The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Learning to Generalize: Meta-Learning for Domain Generalization

Da Li, Yongxin Yang, Yi-Zhe Song
Queen Mary University of London
{da.li, yongxin.yang, yizhe.song}@qmul.ac.uk

Timothy M. Hospedales
The University of Edinburgh
t.hospedales@ed.ac.uk

Abstract

Domain shift refers to the well known problem that a model trained in one source domain performs poorly when applied to a target domain with different statistics. Domain Generalization (DG) techniques attempt to alleviate this issue by producing models which by design generalize well to novel testing domains. We propose a novel meta-learning method for domain generalization. Rather than designing a specific model that is robust to domain shift as in most previous DG work, we propose a model agnostic training procedure for DG. Our algorithm simulates train/test domain shift during training by synthesizing virtual testing domains within each mini-batch. The meta-optimization objective requires that steps to improve training domain performance should also improve testing domain performance. This meta-learning procedure trains models with good generalization ability to novel domains. We evaluate our method and achieve state of the art results on a recent cross-domain image classification benchmark, as well demonstrating its potential on two classic reinforcement learning tasks.

Introduction

Humans are adept at solving tasks under many different conditions. This is partly due to fast adaptation, but also to a lifetime of encountering new task conditions providing the opportunity to develop of strategies that are robust to different task contexts. If a human discovers that their existing strategy fails in a new context they do not just adapt, but further try to update their strategy to be more context independent, so that next time they arrive in a new context they are more likely to succeed immediately. We would like artificial learning agents to solve many tasks under different conditions (domains) and similarly solve the second order task of constructing models that are robust to change of domain and perform well 'out of the box' in new domains. For example we might like computer vision systems to recognize objects immediately and without retraining, when the camera type is changed (Patel et al. 2015), or reinforcement learning trained agents to perform well immediately when placed in a new environment or subjected to changed morphology (Taylor and Stone 2009) – without waiting for adaptation.

Standard learning approaches tend to break down when applied in different conditions (i.e. to data with different statistics) than used for training. This is known as domain or covariate shift (Storkey and Sugiyama 2007), and seriously affects the usefulness of machine learning models as we do not always have access to training data that is exactly representative of the intended testing scenario. Approaches to addressing this issue can be categorized into *domain adaptation* (DA) and *domain generalization* (DG). DA is relatively well studied, and addresses using unlabelled or sparsely labelled data in the target domain to quickly adapt a model trained in a different source domain (Patel et al. 2015; Csurka 2017). The less well studied DG addresses building models that by design function well even in new target/testing domains. In contrast to DA, a DG model is not updated after training, and the issue is how well it works out of the box in a new domain. The few existing DG methods typically train on multiple source domains and propose mechanisms to extract some domain agnostic representation or model that describes common aspects of known domains (Khosla et al. 2012; Muandet, Balduzzi, and Schölkopf 2013; Ghifary et al. 2015; Li et al. 2017). They assume that such a common factor among existing source domains will persist to new testing domains, and thus provide a basis for generalization. DG is a harder problem than DA in that it makes fewer assumptions (target data not required) but for the same reasons, it may be more valuable if solved.

We take a *meta learning* approach to DG. Rather than proposing a specific model suited for DG (Khosla et al. 2012; Ghifary et al. 2015; Li et al. 2017), we propose a model-agnostic training algorithm that trains any given model to be more robust to domain shift. This is related to the long standing idea of *learning to learn* (Thrun and Pratt 1998; Schmidhuber, Zhao, and Wiering 1997), which has recently seen a resurgence of popularity with applications to few-shot learning (Finn, Abbeel, and Levine 2017; Ravi and Larochelle 2017) and learning optimizers (Andrychowicz et al. 2016). The most related of these studies to ours is the MAML approach of (Finn, Abbeel, and Levine 2017). MAML takes a meta-learning approach to few-shot learning by training a single model on a set of source tasks that is only a few gradient descent steps away from a good task-specific model. This meta-optimization objective trains models suited for few-shot fine-tuning to new target tasks.

The DG problem is different because we to transfer across domains rather than tasks, and because DG assumes *zero*, rather than few training examples of the target problem.

Our meta-learning domain generalization approach (MLDG) provides a model agnostic training procedure that improves the domain generality of a base learner. Specifically, MLDG trains a base learner on a set of source domains by synthesizing virtual training and virtual testing domains within each mini-batch. The meta-optimization objective is then: to minimize the loss on the training domains, while also ensuring that the direction taken to achieve this also leads to an improvement in the (virtual) testing loss. We present analyses that give various perspectives on this strategy, including as following an optimization trajectory where the virtual training and virtual testing gradients are aligned. Overall our MLDG approach has several key benefits: As a meta-learning procedure, it does not introduce any new parameters, unlike other model-based DG approaches that grow parameters linearly in the number of source domains (Khosla et al. 2012; Ghifary et al. 2015; Li et al. 2017; Bousmalis et al. 2016) resulting in large numbers of total parameters. Similarly MLDG does not place any constraint on the architecture of the base learner and moreover can be applied to both supervised and reinforcement learning; where prior DG alternatives (Khosla et al. 2012; Ghifary et al. 2015; Li et al. 2017) both constrain the model architecture and address supervised learning.

To summarize our contributions: We develop a gradient-based meta-learning algorithm that trains models for improved domain generalization ability. Our algorithm can train any type of base network and applies to both supervised and reinforcement learning settings. We evaluate our approach on a very recent cross domain image recognition benchmark and achieve state of the art results, as well as demonstrating its promising applicability to two classic reinforcement learning tasks.

Related Work

Multi-Domain Learning (MDL) MDL addresses training a single model that is effective for multiple known domains (Daumé 2007; Yang and Hospedales 2015; Rebuff, Bilen, and Vedaldi 2017; Bilen and Vedaldi 2017; Zhao et al. 2017). Domain generalization often starts with MDL on some source domains but addresses training a model that generalizes well to held out unknown domains.

Domain Generalization Despite the variety of the different methodological tools, most existing DG methods are built on three main strategies. The simplest approach is to train a model for each source domain. When a testing domain comes, estimate the most relevant source domain and use that classifier (Xu et al. 2014). A second approach is to presume that any domain is composed of an underlying globally shared factor and a domain specific component. By factoring out the domain specific and domain-agnostic component during training on source domains, the domain-agnostic component can be extracted and transferred as a model that is likely to work on a new source domain (Khosla et al. 2012; Li et al. 2017). Finally, there is learning a domain-invariant feature representation. If a feature representation can be learned that minimizes the gap between multiple source domains, it should provide a domain independent representation that performs well on a new target domain. This has been achieved with multi-view autoencoders (Ghifary et al. 2015) and mean map embedding-based techniques (Muandet, Balduzzi, and Schölkopf 2013). It has also been achieved based on gradient reversal domain confusion losses in deep networks (Ganin and Lempitsky 2015; Bousmalis et al. 2016). Here multiple source domains are trained with an additional multi-task loss that prefers a shared representation for which domains are indistinguishable. Although initially proposed for DA rather than DG, these approaches can be adapted to the DG setting (Li et al. 2017). In contrast to these studies, ours is the first to addresses domain generalization via meta-learning.

Neural Network Meta-Learning Meta-learning for neural networks has a long history (Thrun and Pratt 1998; Schmidhuber, Zhao, and Wiering 1997), but have resurged in popularity recently. Recent meta-learning studies have focused on learning good weight initializations for few-shot learning (Finn, Abbeel, and Levine 2017; Parisotto, Ba, and Salakhutdinov 2016), meta-models that generate the parameters of other models (Vinyals et al. 2016; Li et al. 2017), or learning transferable optimizers (Ravi and Larochelle 2017; Andrychowicz et al. 2016). Our approach is most related to those that learn transferable weight initializations, notably MAML (Finn, Abbeel, and Levine 2017). In MAML a single shared source model shared is trained using multiple source tasks. The meta-learning process simulates transfer learning by fine-tuning, so the global model is updated to solve each source task in turn based on a few examples and a few gradient descent steps. By training the source model such that all simulated testing tasks fine-tune well, meta-learning produces a source model that is easy to adapt. Both MAML and our MLDG are model agnostic in that they apply to any base architecture and both supervised and to reinforcement learning settings. However, MAML addresses few-shot transfer to new tasks, rather than zero-shot transfer to new domains. In our case a different meta-learning objective is necessary because in DG we will not have access to target examples for fine-tuning during the actual testing. Therefore we propose a new meta-learning objective based around simulating domain shift and training such that steps to improve the source domain also improve the simulated testing domains.

Methodology

Meta-Learning Domain Generalization

In the DG setting, we assume there are S source domains $\mathcal{S}$ and T target domains $\mathcal{T}$. All of them contain the same task (same label space, and input feature space) but have different statistics. We define a single model parametrized as Θ to solve the specified task. DG aims for training Θ on the source domains, such that it generalizes to the target domains. To achieve this, at each learning iteration we split the original S source domains $\mathcal{S}$ into $S-V$ **meta-train** domains $\bar{\mathcal{S}}$ and V **meta-test** domains $\breve{\mathcal{S}}$ (virtual-test domain). This is to mimic real train-test domain-shifts so that over many iter-

Algorithm 1 Meta-Learning Domain Generalization

```
1: procedure MLDG
2:     Input: Domains $\mathcal{S}$
3:     Init: Model parameters $\Theta$. Hyperparameters $\alpha, \beta, \gamma$.
4:     for ite in iterations do
5:         Split: $\bar{\mathcal{S}}$ and $\breve{\mathcal{S}} \leftarrow \mathcal{S}$
6:         Meta-train: Gradients $\nabla_\Theta = \mathcal{F}'_\Theta(\bar{\mathcal{S}}; \Theta)$
7:         Updated parameters $\Theta' = \Theta - \alpha\nabla_\Theta$
8:         Meta-test: Loss is $\mathcal{G}(\breve{\mathcal{S}}; \Theta')$.
9:         Meta-optimization: Update $\Theta$
               $$\Theta = \Theta - \gamma\frac{\partial(\mathcal{F}(\bar{\mathcal{S}}; \Theta) + \beta\mathcal{G}(\breve{\mathcal{S}}; \Theta - \alpha\nabla_\Theta))}{\partial\Theta}$$
10:    end for
11: end procedure
```

Algorithm 2 MLDG for Reinforcement Learning

```
1: procedure MLDG-RL
2:     Input: Environment domains $\mathcal{S}$
3:     Init: Policy params $\Theta$, Hyperparameters $\alpha, \beta, \gamma$.
4:     for ite in iterations do
5:         Split: $\bar{\mathcal{S}}$ and $\breve{\mathcal{S}} \leftarrow \mathcal{S}$
6:         Meta-train:
7:         Collect trajectories $\bar{\tau}$ applying policy $\Theta$ in $\bar{\mathcal{S}}$.
8:         Loss: $\mathcal{F}(\bar{\tau}, \Theta)$.
9:         Gradient: $\nabla_\Theta = \mathcal{F}'_\Theta(\bar{\tau}, \Theta)$.
10:        Updated parameters: $\Theta' = \Theta - \alpha\nabla_\Theta$.
11:        Meta-test:
12:        Collect trajectories $\breve{\tau}$ applying policy $\Theta^{'}$ in $\breve{\mathcal{S}}$.
13:        Loss $\mathcal{G}(\breve{\tau}, \Theta - \alpha\nabla_\Theta)$.
14:        Meta-optimization:
               $$\Theta = \Theta - \gamma\frac{\partial(\mathcal{F}(\bar{\tau}, \Theta) + \beta\mathcal{G}(\breve{\tau}, \Theta - \alpha\mathcal{F}'(\bar{\tau}, \Theta)))}{\partial\Theta}$$
15:    end for
16: end procedure
```

ations we can train a model to achieve good *generalization* in the **final-test** evaluated on target domains $\mathcal{T}$. The overall methodological flow is illustrated schematically in Fig. 1 and summarized in Algorithm 1. This model-agnostic approach can be flexibly applied to both supervised and reinforcement learning as elaborated in the following sections.

Supervised Learning

We first describe how to apply our method to supervised learning. We assume a loss function $l(\hat{y}, y)$ between the predicted and true labels $\hat{y}$ and y. For example in multi-class classification the cross-entropy loss: $l(\hat{y}, y) = -\hat{y}\log(y)$. The process is outlined in the steps below.

Meta-Train The model is updated on all the $S - V$ meta-train domains $\bar{\mathcal{S}}$ in aggregate, and the loss function is,

$$\mathcal{F}(\cdot) = \frac{1}{S-V}\sum_{i=1}^{S-V}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell_\Theta(\hat{y}_j^{(i)}, y_j^{(i)}) \tag{1}$$

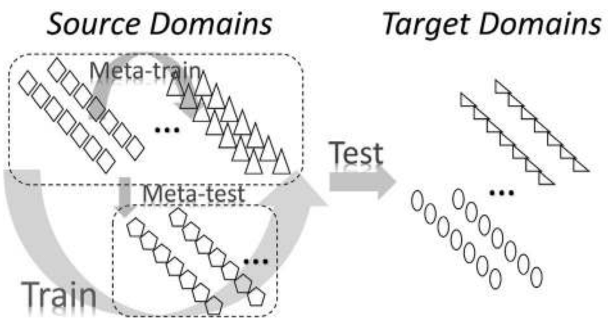


Figure 1: Illustration of our Meta-Learning Domain Generalization method. Symbols represent different data domains.

where $y_j^{(i)}$ indicates the jth point among N_i in the ith domain. The model is parametrized by Θ, so the gradient of Θ calculated respect to this loss function is ∇_Θ, and optimization will update the model as $\Theta^{'} = \Theta - \alpha\nabla_\Theta$.

Meta-Test In each mini-batch the model is also *virtually* evaluated on the V meta-test domains $\breve{\mathcal{S}}$. This meta-test evaluation simulates testing on new domains with different statistics, in order to allow learning to generalize across domains. The loss for the adapted parameters calculated on the meta-test domains is as below,

$$\mathcal{G}(\cdot) = \frac{1}{V}\sum_{i=1}^{V}\frac{1}{N_i}\sum_{j=1}^{N_i}\ell_{\Theta'}(\hat{y}_j^{(i)}, y_j^{(i)}) \tag{2}$$

where, N_i is the number samples of the ith meta-test domain, and the loss on the meta-test domain is calculated using the *updated* parameters $\Theta^{'}$ from meta-train. This means that for optimization with respect to $\mathcal{G}$ we will need the second derivative with respect to Θ.

Summary The meta-train and meta-test are optimized simultaneously, so the final objective is:

$$\underset{\Theta}{\text{argmin}}\ \mathcal{F}(\Theta) + \beta\mathcal{G}(\Theta - \alpha\mathcal{F}'(\Theta)) \tag{3}$$

where α is the meta-train step size and β weights meta-train and meta-test. Objective (Eq. 3) is itself trained by gradient descent (Alg. 1).

Final-Test After Eq. 3 is optimized to convergence on the source domains, we deploy the final model Θ on the truly held-out target domain(s).

Reinforcement Learning

In application to the reinforcement learning (RL) setting, we now assume an agent with a policy π that inputs states x and produces actions a in a sequential decision making task: $a_t = \pi_\Theta(x_t)$. The agent operates in an environment defined by a Markov decision process (MDP) $q(x_{t+1}|x_t, a_t)$ and its goal is to maximize its return, the (potentially discounted) sum of rewards $\mathcal{R} = \sum_t \delta^t R_t(x_t, a_t)$.

While tasks in a supervised learning setting map to reward functions in an RL setting (Finn, Abbeel, and Levine 2017), domains map to solving the same task (reward function) with differences in the environment (MDP or observation function). Thus DG is to achieve an agent with improved generalization ability in the sense of maintaining ability to maximize reward when subject to changes in its operating environment (MDP) – without being allowed any rewarded ($\approx$ supervised domain adaptation (Finn, Abbeel, and Levine 2017; Ammar et al. 2014; Zhao et al. 2017)), or un-rewarded ($\approx$ unsupervised domain adaptation (Finn et al. 2017; Ammar et al. 2015)) trials in the target environment for adaptation. The key idea is still to achieve DG by simulating train-test domain shift during training. Meta-optimization then trains for generalization across environmental conditions. The overall process is summarized in Algorithm 2 and elucidated in the steps below. Note that the MLDG strategy can be straightforwardly applied on-policy with policy-gradient (PG) (Williams 1992), or off-policy with Q-learning (Mnih et al. 2015). For simplicity we describe the PG variant.

Meta-train: In meta-training, the loss function $\mathcal{F}(\cdot)$ now corresponds to the negative return $\mathcal{R}$ of policy π_Θ, averaged over all the meta-training environments in $\bar{\mathcal{S}}$. Update of the policy parameters Θ is performed by REINFORCE (Williams 1992) (or Q-learning (Mnih et al. 2015)), leading to updated parameters $\Theta^{'}$.

Meta-test: Similarly to the SL approach, we now evaluate the model on V meta-test domains $\breve{\mathcal{S}}$. The meta-test loss $\mathcal{G}$ is again the average negative return on meta-test environments. For RL calculating this loss requires rolling out the meta-train updated policy $\Theta^{'}$ in the meta-test domains to collect new trajectories and rewards.

Analysis of MLDG

We provide some analysis to help better understand the proposed method and its motivation. The MLDG objective is:

$$\underset{\Theta}{\text{argmin}}\ \mathcal{F}(\Theta) + \beta\mathcal{G}(\Theta - \alpha\mathcal{F}'(\Theta)) \tag{4}$$

where $\mathcal{F}(.)$ is the loss from the aggregated meta-train domains (Eq. 1), $\mathcal{G}(.)$ is the loss from the aggregated meta-test domains (Eq. 2), and $\mathcal{F}'(\Theta)$ is the gradient of the training loss $\mathcal{F}(\Theta)$ w.r.t 'Θ'. This can be understood as: "*tune such that after updating the meta-train domains, performance is also good on the meta-test domains*".

For another perspective on the MLDG objective, we can do the first order Taylor expansion for the second term, i.e.

$$\mathcal{G}(x) = \mathcal{G}(\dot{x}) + \mathcal{G}'(\dot{x}) \times (x - \dot{x}) \tag{5}$$

where $\dot{x}$ is an arbitrary point that is close to x. The multi-variable form x is a vector and $\mathcal{G}(x)$ is a scalar.

Assume we have $x = \Theta - \alpha\mathcal{F}'(\Theta)$, and we choose the $\dot{x}$ to be Θ. Then, we have

$$\mathcal{G}(\Theta - \alpha\mathcal{F}'(\Theta)) = \mathcal{G}(\Theta) + \mathcal{G}'(\Theta)\cdot(-\alpha\mathcal{F}'(\Theta)) \tag{6}$$

and the objective function becomes

$$\underset{\Theta}{\text{argmin}}\ \mathcal{F}(\Theta) + \beta\mathcal{G}(\Theta) - \beta\alpha(\mathcal{G}'(\Theta)\cdot\mathcal{F}'(\Theta)). \tag{7}$$

This reveals that we want to: (i) *minimize* the loss on both meta-train and meta-test domains, and (ii) *maximize* the dot product of $\mathcal{G}'(\Theta)$ and $\mathcal{F}'(\Theta)$. Minimizing the loss on both domains (i) is intuitive. To understand (ii), recall the dot operation computes the similarity of two vectors: $a \cdot b = ||a||_2||b||_2\cos(\delta)$, where δ is the angle between vectors a and b. If a and b are unit normalized, this computes cosine similarity exactly. Though $\mathcal{G}'(\Theta)$ and $\mathcal{F}'(\Theta)$ are not normalized, the dot product is still larger if these vectors are in a similar direction.

Since $\mathcal{G}'(\Theta)$ and $\mathcal{F}'(\Theta)$ are loss gradients in two sets of domains, then 'similar direction' means the *direction of improvement* in each set of domains is similar. Thus the overall objective can be seen as: "*tune such that both domains' losses are minimized, and also such that they descend in a coordinated way*". In a conventional optimization of $\arg\min_\Theta \mathcal{F}(\Theta) + \mathcal{G}(\Theta)$, there is no such constraint on coordination. The optimizer will happily tune asymmetrically, e.g., focusing on which ever domain is easier to minimize. The regularization provided by the third term in Eq. 7 prefers updates to weights where the two optimization surfaces agree on the gradient. It reduces overfitting to a single domain by finding a route to minimization where both subproblems agree on the direction at all points along the route.

Alternative Variants of MLDG

Based on the discussion above, we propose some variants inspired by the vanilla MLDG method. Variant **MLDG-GC** in Eq. 8 is based on the Taylor expansion and gradient alignment intuition discussed earlier – with the regulariser updated to normalize the gradients so that it indeed computes cosine similarity.

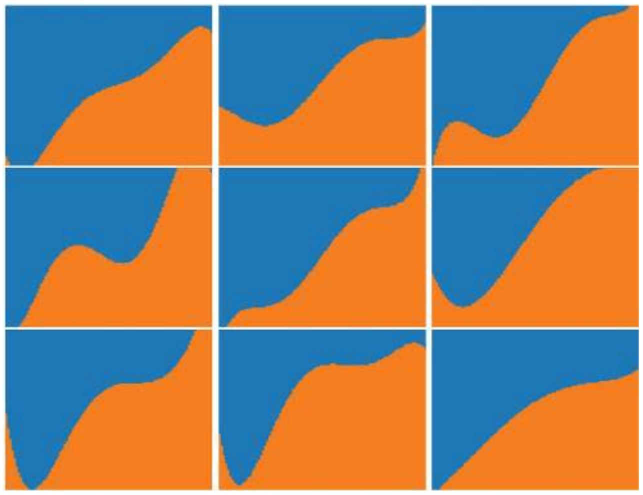
$$\underset{\Theta}{\text{argmin}}\ \mathcal{F}(\Theta) + \beta\mathcal{G}(\Theta) - \beta\alpha\frac{\mathcal{F}'(\Theta)\cdot\mathcal{G}'(\Theta)}{\|\mathcal{F}'(\Theta)\|_2\|\mathcal{G}'(\Theta)\|_2} \tag{8}$$

Another perspective on 'similar direction' gradients is that once meta-train has converged, you also no longer need to update the parameters on the meta-test domains. I.e., at a good solution, meta-test gradients are close to zero. With this intuition variant **MLDG-GN** is proposed in Eq. 9.

$$\underset{\Theta}{\text{argmin}}\ \mathcal{F}(\Theta) + \beta\|\mathcal{G}'(\Theta - \alpha\mathcal{F}'(\Theta))\|_2^2 \tag{9}$$

Clearly MLDG-GN needs a good initialization to be reasonable, so we initialize MLDG-GN with the domain aggregation baseline. In the experiments section we will compare these alternative variants to the initially proposed MLDG.

Related Methods Related to MLDG-GN, squared gradient magnitude loss (SGM) was concurrently proposed in (Hariharan and Girshick 2017) for few-shot recognition. The objective function with SGM loss has the form $\mathcal{F}(\Theta) + \beta\|\mathcal{G}'(\Theta)\|_2^2$. This similar to Eq. 9 when $\alpha = 0$, but the difference is that, $\mathcal{F}$ and $\mathcal{G}$ are classification losses for a large dataset and a small dataset respectively (to simulate the few-shot learning scenario), and there is *no* domain (distribution) shift between these two datasets, though the small one is inadequate to fit the classifier well. These methods are similar in that they are both looking for matched classifiers (between large and small datasets v.s. between meta-train and meta-test datasets), but their motivations are different: to reduce the required training data v.s. to make the model domain invariant.


(a) Synthetic training domains for binary classification

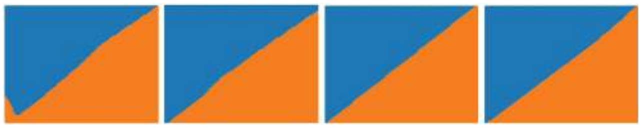

(b) Learned decision boundaries. From left to right: MLP-All; MLDG; MLDG-GC; MLDG-GN.

Figure 2: Synthetic experiment illustrating MLDG.

Experiments

To evaluate our method, we compare it with various alternatives on four different problems, including an illustrative synthetic experiment, a challenging recent computer vision benchmark for multi-class classification across different domains, and two classic reinforcement learning problems, Cart-Pole and Mountain Car. In each case we compare to the baseline of aggregating the data from all source domains to train a single model that ignores domains entirely, as well as various alternative DG methods. As shown in (Li et al. 2017), the former simple baseline can be very effective and outperform many purpose designed DG models.

Experiment I: Illustrative Synthetic Experiment

To illustrate our approach, we construct a synthetic binary classification experiment. We synthesize nine domains by sampling curved deviations from a diagonal line classifier. We treat eight of these as sources for meta-learning and hold out the last for final-test. Fig. 2a shows the nine synthetic domains which are related in form but differ in the details of their decision boundary. A one-hidden layer MLP (50 hidden neurons, RELU activation) is used as the base classifier.
Baselines: **MLP-All**: Simple baseline of aggregating all source domains for training. **MLDG**: Our main proposed MLDG method (Eq. 4). **MLDG-GC** and **MLDG-GN**: variants of our method in Eq. 8 and Eq. 9 respectively.
Results: From the results Fig. 2 we can see that the baseline MLP-ALL over-fits on the training domains. Despite aggregating eight sources, it fits a curve in the bottom left corner rather than the underlying diagonal line. Our methods all draw nearly straight lines. These results illustrate that the MLDG approach helps to avoid overfitting to specific source domains and learn a more generalizable model.

Experiment II: Object Recognition

We next evaluate the efficacy of MLDG on a recent challenging object recognition DG task in computer vision. Specifically, we used the PACS multi-domain recognition benchmark, a new dataset designed for the cross-domain recognition problems (Li et al. 2017)[1]. This dataset has 9991 images in total across 7 categories ('dog', 'elephant', 'giraffe', 'guitar', 'house', 'horse' and 'person') and 4 domains of different stylistic depictions ('Photo', 'Art painting', 'Cartoon' and 'Sketch'). The diverse depiction styles provide a significant domain gap. The goal is to train in set of domains and recognize objects in a disjoint domain. E.g., recognize photos given only various artistic depictions for training.
Baselines: We use the ImageNet pre-trained AlexNet CNN (Krizhevsky, Sutskever, and Hinton 2012) as the base network in each competitor for fair comparison, and compare the following models: **D-MTAE**: a multi-task auto encoder designed for the DG problems (Ghifary et al. 2015). **Deep-All**: Vanilla AlexNet trained on the aggregation of data from all source domains. This baseline that outperforms many prior DG methods as presented in (Li et al. 2017). **DSN**: The domain separation network learns specific and shared representation components for the source and target domains (Bousmalis et al. 2016). We re-purpose the original DSN from the domain adaptation to the DG task. **AlexNet+TF**: the low-rank parametrized network provides prior state of the art on this benchmark (Li et al. 2017).
Settings: We implement MLDG in Tensorflow. We use SGD optimizer with learning rate $5e-4$ (exponential decay is used with decay step $15k$ and decay rate 0.96) and minibatch 64. Meanwhile, parameters α, β, γ are set to $5e-4, 1.0$ and $5e-4$. For final-test, we use the best performing model on the validation set after $45k$ iterations.
Results: The comparison with state of the art on the PACS benchmark is shown in Table 1. From the results, we can see that MLDG surpasses the other baselines including the best prior method AlexNet+TF (Li et al. 2017). We note that this good performance is achieved without any special architecture design and without growing the size of the model in proportion to the number of domains (both of which are required in each of D-MTAE, DSN, and AlexNet+TF). This illustrates the flexibility of MLDG, and also highlights that its scalability compared to alternatives. AlexNet+TF for example requires approximately 2GB of memory per domain with batch size 64, meaning that it cannot be applied to more than 5 source domains on a contemporary GPU.
Analysis of MLDG learning: We next perform some ablation experiments to understand: (i) whether it is important to use MLDG end-to-end way within a CNN, and (ii) verify the impact of the meta-optimization strategy specifically.

To answer the first question of *where* it is important to employ MLDG learning, we compare the variant **MLDG (FC)**: Only apply MLDG learning on the FC layers of AlexNet.

[1]http://sketchx.eecs.qmul.ac.uk

Table 1: Cross-domain recognition accuracy (Multi-class accuracy) on the PACS dataset. Best performance in bold.

	D-MTAE (Ghifary et al. 2015)	Deep-all	DSN (Bousmalis et al. 2016)	AlexNet+TF (Li et al. 2017)	MLDG (CNN)
art_painting	60.27	64.91	61.13	62.86	**66.23**
cartoon	58.65	64.28	66.54	**66.97**	66.88
photo	**91.12**	86.67	83.25	89.50	88.00
sketch	47.86	53.08	58.58	57.51	**58.96**
Ave.	64.48	67.24	67.37	69.21	**70.01**

Table 2: PACS benchmark: Ablation study of MLDG.

	Deep-All	MLDG ($\alpha = 0$)	MLDG (FC)	MLDG (CNN)
art_painting	64.91	64.37	65.54	66.23
cartoon	64.28	65.39	66.37	66.88
photo	86.67	86.67	88.30	88.00
sketch	53.08	55.29	55.34	58.96
Ave.	67.24	67.93	68.89	70.01

Table 3: PACS benchmark: Evaluation of MLDG variants.

	Deep-All	MLDG-GC (Eq. 8)	MLDG-GN (Eq. 9)
art_painting	64.91	64.71	63.64
cartoon	64.28	65.30	63.47
photo	86.67	86.79	87.88
sketch	53.08	56.92	54.94
Ave.	67.24	68.43	67.48

This is in contrast to our full model **MLDG (CNN)** , which applies learning to all layers of AlexNet. Comparing MLDG (FC) to vanilla Deep-All AlexNet in Table 2, we see a benefit of $\approx 1.6\%$ is obtained by MLDG learning on the FC layers. Comparing full MLDG we see that a further $\approx 1.1\%$ benefit is obtained by applying MLDG learning to the convolutional layers, for a total of $\approx 2.7\%$ margin over Deep-All.

To verify the impact of the meta-optimization strategy, we apply MLDG with setting $\alpha = 0$, in which case the objective is merely the sum of the training and validation (meta-test) domains' losses. From the results in Table 2, we see that it performs comparably with Deep-All. Thus the key benefit of MLDG is indeed in the meta-optimization step.

Analysis of MLDG variants: In the Table 3, the original MLDG method is compared to the two variants also proposed in the methodology. In this experiment we found that while the MLDG-GC (cosine) and MLDG-GN (gradient norm) variants provide some benefit compared to Deep-All, the vanilla MLDG performs best.

Experiment III: Cart-Pole

We next demonstrate that MLDG also applies to RL problems. First we study the classic Cart Pole problem (Brockman et al. 2016). The objective is to balance a pole upright by moving a cart. The action space is discrete – left or right. The state it has four elements: the position and velocity of cart and angular position and velocity of the pole.

Settings: We perform two sub-experiments by modifying the OpenAI Gym simulator to provide environments with different properties. In the first we vary one domain factor by changing the pole length. We simulate 9 domains with pole lengths $[0.5, 1.0, \dots, 4.5]$. In the second we vary multiple domain factors – pole length $[0.5, 2.5, 4.5]$ and cart mass $[1, 2, 3]$. In both experiments we randomly choose 6 source domains for training and hold out 3 domains for (true) testing. Since the game can last forever if the pole does not fall, we cap the maximum steps to 200. We train on the observed domains for 500 games per domain. Then, for each held-out domain, we play 500 games, and report the average reward. For fair comparison, the policy architecture for all models is a 1-hidden layer neural network with 50 hidden units. The reward structure is +1 for each time-step the pole is successfully balanced, so the maximum reward is 200. All methods are trained with vanilla REINFORCE policy gradient (Williams 1992).

Table 4: Cart-Pole RL. Domain generalization performance across pole length. Average reward testing on 3 held out domains with random lengths. Upper bound: 200.

Method	RL-Random-Source	RL-All	RL-Undobias
Return	133.74 ± 6.79	97.39 ± 73.49	113.52 ± 11.65
Method	RL-MLDG	RL-MLDG-GC	RL-MLDG-GN
Return	165.34 ± 3.38	129.56 ± 2.51	175.25 ± 3.16

Table 5: Cart-Pole RL. Generalization performance across both pole length and cart mass. Return testing on 3 held out domains with random length and mass. Upper bound: 200.

Method	RL-Random-Source	RL-All	RL-Undobias
Return	98.22 ± 20.35	144.21 ± 9.23	150.46 ± 17.59
Method	RL-MLDG	RL-MLDG-GC	RL-MLDG-GN
Return	170.81 ± 9.90	147.76 ± 4.41	164.97 ± 8.45

Baselines: We compare the following alternative approaches: **RL-All**: The reinforcement-learning analogy to 'Deep-ALL' in the recognition experiment. Trains a single policy by aggregating the reward from all six source domains. **RL-Random-Source**: Different from RL-All, it trains on a single randomly selected source domain. Total training trials are controlled so it gets the same number of trials in one domain as RL-All gets in multiple domains. **RL-Undobias**: Adaptation of the (linear) undo-bias model of (Khosla et al. 2012) updated to non-linear multi-layer network as per (Li et al. 2017). The neural network is trained to factor domain-specific and a single domain-agnostic components on six source domains. The domain agnostic component is then transferred for testing on held out final-testing domains. **RL-MLDG**: Our MLDG. **RL-MLDG-GC**: Our MLDG variant. **RL-MLDG-GN**: Our MLDG variant. In each mini-batch, we split the $S = 6$ source domains into $V = 2$ meta-test and $S - V = 4$ meta-train domains.

Results: All experiments are repeated 10 times to reduce the impact of specific observed/held-out domain sampling. From the results in Tables 4 and 5, we see the impact of domain shift. No methods reach 200 (upper bound

Table 6: Domain generalization performance for mountain car. Failure rate (↓) and reward (↑) on held out testing domains with random mountain heights.

Mountain Car	RL-Random-Source	RL-All	RL-Undobias
Avg. F Rate	0.55 ± 0.07	0.05 ± 0.02	0.08 ± 0.04
Avg. Return	-191.07 ± 3.01	-141.35 ± 2.64	-124.48 ± 3.22
Mountain Car	RL-MLDG	RL-MLDG-GC	RL-MLDG-GN
Avg. F Rate	0.05 ± 0.02	0.0 ± 0.0	1.0 ± 0.0
Avg. Return	-125.73 ± 2.76	-311.80 ± 3.92	-

given the length cap) for unseen domains reliably. However, the proposed MLDG provides the best domain generalization and significantly outperform the baselines. It is interesting to note that RL-Random-Source outperforms RL-All in Table 4, which is different than in vision problems where simply aggregating more domains is usually a reasonable strategy. Although RL-All is exposed to more diverse data, learning a single policy by naively 'averaging' over rewards for multiple distinct problems can sometimes be detrimental (Sung et al. 2017),

Analysis of MLDG variants: Comparing MLDG with its variants MLDG-GC and MLDG-GN we found that MLDG-GN is comparable to vanilla MLDG on this problem, while MLDG-GC is slightly worse.

Experiment IV: Mountain Car

Our second RL experiment is the classic mountain car problem (Brockman et al. 2016). The car is positioned between two mountains, and the agent needs to drive the car (back or forth) so that it can hit the peak of the right mountain. The difficulty of this problem is that the car engine is not strong enough to drive up the right mountain directly. The agent has to figure out a solution of driving up the left mountain to first generate momentum before driving up the right mountain. The state observation in this game consists two elements: the position and velocity of the car. There are three available actions: drive left, do nothing, and drive right.

Settings: We simulate domain bias by randomly drawing the height of the mountains in each domain. Similar to Cart-Pole, we simulate 9 domains in total, and 3 domains are held-out. In contrast to Cart-Pole, it is very difficult for a random policy to finish a full game, as it is likely to be stuck forever. Thus instead of policy gradient, we use Q learning (Watkins and Dayan 1992) for this problem as the base RL algorithm, more specially DQNs (Mnih et al. 2015). For held out domains we play 100 games each without updating. The reward structure is -1 each time step before reaching the target. The Q-network is again a 1 hidden layer MLP.

Baselines: We evaluate the following alternatives **RL-Random-Source**: Trains a single policy on one random source domain. **RL-All**: Trains a single policy on 6 source domains in aggregation. **RL-Undobias**: DG parametrized Q-network adaptation of (Khosla et al. 2012; Li et al. 2017) as per cart-pole. **RL-MLDG**: Our MLDG. And its variants **RL-MLDG-GC** and **RL-MLDG-GN**. In each mini-batch, we split the $S = 6$ source domains into $V = 2$ meta-test domains, and $S - V = 4$ meta-train domains.

Results: All experiments are repeated 10 times to reduce the impact of random observed/held-out domain splits. From the results in Table 6, we again observe the performance drops from observed domains and held-out domains. In this benchmark, succeeding within 110 steps is a good outcome. So a reward of -110 is a good score for within domain evaluation. I.e., in the absence of domain shift. Since it is possible for an agent to never succeed on this benchmark, particularly when testing in a distinct domain from training, we apply a limit of $20,000$ steps maximum. For DG testing, most methods have some failed trials ($> 20,000$ steps) in final-test. The average reward is calculated by ignoring those failed cases. Therefore we report both failure rate and the average reward (negative time to success) in the successful cases. The results show that our vanilla MLDG method outperforms the alternatives: (i) Its average reward is better than RL-All and similar to RL-UndoBias. However (ii) its fail rate is lower than RL-UndoBias. Unlike Cart-Pole here RL-All is more effective than Random-Source.

Analysis of MLDG variants: Only vanilla MLDG performed well here. MLDG-GC had low failure rate but low return, while MLDG-GN had very high failure rate.

Discussion

The experiments show that MLDG-based meta-learning can effectively alleviate domain-shift in diverse problems including supervised and re-reinforcement learning scenarios. Whether training on the aggregate of multiple source domains was a good strategy turned out to be problem dependent (yes for PACS vision benchmark and mountain car, but not for cart pole). The extended variants of the MLDG model MLDG-GC (explicit gradient direction alignment) and MLDG-GN (gradient norm) also had mixed results with MLDG-GC performing second best on PACS, but MLDG-GN performing best on Cart-Pole. Nevertheless the core MLDG strategy was highly effective across all problems and always outperformed prior alternatives.

We note that studies have used the terms 'domain' and 'task' in different ways (Csurka 2017). Some problems we solved here (e.g., poles of different length) have been termed 'tasks' in other studies (Ammar et al. 2014; Zhao et al. 2017), which would use 'domain' to refer to Cart-Pole versus Mountain Car. We use the term domain in the sense of the pattern recognition community (Csurka 2017), where one can learn a model with better 'cross domain generalization'. E.g. a recognition model that is robust to recognizing photos vs sketches; or a policy that is more robust being deployed with poles of a different length than it was trained on. Note that if parameters like pole-length were *observed*, this would be a 'parametrized' or 'contextual' policy situation - for which methods already exist (Kupcsik et al. 2013). But in our case what meta-learning has achieved is to learn a policy that is robust to (i.e., obtains high reward despite of) hidden changes in the underlying MDP. For example balancing poles of diverse but unknown lengths.

Conclusion

We proposed a meta-learning algorithm for domain generalization. Our method trains for domain generalization by meta-optimization on simulated train/test splits with domain-shift. Unlike prior model-based domain generalization approaches, it scales well with number of domains. It is model agnostic so can be applied to different base network types, and both to supervised and reinforcement learning problems. Experimental evaluation shows state of the art results on a recent challenging visual recognition benchmark and promising results on multiple classic RL problems.

Acknowledgements This work was supported by EPSRC (EP/R026173/1) and the European Union's Horizon 2020 research and innovation program under grant agreement No 640891.

References

Ammar, H. B.; Eaton, E.; Ruvolo, P.; and Taylor, M. 2014. Online multi-task learning for policy gradient methods. In *ICML*.

Ammar, H. B.; Eaton, E.; Ruvolo, P.; and Taylor, M. E. 2015. Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment. In *AAAI*.

Andrychowicz, M.; Denil, M.; Gomez, S.; Hoffman, M. W.; Pfau, D.; Schaul, T.; and de Freitas, N. 2016. Learning to learn by gradient descent by gradient descent. In *NIPS*.

Bilen, H., and Vedaldi, A. 2017. Universal representations: The missing link between faces, text, planktons, and cat breeds. In *arXiv 1701.07275*.

Bousmalis, K.; Trigeorgis, G.; Silberman, N.; Krishnan, D.; and Erhan, D. 2016. Domain separation networks. In *NIPS*.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. *OpenAI Gym*.

Csurka, G. 2017. *Domain Adaptation in Computer Vision Applications*. Springer.

Daumé, H. 2007. Frustratingly easy domain adaptation. In *ACL*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*.

Finn, C.; Yu, T.; Fu, J.; Abbeel, P.; and Levine, S. 2017. Generalizing skills with semi-supervised reinforcement learning. In *ICLR*.

Ganin, Y., and Lempitsky, V. 2015. Unsupervised domain adaptation by backpropagation. In *ICML*.

Ghifary, M.; Bastiaan Kleijn, W.; Zhang, M.; and Balduzzi, D. 2015. Domain generalization for object recognition with multi-task autoencoders. In *ICCV*.

Hariharan, B., and Girshick, R. 2017. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*.

Khosla, A.; Zhou, T.; Malisiewicz, T.; Efros, A. A.; and Torralba, A. 2012. Undoing the damage of dataset bias. In *ECCV*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

Kupcsik, A. G.; Deisenroth, M. P.; Peters, J.; and Neumann, G. 2013. Data-efficient generalization of robot skills with contextual policy search. In *AAAI*.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2017. Deeper, broader and artier domain generalization. In *ICCV*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature*.

Muandet, K.; Balduzzi, D.; and Schölkopf, B. 2013. Domain generalization via invariant feature representation. In *ICML*.

Parisotto, E.; Ba, J. L.; and Salakhutdinov, R. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*.

Patel, V.; Gopalan, R.; Li, R.; and Chellappa, R. 2015. Visual domain adaptation: A survey of recent advances. *Signal Processing Magazine*.

Ravi, S., and Larochelle, H. 2017. Optimization as a model for few-shot learning. In *ICLR*.

Rebuff, S.-A.; Bilen, H.; and Vedaldi, A. 2017. Learning multiple visual domains with residual adapters. In *NIPS*.

Schmidhuber, J.; Zhao, J.; and Wiering, M. 1997. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*.

Storkey, A. J., and Sugiyama, M. 2007. Mixture regression for covariate shift. In *NIPS*.

Sung, F.; Zhang, L.; Xiang, T.; Hospedales, T.; and Yang, Y. 2017. Learning to learn: Meta-critic networks for sample efficient learning. In *arXiv 1706.09529*.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *JMLR*.

Thrun, S., and Pratt, L. 1998. *Learning to Learn*. Springer.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; and Wierstra, D. 2016. Matching networks for one shot learning. In *NIPS*.

Watkins, C., and Dayan, P. 1992. Q-learning. *Machine Learning*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

Xu, Z.; Li, W.; Niu, L.; and Xu, D. 2014. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*.

Yang, Y., and Hospedales, T. 2015. A unified perspective on multi-domain and multi-task learning. In *ICLR*.

Zhao, C.; Hospedales, T.; Stulp, F.; and Sigaud, O. 2017. Tensor based knowledge transfer across skill categories for robot control. In *IJCAI*.