GENERALIZING ACROSS DOMAINS VIA CROSS-GRADIENT TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT
We present CrossGrad, a method to use multi-domain training data to learn a classifier that generalizes to new domains. CrossGrad does not need an adaptation phase via labeled or unlabeled data, or domain features in the new domain. Most existing domain adaptation methods attempt to erase domain signals using techniques like domain adversarial training. In contrast, CrossGrad is free to use domain signals for predicting labels, if it can prevent overfitting on training domains. We conceptualize the task in a Bayesian setting, in which a sampling step is implemented as data augmentation, based on domain-guided perturbations of input instances. CrossGrad jointly trains a label and a domain classifier on examples perturbed by loss gradients of each other’s objectives. This enables us to directly perturb inputs, without separating and re-mixing domain signals while making various distributional assumptions. Empirical evaluation on three different applications where this setting is natural establishes that (1) domain-guided perturbation provides consistently better generalization to unseen domains, compared to generic instance perturbation methods, and that (2) data augmentation is a more stable and accurate method than domain adversarial training.

1 INTRODUCTION
We investigate how to train a classification model using multi-domain training data, so as to generalize to labeling instances from unseen domains. This problem arises in many applications, viz., handwriting recognition, speech recognition, sentiment analysis, and sensor data interpretation. In these applications, domains may be defined by fonts, speakers, writers, etc. Most existing work on handling a target domain not seen at training time requires either labeled or unlabeled data from the target domain at test time. Often, a separate “adaptation” step is then run over the source and target domain instances, only after which target domain instances are labeled. In contrast, we consider the situation where, during training, we have labeled instances from several domains which we can collectively exploit so that the trained system can handle new domains without the adaptation step.

1.1 PROBLEM STATEMENT
Let $D$ be a space of domains. During training we get labeled data from a proper subset $D \subset D$ of these domains. Each labeled example during training is a triple $(x, y, d)$ where $x$ is the input, $y \in \mathcal{Y}$ is the true class label from a finite set of labels $\mathcal{Y}$ and $d \in D$ is the domain from which this example is sampled. We must train a classifier to predict the label $y$ for examples sampled from all domains, including the subset $D \setminus \hat{D}$ not seen in the training set. Our goal is high accuracy for both in-domain (i.e., in $\hat{D}$) and out-of-domain (i.e., in $D \setminus \hat{D}$) test instances.

One challenge in learning a classifier that generalizes to unseen domains is that $Pr(y|x)$ is typically harder to learn than $Pr(y|x, d)$. While Yang & Hospedales (2015) addressed a similar setting, they assumed a specific geometry characterizing the domain, and performed kernel regression in this space. In contrast, in our setting, we wish to avoid any such explicit domain representation, appealing instead to the power of deep networks to discover implicit features.

Lacking any feature-space characterization of the domain, conventional training objectives (given a choice of hypotheses having sufficient capacity) will tend to settle to solutions that overfit on the set of domains seen during training. A popular technique in the domain adaptation literature to
generalize to new domains is domain adversarial training (Ganin et al., 2016; Tzeng et al., 2017). As the name suggests, here the goal is to learn a transformation of input $x$ to a domain-independent representation, with the hope that amputating domain signals will make the system robust to new domains. We show in this paper that such training does not necessarily safeguard against over-fitting of the network as a whole. We also argue that even if such overfitting could be avoided, we do not necessarily want to wipe out domain signals, if it helps achieve high accuracy for in-domain test instances.

1.2 Contribution

In a marked departure from domain adaptation via amputation of domain signals, we approach the problem using a form of data augmentation based on domain-guided perturbations of input instances. If we could model exactly how domain signals for $d$ manifest in $x$, we could simply replace these signals with those from suitably sampled other domains $d'$ to perform data augmentation. We first conceptualize this in a Bayesian setting: discrete domain $d$ ‘causes’ continuous multivariate $g$, which, in combination with $y$, ‘causes’ $x$. Given an instance $x$, if we can recover $g$, we can then perturb $g$ to $g'$, thus generating an augmented instance $x'$. Because such perfect domain perturbation is not possible in reality, we first design an (imperfect) domain classifier network to be trained with a suitable loss function. Given an instance $x$, we use the loss gradient w.r.t. $x$ to perturb $x$ in directions that change the domain classifier loss the most. The training loss for the $y$-predictor network on original instances is combined with the training loss on the augmented instances. We call this approach cross-gradient training, which is embodied in a system we describe here, called CROSSGRAD. We carefully study the performance of CROSSGRAD on a variety of domain adaptive tasks: character recognition, handwriting recognition and spoken word recognition. We demonstrate performance gains on new domains without any out-of-domain instances available at training time.

2 Related Work

Domain adaptation has been studied under many different settings: two domains (Ganin et al., 2016; Tzeng et al., 2017) or multiple domains (Mansour et al., 2009; Zhang et al., 2015), with target domain data that is labeled (Daumé III, 2007; Daumé III et al., 2010; Saenko et al., 2010) or unlabeled (Gopalan et al., 2011; Gong et al., 2012; Ganin et al., 2016), paired examples from source and target domain (Peng et al., 2017), or domain features attached with each domain (Yang & Hospedales, 2016). Domain adaptation techniques have been applied to numerous tasks in speech, language processing and computer vision (Woodland, 2001; Saon et al., 2013; Jiang & Zhai, 2007; Daumé III, 2007; Saenko et al., 2010; Gopalan et al., 2011; Li et al., 2013; Huang & Belongie, 2017; Upchurch et al., 2016). However, unlike in our setting, these approaches typically assume the availability of some target domain data which is either labeled or unlabeled.

For neural networks a recent popular technique is domain adversarial networks (DANs) (Tzeng et al., 2015; 2017; Ganin et al., 2016). The main idea of DANs is to learn a representation in the last hidden layer (of a multilayer network) that cannot discriminate among different domains in the input to the first layer. A domain classifier is created with the last layer as input. If the last layer encapsulates no domain information apart from what can be inferred from the label, the accuracy of the domain classifier is low. The DAN approach makes sense when all domains are visible during training. In this paper, our goal is to generalize to unseen domains.

Ganin et al. (2016) tackle a similar problem. They use a DAN to achieve domain generalization. A major flaw in this approach is that DANs can be easily misled by a representation layer that overfits the set of training domains. In the extreme case, a representation that simply outputs label logits via a last linear layer (making the softmax layer irrelevant) can keep both the adversarial loss and label loss small, and yet not be able to generalize to new test domains. In other words, not being able to infer the domain from the last layer does not imply that the classification is domain-robust.

Our Bayesian model to capture the dependence among label, domains, and input is similar to Zhang et al. (2015, Fig. 1d), but the crucial difference is the way the dependence is modeled and estimated.

Since we do not assume any extra information about the test domains, conventional ML approaches for generalizability are also relevant. Of these, the method most related is the adversarial training of (Goodfellow et al., 2014; Miyato et al., 2016) that perturbs examples along the gradient of the clas-
sifier loss. The only extra information we possess is that training data is grouped by domain labels. Our method attempts to model domain variation in a continuous space and project perturbation in that space to the instances.

3 OUR APPROACH

We assume that input objects are characterized by two uncorrelated or weakly correlated tags: their label and their domain. E.g. for a set of typeset characters, the label could be the corresponding character of the alphabet (‘A’, ‘B’ etc) and the domain could be the font used to draw the character. In general, it should be possible to change any one of these, while holding the other fixed.

We use a Bayesian network to model the dependence among the label \( y \), domain \( d \), and input \( x \) as shown in Figure 1. Variables \( y \in \mathcal{Y} \) and \( d \in \mathcal{D} \) are discrete and \( g \in \mathbb{R}^q \). The domain \( d \) induces a set of latent domain features \( g \). The input \( x \) is obtained by a complicated, un-observed mixing\(^1\) of \( y \) and \( g \). In the training sample \( L \), nodes \( y,d,x \) are observed but \( L \) spans only a proper subset \( D \) of the set of all domains \( \tilde{D} \). During inference, only \( x \) is observed and we need to compute the posterior \( \Pr(y|x) \). As reflected in the network, \( y \) is not independent of \( d \) given \( x \). However, since \( d \) is discrete and we observe only a subset of \( d \)’s during training, we need to make additional assumptions to ensure that we can generalize to a new \( d \) during testing. The assumption we make is that integrated over the training domains the distribution \( P(g) \) of the domain features is well-supported in \( L \). More precisely, generalizability of a training set of domains \( D \) to the universe of domains \( \mathcal{D} \) requires that

\[
P(g) = \sum_{d \in D} P(g|d)P(d) \approx \sum_{d \in D} P(g|d) \frac{P(d)}{P(D)} \quad (A1)
\]

Under this assumption \( P(g) \) can be modeled during training, so that during inference we can infer \( y \) for a given \( x \) by estimating

\[
\Pr(y|x) = \sum_{d \in D} \Pr(y|x,d) \Pr(d|x) = \int_{g} \Pr(y|x,g) \Pr(g|x) \approx \Pr(y|x,\hat{g})
\]

where \( \hat{g} = \text{argmax}_g \Pr(g|x) \) is the inferred continuous representation of the domain of \( x \).

We next elaborate on how we estimate \( \Pr(y|x,\hat{g}) \) and \( \hat{g} \) using the domain labeled data \( L = \{(x,y,d)\} \). The main challenge in this task is to ensure that the model for \( \Pr(y|x,\hat{g}) \) is not overfitted on the inferred \( g \)’s of the training domains. In many applications, the per-domain \( \Pr(y|x,d) \) is significantly easier to train. So, an easy local minima is to choose a different \( g \) for each training \( d \) and generate separate classifiers for each distinct training domain. We must encourage the network to stay away from such easy solutions. We strive for generalization by moving along the continuous space \( g \) of domains to sample new training examples from hallucinated domains. Ideally, for each training instance \((x_i,y_i)\) from a given domain \( d_i \), we wish to generate a new \( x' \) by transforming its

\(^1\)The dependence of \( y \) on \( x \) could also be via continuous hidden variables but our model for domain generalization is agnostic of such structure.
(inferred) domain \( g_i \), to a random domain sampled from \( P(g) \), keeping its label \( y_i \) unchanged. Under the domain continuity assumption (A1), a model trained with such an ideally augmented dataset is expected to generalize to domains in \( D \setminus D \).

However, there are many challenges to achieving such ideal augmentation. To avoid changing \( y_i \), it is convenient to draw a sample \( g \) by perturbing \( g_i \). However, \( g_i \) may not be reliably inferred, leading to a distorted sample of \( g \). For example, if the \( g_i \) obtained from an imperfect extraction conceals label information, then big jumps in the approximate \( g \) space could change the label too. We propose a more cautious data augmentation strategy that makes only small moves along the estimated domain features of an input. We arrive at our method in three steps, as follows.

**Domain inference:** We create a model \( G(x) \) to extract domain features \( g \) from an input \( x \). We supervise the training of \( G \) to predict the domain label \( d_i \) as \( S(G(x_i)) \) where \( S \) is a softmax transformation. We use \( J_d \) to denote the cross-entropy loss function of this classifier. Specifically, \( J_d(x_i, d_i) \) is the domain loss at the current instance.

**Domain perturbation:** Given an example \( (x_i, y_i, d_i) \), we seek to sample domain features \( g'_i \) that would correspond to an example \( (x'_i, y_i) \) (i.e., with the same label \( y_i \)), whose domain is as “far” from \( d_i \) as possible. To this end, consider setting \( g'_i = g_i + \epsilon \nabla_{g_i} J_d(x_i, d_i) \), where \( g_i = G(x_i) \) and \( \epsilon \) is a suitable step size. The direction of this perturbation roughly corresponds to changing the domain as much as possible, for a given budget of \( ||g'_i - g_i|| \). However, this approach presupposes that the direction of domain change in our domain classifier is not highly correlated with the direction of label change. To enforce this in our model, we shall train the domain feature extractor \( G \) to avoid domain shifts when the data is perturbed to cause label shifts.

**Recovering perturbed input:** Having sampled \( g'_i \) as above, we define our new augmented input instance \( x'_i \) as

\[
G^{-1}(g'_i) = G^{-1}(G(x_i) + \epsilon \nabla_{g_i} J_d(x_i, d_i))
\]

We show next that the perturbed \( x'_i \) can be approximated by \( x_i + \epsilon \nabla_{x} J_d(x_i, d_i) \).

In this proof we drop the subscript \( i \) for ease of notation. In the forward direction, the relationship between \( \Delta x \) and \( \Delta g \) can be expressed using the Jacobian \( J \) of \( g \) w.r.t. \( x \):

\[
\Delta g = J \Delta x
\]

To invert the relationship for a non-square and possibly low-rank Jacobian, we use the Jacobian transpose method devised for inverse kinematics (Balestrino et al., 1984; Wolovich & Elliott, 1984). Specifically, we write \( \Delta g = g(x') - g(x) \), and recast the problem as trying to minimize the squared L2 error

\[
L_{g}(x) = \frac{1}{2} (g(x') - g(x))^T (g(x') - g(x))
\]

with gradient descent. The gradient of the above expression w.r.t. \( x \) is

\[
\nabla_{x} L_{g} = -\left( \Delta g^T \frac{\partial g}{\partial x} \right)^T = -J^T \Delta g
\]

Hence, the necessary gradient descent step to affect a change of \( \Delta g \) in the domain features would increment \( x \) by \( \epsilon J^T \Delta g \). Assuming local linearity and for a suitable \( \epsilon' \), we can approximate the complete gradient descent process with this single step. The Jacobian, which is a matrix of first partial derivatives, can be computed by back-propagation. Thus we get

\[
\Delta x = \epsilon' J^T \nabla_{g} J_d(g, d) = \epsilon' \sum \frac{\partial g^i}{\partial x} \frac{\partial J_d(g, d)}{\partial g^i}
\]

which, by the chain rule, gives

\[
\Delta x = \epsilon' \nabla_{x} J_d(x, d)
\]

The above development leads to the network sketched in Figure 2, and an accompanying training algorithm, CROSSGRAD, shown in Algorithm 1. Our proposed method integrates data aug-
mentation and batch training as an alternating sequence of steps. We compute not only the standard \( V_X J_l(X, Y; \theta_l) \), but also the complementary \( \nabla_X J_d(X, D; \theta_d) \). Here, \( X, Y, D \) correspond to a minibatch of instances. Next we construct cross-objectives \( \tilde{J}_l \) and \( \tilde{J}_d \), and update their respective parameter spaces.

Algorithm 1 CROSSGRAD training pseudocode.

1: **Input:** Labeled data \( \{(x_i, y_i, d_i)\}_{i=1}^M \), step sizes \( \epsilon_l, \epsilon_d \), learning rate \( \eta \), data augmentation weights \( \alpha_l, \alpha_d \), number of training steps \( n \).
2: **Output:** Label and domain classifier parameters \( \theta_l, \theta_d \)
3: Initialize \( \theta_l, \theta_d \) \( \{J_l, J_d \} \) are loss functions for the label and domain classifiers, respectively
4: for \( n \) training steps do
5: Sample a labeled batch \((X, Y, D)\)
6: \( X_d := X + \epsilon_d \cdot \nabla_X J_d(X, D; \theta_d) \)
7: \( X_l := X + \epsilon_l \cdot \nabla_X J_l(X, Y; \theta_l) \)
8: \( \theta_l \leftarrow \theta_l - \eta \nabla_{\theta_l}(1 - \alpha_l) J_l(X, Y; \theta_l) + \alpha_l J_l(X, Y; \theta_l) \)
9: \( \theta_d \leftarrow \theta_d - \eta \nabla_{\theta_d}(1 - \alpha_d) J_d(X, D; \theta_d) + \alpha_d J_d(X, D; \theta_d) \)
10: end for

If \( y \) and \( d \) are completely correlated, CROSSGRAD reduces to traditional adversarial training. If, on the other extreme, they are perfectly uncorrelated, removing domain signal should work well. The interesting and realistic situation is where they are only partially correlated. CROSSGRAD is designed to handle the whole spectrum of correlations.

4 Experiments

In this section we demonstrate that CROSSGRAD provides effective domain generalization on four different classification tasks under three different model architectures. We provide evidence that our Bayesian characterization of domains as continuous features is responsible for such generalization. We establish that CROSSGRAD’s domain guided perturbations provides a more consistent generalization to new domains than label adversarial perturbation (Goodfellow et al., 2014). Also, we show that DANs, a popular domain adaptation method that suppresses domain signals, provides little improvements over the baseline (Ganin et al., 2016; Tzeng et al., 2017).

We describe the four different datasets and present a summary in Table 3.

**Character recognition across fonts** We created this dataset from Google Fonts\(^3\). The task is to identify the character across different fonts as the domain. The label set consists of twenty-six letters of the alphabet and ten numbers. The data comprises of 109 fonts which are partitioned as 65% train, 25% test and 10% validation folds. For each font and label, two to eighteen images are generated by randomly rotating the image around the center and adding pixel-level random noise. The neural network is LeNet (LeCun et al., 1998) modified to have three pooled convolution layers instead of tanh.

**Handwriting recognition across authors** We use the LipitK dataset that comprises various handwritten characters from the Devanagari script made available by HP Labs as part of the Indic script corpus\(^3\). Each writer of the characters is considered as a domain, and the task is to recognize the character. The images are split as 60% train, 25% test and 15% validation. The neural network is the 2-layer convolutional one used by Motiian et al. (2017).

**MNIST across synthetic domains** This dataset derived from MNIST was introduced by Motiian et al. (2017). Here, labels comprise the 10 digits and domains are created by rotating the images in multiples of 15 degrees: 0, 15, 30, 45, 60 and 75. The domains are labeled with the angle by which they are rotated, e.g., M15, M30, M45. We tested on domain M15 while training on the rest. The network is the 2-layer convolutional one used by Motiian et al. (2017).

\(^3\)http://lipitk.sourceforge.net/hpl-datasets.htm

\(^2\)https://fonts.google.com/?category=Handwriting&subset=latin
Spoken word recognition across users This is the Google Speech Command Dataset⁴ that consists of spoken word commands from a large number of speakers in different acoustic settings. The task is to identify the spoken words. The domains are the speakers. We used 20% of domains each for testing and validation. The number of training domains was 100 for the experiments in Table 4. We also report numbers with varying number of domains later in Table 8. We use the architecture of Sainath & Parada (2015) ⁵.

For all four datasets, the set of domains in the training, test, and validation sets were kept totally disjoint. We selected the various hyper-parameters based on accuracy on the validation set as follows. For \( \text{LABELGRAD} \) the parameter \( \alpha \) was chosen from \( \{0.1, 0.25, 0.75, 0.5, 0.9\} \) and for \( \text{CROSSGRAD} \) we chose \( \alpha_l = \alpha_d \) from the same set of values. We chose \( \epsilon \) ranges so that \( L_\infty \) norm of the perturbations are of similar sizes in \( \text{LABELGRAD} \) and \( \text{CROSSGRAD} \). The multiples in the \( \epsilon \) range came from \( \{0.5, 1, 2, 2.5\} \). The optimizer for the first three datasets is RMS prop with a learning rate (\( \eta \)) of 0.02 whereas for the last Speech dataset it is SGD with \( \eta = 0.001 \) initially and 0.0001 after 15 iterations.

4.1 Overall comparison

In Table 4 we compare \( \text{CROSSGRAD} \) with domain adversarial networks (DAN), label adversarial perturbation (\( \text{LABELGRAD} \)), and a baseline that performs no special training. For the MNIST dataset the baseline is CCSA (Motiian et al., 2017). We observe that, for all four datasets, \( \text{CROSSGRAD} \) provides an accuracy improvement. DAN, which is designed specifically for domain adaptation, is worse than \( \text{LABELGRAD} \), which does not exploit domain signal in any way. While the gap between \( \text{LABELGRAD} \) and \( \text{CROSSGRAD} \) is not dramatic, it is consistent as supported by this table and other experiments that we later describe.

Changing model architecture In order to make sure that these observed trends hold across model architectures, we compare different methods with the model changed to a 2-block ResNet (He et al., 2016) instead of LeNet (LeCun et al., 1998) for the Fonts and Handwriting dataset in Table 5. For both datasets the ResNet model is significantly better than the LeNet model. But even for the higher capacity ResNet model, \( \text{CROSSGRAD} \) surpasses the baseline accuracy while \( \text{LABELGRAD} \) shows little or no improvement.

Table 3: Summary of datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Label</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font</td>
<td>36 characters</td>
<td>109 fonts</td>
</tr>
<tr>
<td>Handwriting</td>
<td>111 characters</td>
<td>74 authors</td>
</tr>
<tr>
<td>MNIST</td>
<td>10 digits</td>
<td>6 rotations</td>
</tr>
<tr>
<td>Speech</td>
<td>12 commands</td>
<td>1888 speakers</td>
</tr>
</tbody>
</table>

Table 4: Accuracy on four different datasets. The baseline for MNIST is CCSA.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Fonts</th>
<th>Handwriting</th>
<th>MNIST</th>
<th>Speech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>68.5</td>
<td>82.5</td>
<td>95.6</td>
<td>72.6</td>
</tr>
<tr>
<td>DAN</td>
<td>68.9</td>
<td>83.8</td>
<td>98.0</td>
<td>70.4</td>
</tr>
<tr>
<td>( \text{LABELGRAD} )</td>
<td>71.4</td>
<td>86.3</td>
<td>97.8</td>
<td>72.7</td>
</tr>
<tr>
<td>( \text{CROSSGRAD} )</td>
<td>72.6</td>
<td>88.6</td>
<td>98.6</td>
<td>73.5</td>
</tr>
</tbody>
</table>

Table 5: Accuracy with varying model architectures.

---


⁵We use the code available in tensorflow https://www.tensorflow.org/versions/master/tutorials/audio_recognition
4.2 Why does CrossGrad work?

We present insights on the working of CrossGrad via experiments on the MNIST dataset where the domains corresponding to image rotations are easy to interpret.

In Figure 6a we show PCA projections of the embeddings for images for three different domains, corresponding to rotations by 30, 45, 60 degrees in green, blue, and yellow respectively. The embeddings of domain 45 (blue) lies in between the embeddings of domains 30 (green) and 60 (yellow) showing that the domain classifier has successfully extracted continuous representation of the domain even when the input domain labels are categorical. Figure 6b shows the same pattern for domains 0, 15, 30. Here again we see that the embedding of domain 15 (blue) lies in-between that of domain 0 (yellow) and 30 (green).

Next, we show that the embeddings, perturbed along gradients of domain loss, does manage to generate images that substitute for the missing domains during training. For example, the embeddings of the domain 45, when perturbed, scatters towards the domain 30 and 60 as can be seen in Figure 6c: note the scatter of perturbed 45 (red) points inside the 30 (green) zone, without any 45 (blue) points. Figure 6d depicts a similar pattern with perturbed domain embedding points (red) scattering towards domains 30 and 0 more than unperturbed domain 15 (blue). For example, between x-axis -1 and 1 dominated by the green domain (domain 30) we see many more red points (perturbed domain 15) than blue points (domain 15). Similarly in the lower right corner of domain 0 shown in yellow. This highlights the mechanism of CrossGrad working; that it is able to augment training with samples closer to unobserved domains.

Finally, we observe in Figure 7 that the embeddings are not correlated with labels. For both domains 30 and 45 the colors corresponding to different labels are not clustered. This is a consequence of CrossGrad’s symmetric training of the domain classifier via label-loss perturbed images.

---

We got similar patterns on t-SNE projections.
4.3 When is Domain Generalization Effective?

We next present a couple of experiments that provide insight into the settings in which CrossGrad is most effective.

First, we show the effect of increasing the number of training domains. Intuitively, we expect CrossGrad to be most useful when training domains are scarce and do not directly cover the test domains. We verified this on the speech dataset where the number of available domains is large. We varied the number of training domains while keeping the test and validation data fixed. Table 8 summarizes our results. Note that CrossGrad outperforms the baseline and LabelGrad most significantly when the number of training domains is small (40). As the training data starts to cover more and more of the possible domain variations, the marginal improvement provided by CrossGrad decreases. In fact, when the models are trained on the full training data (consisting of more than 1000 domains), the baseline achieves an accuracy of 88.3%, and both CrossGrad and LabelGrad provide no gains beyond that. DAN, like in other datasets, provides unstable gains and is difficult to tune. LabelGrad shows smaller relative gains than CrossGrad but follows the same trend of reducing gains with increasing number of domains.

<table>
<thead>
<tr>
<th>Method Name</th>
<th>40 domains</th>
<th>100 domains</th>
<th>200 domains</th>
<th>1000 domains</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>62.2</td>
<td>72.6</td>
<td>79.1</td>
<td>88.3</td>
</tr>
<tr>
<td>LabelGrad</td>
<td>+1.4</td>
<td>+0.1</td>
<td>-0.1</td>
<td>-0.7</td>
</tr>
<tr>
<td>DAN</td>
<td>+0.4</td>
<td>-2.2</td>
<td>+0.7</td>
<td>-3.3</td>
</tr>
<tr>
<td>CrossGrad</td>
<td>+2.3</td>
<td>+0.9</td>
<td>+0.7</td>
<td>-0.4</td>
</tr>
</tbody>
</table>

Table 8: Accuracy on Google Speech Command Task. For each method we show the relative improvement (positive or negative) over the baseline.

In general, how CrossGrad handles multidimensional, non-linear involvement of $g$ in determining $x$ is difficult to diagnose. To initiate a basic understanding of how data augmentation supplies CrossGrad with hallucinated domains, we considered a restricted situation where the discrete domain is secretly a continuous 1-d space, namely, the angle of rotation in MNIST. In this setting, a natural question is, given a set of training domains (angles), which test domains (angles) perform well under CrossGrad?

<table>
<thead>
<tr>
<th>Test domains →</th>
<th>M0</th>
<th>M15</th>
<th>M30</th>
<th>M45</th>
<th>M60</th>
<th>M75</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCSA</td>
<td>84.6</td>
<td>95.6</td>
<td>94.6</td>
<td>82.9</td>
<td>94.8</td>
<td>82.1</td>
</tr>
<tr>
<td>LabelGrad</td>
<td><strong>89.7</strong></td>
<td>97.8</td>
<td>98.1</td>
<td>97.1</td>
<td>96.6</td>
<td><strong>92.1</strong></td>
</tr>
<tr>
<td>DAN</td>
<td>86.7</td>
<td>98.0</td>
<td>97.8</td>
<td>97.4</td>
<td>96.9</td>
<td>89.1</td>
</tr>
<tr>
<td>CrossGrad</td>
<td>88.3</td>
<td><strong>98.6</strong></td>
<td><strong>98.0</strong></td>
<td><strong>97.7</strong></td>
<td><strong>97.7</strong></td>
<td>91.4</td>
</tr>
</tbody>
</table>

Table 9: Accuracy on rotated MNIST.

We conducted leave-one-domain-out experiments by picking one domain as the test domain, and providing the others as training domains. In Table 9 we compare the accuracy of different meth-
ods. We also compare against the numbers reported by the CCSA method of domain generalization (Motiian et al., 2017) as reported by the authors.

It becomes immediately obvious from Table 9 that CROSSGRAD is beaten in only two cases: M0 and M75, which are the two extreme rotation angles. For angles in the middle, CROSSGRAD is able to interpolate the necessary domain representation \( g \) via ‘hallucination’ from other training domains. Recall from Figures 6c and 6d that the perturbed \( g \) during training covers for the missing test domains. In contrast, when M0 or M75 are in the test set, CROSSGRAD’s domain loss gradient does not point in the direction of domains outside the training domains. If and how this insight might generalize to more dimensions or truly categorical domains is left as an open question.

5 CONCLUSION

Domain \( d \) and label \( y \) interact in complicated ways to influence the observable input \( x \). Most domain adaption strategies implicitly consider domain signal to be extraneous and seek to remove its effect to train more robust label predictors. We presented CROSSGRAD, which considers them in a more symmetric manner. CROSSGRAD provides a new data augmentation scheme based on the \( y \) (respectively, \( d \)) predictor using the gradient of the \( d \) (respectively, \( y \)) predictor over the input space, to generate perturbations. Experiments comparing CROSSGRAD with various recent adversarial paradigms show that CROSSGRAD can make better use of partially correlated \( y \) and \( d \), without requiring explicit distributional assumptions about how they affect \( x \). CROSSGRAD is at its best when training domains are scarce and do not directly cover test domains well. Future work includes extending CROSSGRAD to exploit labeled or unlabeled data in the test domain, and integrating the best of the LABELGRAD and CROSSGRAD into a single algorithm.

REFERENCES


