical methodology correct?

I believe that the method and the empirical results are correct. I do not think that all the claims in this paper are proper, e.g. adaptation to a domain or adaptation to a batch.

5. Clarity: Is the paper well written?

Yes

6. Relation to prior work: Is it clearly discussed how this work differs from previous contributions?

No. I think that the authors should focus on the feature extraction and the domain generalization method, e.g. Invariant Risk Minimization, rather than domain adaptation.

7. Reproducibility: Are there enough details to reproduce the major results of this work?

Yes

8. Additional feedback, comments, suggestions for improvement and questions for the authors:

1. This paper considers a varying-weight model g(x, \theta(x)), where x is a mini-batch of inputs. Algorithm 1 is a gradient descent method to directly solve the varying-weight model. Why you call it as a meta-learning algorithm? 2. Is Algorithm 1 essentially different to the feature extraction method? 3. If Algorithm 1 is a batch-adaptation algorithm, can we rewrite z as the index of batches? 4. Please explain the distribution p_emp at Page 4 Line 165.

9. Please provide an "overall score" for this submission.

6: Marginally above the acceptance threshold.

10. Please provide a "confidence score" for your assessment of this submission.

4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.

11. Have the authors adequately addressed the broader impact of their work, including potential negative ethical and societal implications of their work?

Yes

A.2.4 Reviewer 3

1. Summary and contributions: Briefly summarize the paper and its contributions.

This paper presents ARM, a problem formulation for learning models that robustly adapt to test-time distribution shift, by assuming a batch of unlabeled test examples is available. This problem formulation is more practical compared to domain adaptation case. An algorithm to tackle the group shift in this task is proposed by leveraging meta-learning. Then the general algorithm is instantiated based on contextual meta-learning approach. Experiments on several datasets are performed to illustrate the effectiveness of the proposed approach.

2. Strengths: Describe the strengths of the work. Typical criteria include: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the NeurIPS community.

This paper proposes an adaptive risk minimization method for tackling group shift and some specific experiments are conducted to verify its ability.

3. Weaknesses: Explain the limitations of this work along the same axes as above.

1. Though the proposed framework is general, the method for instatiation is relatively simple and more possibility should be investigated. And the presentation should be improved. 2. It is emphasized in the introduction that the hypothesis is that a batch of unlabeled data points is accessed in the training and test stage. Thus, in line 120 and Eq. (1), the learner should not depend on the label y. It would be more consistent to directly introduce the adaptation model h that adapts on the unlabeled test data, instead of the learner h. 3. Algorithm 1 is a general meta-learning approach for optimizing the ARM objective and then in section 4.2, the general algorithm is instantiated based on a contextual meta-learning approach. However, there are some gaps between the general framework and the instantiation. For example, h does not depend on the current status of prediction network, i.e., in line 216, since \psi identically maps to \psi, the h function in fact takes in (x1,. . . ,xk) and produce (\bar{c}). The reason for choosing this instantiation, and whether there are other instantiations based on prior meta-learning methods is not explained. 4. More qualitative and/or intuitive analysis of the algorithm should be added. For example, in figure 2, why the prediction network benefits from the information contained in context vector, or is it related to relative attributes?

4. Correctness: Are the claims and method correct? Is the empirical methodology correct?

relatively correct.

5. Clarity: Is the paper well written?

not well written. Some presentation is not consistent in the whole paper, which makes it confusing.

6. Relation to prior work: Is it clearly discussed how this work differs from previous contributions?

clearly.

7. Reproducibility: Are there enough details to reproduce the major results of this work?

Yes

9. Please provide an "overall score" for this submission.

4: An okay submission, but not good enough; a reject.

10. Please provide a "confidence score" for your assessment of this submission.

4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.

11. Have the authors adequately addressed the broader impact of their work, including potential negative ethical and societal implications of their work?

Yes

A.2.5 Reviewer 4

1. Summary and contributions: Briefly summarize the paper and its contributions.

This paper presents Adaptive Risk Minimization, a framework to learn models that are robust to group shifts in the input distribution. Previous works required access to labeled test examples at training or test time; or have tried to optimize for all possible shifts which tends to degrade performance on real distributions. In contrast, the proposed framework uses meta-learning to learn to adapt the model to test distributions using only a small sample of unlabeled instances at test time.

2. Strengths: Describe the strengths of the work. Typical criteria include: soundness of the claims (theoretical grounding, empirical evaluation), significance and novelty of the contribution, and relevance to the NeurIPS community.

The paper addresses a critical problem in machine learning applications, the shift in the distribution of training and test data. The key contribution of the paper is a framework to adapt models to shifts in distribution using only a set of unlabeled examples from the test distribution. However, training the models requires explicit labels specifying the group to which a training example belongs to.

3. Weaknesses: Explain the limitations of this work along the same axes as above.

The biggest weakness of the paper is the evaluation. The experiments all follow slightly different protocols and not enough care was taken to ensure that the results are directly comparable.

4. Correctness: Are the claims and method correct? Is the empirical methodology correct?

I have some concerns with the experimental protocol.

Why were the baselines on the FEMNIST set to have *more* parameters than the proposed models. How many layers/parameters were added? I am not sure if the differences are due to the merits of the method or due to configuration of the models.

I did not understand the binning strategy used in the CelebA dataset. How does it ensure a fair comparison in this setting?

5. Clarity: Is the paper well written?

The paper is well written and the method is sound.

6. Relation to prior work: Is it clearly discussed how this work differs from previous contributions?

Prior work is adequately discussed and contrasted with the papers contribution.

7. Reproducibility: Are there enough details to reproduce the major results of this work?

Yes

9. Please provide an "overall score" for this submission.

6: Marginally above the acceptance threshold.

10. Please provide a "confidence score" for your assessment of this submission.

4: You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.

11. Have the authors adequately addressed the broader impact of their work, including potential negative ethical and societal implications of their work?

Yes

B Detailed Descriptions of Meta-Learning Approaches

For ARM-CML, we introduce two neural networks: a context network $f_{\text{cont}}(\cdot\,;\phi) : \mathcal{X} \to \mathbb{R}^D$, as mentioned in subsection 3.3, and a *prediction network* $f_{\text{pred}}(\cdot\,,\cdot\,;\theta) : \mathcal{X} \times \mathbb{R}^D \to \mathcal{Y}$, parameterized by θ. As discussed, f_{cont} processes each example $\mathbf{x}_k$ in the mini batch separately to produce contexts $\mathbf{c}_k \in \mathbb{R}^D$ for $k = 1, \ldots, K$, which are averaged together into $\mathbf{c} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{c}_k$. In our experiments, we choose D to be the dimensionality of $\mathbf{x}$, such that we can concatenate each $\mathbf{x}_k$ and $\mathbf{c}$ along the channel dimension to produce the input to f_{pred}. Thus, f_{pred} processes each $\mathbf{x}_k$ separately to produce an estimate of the output $\hat{y}_k$, but it additionally receives $\mathbf{c}$ as input. In this way, f_{cont} can provide information about the entire batch of K unlabeled data points to f_{pred} for predicting the correct outputs.

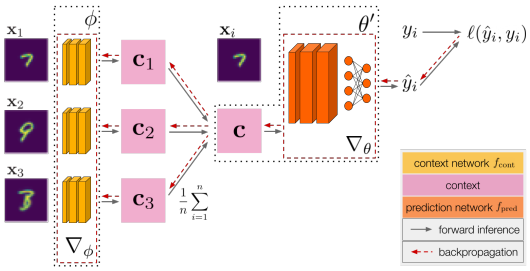

Figure 5: During inference for ARM-CML, the context network produces a vector $\mathbf{c}_k$ for each input image $\mathbf{x}_k$ in the batch, and the average of these vectors is used as the context $\mathbf{c}$ is input to the prediction network. This context may adapt the model by providing helpful information about the underlying test distribution, and this adaptation can aid prediction for difficult or ambiguous examples. During training, we compute the loss of the post adaptation predictions and backpropagate through the inference procedure to update the model.

A schematic of this method is presented in Figure 5. The post adaptation model parameters θ' are $[\theta, \bar{c}]$. Since we only ever use the model after adaptation, both during training and at test time, we can simply define $g(\mathbf{x}; \theta') = f_{\text{pred}}(\mathbf{x}, \mathbf{c}; \theta)$, leaving the model's behavior before adaptation undefined. We then also see that h is a function that takes in $(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K)$ and produces $\left[\theta, \frac{1}{K}\sum_{k=1}^{K} f_{\text{cont}}(\mathbf{x}_k; \phi)\right]$. In the streaming setting tested in subsection 4.4, we keep track of the average context over the previous test points $\mathbf{c}$ and we maintain a counter t of the number of test points seen so far.[1] When we observe a new point $\mathbf{x}$, we increment the counter and update the average context as $\frac{t}{t+1}\mathbf{c} + \frac{1}{t+1} f_{\text{cont}}(\mathbf{x}; \phi)$, and then we make a prediction on $\mathbf{x}$ using this updated context. Notice that, with this procedure, we do not need to store any test points after they are observed, and this procedure results in an equivalent context to ARM-CML in the batch test setting after observing K data points.

At a high level, ARM-BN operates in a similar fashion to ARM-CML, thus we group these methods together into the umbrella of contextual approaches. However, most of the details are different. For ARM-BN, there is no context network, and h has no parameters, i.e., ϕ is empty. The model g is again specified via a prediction network f_{pred}, which must have batch normalization layers. Batch normalization typically tracks a running average of the first and second moments of the activations in these layers, which are then used at test time. Thus, we can view these moments, along with the weights in f_{pred}, as part of θ. ARM-BN instead defines h to swap out these moments for the moments computed via the activations on the test batch. This method is remarkably simple, and in deep learning libraries such as PyTorch [46], the implementation requires the changing of a single line of code. However, as shown in Section 4, this method also performs very well empirically, and it is further boosted by meta-training.

In the streaming setting, ARM-BN is also similar to ARM-CML. Denote the context after seeing t test points as $\mathbf{c} = [\boldsymbol{\mu}, \boldsymbol{\sigma}^2]$, the mean and variance of the batch normalization layer activations on the points so far. Upon seeing a new test point, let $\mathbf{a}$ denote the batch normalization layer activations computed from this new point, and let $\boldsymbol{\mu}_{\mathbf{a}}$ and $\boldsymbol{\sigma}^2_{\mathbf{a}}$ denote the mean and variance of $\mathbf{a}$, respectively. We then update the new context to be $\left[\frac{t}{t+1}\boldsymbol{\mu} + \frac{1}{t+1}\boldsymbol{\mu}_{\mathbf{a}}, \frac{t}{t+1}\boldsymbol{\sigma}^2 + \frac{1}{t+1}\boldsymbol{\sigma}^2_{\mathbf{a}}\right]$. Again note that we do not need to store any test points.

Finally, for ARM-LL, we note that ϕ contains only the parameters of the loss network f_{loss}, and h is defined as

$$h(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi) = \theta - \alpha \nabla_\theta \| [f_{\text{loss}}(g(\mathbf{x}_1; \theta); \phi), \ldots, f_{\text{loss}}(g(\mathbf{x}_K; \theta); \phi)] \|_2 \,.$$

[1] An alternative to maintaining a counter t is to use an exponential moving average, though we do not experiment with this option.

We found that $\alpha = 0.1$ worked well for our experiments. In practice, we also found it was beneficial to take multiple gradient steps, rather than a single gradient step as suggested by this equation. We used 1 gradient steps during meta-training and meta-testing. Finally, for the streaming setting, we can perform a single gradient step with a smaller α after observing each test point, and in an online fashion, we can continually update the model parameters over the course of testing rather than initializing from the meta-learned parameters for each test point.

C Additional Experimental Details

When reporting our results, we run each method across three seeds and report the mean and standard error across seeds. Standard error is calculated as the sample standard deviation divided by the square root of 3. We checkpoint models after every epoch of training, and at test time, we evaluate the checkpoint with the best worst case validation accuracy. Training hyperparameters and details for how we evaluate validation and test accuracy are provided for each experimental domain below.

C.1 Rotated MNIST details

We construct a training set of 32292 data points by replicating 90% of the original training set – separating out a validation set – and then applying random rotations to each image. The rotations are not dependent on the image or label, but certain rotations are sampled much less frequently than others. In particular, rotations of 0 through 20 degrees, inclusive, have 7560 data points each, 30 through 50 degrees have 2160 points each, 60 through 80 have 648, 90 through 110 have 324 each, and 120 to 130 have 108 points each.

We train all models for 200 epochs with mini batch sizes of 50. We use Adam updates with learning rate 0.0001 and weight decay 0.0001. We construct an additional level of mini batching for our method as described in subsection 3.3, such that the batch dimensions of the data mini batches is 6×50 rather than just 50, and each of the inner mini batches contain examples from the same rotation. We refer to the outer batch dimension as the *meta batch size* and the inner dimension as the batch size. All methods are still trained for the same number of epochs and see the same amount of data. Finally, DRNN uses an additional learning rate hyperparameter for their robust loss, which we set to 0.01 across all experiments [53].

Due to the large number of groups in this setting, we only compute validation accuracy every 10 epochs. When computing validation accuracy, we estimate accuracy on each rotation by randomly sampling 300 of the held out 6000 original training points and applying the specific rotation, resampling for each validation evaluation. This is effectively the same procedure as the test evaluation, which randomly samples 3000 of the 10000 test points and applies a specific rotation.

We retain the original $28 \times 28 \times 1$ dimensionality for the MNIST images, and we divide inputs by 256. We use convolutional neural networks for all methods with varying depths to account for parameter fairness. For ERM, the UW baseline, and DRNN, the network has four convolution layers with 128 filters of size 5×5, followed by 4×4 average pooling, one fully connected layer of size 200, and a linear output layer. Rectified linear unit (ReLU) nonlinearities are used throughout, and batch normalization [25] is used for the convolution layers. The first two convolution layers use padding to preserve the input height and width, and the last two convolution layers use 2×2 max pooling. For our method and context ablation, we remove the first two convolution layers for the prediction network, but we incorporate a context network. The context network uses two convolution layers with 64 filters of size 5×5, with ReLU nonlinearities, batch normalization, and padding, followed by a final convolution layer with padding. This last layer has number of filters, of size 5×5, equal to 12 in the case of MNIST, 3 for CIFAR and Tiny ImageNet-C, and 1 for FEMNIST.

C.2 FEMNIST details

FEMNIST, and EMNIST in general, is a significantly more challenging dataset compared to MNIST due to its larger label space (62 compared to 10 classes), label imbalance (almost half of the data points are digits), and inherent ambiguities (e.g., lowercase versus uppercase "o") [9]. In processing the FEMNIST dataset,[2] we filter out users with fewer than 100 examples, leaving 262, 50, and 35

[2] https://github.com/TalwalkarLab/leaf/tree/master/data/femnist.

unique users and a total of 62732, 8484, and 8439 data points in the training, validation, and test splits, respectively. The smallest users contain 104, 119, and 140 data points, respectively. We keep all hyperparameters the same as MNIST, except we set the meta batch size for our method to be 2.

We additionally compare to q-FedAvg on this domain, as this method is specifically designed for federated learning settings [32]. We modify the authors' publicly available code[3] to run experiments in our setting, and we will make this fork available upon publication along with our own code base. This method follows its own update rule and hyperparameter settings, and we separately optimize the hyperparameters for q-FedAvg as described in Li et al. [32]. Specifically, we first set $q = 0$ and sweep learning rate values between 0.0001 and 1.0, and then we sweep $q \in \{0.001, 0.01, 0.1, 0.5, 1, 2, 5, 10, 15\}$ with the optimal learning rate. With this procedure, we set learning rate to be 0.8 and q to be 0.001.

We compute validation accuracy every epoch by iterating through the data of each validation user once, and this procedure is the same as test evaluation. Note that all methods will sometimes receive small batch sizes as each user's data size may not be a multiple of 50, and though this may affect ARM methods, we demonstrate in subsection 4.4 that ARM-CML can adapt using batch sizes much smaller than 50. The network architectures are the same as the architectures used for rotated MNIST.

C.3 CIFAR-10-C and Tiny ImageNet-C details

For both CIFAR-10-C and Tiny ImageNet-C, we construct training, validation, and test sets with 56, 17, and 22 groups, respectively. Each group is based on type and severity of corruption. We split groups such that corruptions in the training, validation, and test sets are disjoint. Specifically, the training set consists of Gaussian noise, shot noise, defocus blur, glass blur, zoom blur, snow, frost, brightness, contrast, and pixelate corruptions of all severity levels. Similarly, the validation set consists of speckle noise, Gaussian blur, and saturate corruptions, and the test set consists of impulse noise, motion blur, fog, and elastic transform corruptions of all severity levels. For two corruptions, spatter and JPEG compression, we include lower severities (1-3) in the training set and higher severities (4-5) in the validation and test sets. For the training and validation sets, each group consists of 1000 images for CIFAR-10-C and 2000 images for Tiny ImageNet-C, giving training sets of size 56000 and 112000, respectively. We use the full test set of 10000 images for each group, giving a total of 220000 test images for both CIFAR-10-C and Tiny ImageNet-C.

In these experiments, we train ResNet-50 [18] models with a support size of 50 and meta batch size of 6. As described above, the context ablation and ARM-CML additionally use small convolutional context networks, and the learned loss ablation and ARM-LL use small fully connected loss networks. The images are normalized by the ImageNet mean and standard deviation before they are passed through the model. For CIFAR-10-C, we train models from scratch for 100 epochs, and for Tiny ImageNet-C we fine tune a pretrained model for 50 epochs. We use stochastic gradient descent with learning rate 0.01, momentum 0.9, and weight decay 0.0001. We evaluate validation accuracy after every epoch and perform model selection based on the worst case accuracy over groups. We perform test evaluation by randomly sampling 3000 images from each group and computing worst case and average classification accuracy across groups.

[3] https://github.com/litian96/fair_flearn