Adaptive Risk Minimization: A Meta-Learning Approach for Tackling Group Distribution Shift

Marvin Zhang [*1] **Henrik Marklund** [*2] **Nikita Dhawan** [*1] **Abhishek Gupta** [1] **Sergey Levine** [1] **Chelsea Finn** [2]

Abstract

A fundamental assumption of most machine learning algorithms is that the training and test data are drawn from the same underlying distribution. However, this assumption is violated in almost all practical applications: machine learning systems are regularly tested under *distribution shift*, due to changing temporal correlations, atypical end users, or other factors. In this work, we consider the setting where the training data are structured into groups and there may be multiple test time shifts, corresponding to new groups or group distributions. Most prior methods aim to learn a single robust model or invariant feature space to tackle this group shift. In contrast, we aim to learn models that *adapt at test time* to shift using unlabeled test points. Our primary contribution is to introduce the framework of adaptive risk minimization (ARM), in which models are optimized for post adaptation performance on training batches sampled from different groups, which simulate group shifts that may occur at test time. We use meta-learning to solve the ARM problem, and compared to prior methods for robustness, invariance, and adaptation, ARM methods provide consistent gains of 1-4% test accuracy on image classification problems exhibiting group shift.

1. Introduction

The standard assumption in empirical risk minimization (ERM) is that the data distribution at test time will match the training distribution. When this assumption does not hold, the performance of standard ERM methods typically deteriorates rapidly, and this setting is commonly referred to as distribution or dataset *shift* (Quiñonero Candela et al., 2009; Lazer et al., 2014). As an example we will study in detail later, consider a handwriting classification system that, after training on a database of past users, is deployed to new end users. Each user induces a data distribution that does not match the global training distribution, and the new users that have peculiar handwriting represent particularly extreme shifts that deteriorate model performance. This test scenario must be carefully considered when building machine learning systems for real world applications.

Algorithms for handling distribution shift have been studied under a number of frameworks (Quiñonero Candela et al., 2009). One commonly used assumption within these frameworks is that the training data are provided in *groups* and distributions at test time will represent new groups – as in domain generalization (Blanchard et al., 2011; Gulrajani & Lopez-Paz, 2021) – or new group distributions – as in group distributionally robust optimization (DRO) (Hu et al., 2018; Sagawa et al., 2020). Constructing training groups in practice is generally accomplished by using meta-data, which exists for many commonly used datasets. Thus, frameworks that use this assumption are still applicable to a wide range of realistic distribution shift problems.

However, many prior benchmarks in this setting are geared toward discovering *invariances* – i.e., where there is a consistent input-output relationship across all groups, and the goal is to learn this relationship while identifying and ignoring the spurious correlations within the groups (see, e.g., Gulrajani & Lopez-Paz (2021)). This assumption motivates methods that, through techniques including robust optimization and learning an invariant feature space, aim for generalization to shifts by discovering this relationship (Li et al., 2018b; Arjovsky et al., 2019; Sagawa et al., 2020). This goal is appealing in that it makes minimal assumptions about the information provided at test time – in particular, these methods do not require test labels, which are often not available, and the learned model can be immediately applied to predict on a single test point. Nevertheless, these methods also have limitations, such as in dealing with problems where the input-output relationship varies across groups.

In this paper, we instead focus on methods that aim to *adapt* at test time to group shift. To do so, we study problems in which it is both feasible and helpful (and perhaps even necessary) to assume access to a batch or stream of *unlabeled* data at test time. Leveraging this assumption does not require labels for any test data and is feasible in many

[*]Equal contribution [1]University of California, Berkeley [2]Stanford University. Correspondence to: Marvin Zhang <marvin@eecs.berkeley.edu>.

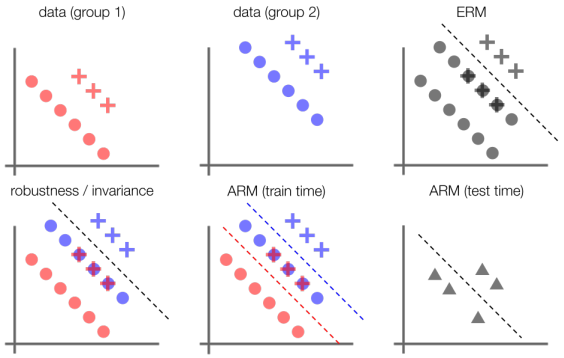

Figure 1. A toy problem for which adaptation is crucial to good predictive performance, discussed in detail in Section 2. Non adaptive models are unable to deal with the ambiguous data points, whereas adaptive models trained via ARM methods can leverage group information and unlabeled test batches to succeed.

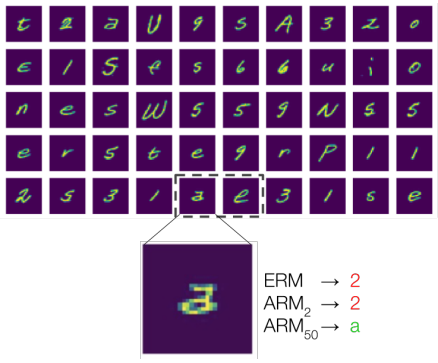

Figure 2. An example of ambiguous data points in a handwriting classification application. A model trained via an ARM method, when given a batch of 50 unlabeled test points, is able to adapt to output the correct label "a" on the ambiguous example, shown enlarged, whereas other models incorrectly output "2".

practical setups. For example, in a handwriting recognition application, we do not access single handwritten characters from an end user, but rather collections of characters such as sentences or paragraphs. Unlabeled adaptation has been shown empirically to be useful for distribution shift problems (Sun et al., 2020; Schneider et al., 2020; Wang et al., 2021), such as for dealing with image corruptions (Hendrycks & Dietterich, 2019), and we identify additional problems where adaptation is crucial in Section 6.

The main contribution of this paper is to introduce the framework of adaptive risk minimization (ARM), which proposes the following objective: optimize the model such that it can maximally leverage the unlabeled adaptation phase to handle group distribution shift. We tackle this objective by introducing a general *meta-learning* algorithm and instantiating a set of methods that, given a set of training groups, meta-learns a model that is adaptable to these groups. Our experiments demonstrate that the proposed ARM methods, by leveraging the meta-training phase, are able to consistently outperform prior methods for handling shift by 1-4% test accuracy in image classification settings including benchmarks for federated learning (Caldas et al., 2019) and image classifier robustness (Hendrycks & Dietterich, 2019).

2. Illustrative Examples

Consider the 2D binary classification problem depicted in Figure 1, where the label is indicated by a circle or cross. Here, the training examples come from two groups (color coded red and blue), and the challenge in this problem is that some data points are ambiguous: identical points can have different labels depending on the group. When learning a model on this data, we may intuitively expect better performance if we use a learning method that uses knowledge of the groups during training. However, any method that learns a non adaptive model, i.e., a model which represents a single decision boundary, will still misclassify half of the ambiguous training points *at best*. For example, robustness methods that optimize for worst case group performance would sacrifice correctness on group 1 in order to succeed on the majority points in group 2. Invariance methods would fare no better, as there is no feature space that can separate the ambiguous points and make them classifiable.

As we will discuss in Section 5, ARM methods can solve this problem by learning adaptive models, which can represent different decision rules for different groups. Unlike prior methods for test time adaptation, ARM methods leverage the training groups to specifically train the model to adapt to group shift. At test time, the model adapts by observing unlabeled points (shown as triangles), and it learns to leverage this adaptation through the training phase.

In Figure 2, we highlight a similar example for handwriting classification, a problem setting that we evaluate quantitatively in Section 6. We visualize a batch of 50 examples from a test user, and we highlight an ambiguous example which may be either a "2" or an "a" depending on the user's handwriting. An ERM trained model and an ARM trained model, when only given two unlabeled examples for adaptation, incorrectly classify this example as "2". However, when given access to the entire batch of 50 images, which contain examples of class "2" and "a" from this user, the ARM trained model successfully adapts its prediction to "a", which is the correct label. Beyond problems involving ambiguous data points, we also observe the benefits of ARM methods in a problem from the DomainBed benchmark suite (Gulrajani & Lopez-Paz, 2021), discussed in Appendix C, and additional problems in Section 6.

3. Related Work

A number of prior works have studied distribution shift in various forms (Quiñonero Candela et al., 2009). In this section, we review prior work in domain generalization, group DRO, meta-learning, and adaptation.

Invariance and robustness to groups. As discussed above, a number of frameworks leverage training groups to address test time shift. The terminology in prior work is scattered and, depending on the application, includes terms such as "domains", "datasets", "subpopulations", and "users"; in this work, we adopt the term "groups" which we believe is an appropriate unifying term. A number of testbeds for this problem setting have been proposed for image classification, including generalizing to new datasets (Fang et al., 2013), new image types (Li et al., 2017a; Peng et al., 2019), and underrepresented demographics (Sagawa et al., 2020).

Prior benchmarks typically assume the existence of a consistent input-output relationship across groups that is learnable by the specified model, thus motivating methods such as learning an invariant feature space to model all the groups (Li et al., 2018b;c; Arjovsky et al., 2019) or optimizing for worst case group performance (Hu et al., 2018; Sagawa et al., 2020). Gulrajani & Lopez-Paz (2021) provide a comprehensive survey of many of these benchmarks and find that, surprisingly, ERM is competitive with the state of the art across all the benchmarks considered. In Appendix C, we discuss this finding as well as the performance of an ARM method on this benchmark suite. In Section 6, we identify different problems for which adaptation is helpful, and we find that ARM methods consistently outperform ERM and other non adaptive group based learning methods.

Meta-learning. Meta-learning (Schmidhuber, 1987; Bengio et al., 1992; Thrun & Pratt, 1998; Hochreiter et al., 2001) has been most extensively studied in the context of few shot supervised learning methods (Santoro et al., 2016; Vinyals et al., 2016; Ravi & Larochelle, 2017; Finn et al., 2017; Snell et al., 2017), i.e., *labeled* adaptation. Our aim is not to address few-shot recognition problems, as in prior meta-learning work, nor to propose a novel meta-learning algorithm, but rather to extend meta-learning paradigms to problems requiring unlabeled adaptation, with the primary goal of tackling distribution shift. This aim differs from previous work in meta-learning for domain generalization (Li et al., 2018a; Dou et al., 2019), which seek to meta-train models for non adaptive generalization performance. We discuss in Section 5 how paradigms such as contextual meta-learning (Garnelo et al., 2018; Requeima et al., 2019) can be readily extended using the ARM framework.

Some other meta-learning methods adapt using both labeled and unlabeled data, either in the semi supervised learning setting (Ren et al., 2018; Zhang et al., 2018; Li et al., 2019) or the transductive learning setting (Liu et al., 2019; Antoniou & Storkey, 2019; Hu et al., 2020). These works do not focus on the same setting of distribution shift and all assume access to labeled data for adaptation. Prior works in meta-learning for unlabeled adaptation include Yu et al. (2018), who adapt a policy to imitate human demonstrations in the context of robotic learning; Metz et al. (2019), who meta-learn an update rule for unsupervised representation learning, though they still require labels to learn a predictive model; and Alet et al. (2020), who meta-learn adaptive models based on task specific unsupervised objectives. Unlike these prior works, we propose a general framework for tackling distribution shift problems by meta-learning unsupervised adaptation strategies. This framework simplifies the extension of meta-learning paradigms to these problems, encapsulates previous approaches such as the gradient based meta-learning approach of Yu et al. (2018), and sheds light on how to improve existing strategies such as adaptation via batch normalization (Li et al., 2017b).

Adaptation to shift. Unlabeled adaptation has primarily been studied separately from meta-learning. Domain adaptation is a prominent framework that assumes access to test examples at *training* time (Csurka, 2017; Wilson & Cook, 2020), similar to transductive learning (Vapnik, 1998). As such, most domain adaptation methods consider the problem setting where there is a single test distribution (Shimodaira, 2000; Daumé III, 2007; Gong et al., 2017; Ganin & Lempitsky, 2015; Tzeng et al., 2017) and some are difficult to apply to problems where there are multiple test distributions. Certain domain adaptation methods have also been applied in the domain generalization setting, such as methods for learning invariant features (Ganin & Lempitsky, 2015; Li et al., 2018b), and we compare to these methods in Section 6.

Blanchard et al. (2011) and Blanchard et al. (2021) provide a theoretic study of domain generalization and establish favorable generalization bounds for models that can adapt to group shift at test time. In comparison, our work establishes a framework that makes explicit the connection between adaptation to group shift and meta-learning, allowing us to devise new methods in a straightforward and principled manner. These methods are amenable to expressive models such as deep neural networks, which enables us to study real world problems with raw image observations.

Adaptation at test time has also been studied from the perspective of dealing with label shift (Royer & Lampert, 2015; Lipton et al., 2018; Sulc & Matas, 2019), as well as crafting favorable inductive biases for the domain of interest. For example, for image classification, techniques such as using normalization statistics of the test inputs (Li et al., 2017b) and optimizing self-supervised surrogate losses (Sun et al., 2020) have proven effective for dealing with shift. We compare against these prior methods in Section 6.

4. Preliminaries and Notation

In this section, we discuss the group distribution shift problem setting and formally describe adaptive models. In Section 5, we discuss how adaptive models can be meta-trained via the ARM objective and approach, and we instantiate ARM methods which we empirically evaluate in Section 6.

Let $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ represent the input and output, respectively. We assume that the training data points are generated by repeated runs of the following process: first, a data distribution $p_{\mathbf{x}y} \in \mathcal{P}_{\mathbf{x}y}$ is sampled from a set of distributions, and then data is sampled from $p_{\mathbf{x}y}$. We refer to each $p_{\mathbf{x}y}$ as a group, e.g., a different dataset or user, and at training time, the labeled training data are organized by group. An equivalent characterization which we will use for clarity is that, within the training set, there are S groups, and each data point $(\mathbf{x}^{(i)}, y^{(i)})$ is annotated with a group label $z^{(i)}$. Each $z^{(i)}$ is an integer that takes on a value between 1 and S, indicating which $p_{\mathbf{x}y}$ generated the corresponding training point. At test time, there may be multiple evaluation settings, where each setting is considered separately and contains unlabeled data sampled via a new run of the same generative process. This data may represent, e.g., a new dataset or user, and the test groups are likely to be distinct from the training groups when $|\mathcal{P}_{\mathbf{x}y}|$ is large or infinite.

Our formal goal is to optimize for expected performance, e.g., classification accuracy, at test time. To do so, we will consider predictive models of the form $f : \mathcal{X} \times \mathcal{P}_{\mathbf{x}} \to \mathcal{Y}$, where the model f takes in not just an input $\mathbf{x}$ but also the marginal input distribution $p_{\mathbf{x}} \in \mathcal{P}_{\mathbf{x}}$ that $\mathbf{x}$ was sampled from. We refer to f as an *adaptive* model, as it has to opportunity to use $p_{\mathbf{x}}$ to adapt its predictions on $\mathbf{x}$. The underlying assumption is that $p_{\mathbf{x}}$ provides information about the conditional output distribution, i.e., $p_{\mathbf{x}}$ is used as a surrogate input in place of $p_{\mathbf{x}y}$. In the worst case, if $p_{\mathbf{x}}$ and $p_{y|\mathbf{x}}$ are sampled independently, then the model does not benefit at all from knowing $p_{\mathbf{x}}$ (Blanchard et al., 2021). In many problems, however, we expect knowledge about $p_{\mathbf{x}}$ to be useful, e.g., for resolving ambiguity as described in Section 2.

Practically, we cannot input $p_{\mathbf{x}}$ to f in most cases. Similar to Blanchard et al. (2021), we instead replace $p_{\mathbf{x}}$ with a batch of inputs $\mathbf{x}_1, \ldots, \mathbf{x}_K$ where K is a hyperparameter. This batch serves as an empirical approximation $\hat{p}_{\mathbf{x}}$ to $p_{\mathbf{x}}$, and we assume that a batch of K unlabeled points will also be available at test time for adaptation. Though the exposition will focus on this test setting for clarity, we also experiment in Section 6 with the *streaming* setting where the test inputs are observed one at a time and adaptation occurs incrementally as the test batch grows. Thus, the assumption of test batches is not crucial. Notice that, if we instead passed in an approximation $\hat{p}_{\mathbf{x}y}$ of $p_{\mathbf{x}y}$ to the model, such as a batch of *labeled* data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_K, y_K)$, then this setup would resemble the standard few shot meta-learning problem (Vinyals et al., 2016). In the next section, we expand on this connection to develop the ARM framework, which then allows us to bring forward tools from meta-learning to tackle group distribution shift problems.

5. Adaptive Risk Minimization

In this section, we formally describe the ARM framework, which defines an objective for training adaptive models to tackle group shift. Furthermore, we propose a general meta-learning algorithm as well as specific methods for optimizing the ARM objective. In Section 6, we test these ARM methods on problems for which unlabeled adaptation can be leveraged for better test performance.

5.1. Devising the ARM objective

Motivated by Section 4, the goal of this work is to learn models $f : \mathcal{X} \times \mathcal{X}^K \to \mathcal{Y}$ that can adapt using batches of unlabeled data to handle group distribution shift. As noted, meta-learning methods for labeled adaptation study a similar form of model, and a common approach in many of these methods is to define f such that it is composed of two parts: first, a *learner* which ingests the data and produces parameters, and second, a *prediction model* which uses these parameters to make predictions (Vinyals et al., 2016; Finn et al., 2017). We will follow a similar strategy which, as we will discuss in subsection 5.2, allows us to easily extend and design meta-learning methods towards our goal.

In particular, we will decompose the model f into two modules: a standard *prediction model* $g(\cdot\,; \theta) : \mathcal{X} \to \mathcal{Y}$, that is parameterized by $\theta \in \Theta$ and predicts y given $\mathbf{x}$, and an *adaptation model* $h(\cdot, \cdot\,; \phi) : \Theta \times \mathcal{X}^K \to \Theta$, which is parameterized by ϕ. h, which is analogous to the learner in meta-learning, takes as input the prediction model parameters θ and K unlabeled data points, which serve as the empirical input distribution $\hat{p}_{\mathbf{x}}$, and produces updated parameters θ' after adapting using the K points.[1] h can be initialized as a variety of different adaptation procedures, and we defer this discussion to subsection 5.2. Our goal is to meta-learn ϕ and θ such that h can adapt g using unlabeled training data sampled according to a particular group z. This motivates the ARM objective, given by

$$\min_{\theta,\phi} \mathbb{E}_{p_z}\left[\mathbb{E}_{p_{\mathbf{x}y|z}}\left[\frac{1}{K}\sum_{k=1}^{K} \ell(g(\mathbf{x}_k; \theta'), y_k)\right]\right], \quad (1)$$
$$\text{where } \theta' = h(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi)\,.$$

[1] An alternate characterization in meta-learning is that the learner takes in only the data points to produce a set of model parameters (Vinyals et al., 2016), and we also devise methods of this form in subsection 5.2. These methods are also captured by this formalism as h can choose to simply ignore the input θ.

Mimicking the generative process from Section 4 that we assume generated the training data, p_z is a categorical distribution over $\{1, \dots, S\}$ which places uniform probability mass on each training group, and $p_{\mathbf{x}y|z}$ assigns uniform probability to only the data points within a particular group. Theoretically, we expect the trained models to perform well at test time if the test groups are sampled independently and identically – i.e., from the same distribution over $\mathcal{P}_{\mathbf{x}y}$ – as the training groups. In practice, similar to how meta-learned few shot classification models are typically evaluated on new and unseen meta-test classes (Vinyals et al., 2016; Finn et al., 2017), we empirically show in Section 6 that the trained models can generalize to test groups that are not sampled identically to the training groups.

Standard few shot meta-learning formulations must use disjoint data batches for adaptation and meta-training to avoid label memorization (Vinyals et al., 2016). Since labels are not used during adaptation in the ARM framework, we train the adapted prediction model using the same K examples that are used for adaptation. The labels for these examples are used in the meta-training update but not the adaptation itself. Thus, the adaptation matches the ARM setting at test time, in which h adapts g on the same unlabeled test points that the adapted prediction model then predicts on.

5.2. Optimizing the ARM objective

Algorithm 1 Meta-Learning for ARM

```
// Training procedure
```
Require: # training steps T, batch size K, learning rate η

1: **Initialize:** θ, ϕ
2: **for** $t = 1, \dots, T$ **do**
3: Sample z uniformly from training groups
4: Sample $(\mathbf{x}_k, y_k) \sim p(\cdot, \cdot|z)$ for $k = 1, \dots, K$
5: $\theta' \leftarrow h(\theta, \mathbf{x}_1, \dots, \mathbf{x}_K; \phi)$
6: $(\theta, \phi) \leftarrow (\theta, \phi) - \eta \nabla_{(\theta,\phi)} \sum_{k=1}^{K} \ell(g(\mathbf{x}_k; \theta'), y_k)$

```
// Test time adaptation procedure
```
Require: θ, ϕ, test batch $\mathbf{x}_1, \dots, \mathbf{x}_K$

7: $\theta' \leftarrow h(\theta, \mathbf{x}_1, \dots, \mathbf{x}_K; \phi)$
8: $\hat{y}_k \leftarrow g(\mathbf{x}_k; \theta')$ for $k = 1, \dots, K$

Algorithm 1 presents a general meta-learning approach for optimizing the ARM objective. As described above, h outputs updated parameters θ' using an unlabeled batch of data (line 5). We assume that h is differentiable with respect to θ and ϕ, thus we can use gradient updates to meta-train both θ and ϕ for *post adaptation* performance on a mini batch of data sampled according to a particular group z (line 6). This adaptation is performed using unlabeled data, mimicking the test time procedure (lines 7-8). In practice, we sample mini batches of groups rather than just one group (line 3), to provide a better gradient signal for optimizing ϕ and θ.

Together, Equation 1 and Algorithm 1 shed light on a number of ways to devise methods for solving the ARM problem. First, we can extend meta-learning paradigms to the ARM problem setting, and any paradigm in which the adaptation model h can be augmented to operate on unlabeled data is readily applicable. As an example, we propose the ARM-CML method, which is inspired by recent works in contextual meta-learning (Garnelo et al., 2018; Requeima et al., 2019). Second, we can enhance prior unlabeled adaptation methods by incorporating a meta-training phase that allows the model to better leverage the adaptation. To this end, we propose the ARM-BN method, based on the general approach of adapting using statistics of the test inputs (Li et al., 2017b; Schneider et al., 2020; Kaku et al., 2020; Nado et al., 2020). Third, we can incorporate existing methods for meta-learning unlabeled adaptation to solve group distribution shift problems. We demonstrate this by proposing the ARM-LL method, which is based on the method from Yu et al. (2018) for robotic imitation learning. In Appendix A, we provide complete details for these ARM methods.

ARM-CML. In ARM-CML, the parameters ϕ of h define the weights of a *context network*, which processes each example $\mathbf{x}_k$ separately to produce vectors $\mathbf{c}_k \in \mathbb{R}^D$ for $k = 1, \dots, K$, where D is a hyperparameter. This is different from prior contextual meta-learning approaches, in which the context network processes both inputs and outputs to produce each $\mathbf{c}_k$. The average vector $\mathbf{c} \triangleq \frac{1}{K} \sum_{k=1}^{K} \mathbf{c}_k$ is then passed to g as an additional input. In this setup, h can learn to provide useful information about the entire batch of K unlabeled points to g for predicting the correct outputs.

ARM-BN. ARM-BN is a particularly simple method that is applicable for any model g that is parameterized by a deep neural network with batch normalization (BN) layers (Ioffe & Szegedy, 2015). Practically, training g via ARM-BN follows the same protocol as Ioffe & Szegedy (2015) except for two key differences: first, the training batches are sampled from a single group, rather than from the entire dataset, and second, the normalization statistics are recomputed at test time rather than using a training running average. As noted, this second difference has been explored by several works as a method for test time adaptation, but the first difference is novel to ARM-BN. Following Algorithm 1, ARM-BN defines a meta-training procedure, in which g learns to adapt – i.e., compute normalization statistics – using batches of training points sampled from the same group. We empirically show in Section 6 that, for problems where BN adaptation already has a favorable inductive bias, such as for image classification, further introducing meta-training boosts its performance. We believe that other test time adaptation methods, such as those based on optimizing surrogate losses (Sun et al., 2020; Wang et al., 2021), may similarly benefit from their corresponding meta-training procedure.

ARM-LL. ARM-LL follows the gradient based meta-learning paradigm (Finn et al., 2017) and learns parameters θ that are amenable to gradient updates on a loss function in order to quickly adapt to a new problem. In other words, h produces $\theta' = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K; \phi)$, where α is a hyperparameter. Note that the loss function $\mathcal{L}$ used in the gradient updates is different from the original supervised loss function ℓ, in that it operates on only the inputs $\mathbf{x}$, rather than the input output pairs that ℓ receives. We follow the general implementation of Yu et al. (2018), in that the adaptation model parameters ϕ define the weights of a *loss network*, which takes in the output of g and produces a scalar, and $\mathcal{L}$ is defined as the ℓ_2-norm of these scalar outputs across the batch of inputs. We defer full details of ARM-LL to Appendix A and Yu et al. (2018).

6. Experiments

Our experiments are designed to answer the following:

1. Do ARM methods learn models that can leverage unlabeled adaptation to tackle group distribution shift?
2. How do ARM methods compare to prior methods for robustness, invariance, and adaptation?
3. Can ARM methods succeed in the streaming test setting?

6.1. Evaluation domains and protocol

We evaluate on four image classification problems, which we present below and describe in full detail in Appendix B. We choose these testbeds due to the potential for adaptation to improve test performance, and this motivation differs from prior benchmarks such as those compiled by DomainBed (Gulrajani & Lopez-Paz, 2021). In Appendix C, we compare our testbeds to DomainBed and other group robustness benchmarks, and we briefly discuss the ARM results in Gulrajani & Lopez-Paz (2021).

Rotated MNIST. We study a modified version of MNIST where images are rotated in 10 degree increments, from 0 to 130 degrees. We use only 108 training data points for each of the 2 smallest groups (120 and 130 degrees), and 324 points each for rotations 90 to 110, whereas the overall training set contains 32292 points. In this setting, we hypothesize that adaptation can specialize the model to specific groups, in particular the *rare groups* in the training set. For each test evaluation, we generate images from the MNIST test set with a certain rotation. Since we compare against methods for robustness, we measure both worst case and average accuracy across groups.

Federated Extended MNIST (FEMNIST). The extended MNIST (EMNIST) dataset (Cohen et al., 2017) consists of images of handwritten uppercase and lowercase letters, in addition to digits. FEMNIST (Caldas et al., 2019) is a version of EMNIST that associates each handwritten character with its user. We measure each method's worst case and average accuracy across 35 test users, which are held out and thus *disjoint* from the training users. As discussed in Section 2, adaptation may help for this problem for specializing the model and resolving ambiguous data points.

Corrupted image datasets. CIFAR-10-C and Tiny ImageNet-C (Hendrycks & Dietterich, 2019) augment the CIFAR-10 (Krizhevsky, 2009) and Tiny ImageNet test sets, respectively, with common image corruptions that vary in type and severity. The goal of Hendrycks & Dietterich (2019) is to benchmark robustness to these corruptions without seeing *any* corruptions during training, thus successful methods for this problem typically rely on heuristics designed specifically for image classification. For example, prior work has shown that carefully designed test time adaptation procedures are effective for dealing with corruptions (Sun et al., 2020; Schneider et al., 2020; Wang et al., 2021). One possible reason for this phenomenon is that convolutional networks typically rely on texture (Geirhos et al., 2019), which is distorted by corruptions, thus adaptation can help the model to specialize to each corruption type.

We study whether meta-training for adaptation performance can further improve upon these results. To do so, we modify the protocol from Hendrycks & Dietterich (2019) to fit into the ARM problem setting by using a set of 56 corruptions for the training data, and we define each corruption to be a group. We use a disjoint set of 22 test corruptions that are mostly of different types from the training corruptions (and thus not sampled identically), and we measure worst case and average accuracy across the test corruptions. This modification allows us to study, for both ARM and prior methods, whether seeing corruptions at training time can help the model deal with new corruptions at test time.

6.2. Comparisons and ablations

We compare the ARM methods against several prior methods designed for robustness, invariance, and adaptation. We describe the comparisons here and in Appendix B.

Group robustness and invariance. Sagawa et al. (2020) recently proposed a state-of-the-art method for group robustness, and we refer to this approach as distributionally robust neural networks (DRNN). Their work also evaluates a strong upweighting (UW) baseline that samples uniformly from each group, and so we also evaluate this approach in our experiments. Additionally, we compare to domain adversarial neural networks (DANN) (Ganin & Lempitsky, 2015) and maximum mean discrepancy (MMD) feature learning (Li et al., 2018b), two state-of-the-art methods for adversarial learning of invariant predictive features.

Test time adaptation. We evaluate the general approach of using test batches to compute BN statistics (Li et al., 2017b;

	MNIST		FEMNIST		CIFAR-10-C		Tiny ImageNet-C	
Method	**WC**	**Avg**	**WC**	**Avg**	**WC**	**Avg**	**WC**	**Avg**
ERM	74.3 ± 1.7	93.6 ± 0.4	62.9 ± 1.9	80.1 ± 0.9	49.6 ± 0.1	69.8 ± 0.4	19.3 ± 0.5	41.4 ± 0.2
UW*	80.2 ± 0.1	94.8 ± 0.2	61.8 ± 0.9	80.1 ± 0.3	—	—	—	—
DRNN	79.3 ± 1.1	94.8 ± 0.1	58.1 ± 0.7	74.4 ± 0.8	44.5 ± 0.5	70.7 ± 0.6	19.9 ± 0.3	41.6 ± 0.2
DANN	80.3 ± 1.4	95.1 ± 0.1	66.5 ± 0.4	82.3 ± 0.4	42.4 ± 0.2	69.1 ± 0.4	20.5 ± 0.1	41.9 ± 0.2
MMD	82.8 ± 2.0	95.6 ± 0.3	65.3 ± 1.6	81.6 ± 0.6	43.2 ± 0.4	70.4 ± 0.3	20.2 ± 0.0	39.9 ± 0.5
BN adaptation	75.1 ± 0.2	93.9 ± 0.1	66.9 ± 0.8	81.1 ± 0.3	62.5 ± 0.2	79.4 ± 0.3	23.9 ± 0.2	42.8 ± 0.2
TTT	81.1 ± 0.3	95.4 ± 0.1	64.1 ± 0.2	83.4 ± 0.1	66.6 ± 0.6	75.6 ± 0.8	19.7 ± 0.4	41.4 ± 0.3
CML ablation	78.2 ± 0.6	94.2 ± 0.1	64.4 ± 0.7	81.5 ± 0.7	47.8 ± 0.1	68.2 ± 0.1	19.6 ± 0.4	42.3 ± 0.2
LL ablation	82.4 ± 0.3	94.8 ± 0.2	61.9 ± 0.2	79.3 ± 0.6	61.5 ± 0.2	68.3 ± 0.5	25.8 ± 0.4	41.7 ± 0.1
ARM-CML	$\mathbf{88.7 \pm 0.6}$	$\mathbf{96.7 \pm 0.1}$	67.8 ± 1.3	$\mathbf{85.7 \pm 0.3}$	67.7 ± 0.5	79.2 ± 0.3	21.4 ± 0.2	43.3 ± 0.4
ARM-BN	82.8 ± 0.4	95.3 ± 0.1	$\mathbf{72.6 \pm 0.3}$	$\mathbf{85.7 \pm 0.1}$	$\mathbf{71.1 \pm 0.1}$	$\mathbf{80.9 \pm 0.2}$	$\mathbf{27.7 \pm 0.2}$	$\mathbf{44.9 \pm 0.2}$
ARM-LL	87.2 ± 0.5	96.3 ± 0.2	69.6 ± 2.1	$\mathbf{85.6 \pm 0.5}$	66.9 ± 0.2	75.7 ± 0.3	27.1 ± 0.3	44.2 ± 0.4

Table 1. Worst case (WC) and average (Avg) top 1 accuracy on rotated MNIST, FEMNIST, CIFAR-10-C, and Tiny ImageNet-C, where means and standard errors are reported across three separate runs of each method. Vertical lines separate methods that make use of (from top to bottom): neither, training groups, test batches, or both. ARM methods consistently achieve greater robustness, measured by WC, and Avg performance compared to prior methods. *UW is identical to ERM for CIFAR-10-C and Tiny ImageNet-C.

Schneider et al., 2020; Kaku et al., 2020; Nado et al., 2020), which we term BN adaptation. We also compare to test time training (TTT) (Sun et al., 2020), which adapts the model at test time using a self-supervised rotation prediction loss. These methods have previously achieved strong results for image classification, likely because adaptation procedures such as batch normalizing activations in convolution layers and predicting image rotations constitute favorable inductive biases for this domain. In other words, these procedures have been carefully designed to correlate closely with improving the true loss (Sun et al., 2020).

Robustness and invariance methods assume access to training groups but not test batches, whereas adaptation methods assume the opposite. Thus, at a high level, we can view the comparisons to these methods as evaluating the importance of each of these assumptions for the specified problems.

Ablations. We also include ablations of the ARM-CML and ARM-LL methods, which sample training batches of unlabeled examples uniformly rather than sampling from a single group. These "context ablation" and "learned loss ablation" are similar to test time adaptation methods in that they do not require training groups. However, these methods lack the inductive bias of BN adaptation and TTT, as they instead use learned context and loss networks.

6.3. Quantitative evaluation and comparisons

The results are presented in Table 1. From these results, we highlight several key takeaways:

ARM methods improve robustness and performance. Across all of our experiments, ARM methods consistently increase both worst case and average accuracy compared to all other methods. ARM-BN in particular achieves the best performance on most domains, demonstrating the effectiveness of training the model to make even greater use of an already strong adaptation procedure. The exception to this is the rotated MNIST experiment, where ARM-CML and ARM-LL outperform ARM-BN. This testbed represents a less realistic shift compared to the user and corruption shifts in the other experiments, thus BN adaptation itself seems to provide less benefit. ARM-CML and ARM-LL instead use learned adaptation procedures and therefore are able to generally improve upon the other methods for almost all metrics. We suspect that these more expressive methods could perform better than ARM-BN for larger datasets or other modalities such as video sequences (Yu et al., 2018).

Robust optimization methods suffer from pessimism and training difficulties. DRNN generally results in worse average case and, surprisingly, worst case performance, which may be due to the differences between our problems and prior group DRO settings, as we discuss in Appendix C. In particular, methods such as DRNN were originally evaluated in settings where the training and test groups were semantically the same (Sagawa et al., 2020), such as our rotated MNIST setup, whereas our FEMNIST setup tests on held out users and our CIFAR-10-C and Tiny ImageNet-C setups test on held out corruptions. For FEMNIST, we also test q-FedAvg (Li et al., 2020), a state-of-the-art method for fair federated learning. q-FedAvg was also originally evaluated with the same users at training and test time, and in our setup, this method also performs poorly, achieving 58.2 ± 1.0 worst case and 80.8 ± 0.3 average accuracy.

In these experiments, we see that DANN and MMD mostly outperform robustness methods, though the overall performance of these methods for learning invariant features across groups is still worse than ARM methods. Invariance methods have primarily been tested in settings with an order of magnitude fewer source domains – usually 4 to 6, rather

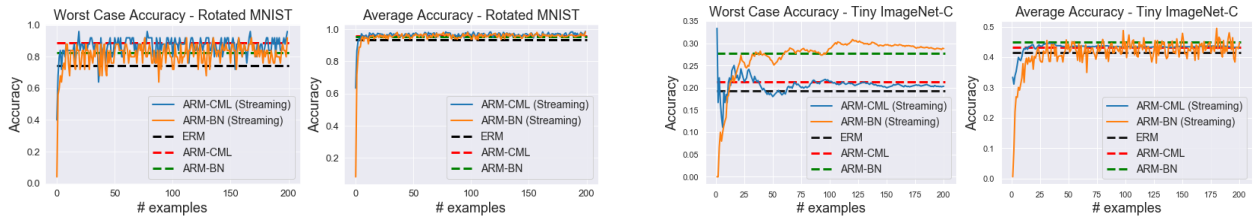

Figure 3. In the streaming setting, ARM methods reach strong performance on rotated MNIST (left) and Tiny ImageNet-C (right), after fewer than 10 and 25 data points, respectively, despite meta-training with batch sizes of 50 for both domains. This highlights that the trained models are able to adapt with small test batches and can operate successfully in the standard streaming evaluation setting.

than tens or hundreds – and more data per domain (Gulrajani & Lopez-Paz, 2021), and employing techniques such as an adversarial domain classifier may be less effective in general as the number of domains increases.

With an appropriate inductive bias, test time adaptation methods perform well. In our evaluation, one of the strongest prior methods is the simple BN adaptation method. This method performs well across all metrics, though it typically still lags behind ARM methods and ARM-BN in particular. As discussed above, we believe that this adaptation procedure performs well as it constitutes an inductive bias that is well suited for image classification. TTT offers additional support for this hypothesis: this method works notably well for rotated MNIST, possibly because the inductive bias associated with the auxiliary task of rotating images specifically robustifies the classifier to rotation shift. ARM methods are comparatively less reliant on a favorable inductive bias and thus consistently attain better results.

In summary, in our experiments, we observe poor performance from robustness methods, varying performance from invariance and adaptation methods, and the strongest performance from ARM methods. As ARM methods also make the strongest assumptions, we next present some findings on how the test batch assumption may be loosened.

6.4. Loosening the test batch assumption

We investigate the effectiveness of ARM methods in the streaming test time setting, where test points are observed one at a time rather than in batches. In Appendix D, we also study whether the training group assumption can be loosened, by instead using unsupervised learning techniques to discover group structure in the training data.

When we cannot access a batch of test points all at once, we can augment the proposed ARM methods to be sequential. For example, ARM-CML and ARM-BN can update their average context and normalization statistics, respectively, after observing each new test point. In Figure 3, we visualize the performance of both of these methods, using models that were meta-trained with batch sizes of 50 but evaluated in this streaming setting. We see that both ARM-CML and ARM-BN are able to achieve near their original worst case and average accuracy within observing 10 data points, for rotated MNIST, and 25 data points, for Tiny ImageNet-C, well before the training batch size of 50. This result demonstrates that ARM methods are applicable for problems where test points must be observed one at a time, provided that the model is permitted to adapt using each point. We describe in detail how each ARM method can be applied to the streaming setting in Appendix A.

7. Discussion and Future Work

We presented adaptive risk minimization (ARM), a framework and problem formulation for learning models that can adapt in the face of group distribution shift at test time using only a batch of unlabeled test examples. We devised an algorithm and instantiated a set of methods for optimizing the ARM objective that meta-learns models that are adaptable to different groups of training data. Empirically, we observed that ARM methods consistently improve performance in terms of both average and worst case metrics, as compared to a number of prior approaches for handling group shift.

Though we provided contextual meta-learning as a concrete example, a number of other meta-learning paradigms would also be interesting to extend to the ARM setting. For example, few shot generative modeling objectives would be a natural fit for unlabeled adaptation (Edwards & Storkey, 2017; Hewitt et al., 2018; Wu et al., 2019). Another exciting direction for future work is to further explore the problem setting where groups are not provided at training time. As discussed in Appendix D, in this setting, we can instead construct groups via unsupervised learning techniques. Similar to Hsu et al. (2019), one promising approach is to generate a diverse set of groupings in order to learn generally effective adaptation strategies. Robustness and invariance methods cannot be used easily with multiple different groupings, learned or otherwise, as techniques such as group weighted loss functions (Sagawa et al., 2020) and group classifiers (Ganin & Lempitsky, 2015) are not immediately extendable to this setup. Thus, ARM methods may be uniquely suited to be paired with unsupervised group learning.

Acknowledgements.

MZ thanks Matt Johnson and Sharad Vikram for helpful discussions and is supported by an NDSEG fellowship. HM is funded by a scholarship from the Dr. Tech. Marcus Wallenberg Foundation for Education in International Industrial Entrepreneurship. AG is supported by an NSF graduate research fellowship. CF is a CIFAR Fellow in the Learning in Machines and Brains program. This research was supported by the DARPA Assured Autonomy and Learning with Less Labels programs.

References

Alet, F., Kawaguchi, K., Bauza, M., Kuru, N., Lozano-Pérez, T., and Kaelbling, L. Tailoring: Encoding inductive biases by optimizing unsupervised objectives at prediction time. *arXiv preprint arXiv:2009.10623*, 2020.

Antoniou, A. and Storkey, A. Learning to learn via self-critique. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Bengio, S., Bengio, Y., Cloutier, J., and Gecsei, J. On the optimization of a synaptic learning rule. In *Optimality in Artificial and Biological Neural Networks*, 1992.

Blanchard, G., Lee, G., and Scott, C. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

Blanchard, G., Deshmukh, A., Dogan, U., Lee, G., and Scott, C. Domain generalization by marginal transfer learning. *Journal of Machine Learning Research (JMLR)*, 2021.

Caldas, S., Duddu, S., Wu, P., Li, T., Konečný, J., McMahan, H., Smith, V., and Talwalkar, A. LEAF: A benchmark for federated settings. In *Workshop on Federated Learning for Data Privacy and Confidentiality*, 2019.

Cohen, G., Afshar, S., Tapson, J., and van Schaik, A. EMNIST: An extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.

Csurka, G. Domain adaptation for visual applications: A comprehensive survey. *arXiv preprint arXiv:1702.05374*, 2017.

Daumé III, H. Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, 2007.

Dou, Q., Castro, D., Kamnitsas, K., and Glocker, B. Domain generalization via model-agnostic learning of semantic features. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Edwards, H. and Storkey, A. Towards a neural statistician. In *International Conference on Learning Representations (ICLR)*, 2017.

Fang, C., Xu, Y., and Rockmore, D. Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Ganin, Y. and Lempitsky, V. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning (ICML)*, 2015.

Garnelo, M., Rosenbaum, D., Maddison, C., Ramalho, T., Saxton, D., Shanahan, M., Teh, Y., Rezende, D., and Eslami, S. Conditional neural processes. In *International Conference on Machine Learning (ICML)*, 2018.

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F., and Brendel, W. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

Gong, B., Shi, Y., Sha, F., and Grauman, K. Geodesic flow kernel for unsupervised domain adaptation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. In *International Conference on Learning Representations (ICLR)*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

Hewitt, L., Nye, M., Gane, A., Jaakkola, T., and Tenenbaum, J. The variational homoencoder: Learning to learn high capacity generative models from few examples. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2018.

Hochreiter, S., Younger, A., and Conwell, P. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks (ICANN)*, 2001.

Hsu, K., Levine, S., and Finn, C. Unsupervised learning via meta-learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Hu, S., Moreno, P., Xiao, Y., Shen, X., Obozinski, G., Lawrence, N., and Damianou, A. Empirical Bayes transductive meta-learning with synthetic gradients. In *International Conference on Learning Representations (ICLR)*, 2020.

Hu, W., Niu, G., Sato, I., and Sugiyama, M. Does distributionally robust supervised learning give robust classifiers? In *International Conference on Machine Learning (ICML)*, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, 2015.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with Gumbel-softmax. In *International Conference on Learning Representations (ICLR)*, 2017.

Kaku, A., Mohan, S., Parnandi, A., Schambra, H., and Fernandez-Granda, C. Be like water: Robustness to extraneous variables via adaptive feature normalization. *arXiv preprint arXiv:2002.04019*, 2020.

Kingma, D. and Welling, M. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Lazer, D., Kennedy, R., King, G., and Vespignani, A. The parable of Google flu: Traps in big data analysis. *Science*, 2014.

Li, D., Yang, Y., Song, Y., and Hospedales, T. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision (ICCV)*, 2017a.

Li, D., Yang, Y., Song, Y., and Hospedales, T. Learning to generalize: Meta-learning for domain generalization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2018a.

Li, H., Pan, S., Wang, S., and Kot, A. Domain generalization with adversarial feature learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018b.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Li, X., Sun, Q., Liu, Y., Zhou, Q., Zheng, S., Chua, T., and Schiele, B. Learning to self-train for semi-supervised few-shot classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Li, Y., Wang, N., Shi, J., Liu, J., and Hou, X. Revisiting batch normalization for practical domain adaptation. In *International Conference on Learning Representations Workshop (ICLRW)*, 2017b.

Li, Y., Tian, X., Gong, M., Liu, Y., Liu, T., Zhang, K., and Tao, D. Deep domain generalization via conditional invariant adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2018c.

Lipton, Z., Wang, Y., and Smola, A. Detecting and correcting for label shift with black box predictors. In *International Conference on Machine Learning (ICML)*, 2018.

Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S., and Yang, Y. Learning to propagate labels: Transductive propagation network for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Maddison, C., Mnih, A., and Teh, Y. The Concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

Metz, L., Maheswaranathan, N., Cheung, B., and Sohl-Dickstein, J. Meta-learning update rules for unsupervised representation learning. In *International Conference on Learning Representations (ICLR)*, 2019.

Nado, Z., Padhy, S., Sculley, D., D'Amour, A., Lakshminarayanan, B., and Snoek, J. Evaluating prediction-time batch normalization for robustness under covariate shift. *arXiv preprint arXiv:2006.10963*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

Quiñonero Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J., Larochelle, H., and Zemel, R. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2018.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Rezende, D., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

Royer, A. and Lampert, C. Classifier adaptation at prediction time. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

Sagawa, S., Koh, P., Hashimoto, T., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*, 2020.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. Meta-learning with memory-augmented neural networks. In *International Conference on Machine Learning (ICML)*, 2016.

Schmidhuber, J. Evolutionary principles in self-referential learning. *Diploma thesis, Institut f. Informatik, Tech. Univ. Munich*, 1987.

Schneider, S., Rusak, E., Eck, L., Bringmann, O., Brendel, W., and Bethge, M. Improving robustness against common corruptions by covariate shift adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference (JSPI)*, 2000.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Sulc, M. and Matas, J. Improving CNN classifiers by estimating test-time priors. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning (ICML)*, 2020.

Thrun, S. and Pratt, L. *Learning to Learn*. Springer Science & Business Media, 1998.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Vapnik, V. *Statistical Learning Theory*. Wiley New York, 1998.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, K., and Wierstra, D. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Wang, D., Shelhamer, E., Liu, S., Olshausen, B., and Darrell, T. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Wilson, G. and Cook, D. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2020.

Wu, M., Choi, K., Goodman, N., and Ermon, S. Meta-amortized variational inference and learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Yu, T., Finn, C., Xie, A., Dasari, S., Abbeel, P., and Levine, S. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Robotics: Science and Systems (RSS)*, 2018.

Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., and Song, Y. MetaGAN: An adversarial approach to few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

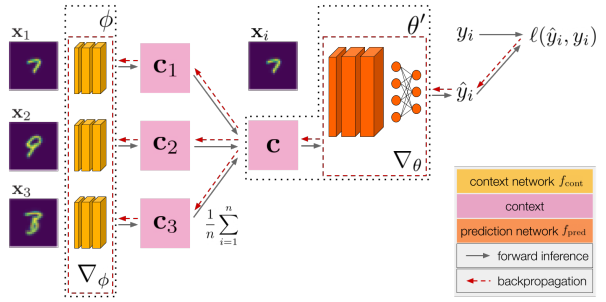

Figure 4. During inference for ARM-CML, the context network produces a vector $\mathbf{c}_k$ for each input image $\mathbf{x}_k$ in the batch, and the average of these vectors is used as the context $\mathbf{c}$ is input to the prediction network. This context may adapt the model by providing helpful 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 to update θ and ϕ.

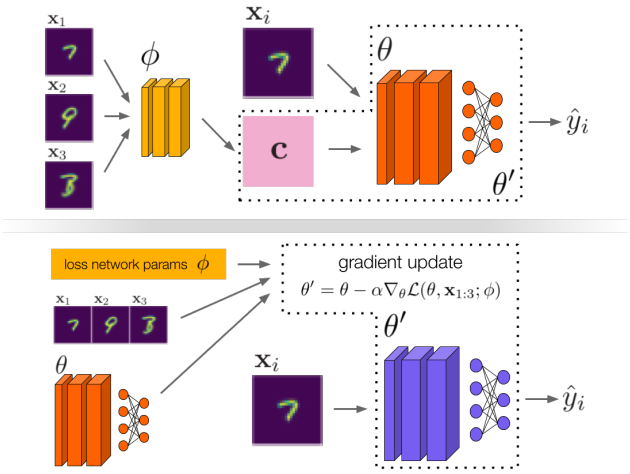

Figure 5. Schematics of the two broad classes of approaches we consider. In the contextual approach (top), $\mathbf{x}_1, \ldots, \mathbf{x}_K$ are summarized into a context $\mathbf{c}$, and we propose two methods for this summarization, either through a separate context network or using batch normalization activations in the model itself. $\mathbf{c}$ can then be used by the model to infer additional information about the input distribution. In the gradient based approach (bottom), an unlabeled loss function $\mathcal{L}$ is used for gradient updates to the model parameters, in order to produce parameters that are specialized to the test inputs and can produce more accurate predictions.

A. Detailed Descriptions of ARM Methods

For ARM-CML, we introduce two neural networks: a context network $\text{f}_{\text{cont}}(\cdot\,;\phi) : \mathcal{X} \to \mathbb{R}^D$, parameterized by the adaptation model parameters ϕ, and a *prediction network* $\text{f}_{\text{pred}}(\cdot\,,\cdot\,;\theta) : \mathcal{X} \times \mathbb{R}^D \to \mathcal{Y}$, parameterized by θ. As discussed, f_{cont} processes each example $\mathbf{x}_k$ in the mini batch separately to produce $\mathbf{c}_k \in \mathbb{R}^D$ 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.

A schematic of this method is presented in Figure 4. The post adaptation model parameters θ' are $[\theta, \mathbf{c}]$. Since we only ever use the model after adaptation, both during training and at test time, we can simply specify $g(\mathbf{x}; \theta') = \text{f}_{\text{pred}}(\mathbf{x}, \mathbf{c}; \theta)$. Though not strictly necessary, we could define the behavior of g before adaptation, i.e., with unadapted parameters θ, as using a running average of the context computed throughout the course of training, similar to BN. We then also see that h is a function that takes in $(\theta, \mathbf{x}_1, \ldots, \mathbf{x}_K)$ and produces $\left[\theta, \frac{1}{K}\sum_{k=1}^{K} \text{f}_{\text{cont}}(\mathbf{x}_k; \phi)\right]$. In the streaming setting, 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.[2] When we observe a new point $\mathbf{x}$, we increment the counter and update the average context as $\frac{t}{t+1}\mathbf{c} + \frac{1}{t+1}\text{f}_{\text{cont}}(\mathbf{x}; \phi)$, 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, shown in Figure 5 (top). The interpretation of ARM-BN through the contextual approach is that the running statistics used by standard BN are learned parameters of g, and h replaces these parameters with statistics computed on the batch of inputs, which then serves as the context $\mathbf{c}$. Thus, 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 BN layers. BN 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 defines h such that it swaps out these moments for the moments computed via the activations on the test batch, thus giving us adapted parameters θ'. This method is remarkably simple, and in deep learning libraries such as PyTorch (Paszke et al., 2019), the implementation requires the changing of a single line of code. However, as shown in Section 6, this method also performs very well empirically, and it is further boosted by meta-training.

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

In the streaming setting, ARM-BN is similar to ARM-CML, though slightly more complex due to the requirement of computing second moments. Denote the context after seeing t test points as $\mathbf{c} = [\boldsymbol{\mu}, \boldsymbol{\sigma}^2]$, the mean and variance of the BN layer activations on the points so far. Upon seeing a new test point, let $\mathbf{a}$ denote the BN layer activations computed from this new point, with size h. We then update $\mathbf{c}$ to be

$$\left[\frac{ht}{h(t+1)}\boldsymbol{\mu} + \frac{\sum \mathbf{a}}{h(t+1)}, \frac{ht}{h(t+1)}(\boldsymbol{\sigma}^2 + \boldsymbol{\mu}^2) + \frac{\sum \mathbf{a}^2}{h(t+1)} - \left(\frac{ht}{h(t+1)}\boldsymbol{\mu} + \frac{\sum \mathbf{a}}{h(t+1)}\right)^2\right].$$

Again note that we do not store any test points and that we arrive at the same context as the batch test setting after observing K data points.

Finally, ARM-LL is depicted in Figure 5 (bottom). We define g to produce output features $\mathbf{o} \in \mathbb{R}^{|\mathcal{Y}|}$ that are used as logits when making predictions. We then define the unlabeled loss function $\mathcal{L}$ to be the composition of g and a *loss network* $\mathrm{f}_{\text{loss}}(\cdot\,; \phi) : \mathbb{R}^{|\mathcal{Y}|} \to \mathbb{R}$, which takes in the output features from g and produces a scalar. Similar to f_{cont} in ARM-CML, f_{loss} is parameterized by ϕ, as h has no additional parameters. As we desire that the loss is bounded below, we use the ℓ_2-norm of these scalars across the batch of inputs is used as the loss for updating θ. In other words,

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

We found that $\alpha = 0.1$ worked well for our experiments. We used 1 gradient step for both meta-training and meta-testing. Finally, though we did not evaluate ARM-LL in the streaming setting, in principle this method can be extended to this setting by performing a gradient step with a smaller α after observing each test point. In an online fashion, similar to Sun et al. (2020), we can continually update the model parameters over the course of testing rather than initializing from the meta-learned parameters for each test point.

B. 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 $\sqrt{3}$. 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. All hyperparameter settings were selected in preliminary experiments using validation accuracy only.

We also provide details for how we constructed the training, validation, and test splits for each dataset. These splits were designed without any consideration for the train, validation, and test accuracies of any method. All of these design choices were made either intuitively – such as maintaining the original data splits for MNIST – or randomly – such as which users were selected for which splits in FEMNIST – or with a benign alternate purpose – such as choosing disjoint sets of corruptions with mostly different types.

B.1. Rotated MNIST details

We construct a training set of 32292 data points using 90% of the original training set – separating out a validation set – by sampling and 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. We construct an additional level of mini batching for our method as described in Section 5, such that the batch dimensions of the data mini batches are 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. DRNN uses an additional learning rate hyperparameter for their robust loss, which we set to 0.01 across all experiments (Sagawa et al., 2020).

We compute validation accuracy every 10 epochs. We estimate validation 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 all methods that do not use a context network, the network has four convolution layers with 128 5×5 filters, 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 BN (Ioffe & Szegedy, 2015) 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 ARM-CML and the 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, BN, and padding, followed by a final convolution layer with 12 5×5 filters with padding.

B.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") (Cohen et al., 2017). In processing the FEMNIST dataset,[3] we filter out users with fewer than 100 examples, leaving 262, 50, and 35 unique users and a total of 62732, 8484, and 8439 data points in the training, validation, and test splits, respectively. The smallest users in each split contain 104, 119, and 140 data points, respectively. We keep all hyperparameters the same as for rotated MNIST, except we set the meta batch size for ARM methods to be 2.

We additionally compare to q-FedAvg on this domain, as this method is specifically designed for federated learning settings (Li et al., 2020). We modify the authors' publicly available code[4] to run experiments in our setting. 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. (2020). 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. Though this may affect ARM methods, we demonstrate in Section 6 that ARM-CML and ARM-BN can adapt using batch sizes much smaller than 50, such as in the streaming test setting. The network architectures are the same as the architectures used for rotated MNIST, except that, when applicable, the last layer of the context network has only 1 filter of size 5×5..

B.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. In this way, we are constructing a more challenging test setting, in which the test groups are not sampled identically as the training groups, since the corruption types are largely different between the two 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 datasets.

In these experiments, we train ResNet-50 (He et al., 2016) models with a support size of 50 and meta batch size of 6. 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 accuracy across groups.

C. Contrasting with Prior Benchmarks

One aim of this work is to identify problems for which unlabeled adaptation is feasible, helpful, and potentially crucial, inspired by important real world problem settings such as federated learning. Thus, the problems we focus on will naturally differ from prior work in group distribution shift, which have different implicit and explicit goals when designing and choosing benchmarks. In particular, as discussed in Section 1, many prior benchmarks assume the existence of a consistent input-output relationship across groups, for which various methods can be designed to try and better uncover this relationship. Compared to problems where adaptation is important, we can roughly characterize these benchmarks as having a conditional distribution $p(y|\mathbf{x})$ that is more stable across groups and thus does not depend as much on the marginal $p(\mathbf{x})$. As Blanchard et al. (2021) note informally, and as mentioned in Section 4, the less information the marginal provides about the conditional, the less we expect domain generalization strategies to improve over ERM. Indeed, Gulrajani & Lopez-Paz (2021) provide a comprehensive survey of domain generalization benchmarks and find that, though ERM is very occasionally outperformed by

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

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

certain methods on certain benchmarks, ERM is competitive with the state of the art on average across the benchmarks.

Gulrajani & Lopez-Paz (2021) evaluate ARM-CML across the whole suite and find that, interestingly, ARM-CML handily outperforms ERM and all prior methods on the colored MNIST benchmark originally introduced by Arjovsky et al. (2019). This benchmark modifies MNIST, first by introducing color and then by splitting the data into three groups. The task is binary classification, and the noisy relationship between the digit class and binary label is consistent across groups. The noisy relationship between the color and binary label varies across groups, but, importantly, the correlation between color and label is *stronger*. For a non adaptive model, the goal as originally proposed in Arjovsky et al. (2019) is to disregard color and learn the invariant relationship between digits and labels. Irrespective of the original motivations, though, an adaptive model is in theory capable of learning a more flexible classification strategy for this problem, in that it may leverage information about the current group in order to produce better predictions. This is similar to the toy example discussed in Section 2: ARM methods can use the training groups to learn an adaptive model, which then changes its decision rule based on unlabeled test data. Viewed this way, it becomes clear why ARM-CML can learn a more performant solution for the colored MNIST problem. This result provides further motivation for identifying and studying problems for which adaptation can be beneficial, alongside other benchmarks geared toward discovering invariances.

Specifically when viewing learning adaptation as a meta-learning problem, as in this work, we may pose additional hypotheses about a problem's desired properties. For example, in meta-learning, each task is viewed as a "higher level" data point, and this generally motivates constructing many different tasks so as to prevent the learner from overfitting to the tasks. We extend this intuition to our work in that our problems have tens to hundreds of groups, whereas the benchmarks in DomainBed have between 3 to 6 groups (and significantly more data per group). This is also true for prior testbeds used in group DRO (Sagawa et al., 2020), which further have a couple of important differences. First, as discussed in Section 6, group DRO testbeds typically use the same training and test groups and measure worst case performance, which differs from domain generalization, meta-learning, and most problems considered in this work, which construct or hold out disjoint sets of groups for testing. Second, prior group DRO testbeds use label information to construct groups, in that data within each group will all have the same label. This is not an issue for non adaptive models, however, classification in this setup becomes much easier for adaptive models and particularly if training with an ARM method, as the model simply needs to learn to adapt to output a constant label. Thus, in this work, we identify and set up problems which are distinct from both prior work in domain generalization and group DRO, in order to properly evaluate ARM and prior methods in settings for which adaptation is beneficial.

Method	WC	Avg
ERM	74.3 ± 1.7	93.6 ± 0.4
TTT	$\mathbf{81.1 \pm 0.3}$	$\mathbf{95.4 \pm 0.1}$
ARM-CML	$\mathbf{81.7 \pm 0.3}$	$\mathbf{95.2 \pm 0.3}$

Table 2. Using learned groups, ARM-CML outperforms ERM and matches the performance of TTT on rotated MNIST. This result may be improved by techniques for learning more diverse groups.

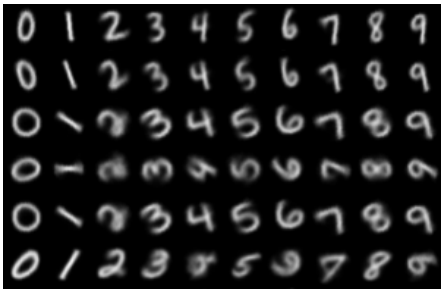

Figure 6. VAE samples conditioned on different values of y (x axis) and c (y axis). The VAE learns to use c to represent rotations.

D. Loosening the Training Group Assumption

In the case of unknown groups, one option is to use unsupervised learning techniques to discover group structure in the training data. To test this option, we focus on rotated MNIST and ARM-CML, which performs the best on this dataset, and train a variational autoencoder (VAE) (Kingma & Welling, 2014; Rezende et al., 2014) with discrete latent variables (Jang et al., 2017; Maddison et al., 2017) using the training images and labels. We define the latent variable, which we denote as c to differentiate from the group z, to be Categorical with 12 possible discrete values, which we purposefully choose to be smaller than the number of rotations. The VAE is not given any information about the ground truth z; however, we weakly encode the notion that c is independent of y by conditioning the decoder on the label. We use the VAE inference network to assign groups to the training data, and we run ARM-CML using these learned groups. In Table 2, we see that ARM-CML in this setting outperforms ERM and is competitive with TTT, which as discussed earlier encodes a strong inductive bias, via rotation prediction, for solving this task. Figure 6 visualizes samples from the VAE for different values of y and c, which shows that the VAE learns to encode rotation information using c. This result suggests that, when group information is not provided, a viable approach is to learn groups which then enables the use of ARM methods.