This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Addressing Model Vulnerability to Distributional Shifts over Image Transformation Sets

Riccardo Volpi[1], Vittorio Murino[1,2,*]
[1]Istituto Italiano di Tecnologia [2]Università di Verona
{riccardo.volpi,vittorio.murino}@iit.it

Abstract

We are concerned with the vulnerability of computer vision models to distributional shifts. We formulate a combinatorial optimization problem that allows evaluating the regions in the image space where a given model is more vulnerable, in terms of image transformations applied to the input, and face it with standard search algorithms. We further embed this idea in a training procedure, where we define new data augmentation rules according to the image transformations that the current model is most vulnerable to, over iterations. An empirical evaluation on classification and semantic segmentation problems suggests that the devised algorithm allows to train models that are more robust against content-preserving image manipulations and, in general, against distributional shifts[1].

1. Introduction

When designing a machine learning system, we generally desire it may perform well on a wide realm of different domains. However, the training data at disposal is typically defined by samples from a limited number of distributions, resulting in unsatisfactory performance when the model has to process data from unseen distributions [24, 7, 5, 53]. This problem is typically referred to as *distributional shift* or *domain shift*, and it was shown to affect models even in cases where the distance between training and testing domain is—apparently—very limited [44, 45].

This vulnerability also affects the robustness of machine learning models against input manipulations [17, 22, 23], potentially leading to harmful situations. As a concrete example, consider the algorithms that analyze images uploaded to social networks in order to evaluate, *e.g.*, if an image contains violence or adult content. The huge set of image modifications that users might carry out can make the underlying learning systems fail in several ways if they, accidentally or with malicious intent, cause a *shift* that the models are not able to figure out. Recognizing this weakness of modern learning systems, an important research direction is defining methods to understand *a priori* which distributional shifts will lead to a fail of the model.

Figure 1. *(Top)* Image subdivided in 2 parts, in which the left part is original, and the right part was subject to a content-preserving (appearance) transformation. Image transformations can cause distributional shifts that models are not able to handle *(bottom-left)*. Models trained with the methods proposed in this paper are more robust against a variety of image transformations *(bottom-right)*

In this paper, we start from this idea, and develop methods to evaluate and improve the performance of machine learning models for vision tasks, when the input can be modified through a series of content-preserving image transformations. By "content-preserving" [17], we intend transformations that do not modify an image content, but only the way it is portrayed (*e.g.*, modifying RGB intensities, enhancing contrast, applying filters, etc).

We cast this problem in terms of combinatorial optimization. Given a black-box model, a bunch of samples, and a set of image transformations, our goal is to individuate the distributional shifts that the model is most vulnerable to when image transformation tuples (namely, concatenations of transformations) are applied. To find these tuples, we investigate two different search algorithms—random search and evolution-based search—showing that it is easy to find tuples that severely deteriorate the model performance for a variety of tasks, such as face detection, semantic scene segmentation, and classification. The main application

*VM is also with Huawei Technologies (Ireland) Co., Ltd., Dublin.
[1]Code at github.com/ricvolpi/domain-shift-robustness

for this method as-is, is to evaluate the vulnerability of a machine learning model *before* its deployment. Coupled with proper transformation sets, this tool can indeed be used to verify the robustness of a model under a broad variety of input manipulations and visual conditions.

Furthermore, we introduce a training procedure to learn more robust models against this class of transformations. We design an algorithm where new data augmentation rules are included over iterations, in order to cover the distributional shifts where the current model is more vulnerable. We show that models trained in this way are more robust against content-preserving input image manipulations and, moreover, better generalize to unseen scenarios at test time.

1.1. Background and related work

Vulnerability of learning systems. Recently, the vulnerability of learning systems in different scenarios has gained a lot of attention, in particular in relations with computer vision models (typically, deep convolutional neural networks, or "ConvNets" [28]). A widely studied area is the one related to defense against adversarial perturbations. Gilmer et al. [17] makes a distinction between adversarial samples that are merely *content-preserving* or also *indistinguishable* from the originals. The latter case takes into account imperceptible (to human eye) input perturbations that make a model fail. This paradigm has been extensively studied in a substantial body of works (*e.g.*, [18, 49, 30, 20, 39, 36]).

Instead, we can include a broader range of transformations in the "content-preserving" class. Given some input, the content-preserving transformations are the ones that do not change its content, even if the appearance may change significantly. For example, Gilmer et al. [17] explore the performance of a classifier trained on MNIST [29] when the input is modified by altering the background or adding random lines. Brown et al. [9] show that we can cause model failure by including adversarial patches in an image. Hosseini et al. [22] realize that vision models are vulnerable to negative images. The same research group, in an other work [23], shows that we can find hue and saturation shifts for a given image that a model is vulnerable to. Furthermore, recent works [21, 16] show that state-of-the-art ImageNet [13] models are vulnerable towards simple image modifications. In particular, Hendrycks et al. [21] have found that these models are not resistant towards basic noise sources, and Geirhos et al. [16] have shown that these models are biased towards the texture of the objects.

As stated by Gilmer et al. [17], we also deem that "*the space of content-preserving image transformations remains largely unexplored in the literature*". One of the aims of this work is to help filling this gap, proposing methods to study, generate, and be robust against content-preserving image transformations. Differently from previous works [17, 23], we are not interested in finding adversarial transformations for *single* images. We are instead interested in discovering the distributional shifts that a model is *in general* more vulnerable, applying the same transformation to all the images in the provided set. In this sense, this work is related to Moosavi-Dezfooli et al. [39], where a single, imperceptible perturbation that fools ImageNet models is found.

Robustness against distributional shifts. There is a significant body of works whose goal is overcoming issues related to distributional shift.

One of the main research direction is *domain adaptation* [24, 7, 47, 15, 54, 50, 40, 55], where the goal is to better generalize on domains of interest for which only unlabeled data are available. While there are algorithms that tackle this problem with remarkable results across a variety of tasks, the assumption of an *a priori* fixed target distribution is often too strong. In *domain generalization* [31, 41, 42, 48, 37, 56, 32, 34, 56] the problem of dealing with unseen distributions is coped. Usually, the proposed algorithms start from the assumption that the training dataset comprises a number of different populations. One exception is the method proposed by Volpi et al. [56], where the authors introduce a worst-case formulation that improves generalization performance across distributions close to the training one in the semantic space, using a single-source distribution as starting point. Tobin et al. [52] introduce *domain randomization* for models trained through simulated data. It generates a randomized variety of visual conditions during training, hoping to better generalize when coping with real data.

In this context, the method devised by Volpi et al. [56] is the most related approach to the proposed training strategy (detailed in Section 4) since they are aimed at learning models that better generalize to unseen scenarios, without any assumptions on the number of data populations in the training set. As results will show, the competing algorithm [56] results in models that are only slightly more resistant than the Empirical Risk Minimization (ERM) baseline in the testbed presented in Section 3, and significantly less performing than models trained through the Algorithm proposed in this work in domain generalization settings.

2. Problem formulation

Let $\mathbb{M}$ be a model that takes in input images and provides an output according to the given task. Let $D = \{(x^{(i)}, y^{(i)})\}_{i=0}^{N_D} \sim P(X, Y)$ be a set of datapoints with their labeling, drawn from some data distribution. Finally, let $\mathbb{T} = \{(\tau^{(j)}, l_j^{(k)}), j = 0 \dots N_T, k = 0 \dots N_j\}$ be a set where each object $t = (\tau^{(j)}, l_j^{(k)})$ is a data transformation τ with a related magnitude l. The transformations give in output datapoints in the same format as the input ones (RGB

images throughout this work)[2]. The transformations can be concatenated and repetitions are allowed; we define a composite transformations as a transformation tuple. We define the set of all the possible transformation tuples that one can obtain by combining objects in $\mathbb{T}$ as

$$\mathbb{T}_N = \{\text{all } N-\text{tuples from } \mathbb{T}\} \tag{1}$$

A tuple $T \in \mathbb{T}_N$ is the concatenation of N objects from $\mathbb{T}$, and we define it as $T = (t_1, \dots, t_N)$, with $t_n \in \mathbb{T}$. When we apply the tuple T to a datapoint x, we apply all the transformations from t_1 to t_N. Armed with this set, we propose the following combinatorial optimization problem

$$\min_{T^* \in \mathbb{T}_N} f\big(\mathbb{M}, T^*, D\big) \tag{2}$$

where f is a fitness function that measures the performance of a model $\mathbb{M}$ when provided with some labelled datapoints D, transformed according to the tuple T^*. Assuming that the maximum and minimum values for the metric associated with f are 1 and 0, respectively, we have

$$f : T \longrightarrow [0, 1] \subset \mathbb{R}^1$$

Intuitively, the $N-$tuples that induce lower f values, are the ones that a model $\mathbb{M}$ is more vulnerable to, with respect to the chosen metric. For classifiers, the optimization problem 2 assumes the form

$$\min_{T^* \in \mathbb{T}_N} f := \frac{1}{N_D} \sum_{i=0}^{N_D} \mathbb{1}\big\{y^{(i)} = \mathbb{M}\big(T^*(x^{(i)})\big)\big\} \tag{3}$$

In general, one can define an instance of problem 2 if provided with a set of annotated samples D, a transformation set $\mathbb{T}$ (and, consequently, a tuple set $\mathbb{T}_N$), a model $\mathbb{M}$, a measure to evaluate the performance of the model, and, consequently, a fitness function f. It is not required to have access to the model parameters: it can be a black-box. A legit critique to this formulation is that we are not constraining the transformation tuples to be content-preserving. For instance, in the classification problem 3, a proper formulation would include a constraint similar to the following:

$$\mathcal{O}\big(x^{(i)}\big) = \mathcal{O}\big(T(x^{(i)})\big) \quad \forall i,$$

which means that an oracle $\mathcal{O}(.)$ would classify the transformed images in the same way as the original images. In this work, we do not explicitly constraint the transformation tuples to be content-preserving through the optimization problem. We satisfy the constraint by properly defining the set of available image transformations, *e.g.*, focusing on simple color transformations such as RGB enhancement, contrast/brightness adjustments, and setting a proper value for N in $\mathbb{T}_N$. Explicitly imposing the constraint is an important research direction, since it would allow to consider more complex sets, and we reserve it for future work.

[2]To provide a practical example, one object from $\mathbb{T}$ might be the *"brightness"* operation, and the intensity level might be $+6\%$.

2.1. Transformation set and size of the search space

Given a transformation set $\mathbb{T}$ with N_T available transformations $\tau^{(j)}$, where the j_{th} has N_j available magnitude values, the size of $\mathbb{T}_N$, and consequently the size of the search space of the optimization problem 2, is $S = (\sum_{j=0}^{N_T} N_j)^N$.

In this work, we consider a transformation set $\mathbb{T}$ including standard image transformations from the Python library Pillow [2], as done by Cubuk et al. [12], and a few more we included. It is defined by the following transformations, with the number of available intensity levels indicated in parenthesis: *autocontrast (20), sharpness (20), brightness (20), color (20), contrast (20), grayscale conversion (1), R-channel enhancer (30), G-channel enhancer (30), B-channel enhancer (30), solarize (20)*. The description of the various transformations is reported in the Supplementary Material, as well as the ranges of intensity levels. This set results in a search space with size $S = 211^N$. Throughout this work, we will consider tuple sets $\mathbb{T}_N$ with $N = 3$ and $N = 5$, resulting in search spaces with size in the order of $\sim 10^6$ and $\sim 10^{12}$, respectively.

3. Searching worst-case image transformations

In this section, we analyze different solutions to face the combinatorial optimization problem 2. Specifically, the two approaches rely on random search and evolution-based [38] search. We provide a proof of concept experiment on MNIST models, and report a more exhaustive experimental evaluation in Section 5.

3.1. Random search.

Facing the optimization problem 2 through random search is important for several reasons. First, it is the simplest approach that we can adopt, thus it is worth to be explored. Further, random search is often a very strong baseline to compare against, as shown, *e.g.*, in hyper-parameter optimization [6] and neural architecture search [33]. Finally, it sheds light on a relevent question: *how is a model affected by random image transformations?*

The idea is to evaluate the fitness function f over an arbitrary number of random transformation tuples, thus the implementation is straightforward. For clarity and reproducibility, we detail it step-by-step on Algorithm 1. In the following, we will refer to this procedure as RS (short for Random Search)

3.2. Evolution-based search.

We define a simple genetic algorithm [38], aimed at minimizing the objective in problem 2. Each individual of the population is defined by a transformation $N-$tuple from a set $\mathbb{T}_N$. We define standard *Selection*, *Crossover* and *Mutation* operations. For a detailed explanation of genetic algorithms and the definitions we provided, we refer to [38]. In

Algorithm 1 RS (Random Search)

Input: N-tuple set $\mathbb{T}_N$, model $\mathbb{M}$, dataset $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N_D}$, fitness function f.
Output: transformation $T \in \mathbb{T}_N$
Initialize: $f_{min} \leftarrow 1$.
for $k = 1, ..., K$ **do**
    Sample T^* uniformly from $\mathbb{T}_N$
    $f^* \leftarrow f(\mathbb{M}, T^*, D)$
    if $f^* < f_{min}$ **then**
        $T \leftarrow T^*$

the following, we briefly discuss how we use these concepts in our framework.

- **Selection.** Given in input a population $\texttt{pop} = \{T^p\}_{p=1}^P$, the fitness score of each individual $\texttt{fit} = \{f^p\}_{p=1}^P$, and a positive integer $\hat{P}$, returns in output a population of $\hat{P}$ individuals sampled from $\texttt{pop}$ with individual probabilities proportional to $\frac{1}{f^p}$.

- **Crossover.** Given in input two initialized populations $\texttt{pop1} = \{T^p\}_{p=1}^P$, where $T^p = (t_1^p, \ldots, t_N^p)$ and $\texttt{pop2} = \{\tilde{T}^p\}_{p=1}^P$, where $\tilde{T}^p = (\tilde{t}_1^p, \ldots, \tilde{t}_N^p)$, for each couple of elements $\{(T^p, \tilde{T}^p)\}_{p=1}^P$ we uniformly draw an integer $n \in [1, N]$ and return the following two individuals: $T^{p,1} = (t_1^p, \ldots, t_n^p, \tilde{t}_{n+1}^p, \ldots, \tilde{t}_N^p)$ and $T^{p,2} = (\tilde{t}_1^p, \ldots, \tilde{t}_n^p, t_{n+1}^p, \ldots, t_N^p)$. The output is the population defined by the $2P$ new individuals.

- **Mutation.** Given in input an initialized population $\texttt{pop1} = \{T^p\}_{p=1}^P$ and a mutation rate η, it changes each transformation of each tuple in $\texttt{pop}$ with probability η, sampling from $\mathbb{T}$.

Endowed of these methods, we implement an evolution-based search procedure, detailed in Algorithm 2. The complexity is $\mathcal{O}(PK)$, where P is the population size and K is the number of evolutionary steps. Notice that the operations associated with lines 5 and 11, namely computing the fitness function value for each transformation in the population, constitute the computationally expensive part of the algorithm. For each run, we perform $P(K+1)$ fitness function evaluations. In the following, we will refer to this procedure as ES (short for Evolution-based Search).

3.3. Proof of concept: MNIST

The MNIST dataset [29] is defined by 28×28 pixel images, representing white digits on a black background. It is divided into a $50,000$ sample training set and a $10,000$ sample test set. In our experiments, we train a small ConvNet (*conv-pool-conv-pool-fc-fc-softmax*) on the whole training set, via backpropagation [46]. We resize the images to 32×32 pixels, in order to be comparable with other digit datasets (in view of the domain generalization experiments

Algorithm 2 ES (Evolution-based Search)

1: **Input:** N-tuple set $\mathbb{T}_N$, model $\mathbb{M}$, dataset $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N_D}$, fitness function f, population size P, mutation rate η.
2: **Output:** transformation $T \in \mathbb{T}_N$
3: **Initialize:** $f_{min} \leftarrow 1$.
4: $\texttt{pop} \leftarrow \{T^p\}_{p=1}^P$ sampling from $\mathbb{T}_N$
5: $\texttt{fit} \leftarrow \{f^p \leftarrow f(\mathbb{M}, T^p, D)\}_{p=1}^P$
6: **for** $k = 1, ..., K$ **do**
7: $\texttt{newpop1} \leftarrow$ **Select**$(\texttt{pop}, \texttt{fit}, \frac{P}{2})$
8: $\texttt{newpop2} \leftarrow$ **Select**$(\texttt{pop}, \texttt{fit}, \frac{P}{2})$
9: $\texttt{pop} \leftarrow$ **Crossover**$(\texttt{newpop1}, \texttt{newpop2})$
10: $\texttt{pop} \leftarrow$ **Mutation**$(\texttt{pop}, \eta)$
11: $\texttt{fit} \leftarrow \{f^p \leftarrow f(\mathbb{M}, T^p, D)\}_{p=1}^P$
12: **for** $p = 1, ..., P$ **do**
13: **if** $\texttt{fit}[p] < f_{min}$ **then**
14: $T \leftarrow \texttt{pop}[p]$

reported in Section 5). We apply the search algorithms (RS and ES) on problem 2 using $1,000$ samples from the test set. We set $N = 3$, namely, we use transformation tuples defined by three transformations.

The blue curve in Figure 2 is the density plot associated with all the fitness function values obtained while running RS for $K = 10,000$ iterations, using as model $\mathbb{M}$ the trained ConvNet—that achieves 99.3% accuracy on the clean test set. The accuracy values are reported on the x-axis. Values lower than the one indicated by the black flag have less than 0.1% probability to be achieved by transforming the input through transformation tuples sampled from $\mathbb{T}_N$. This plot provides a glance on the vulnerability of MNIST models to the image transformations included in our set. It shows that there is a substantial mass of transformation tuples that the model is resistant to, but, even though with lower probability to be sampled, there are transformation tuples against which the model is severely vulnerable.

Table 1 (RS row) shows the minimum accuracy obtained in $10,000$ evaluations of the fitness function f, averaged over 6 different models. We report results associated with both models trained via standard ERM (homonymous column) and models trained through the method proposed by Volpi et al. [56] (*"ADA"* column). As one can observe, both types of models are severely vulnerable to the transformation tuples found through RS. For comparison with previous work, we also report results obtained on negative images [22] and on images with random hue/value perturbations [23] (for the latter, we used the original code).

We proceed by approaching problem 2 through ES, setting population size $P = 10$, number of generations $K = 99$ and mutation rate $\eta = 0.1$. With this setting, the number of fitness evaluations is $1,000$. The red flags in Figure 2 indicate the f_{min} values achieved on 6 different runs, using the same ConvNet as in the RS experiment. A comparison

Performance of MNIST models

Test	Training procedure				
	ERM	ADA [56]	RDA	RSDA	ESDA
Original	.993 ± .001	.992 ± .001	.993 ± .001	.992 ± .001	.993 ± .001
RS	.160 ± .024	.192 ± .025	.941 ± .011	.977 ± .001	.979 ± .003
ES	.122 ± .028	.177 ± .042	.927 ± .007	.979 ± .005	.977 ± .000
Neg. [22]	.436 ± .042	.448 ± .046	.991 ± .001	.992 ± .001	.992 ± .001
SAE [23]	.979 ± .005	.980 ± .005	.985 ± .004	.974 ± .007	.971 ± .016

Table 1. Accuracy values associated with MNIST models in different training/testing conditions, averaged over 6 different training runs. Each row is associated with a different test: *Original* refers to performance achieved on clean test samples; *RS* and *ES* refer to results obtained applying the transformations found via RS and ES, respectively; *Neg* and *SAE* are related to adversarial attacks detailed in [22] and [23], respectively. Each column is related to models trained with a different procedure.

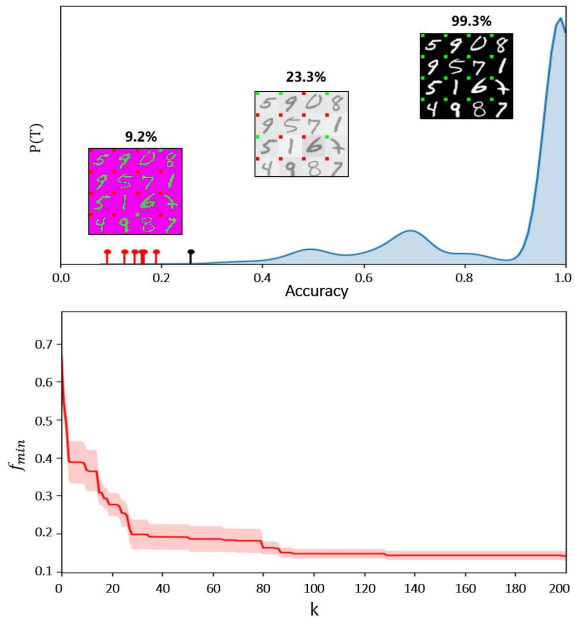

Figure 2. *Top.* The blue curve represents the density plot related to $10,000$ transformation tuples uniformly sampled for RS. The black flag represents the 0.1% probability threshold for RS. The red flags represent the accuracy values obtained via ES. Three examples of sample sets resulting from different transformations are reported, with green and red squares indicating whether the sample is mis-classified or not, respectively. *Bottom.* Evolution of f_{min} value averaged over 6 runs of ES.

between the 0.1% threshold (black flag) and the results obtained via evolution shows that ES allows to efficiently find low-probability transformation tuples that the model is most vulnerable to. Furthermore, even though we set $K = 99$, EF can find transformation tuples that go beyond the 0.1% threshold in less iterations. We report this evidence in Figure 2 (bottom), which shows the evolution of f_{min} during 200 iterations of ES. Comparing this result with the ones pictured in Figure 2 (top), one can observe that even by setting $K \simeq 30$ (310 fitness function evaluations), ES outperforms the 0.1% threshold in the RS results. We report numerical results in Table 1 (ES row), where we average

Algorithm 3 Robust Training

Input: $D = \{x^{(i)}, y^{(i)}\}_{i=1}^{N_D}$, initialized weights θ_0, $N-$tuple set $\mathbb{T}_N$, initialized data augmentation set $\mathbb{T}_{tr}$, learning rate α.
Output: learned weights θ
Initialize: $\theta \leftarrow \theta_0$
for $h = 1, ..., H$ **do**
  for $j = 1, ..., J$ **do**
    Sample (x, y) uniformly from D
    Sample T uniformly from $\mathbb{T}_{tr}$
    $\theta \leftarrow \theta - \alpha \nabla_\theta \ell(\theta; (T\{x\}, y))$
  Find $T^* \in \mathbb{T}_N$ by running RS or ES on a subset of D
  Append T^* to $\mathbb{T}_{tr}$
while training is not done **do**
  Sample (x, y) uniformly from dataset
  Sample T uniformly from $\mathbb{T}_{tr}$
  $\theta \leftarrow \theta - \alpha \nabla_\theta \ell(\theta; (T\{x\}, y))$

over 6 models the lowest f_{min} achieved over 6 runs of ES with different initializations. In Section 5, we will provide a more exhaustive analysis of the efficacy of RS and ES to approach different instances of problem 2.

4. Training more robust models

In this section, we detail two straightforward methods devised to train models that are robust against content-preserving transformations from a given set.

The simplest approach that one can devise is likely the following: given a set $\mathbb{T}_N$, we can perform data augmentation by sampling transformation tuples $T \in \mathbb{T}_N$ and applying them to the training images throughout the training procedure. We term this method *Randomized Data Augmentation*, in short RDA. This technique can be interpreted as an application of domain randomization [52] to real data instead of simulated ones.

Drawing inspiration from the literature related to adversarial robustness [18, 49, 56], where a loss is minimized with respect to adversarially perturbed inputs, we devise a method that is more effective than RDA in our setting. We propose a training procedure where transformation tuples that the current model is most vulnerable to are searched throughout the training procedure (via RS or ES), and data augmentation is performed according to the so-found transformations. We implement this idea as follows: (a) we define a transformation set to sample from during training (the "data augmentation set" $\mathbb{T}_{tr}$), that at the beginning of training only comprises the *identity* transformation; (b) we train the network via gradient descent updates [10], augmenting samples by applying transformations uniformly sampled from $\mathbb{T}_{tr}$ (in this work, the loss ℓ used is the cross-entropy function between the output of the model and the ground truth labels); (c) we run RS or ES, using appropriate fitness function f and tuple set $\mathbb{T}_N$, and append the so-

found transformation tuple to $\mathbb{T}_{tr}$. We alternate between steps (b) and (c) for the desired number of times, and (d) we repeat step (b) until the value of the loss ℓ is satisfactory. The procedure is also detailed in Algorithm 3.

As results will show, the latter method performs significantly better than RDA in several settings. In the next Sections, we will refer to this method as RSDA or ESDA, short for *Random Search Data Augmentation* and *Evolution-based Search Data Augmentation*, respectively

5. Experiments

In Section 3.3, we provided a first evidence that the problem formulation introduced in Section 2 can be useful to detect harmful distributional shifts for a given model—in terms of image transformations. In Section 4, we introduced different methods to train more robust models.

In this section, we further validate the effectiveness of RS and ES on different instances of problem 2, associated with models for classification, semantic segmentation and face detection. Furthermore, we evaluate the performance of classification and semantic segmentation models trained through RDA, RSDA and ESDA, assessing both their robustness against image transformations and their domain generalization properties. When we search for transformations while running RSDA and ESDA (Algorithm 3, line 9), we set $K = 100$ for RS and $K = 10$ for ES. When we apply ES, we set number of individual $P = 10$ and mutation rate $\eta = 0.1$ throughout the entire analysis. We use accuracy as evaluation metric in all the experiments.

5.1. Digit Recognition

Experimental setup. We adopt the same experimental setting detailed in Section 3.3. We train models via ERM and RDA for 10^6 gradient descent updates. When we train models through RSDA/ESDA, we set $J = 10^4$ and $H = 100$, running a total of 10^6 weight updates also in this case. We use a subset of $1,000$ samples from the training set when we run RS/ES (Algorithm 3, line 9). In all the experiments, we set the size of the transformation tuples as $N = 3$. We use Adam [26] as optimizer, with learning rate $\alpha = 3 \cdot 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

In addition to assessing model vulnerability against the transformations found via RS and ES, we also evaluate the domain generalization capabilities of MNIST models, testing on different, unseen digit datasets (SVHN [43], SYN [15], MNIST-M [15], USPS [14]), following the evaluation protocol used by Volpi et al. [56]. Samples from every dataset were resized to 32×32 pixels and treated as RGB images, to be comparable. Notice that we do not use any sample from other dataset than MNIST during training.

Results. In Section 3.3 we showed that our setup allows to find transformation tuples that lower the accuracy of MNIST models to values as low as $\sim 12\%$ (Table 1 – ES row, ERM column). We are now interested in evaluating the performance on MNIST models trained through the methods detailed in Section 4 (RDA, RSDA and ESDA). Table 1 (last three columns) shows the performance of models trained with the proposed methods. The most robust model is the one trained through ESDA, for which the accuracy related to each testing case is greater than 97%. All our models are resistant to the negative operation applied to the images [22], with accuracy values greater than 99%. An important result is that there is not a statistically significant accuracy loss on original samples (Table 1 – first row).

Domain generalization performance of MNIST models

Training Method	Testing dataset: SVHN	SYN	MNIST-M	USPS
ERM	$.365 \pm .021$	$.477 \pm .015$	$.590 \pm .012$	$.812 \pm .013$
ADA [56]	$.391 \pm .017$	$.482 \pm .019$	$.595 \pm .013$	$.819 \pm .016$
RDA	$.395 \pm .017$	$.603 \pm .005$	$.751 \pm .013$	$.832 \pm .013$
RSDA	$.474 \pm .048$	$.620 \pm .012$	$.815 \pm .016$	$.831 \pm .012$
ESDA	$.489 \pm .052$	$.622 \pm .013$	$.816 \pm .016$	$.840 \pm .012$

Table 2. Performance of MNIST models trained with different methods *(rows)* and evaluated on test samples from different digit datasets *(columns)*. Results computed by averaging over 6 different runs.

Having confirmed that we can train more robust models against the types of perturbations introduced in this work, we are interested in evaluating the performance in the domain generalization testbed; Table 2 reports our findings. Also in this setting, we observe that models trained via ESDA are the most robust against distributional shifts. Models trained via RSDA are slightly less performing, but significantly more robust than the ones trained via RDA in different test cases. The more significant result is that, when testing on SVHN, there is $\sim 10\%$ gap when comparing RDA and ESDA. Furthermore, despite the transformation set $\mathbb{T}$ used is biased towards color transformations, we can observe improved performance with respect to ERM also when testing on USPS, whose samples differ from MNIST ones only in their shape.

5.2. CIFAR-10 Classification

Experimental setup. We use the CIFAR-10 [27] dataset, and train Wide Residual Network models (WRN [57]) on the provided training set. We have chosen this class of models because they are broadly used in the community and they provide results competitive with the state of the art on CIFAR-10. We train networks with $16-$layers and set the width to 4, choosing a trade-off between accuracy and training/testing speed, among the recipes proposed in the original work [57]. We use the original code provided by the authors [3].

When training ERM and RDA the models, we follow the procedure proposed in [57], and run stochastic gradient descent with momentum 0.9 for 200 epochs, starting with a learning rate 0.1 and decaying it at epochs 60, 120 and 160. When training RSDA and ESDA models, we observed that the learning procedure is eased if the new augmentation rules are included while we are training the model with a large learning rate. For this reason, we start the learning rate decay after having searched for a satisfactory number of transformations. In the results proposed in this section, we search for 100 different ones, and each search procedure is followed by 5 epochs of training.

We set the size of the tuples in $\mathbb{T}_N$ as $N = 5$; with respect to the transformation set $\mathbb{T}$ described in Section 2.1, we do not include *solarize* and *grayscale*. When we search for transformations, we use $2,000$ samples drawn from the training set. When we test the models, we search for transformations through RS and ES using the whole test set. We run RS with $K = 1,000$ iterations and three runs of ES with $K = 100$ iterations; the results reported in the next paragraph are associated with the optimal f_{min} found. In addition to testing the model vulnerability against such transformations, we also evaluate the domain generalization capabilities of WRN models, assessing the performance on CIFAR-10.1 [44] dataset and on STL [11] dataset. We remove samples associated with the class "monkey", not present in the CIFAR-10 dataset, and resize images to 32×32, to be comparable.

Results. Table 3 reports the achieved results. The "ERM" column, which shows the results obtained by testing baseline models in different conditions, confirms the results we observed in the MNIST experiment, although with less dramatic effects. Indeed, we can find transformation tuples that the model is significantly vulnerable to, by using RS ($\sim 72\%$) and ES ($\sim 56\%$, with a larger standard deviation). Concerning models trained with our methods, also in this experiment RDA represents an effective strategy, but RSDA and ESDA allow to train more robust models, with respect to the transformations we are testing against.

Furthermore, the last row, reporting results obtained when testing on STL dataset, confirms the domain generalization capabilities of models trained with our method; using Algorithm 3, we can observe $\sim 7\%$ improvement in accuracy, when compared against ERM. When testing on CIFAR 10.1, the benefits are less marked, but still noticeable in the RSDA case. Each accuracy value reported was obtained by averaging over 3 different runs.

5.3. Semantic Scene Segmentation

Experimental setup. We train FC-DenseNet [25] models on the CamVid [8] dataset. We use the $103-$layer version of the model, relying on an open-source implementation [4].

Performance of CIFAR-10 models

	Training procedure			
Test	**ERM**	**RDA**	**RSDA**	**ESDA**
Original	$.946 \pm .000$	$.944 \pm .002$	$.950 \pm .002$	$.946 \pm .000$
RS	$.724 \pm .026$	$.899 \pm .004$	$.904 \pm .016$	$.915 \pm .000$
ES	$.565 \pm .149$	$.862 \pm .012$	$.867 \pm .050$	$.913 \pm .004$
10.1 [44]	$.872 \pm .004$	$.873 \pm .007$	$.886 \pm .009$	$.878 \pm .003$
STL [11]	$.466 \pm .009$	$.503 \pm .009$	$.526 \pm .007$	$.534 \pm .009$

Table 3. Performance of CIFAR-10 models trained with different methods *(columns)* and tested in different conditions *(rows)*. The *10.1* row reports results obtained by testing on CIFAR-10.1 [44]; the *STL* row reports the ones related to STL [11]. Results computed by averaging over 3 runs.

Also in this case, the choice of the model is due to its success with respect to the analyzed benchmark. The CamVid dataset contains 367 training images, 101 validation images and 233 testing images from 32 classes. We lower the sample resolution from 960×720 to 480×360, and train the models for 300 epochs.

When we train using RSDA/ESDA, we run RS and ES on 30 samples from the training set, and search for new transformations every 10 epochs. We set the size of the $N-$tuples as $N = 5$. With respect to the transformation set $\mathbb{T}$ introduced in Section 2.1, we do not include *solarize* and *grayscale*. When we test the vulnerability of the models, we run RS (with $K = 500$) and three different runs of ES (with $K = 50$) on 30 samples from the test set. As for previous experiments, we report results related to the minimum f_{min} values found. Notice that the output of semantic segmentation models is richer than the output of classification models, since a prediction is associated with each pixel; indeed, 30 samples lead to $30 \cdot 480 \cdot 360$ pixel predictions. We use pixel accuracy as a metric [35]. We use RMSprop [51] as optimizer, with decay 0.001.

Results. Table 4 reports the results we obtained. They confirm the higher level of robustness of models trained via RDA, RSDA and ESDA. In this experiment though, we can observe a narrower gap between RDA and RSDA/ESDA.

Figure 3 shows the output of a model trained via ESDA (middle) and the output of a model trained via standard ERM (bottom), when the original input (first column, top) is perturbed with different image transformations (top). These results not only qualitatively show the better performance of ESDA, but also that the transformation tuples we are sampling from $\mathbb{T}_N$ are realistic approximations of possible visual conditions that a vision module (for instance, for a self-driving car) might encounter. For example, images in the middle row, second and third column, can be considered as simulations of the light conditions that one could encounter during dawn or sunset—and in which the baseline model performs poorly.

Performance of CamVid models				
	Training procedure			
Test	ERM	RDA	RSDA	ESDA
Original	$.862 \pm .007$	$.851 \pm 004$	$.854 \pm 003$	$.851 \pm .002$
RS	$.458 \pm .027$	$.812 \pm .007$	$.825 \pm .009$	$.820 \pm .007$
ES	$.311 \pm .013$	$.811 \pm .011$	$.824 \pm .008$	$.822 \pm .008$

Table 4. Performance of CamVid models trained with different methods *(columns)* and tested in different conditions *(rows)*. Results computed by averaging over 3 different runs.

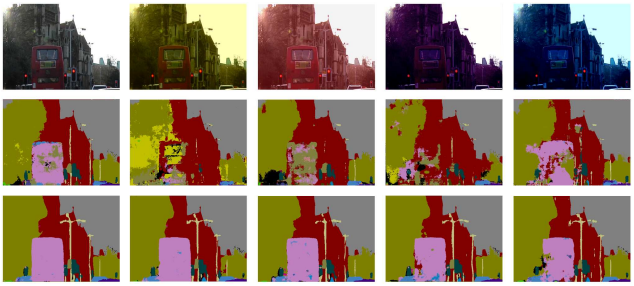

Figure 3. A sample from CamVid *(first column, first row)* modified with various image transformations. *Second* and *last rows* report the outputs of models trained via standard ERM and via Algorithm 3, respectively.

5.4. Face detection

Experimental setup. We test our search methods on a widely used API for face detection [1], that takes RGB images as input and provides in output the locations of the faces in the image. We use four subsets of $1,000$ images uniformly sampled from the MS-Celeb-1M [19], resized to 64×64, as datapoints in input to RS and ES. Each image contains one celebrity face, thus the API gives in output a single location if it detects a face, or nothing otherwise. In practical terms, due to the nature of the input, we can interpret the API as a binary model and test it through the optimization problem 3. We set the number of transformations as $N = 3$ and $N = 5$. For each N value and each subset of faces, we run RS with $K = 15,000$ iterations, and run 6 different runs of ES with $K = 100$. We average results over the optimal f_{min} values obtained in the four subsets. With respect to the transformation set $\mathbb{T}$ depicted in Section 2.1, we do not include *solarize*.

Results. Table 5 reports the accuracy values obtained, and Figure 4 reports different examples of faces modified through the transformation tuples found via RS and ES. Green and red squares indicate whether the API has detected or not a face, respectively. Qualitatively, we observed that the model tends to fail when the input manipulation is such that some facial features are no longer visible or deteriorated (for example, the nose). The importance of these vulnerabilities depends on the different API use cases. For example, vulnerability to some grayscale tones might not matter for a model that deals with images recorded in the streets, but it might matter for a social network application. Vulnerability to extreme brightness conditions can be harmful for a street camera, where the broad variety of possible visual conditions might not allow to have a proper view of the facial features. One strength of the search methods we proposed is that they allow users to set transformation sets according to the applications they are concerned about.

Performance of Face Detection API [1]				
Original Accuracy	RS N=3	ES N=3	RS N=5	ES N=5
$.878 \pm .007$	$.789 \pm .010$	$.705 \pm .069$	$.516 \pm .014$	$.174 \pm .134$

Table 5. Accuracy of the Face Detection API in different testing conditions, analogously to the previous tests reported in this work.

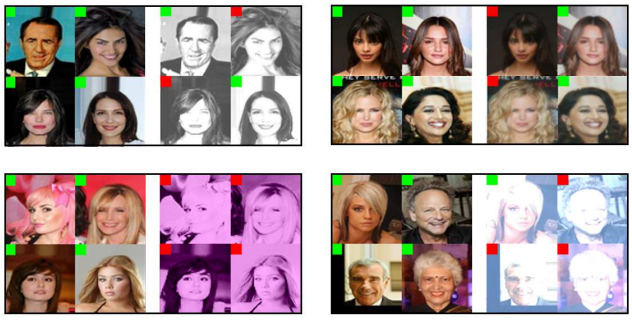

Figure 4. Four different examples of transformations found via RS and ES. Green and red squares indicate whether the API has detected the face or not, respectively.

6. Conclusions

We propose a combinatorial optimization problem to find distributional shifts that a given model is vulnerable to, in terms of $N-$tuples of image transformations. We show that random search and, in particular, evolution-based search are effective approaches to face this problem. Further, we show that the same search algorithms can be exploited in a training procedure, where harmful distributional shifts are searched and harnessed. We report results for a variety of tasks (classification, segmentation and face detection), showing that the problem formulation is flexible and can be adopted in different circumstances.

Among others, some valuable directions for future works consist in (i) the implementation of more effective methods to approach the optimization problem 2, in order to find more harmful transformations with reduced computational cost, (ii) the analysis of more complex transformation sets, and (iii) the definition of a proper content-preserving constraint in the optimization problem.

Acknowledgments. We are grateful to Jacopo Cavazza and Federico Marmoreo for helpful discussions concerning the problem formulation proposed in this work.

References

[1] Face recognition. https://github.com/ageitgey/face_recognition. 8

[2] Python imaging library. https://github.com/python-pillow/Pillow. 3

[3] Pytorch training code for wide residual networks. https://github.com/szagoruyko/wide-residual-networks/tree/master/pytorch. 6

[4] Semantic segmentation suite. https://github.com/GeorgeSeif/Semantic-Segmentation-Suite. 7

[5] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 137–144. MIT Press, 2007. 1

[6] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13:281–305, Feb. 2012. 3

[7] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 120–128, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics. 1, 2

[8] Gabriel J. Brostow, Jamie Shotton, Julien Fauqueur, and Roberto Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV (1)*, pages 44–57, 2008. 7

[9] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *CoRR*, abs/1712.09665, 2017. 2

[10] Augustin-Louis Cauchy. Méthode générale pour la résolution des systémes d'équations simultanées. *Comptes Rendus Hebd. Séances Acad. Sci.*, 25:536–538, 1847. 5

[11] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudk, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. 7

[12] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. 3

[13] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2

[14] J. S. Denker, W. R. Gardner, H. P. Graf, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, H. S. Baird, and I. Guyon. Advances in neural information processing systems 1. chapter Neural Network Recognizer for Hand-written Zip Code Digits, pages 323–331. 1989. 6

[15] Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1180–1189, 2015. 2, 6

[16] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019. 2

[17] Justin Gilmer, Ryan P. Adams, Ian J. Goodfellow, David Andersen, and George E. Dahl. Motivating the rules of the game for adversarial example research. *CoRR*, abs/1807.06732, 2018. 1, 2

[18] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 2, 5

[19] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. MS-Celeb-1M: A dataset and benchmark for large scale face recognition. In *European Conference on Computer Vision*. Springer, 2016. 8

[20] Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2017. 2

[21] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 2

[22] Hossein Hosseini and Radha Poovendran. Deep neural networks do not recognize negative images. *CoRR*, abs/1703.06857, 2017. 1, 2, 4, 5, 6

[23] Hossein Hosseini and Radha Poovendran. Semantic adversarial examples. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018. 1, 2, 4, 5

[24] Hal Daumé III and Daniel Marcu. Domain adaptation for statistical classifiers. *CoRR*, abs/1109.6341, 2011. 1, 2

[25] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *arXiv e-prints*, abs/1611.09326, 2016. 7

[26] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6

[27] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009. 6

[28] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 2

[29] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998. 2, 4

[30] Jaeho Lee and Maxim Raginsky. Minimax statistical learning and domain adaptation with wasserstein distances. *arXiv preprint arXiv:1705.07815*, 2017. 2

[31] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. *The IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[32] Da Li, Jianshu Zhang, Yongxin Yang, Cong Liu, Yi-Zhe Song, and Timothy M. Hospedales. Episodic training for domain generalization. *CoRR*, abs/1902.00113, 2019. 2

[33] Liam Li and Ameet Talwalkar. Random search and reproducibility for neural architecture search. *CoRR*, abs/1902.07638, 2019. 3

[34] Yiying Li, Yongxin Yang, Wei Zhou, and Timothy M. Hospedales. Feature-critic networks for heterogeneous domain generalization. *CoRR*, abs/1901.11448, 2019. 2

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. 7

[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 2

[37] Massimiliano Mancini, Samuel Rota Bulò, Barbara Caputo, and Elisa Ricci. Robust place categorization with deep domain generalization. *IEEE Robotics and Automation Letters*, 3(3):2093–2100, July 2018. 2

[38] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA, 1998. 3

[39] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94, 2017. 2

[40] Pietro Morerio, Jacopo Cavazza, and Vittorio Murino. Minimal-entropy correlation alignment for unsupervised deep domain adaptation. *International Conference on Learning Representations*, 2018. 2

[41] Saeid Motiian, Marco Piccirilli, Donald A. Adjeroh, and Gianfranco Doretto. Unified deep supervised domain adaptation and generalization. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[42] Krikamol Muandet, David Balduzzi, and Bernhard Schlkopf. Domain generalization via invariant feature representation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 10–18, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. 2

[43] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. 6

[44] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do CIFAR-10 classifiers generalize to cifar-10? *CoRR*, abs/1806.00451, 2018. 1, 7

[45] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *CoRR*, abs/1902.10811, 2019. 1

[46] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986. 4

[47] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 213–226, Berlin, Heidelberg, 2010. Springer-Verlag. 2

[48] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. Generalizing across domains via cross-gradient training. In *International Conference on Learning Representations*, 2018. 2

[49] Aman Sinha, Hongseok Namkoong, and John Duchi. Certifiable distributional robustness with principled adversarial training. In *International Conference on Learning Representations*, 2018. 2, 5

[50] Baochen Sun and Kate Saenko. Deep CORAL: correlation alignment for deep domain adaptation. In *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III*, pages 443–450, 2016. 2

[51] Tijmen Tieleman and Geoffrey E. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning, 2012. 7

[52] Joshua Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *CoRR*, abs/1703.06907, 2017. 2, 5

[53] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 1521–1528, Washington, DC, USA, 2011. IEEE Computer Society. 1

[54] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[55] Riccardo Volpi, Pietro Morerio, Silvio Savarese, and Vittorio Murino. Adversarial feature augmentation for unsupervised domain adaptation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[56] Riccardo Volpi*, Hongseok Namkoong*, Ozan Sener, John Duchi, Vittorio Murino, and Silvio Savarese. Generalizing to unseen domains via adversarial data augmentation. In *Advanced in Neural Information Processing Systems (NeurIPS) 32*, December 2018. 2, 4, 5, 6

[57] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 6, 7