Self-Challenging Improves Cross-Domain Generalization

Zeyi Huang*, Haohan Wang*, Eric P. Xing, and Dong Huang

School of Computer Science, Carnegie Mellon University
{zeyih@andrew, haohanw@cs, epxing@cs, donghuang@}.cmu.edu

Abstract. Convolutional Neural Networks (CNN) conduct image classification by activating dominant features that correlated with labels. When the training and testing data are under similar distributions, their dominant features are similar, leading to decent test performance. The performance is nonetheless unmet when tested with different distributions, leading to the challenges in cross-domain image classification. We introduce a simple training heuristic, Representation Self-Challenging (RSC), that significantly improves the generalization of CNN to the out-of-domain data. RSC iteratively challenges (discards) the dominant features activated on the training data, and forces the network to activate remaining features that correlate with labels. This process appears to activate feature representations applicable to out-of-domain data without prior knowledge of the new domain and without learning extra network parameters. We present the theoretical properties and conditions of RSC for improving cross-domain generalization. The experiments endorse the simple, effective, and architecture-agnostic nature of our RSC method.

Keywords: cross-domain generalization, robustness

1 Introduction

Imagine teaching a child to visually differentiate "dog" from "cat": when presented with a collection of illustrations from her picture books, she may immediately answer that "cats tend to have chubby faces" and end the learning. However, if we continue to ask for more differences, she may start to notice other features like ears or body-size. We conjecture this follow-up challenge question plays a significant role in helping human reach the remarkable generalization ability. Most people should be able to differentiate "cat" from "dog" visually even when the images are presented in irregular qualities. After all, we did not stop learning after we picked up the first clue when we were children, even the first clue was good enough to help us recognize all the images in our textbook.

Nowadays, deep neural networks have exhibited remarkable empirical results over various computer vision tasks, yet these impressive performances seem unmet when the models are tested with the samples in irregular qualities [32] (*i.e.*,

* equal contribution; codes are available at https://github.com/DeLightCMU/RSC

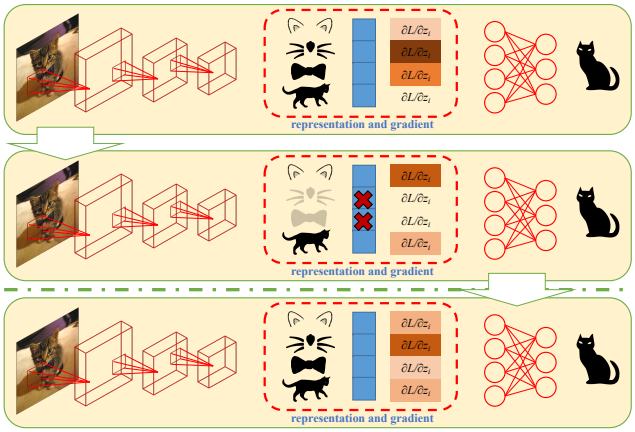

Fig. 1. The essence of our Representation Self-Challenging (RSC) training method: top two panels: the algorithm mutes the feature representations associated with the highest gradient, such that the network is forced to predict the labels through other features; bottom panel: after training, the model is expected to leverage more features for prediction in comparison to models trained conventionally.

out-of-domain data, samples collected from the distributions that are similar to, but different from the distributions of the training samples). To account for this discrepancy, technologies have been invented under the domain adaptation regime [2, 3], where the goal is to train a model invariant to the distributional differences between the source domain (*i.e.*, the distribution of the training samples) and the target domain (*i.e.*, the distribution of the testing samples) [5, 33].

As the influence of machine learning increases, the industry starts to demand the models that can be applied to the domains that are not seen during the training phase. Domain generalization [18], as an extension of domain adaptation, has been studied as a response. The central goal is to train a model that can align the signals from multiple source domains.

Further, Wang *et al.* extend the problem to ask how to train a model that generalizes to an arbitrary domain with only the training samples, but not the corresponding domain information, as these domain information may not be available in the real world [31]. Our paper builds upon this set-up and aims to offer a solution that allows the model to be robustly trained without domain information and to empirically perform well on unseen domains.

In this paper, we introduce a simple training heuristic that improves cross-domain generalization. This approach discards the representations associated with the higher gradients at each epoch, and forces the model to predict with remaining information. Intuitively, in a image classification problem, our heuristic works like a "self-challenging" mechanism as it prevents the fully-connected layers to predict with the most predictive subsets of features, such as the most frequent color, edges, or shapes in the training data. We name our method Representation Self Challenging (RSC) and illustrate its main idea in Figure 1.

We present mathematical analysis that RSC induces a smaller generalization bound. We further demonstrate the empirical strength of our method with domain-agnostic cross-domain evaluations, following previous setup [31]. We also conduct ablation study to examine the alignment between its empirical performance and our intuitive understanding. The inspections also shed light upon the choices of its extra hyperparameter.

2 Related Work

We summarize the related DG works from two perspectives: learning domain invariant features and augmenting source domain data. Further, as RSC can be broadly viewed as a generic training heuristic for CNN, we also briefly discuss the general-purpose regularizations that appear similar to our method.

DG through Learning Domain Invariant Features: These methods typically minimize the discrepancy between source domains assuming that the resulting features will be domain-invariant and generalize well for unseen target distributions. Along this track, Muandet *et al.* employed Maximum Mean Discrepancy (MMD) [18]. Ghifary *et al.* proposed a multi-domain reconstruction auto-encoder [10]. Li *et al.* applied MMD constraints to an autoencoder via adversarial training [15].

Recently, meta-learning based techniques start to be used to solve DG problems. Li *et al.* alternates domain-specific feature extractors and classifiers across domains via episodic training, but without using inner gradient descent update [14]. Balaji *et al.* proposed MetaReg that learns a regularization function (e.g., weighted ℓ_1 loss) particularly for the networks classification layer, while excluding the feature extractor [1].

Further, recent DG works forgo the requirement of source domains partitions and directly learn the cross-domain generalizable representations through a mixed collection of training data. Wang *et al.* extracted robust feature representation by projecting out superficial patterns like color and texture [31]. Wang *et al.* penalized models tendency in predicting with local features in order to extract robust globe representation [30]. RSC follows this more recent path and directly activates more features in all source domain data for DG without knowledge of the partition of source domains.

DG through Augmenting Source Domain: These methods augment the source domain to a wider span of the training data space, enlarging the possibility of covering the span of the data in the target domain. For example, An auxiliary domain classifier has been introduced to augment the data by perturbing input data based on the domain classification signal [23]. Volpi *et al.* developed an adversarial approach, in which samples are perturbed according to fictitious target distributions within a certain Wasserstein distance from the source [29]. A recent method with state-of-the art performance is JiGen [4], which leverages self-supervised signals by solving jigsaw puzzles.

Key difference: These approaches usually introduce a model-specific DG model and rely on prior knowledge of the target domain, for instance, the tar-

get spatial permutation is assumed by JiGen [4]. In contrast, RSC is a model-agnostic training algorithm that aims to improve the cross-domain robustness of any given model. More importantly, RSC does not utilize any knowledge of partitions of domains, either source domain or target domain, which is the general scenario in real world application.

Generic Model Regularization: CNNs are powerful models and tend to overfit on source domain datasets. From this perspective, model regularization, *e.g.*, weight decay [19], early stopping, and shake-shake regularization [8], could also improve the DG performance. Dropout [25] mutes features by randomly zeroing each hidden unit of the neural network during the training phase. In this way, the network benefit from the assembling effect of small subnetworks to achieve a good regularization effect. Cutout [6] and HaS [24] randomly drop patches of input images. SpatialDropout [26] randomly drops channels of a feature map. DropBlock [9] drops contiguous regions from feature maps instead of random units. DropPath [11] zeroes out an entire layer in training, not just a particular unit. MaxDrop [20] selectively drops features of high activations across the feature map or across the channels. Adversarial Dropout [21] dropouts for maximizing the divergence between the training supervision and the outputs from the network. [12] leverages Adversarial Dropout [21] to learn discriminative features by enforcing the cluster assumption.

Key difference: RSC differs from above methods in that RSC locates and mutes most predictive parts of feature maps by gradients instead of randomness, activation or prediction divergence maximization. This selective process plays an important role in improving the convergence, as we will briefly argue later.

3 Method

Notations: $(\mathbf{x}, \mathbf{y})$ denotes a sample-label pair from the data collection $(\mathbf{X}, \mathbf{Y})$ with n samples, and $\mathbf{z}$ (or $\mathbf{Z}$) denotes the feature representation of $(\mathbf{x}, \mathbf{y})$ learned by a neural network. $f(\cdot;\theta)$ denotes the CNN model, whose parameters are denoted as θ. $h(\cdot;\theta^{\text{top}})$ denotes the task component of $f(\cdot;\theta)$; $h(\cdot;\theta^{\text{top}})$ takes $\mathbf{z}$ as input and outputs the logits prior to a softmax function; θ^{top} denotes the parameters of $h(\cdot;\theta^{\text{top}})$. $l(\cdot,\cdot)$ denotes a generic loss function. RSC requires one extra scalar hyperparameter: the percentage of the representations to be discarded, denoted as p. Further, we use $\widehat{\cdot}$ to denote the estimated quantities, use $\widetilde{\cdot}$ to denote the quantities after the representations are discarded, and use t in the subscript to index the iteration. For example, $\widehat{\theta}_t$ means the estimated parameter at iteration t.

3.1 Self-Challenging Algorithm

As a generic deep learning training method, RSC solves the same standard loss function as the ones used by many other neural networks, *i.e.*,

$$\widehat{\theta} = \arg\min_{\theta} \sum_{\langle \mathbf{x},\mathbf{y}\rangle \sim \langle \mathbf{X},\mathbf{Y}\rangle} l(f(\mathbf{x};\theta),\mathbf{y}),$$

but RSC solves it in a different manner.

At each iteration, RSC inspects the gradient, identifies and then mutes the most predictive subset of the representation $\mathbf{z}$ (by setting the corresponding values to zero), and finally updates the entire model.

This simple heuristic has three steps (for simplicity, we drop the indices of samples and assume the batch size is 1 in the following equations):

1. Locate: RSC first calculates the gradient of upper layers with respect to the representation as follows:

$$\mathbf{g}_{\mathbf{z}} = \partial(h(\mathbf{z}; \widehat{\theta}_t^{\text{top}}) \odot \mathbf{y})/\partial \mathbf{z}, \tag{1}$$

   where $\odot$ denotes an element-wise product. Then RSC computes the $(100 - p)^{\text{th}}$ percentile, denoted as q_p. Then it constructs a masking vector $\mathbf{m}$ in the same dimension of $\mathbf{g}$ as follows. For the i^{th} element:

$$\mathbf{m}(i) = \begin{cases} 0, & \text{if} \quad \mathbf{g}_{\mathbf{z}}(i) \geq q_p \\ 1, & \text{otherwise} \end{cases} \tag{2}$$

   In other words, RSC creates a masking vector $\mathbf{m}$, whose element is set to 0 if the corresponding element in $\mathbf{g}$ is one of the top p percentage elements in $\mathbf{g}$, and set to 1 otherwise.
2. Mute: For every representation $\mathbf{z}$, RSC masks out the bits associated with larger gradients by:

$$\tilde{\mathbf{z}} = \mathbf{z} \odot \mathbf{m} \tag{3}$$

3. Update: RSC computes the softmax with perturbed representation with

$$\tilde{\mathbf{s}} = \text{softmax}(h(\tilde{\mathbf{z}}; \widehat{\theta}_t^{\text{top}})), \tag{4}$$

   and then use the gradient

$$\tilde{\mathbf{g}}_\theta = \partial l(\tilde{\mathbf{s}}, \mathbf{y})/\partial \widehat{\theta}_t \tag{5}$$

   to update the entire model for $\widehat{\theta}_{t+1}$ with optimizers such as SGD or ADAM.

We summarize the procedure of RSC in Algorithm 1. No that operations of RSC comprise of only few simple operations such as pooling, threshold and element-wise product. Besides the weights of the original network, no extra parameter needs to be learned.

3.2 Theoretical Evidence

To expand the theoretical discussion smoothly, we will refer to the "dog" vs. "cat" classification example repeatedly as we progress. The basic set-up, as we introduced in the beginning of this paper, is the scenario of a child trying to learn the concepts of "dog" vs. "cat" from illustrations in her book: while the

Algorithm 1: RSC Update Algorithm

Input: data set $\langle \mathbf{X}, \mathbf{Y} \rangle$, percentage of representations to discard p, other configurations such as learning rate η, maximum number of epoches T, *etc*;
Output: Classifier $f(\cdot; \widehat{\theta})$;
random initialize the model $\widehat{\theta}_0$;
while $t \leq T$ **do**
  for *every sample (or batch)* $\mathbf{x}, \mathbf{y}$ **do**
    calculate $\mathbf{z}$ through forward pass;
    calculate $\mathbf{g}_{\mathbf{z}}$ with Equation 1;
    calculate q_p and $\mathbf{m}$ as in Equation 2;
    generate $\tilde{\mathbf{z}}$ with Equation 3;
    calculate gradient $\tilde{\mathbf{g}}_\theta$ with Equation 4 and Equation 5;
    update $\widehat{\theta}_{t+1}$ as a function of $\widehat{\theta}_t$ and $\tilde{\mathbf{g}}_\theta$
  end
end

hypothesis "cats tend to have chubby faces" is good enough to classify all the animals in her picture book, other hypotheses mapping ears or body-size to labels are also predictive.

On the other hand, if she wants to differentiate all the "dogs" from "cats" in the real world, she will have to rely on a complicated combination of the features mentioned about. Our main motivation of this paper is as follows: this complicated combination of these features is already illustrated in her picture book, but she does not have to learn the true concept to do well in her finite collection of animal pictures.

This disparity is officially known as "covariate shift" in domain adaptation literature: the conditional distribution (*i.e.*, the semantic of a cat) is the same across every domain, but the model may learn something else (*i.e.*, chubby faces) due to the variation of marginal distributions.

With this connection built, we now proceed to the theoretical discussion, where we will constantly refer back to this "dog" vs. "cat" example.

Background As the large scale deep learning models, such as AlexNet or ResNet, are notoriously hard to be analyzed statistically, we only consider a simplified problem to argue for the theoretical strength of our method: we only concern with the upper layer $h(\cdot; \theta^{\text{top}})$ and illustrate that our algorithm helps improve the generalization of $h(\cdot; \theta^{\text{top}})$ when $\mathbf{Z}$ is fixed. Therefore, we can directly treat $\mathbf{Z}$ as the data (features). Also, for convenience, we overload θ to denote θ^{top} within the theoretical evidence section.

We expand our notation set for the theoretical analysis. As we study the domain-agnostic cross-domain setting, we no longer work with $i.i.d$ data. Therefore, we use $\mathcal{Z}$ and $\mathcal{Y}$ to denote the collection of distributions of features and labels respectively. Let Θ be a hypothesis class, where each hypothesis $\theta \in \Theta$ maps $\mathcal{Z}$ to $\mathcal{Y}$. We use a set $\mathcal{D}$ (or $\mathcal{S}$) to index $\mathcal{Z}$, $\mathcal{Y}$ and θ. Therefore, $\theta^\star(\mathcal{D})$

denotes the hypothesis with minimum error in the distributions specified with $\mathcal{D}$, but with no guarantees on the other distributions.

e.g., $\theta^\star(\mathcal{D})$ can be "cats have chubby faces" when $\mathcal{D}$ specifies the distribution to be picture book.

Further, $\theta^\star$ denotes the classifier with minimum error on every distribution considered. If the hypothesis space is large enough, $\theta^\star$ should perform no worse than $\theta^\star(\mathcal{D})$ on distributions specified by $\mathcal{D}$ for any $\mathcal{D}$.

e.g., $\theta^\star$ is the true concept of "cat", and it should predict no worse than "cats have chubby faces" even when the distribution is picture book.

We use $\widehat{\theta}$ to denote any ERM and use $\widehat{\theta}_{\text{RSC}}$ to denote the ERM estimated by the RSC method. Finally, following conventions, we consider $l(\cdot,\cdot)$ as the zero-one loss and use a shorthand notation $L(\theta;\mathcal{D}) = \mathbb{E}_{\langle \mathbf{z},\mathbf{y}\rangle \sim \langle \mathcal{Z}(\mathcal{D}),\mathcal{Y}(\mathcal{D})\rangle} l(h(\mathbf{z};\theta),\mathbf{y})$ for convenience, and we only consider the finite hypothesis class case within the scope of this paper, which leads to the first formal result:

Corollary 1. *If*

$$|e(\mathbf{z}(\mathcal{S});\theta^\star_{\text{RSC}}) - e(\tilde{\mathbf{z}}(\mathcal{S});\theta^\star_{\text{RSC}})| \leq \xi(p), \tag{6}$$

where $e(\cdot;\cdot)$ is a function defined as

$$e(\mathbf{z};\theta^\star) := \mathbb{E}_{\langle \mathbf{z},\mathbf{y}\rangle \sim \mathcal{S}} l(f(\mathbf{z};\theta^\star);\mathbf{y})$$

*and $\xi(p)$ is a small number and a function of RSC's hyperparameter p; $\tilde{\mathbf{z}}$ is the perturbed version of $\mathbf{z}$ generated by RSC, it is also a function of p, but we drop the notation for simplicity. If Assumptions **A1**, **A2**, and **A3** (See Appendix) hold, we have, with probability at least $1-\delta$*

$$\begin{aligned} &L(\widehat{\theta}_{\text{RSC}}(\mathcal{S});\mathcal{S}) - L(\theta^\star_{\text{RSC}}(\mathcal{S});\mathcal{D}) \\ &\qquad \leq (2\xi(p)+1)\sqrt{\frac{2(\log(2|\Theta_{\text{RSC}}|) + \log(2/\delta))}{n}} \end{aligned}$$

As the result shows, whether RSC will succeed depends on the magnitude of $\xi(p)$. The smaller $\xi(p)$ is, the tighter the bound is, the better the generalization bound is. Interestingly, if $\xi(p) = 0$, our result degenerates to the classical generalization bound of *i.i.d* data.

While it seems the success of our method will depend on the choice of Θ to meet Condition 6, we will show RSC is applicable in general by presenting it forces the empirical counterpart $\widehat{\xi}(p)$ to be small. $\widehat{\xi}(p)$ is defined as

$$\widehat{\xi}(p) := |h(\widehat{\theta}_{\text{RSC}},\mathbf{z}) - h(\widehat{\theta}_{\text{RSC}},\tilde{\mathbf{z}})|,$$

where the function $h(\cdot,\cdot)$ is defined as

$$h(\widehat{\theta}_{\mathrm{RSC}}, \mathbf{z}) = \sum_{(\mathbf{z},\mathbf{y})\sim\mathcal{S}} l(f(\mathbf{z}; \widehat{\theta}_{\mathrm{RSC}}); \mathbf{y}). \tag{7}$$

We will show $\widehat{\xi}(p)$ decreases at every iteration with more assumptions:

A4: Discarding the most predictive features will increase the loss at current iteration.

A5: The learning rate η is sufficiently small (η^2 or higher order terms are negligible).

Formally,

Corollary 2. *If Assumption **A4** holds, we can simply denote*

$$h(\widehat{\theta}_{\mathrm{RSC}}(t), \tilde{\mathbf{z}}_t) = \gamma_t(p) h(\widehat{\theta}_{\mathrm{RSC}}(t), \mathbf{z}_t),$$

*where $h(\cdot,\cdot)$ is defined in Equation 7. $\gamma_t(p)$ is an arbitrary number greater than 1, also a function of RSC's hyperparameter p. Also, if Assumption **A5** holds, we have:*

$$\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t+1)) = \Gamma(\widehat{\theta}_{\mathrm{RSC}}(t)) - (1 - \frac{1}{\gamma_t(p)})||\tilde{\mathbf{g}}||_2^2 \eta$$

where

$$\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t)) := |h(\widehat{\theta}_{\mathrm{RSC}}(t), \mathbf{z}_t) - h(\widehat{\theta}_{\mathrm{RSC}}(t), \tilde{\mathbf{z}}_t)|$$

t denotes the iteration, $\mathbf{z}_t$ (or $\tilde{\mathbf{z}}_t$) denotes the features (or perturbed features) at iteration t, and $\tilde{\mathbf{g}} = \partial h(\widehat{\theta}_{\mathrm{RSC}}(t), \tilde{\mathbf{z}}_t)/\partial\widehat{\theta}_{\mathrm{RSC}}(t)$

Notice that $\widehat{\xi}(p) = \Gamma(\widehat{\theta}_{\mathrm{RSC}})$, where $\widehat{\theta}_{\mathrm{RSC}}$ is $\widehat{\theta}_{\mathrm{RSC}}(t)$ at the last iteration t. We can show that $\widehat{\xi}(p)$ is a small number because $\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t))$ gets smaller at every iteration. This discussion is also verified empirically, as shown in Figure 2.

The decreasing speed of $\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t))$ depends on the scalar $\gamma_t(p)$: the greater $\gamma_t(p)$ is, the faster $\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t))$ descends. Further, intuitively, the scale of $\gamma_t(p)$ is highly related to the mechanism of RSC and its hyperparameter p. For example, RSC discards the most predictive representations, which intuitively guarantees the increment of the empirical loss (Assumption **A4**).

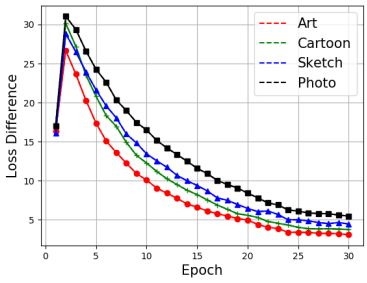

Fig. 2. $\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t))$, i.e., "Loss Difference", plotted for the PACS experiment (details of the experiment setup will be discussed later). Except for the first epoch, $\Gamma(\widehat{\theta}_{\mathrm{RSC}}(t))$ decreases consistently along the training process.

Finally, the choice of p governs the increment of the empirical loss: if p is small, the perturbation will barely affect the model, thus the increment will be small; while if p is large, the perturbation can alter the model's response dramatically, leading to significant ascend of the loss. However, we cannot blindly choose the largest possible p because if p is too large, the model may not be able to learn anything predictive at each iteration.

In summary, we offer the intuitive guidance of the choice of hyperparamter p: for the same model and setting,

- the smaller p is, the smaller the training error will be;
- the bigger p is, the smaller the (cross-domain) generalization error (*i.e.*, difference between testing error and training error) will be.

Therefore, the success of our method depends on the choice of p as a balance of the above two goals.

3.3 Engineering Specification & Extensions

For simplicity, we detail the RSC implementation on a ResNet backbone + FC classification network. RSC is applied to the training phase, and operates on the last convolution feature tensor of ResNet. Denote the feature tensor of an input sample as $\mathbf{Z}$ and its gradient tensor of as $\mathbf{G}$. $\mathbf{G}$ is computed by back propagating the classification score with respect to the ground truth category. Both of them are of size $[7 \times 7 \times 512]$.

Spatial-wise RSC: In the training phase, global average pooling is applied along the channel dimension to the gradient tensor $\mathbf{G}$ to produce a weighting matrix w_i of size $[7 \times 7]$. Using this matrix, we select top p percentage of the $7 \times 7 = 49$ cells, and mute its corresponding features in $\mathbf{Z}$. Each of the 49 cells correspond to a $[1 \times 1 \times 512]$ feature vector in $\mathbf{Z}$. After that, the new feature tensor $\mathbf{Z_{new}}$ is forwarded to the new network output. Finally, the network is updated through back-propagation. We refer this setup as spatial-wise RSC, which is the default RSC for the rest of this paper.

Channel-wise RSC: RSC can also be implemented by dropping features of the channels with high-gradients. The rational behind the channel-wise RSC lies in the convolutional nature of DNNs. The feature tensor of size $[7 \times 7 \times 512]$ can be considered a decomposed version of input image, where instead of the RGB colors, there are 512 different characteristics of the each pixels. The C characteristics of each pixel contains different statistics of training data from that of the spatial feature statistics.

For channel-wise RSC, global average pooling is applied along the spatial dimension of $\mathbf{G}$, and produce a weighting vector of size $[1 \times 512]$. Using this vector, we select top p percentage of its 512 cells, and mute its corresponding features in $\mathbf{Z}$. Here, each of the 512 cells correspond to a $[7 \times 7]$ feature matrix in $\mathbf{Z}$. After that, the new feature tensor $\mathbf{Z_{new}}$ is forwarded to the new network output. Finally, the network is updated through back-propagation.

Batch Percentage: Some dropout methods like curriculum dropout [17] do not apply dropout at the beginning of training, which improves CNNs by

learning basic discriminative clues from unchanged feature maps. Inspired by these methods, we randomly apply RSC to some samples in each batch, leaving the other unchanged. This introduces one extra hyperparameter, namely Batch Percentage: the percentage of samples to apply RSC in each batch. We also apply RSC to top percentage of batch samples based on cross-entropy loss. This setup is slightly better than randomness.

Detailed ablation study on above extensions will be conducted in the experiment section below.

4 Experiments

4.1 Datasets

We consider the following four data collections as the battleground to evaluate RSC against previous methods.

- **PACS** [13]: seven classes over four domains (Artpaint, Cartoon, Sketches, and Photo). The experimental protocol is to train a model on three domains and test on the remaining domain.
- **VLCS** [27]: five classes over four domains. The domains are defined by four image origins, *i.e.*, images were taken from the PASCAL VOC 2007, LabelMe, Caltech and Sun datasets.
- **Office-Home** [28]: 65 object categories over 4 domains (Art, Clipart, Product, and Real-World).
- **ImageNet-Sketch** [30]: 1000 classes with two domains. The protocol is to train on standard ImageNet [22] training set and test on ImageNet-Sketch.

4.2 Ablation Study

We conducted five ablation studies on possible configurations for RSC on the PACS dataset [13]. All results were produced based on the ResNet18 baseline in [4] and were averaged over five runs.

(1) Feature Dropping Strategies (Table 1). We compared the two attention mechanisms to select the most discriminative spatial features. The "Top-Activiation" [20] selects the features with highest norms, whereas the "Top-Gradient" (default in RSC) selects the features with high gradients. The comparison shows that "Top-Gradient" is better than "Top-Activation", while both are better than the random strategy. Without specific note, we will use "Top-Gradient" as default in the following ablation study.

(2) Feature Dropping Percentage (choice of p) (Table 2): We ran RSC at different dropping percentages to mute spatial feature maps. The highest average accuracy was reached at $p = 33.3\%$. While the best choice of p is data-specific, our results align well with the theoretical discussion: the optimal p should be neither too large nor too small.

(3) Batch Percentage (Table 3): RSC has the option to be only randomly applied to a subset of samples in each batch. Table 3 shows that the performance

Feature Drop Strategies	backbone	artpaint	cartoon	sketch	photo	Avg ↑
Baseline [4]	ResNet18	78.96	73.93	70.59	**96.28**	79.94
Random	ResNet18	79.32	75.27	74.06	95.54	81.05
Top-Activation	ResNet18	80.31	76.05	76.13	**95.72**	82.03
Top-Gradient	ResNet18	**81.23**	**77.23**	**77.56**	95.61	**82.91**

Table 1. Ablation study of Spatial-wise RSC on Feature Dropping Strategies. Feature Dropping Percentage 50.0% and Batch Percentage 50.0%.

Feature Dropping Percentage	backbone	artpaint	cartoon	sketch	photo	Avg↑
66.7%	ResNet18	80.11	76.35	76.24	95.16	81.97
50.0%	ResNet18	81.23	77.23	77.56	95.61	82.91
33.3%	ResNet18	**82.87**	**78.23**	**78.89**	95.82	**83.95**
25.0%	ResNet18	81.63	78.06	78.12	96.06	83.46
20.0%	ResNet18	81.22	77.43	77.83	96.25	83.18
13.7%	ResNet18	80.71	77.18	77.12	**96.36**	82.84

Table 2. Ablation study of Spatial-wise RSC on Feature Dropping Percentage. We used "Top-Gradient" and fixed the Batch Percentage (50.0%) here.

Batch Percentage	backbone	artpaint	cartoon	sketch	photo	Avg↑
50.0%	ResNet18	**82.87**	78.23	78.89	95.82	83.95
33.3%	ResNet18	82.32	**78.75**	**79.56**	96.05	**84.17**
25.0%	ResNet18	81.85	78.32	78.75	**96.21**	83.78

Table 3. Ablation study of Spatial-wise RSC on Batch Percentage. We used "Top-Gradient" and fixed Feature Dropping Percentage (33.3%).

Method	backbone	artpaint	cartoon	sketch	photo	Avg↑
Spatial	ResNet18	82.32	78.75	79.56	**96.05**	84.17
Spatial+Channel	ResNet18	**83.43**	**80.31**	**80.85**	95.99	**85.15**

Table 4. Ablation study of Spatial-wise RSC verse Spatial+Channel RSC. We used the best strategy and parameter by Table 3:"Top-Gradient", Feature Dropping Percentage(33.3%) and Batch Percentage(33.3%).

Method	backbone	artpaint	cartoon	sketch	photo	Avg↑
Baseline [4]	ResNet18	78.96	73.93	70.59	**96.28**	79.94
Cutout[6]	ResNet18	79.63	75.35	71.56	95.87	80.60
DropBlock[9]	ResNet18	80.25	77.54	76.42	95.64	82.46
AdversarialDropout[21]	ResNet18	82.35	78.23	75.86	96.12	83.07
Random(S+C)	ResNet18	79.55	75.56	74.39	95.36	81.22
Top-Activation(S+C)	ResNet18	81.03	77.86	76.65	96.11	82.91
RSC: Top-Gradient(S+C)	ResNet18	**83.43**	**80.31**	**80.85**	95.99	**85.15**

Table 5. Ablation study of Dropout methods. "S" and "C" represent spatial-wise and channel-wise respectively. For fair comparison, results of above methods are report at their best setting and hyperparameters. RSC used the hyperparameters selected in above ablation studies:"Top-Gradient", Feature Dropping Percentage (33.3%) and Batch Percentage (33.3%).

is relatively constant. Nevertheless we still choose 33.3% as the best option on the PACS dataset.

(4) Spatial-wise plus Channel-wise RSC (Table 4): In "Spatial+Channel", both spatial-wise and channel-wise RSC were applied on a sample at 50% probability, respectively. (Better options of these probabilities could be explored.) Its

improvement over Spatial-wise RSC indicates that it further activated features beneficial to target domains.

PACS	backbone	artpaint	cartoon	sketch	photo	Avg ↑
Baseline[4]	AlexNet	66.68	69.41	60.02	89.98	71.52
Hex[31]	AlexNet	66.80	69.70	56.20	87.90	70.20
PAR[30]	AlexNet	66.30	66.30	64.10	89.60	72.08
MetaReg[1]	AlexNet	69.82	70.35	59.26	**91.07**	72.62
Epi-FCR[14]	AlexNet	64.70	72.30	65.00	86.10	72.00
JiGen[4]	AlexNet	67.63	71.71	65.18	89.00	73.38
MASF[7]	AlexNet	70.35	72.46	**67.33**	90.68	75.21
RSC(ours)	AlexNet	**71.62**	**75.11**	66.62	90.88	**76.05**
Baseline[4]	ResNet18	78.96	73.93	70.59	**96.28**	79.94
MASF[7]	ResNet18	80.29	77.17	71.69	94.99	81.03
Epi-FCR[14]	ResNet18	82.10	77.00	73.00	93.90	81.50
JiGen[4]	ResNet18	79.42	75.25	71.35	96.03	80.51
MetaReg[1]	ResNet18	**83.70**	77.20	70.30	95.50	81.70
RSC(ours)	ResNet18	83.43	**80.31**	**80.85**	95.99	**85.15**
Baseline[4]	ResNet50	86.20	78.70	70.63	97.66	83.29
MASF[7]	ResNet50	82.89	80.49	72.29	95.01	82.67
MetaReg[1]	ResNet50	87.20	79.20	70.30	97.60	83.60
RSC(ours)	ResNet50	**87.89**	**82.16**	**83.35**	**97.92**	**87.83**

Table 6. DG results on PACS[13] (Best in bold).

VLCS	backbone	Caltech	Labelme	Pascal	Sun	Avg ↑
Baseline[4]	AlexNet	96.25	59.72	70.58	64.51	72.76
Epi-FCR[14]	AlexNet	94.10	64.30	67.10	65.90	72.90
JiGen[4]	AlexNet	96.93	60.90	70.62	64.30	73.19
MASF[7]	AlexNet	94.78	**64.90**	69.14	67.64	74.11
RSC(ours)	AlexNet	**97.61**	61.86	**73.93**	**68.32**	**75.43**

Table 7. DG results on VLCS [27] (Best in bold).

(5) Comparison with different dropout methods (Table 5): Dropout has inspired a number of regularization methods for CNNs. The main differences between those methods lie in applying stochastic or non-stochastic dropout mechanism at input data, convolutional or fully connected layers. Results shows that our gradient-based RSC is better. We believe that gradient is an efficient and straightforward way to encode the sensitivity of output prediction. To the best of our knowledge, we compare with the most related works and illustrate the impact of gradients. (a) Cutout [6]. Cutout conducts random dropout on input images, which shows limited improvement over the baseline. (b) DropBlock [9]. DropBlock tends to dropout discriminative activated parts spatially. It is better than random dropout but inferior to non-stochastic dropout methods in Table 5 such as AdversarialDropout, Top-Activation and our RSC. (c) AdversarialDropout [21, 12]. AdversarialDropout is based on divergence maximization, while RSC is based on top gradients in generating dropout masks. Results show evidence that the RSC is more effective than AdversarialDropout. (d) Random and Top-Activation dropout strategies at their best hyperparameter settings.

Office-Home	backbone	Art	Clipart	Product	Real	Avg ↑
Baseline[4]	ResNet18	52.15	45.86	70.86	73.15	60.51
JiGen[4]	ResNet18	53.04	47.51	71.47	72.79	61.20
RSC(ours)	ResNet18	**58.42**	**47.90**	**71.63**	**74.54**	**63.12**

Table 8. DG results on Office-Home [28] (Best in bold).

ImageNet-Sketch	backbone	Top-1 Acc ↑	Top-5 Acc ↑
Baseline[31]	AlexNet	12.04	24.80
Hex[31]	AlexNet	14.69	28.98
PAR [30]	AlexNet	15.01	29.57
RSC(ours)	AlexNet	**16.12**	**30.78**

Table 9. DG results on ImageNet-Sketch [30].

4.3 Cross-Domain Evaluation

Through the following experiments, we used "Top-Gradient" as feature dropping strategy, 33.3% as Feature Dropping Percentages, 33.3% as Batch Percentage, and Spatial+Channel RSC. All results were averaged over five runs. In our RSC implementation, we used the SGD solver, 30 epochs, and batch size 128. The learning rate starts with 0.004 for ResNet and 0.001 for AlexNet, learning rate decayed by 0.1 after 24 epochs. For PACS experiment, we used the same data augmentation protocol of randomly cropping the images to retain between 80% to 100%, randomly applied horizontal flipping and randomly (10% probability) convert the RGB image to greyscale, following [4].

In Table. 6,7,8, we compare RSC with the latest domain generalization work, such as Hex [31], PAR [30], JiGen [4] and MetaReg [1]. All these work only report results on different small networks and datasets. For fair comparison, we compared RSC to their reported performances with their most common choices of DNNs (*i.e.*, AlexNet, ResNet18, and ResNet50) and datasets. RSC consistently outperforms other competing methods.

The empirical performance gain of RSC can be better appreciated if we have a closer look at the PACS experiment in Table. 6. The improvement of RSC from the latest baselines [4] are significant and consistent: 4.5 on AlexNet, 5.2 on ResNet18, and 4.5 on ResNet50. It is noticeable that, with both ResNet18 and ResNet50, RSC boosts the performance significantly for sketch domain, which is the only colorless domain. The model may have to understand the semantics of the object to perform well on the sketch domain. On the other hand, RSC performs only marginally better than competing methods in photo domain, which is probably because that photo domain is the simplest one and every method has already achieved high accuracy on it.

5 Discussion

Standard ImageNet Benchmark: With the impressive performance observed in the cross-domain evaluation, we further explore to evaluate the benefit of RSC with other benchmark data and higher network capacity.

ImageNet	backbone	Top-1 Acc ↑	Top-5 Acc ↑	#Param. ↓
Baseline	ResNet50	76.13	92.86	25.6M
RSC(ours)	ResNet50	77.18	93.53	25.6M
Baseline	ResNet101	77.37	93.55	44.5M
RSC(ours)	ResNet101	78.23	94.16	44.5M
Baseline	ResNet152	78.31	94.05	60.2M
RSC(ours)	ResNet152	78.89	94.43	60.2M

Table 10. Generalization results on ImageNet. Baseline was produced with official Pytorch implementation and their ImageNet models.

We conducted image classification experiments on the Imagenet database[22]. We chose three backbones with the same architectural design while with clear hierarchies in model capacities: ResNet50, ResNet101, and ResNet152. All models were finetuned for 80 epochs with learning rate decayed by 0.1 every 20 epochs. The initial learning rate for ResNet was 0.01. All models follow extra the same training prototype in default Pytorch ImageNet implementation, using original batch size of 256, standard data augmentation and 224×224 as input size.

The results in Table 10 shows that RSC exhibits the ability reduce the performance gap between networks of same family but different sizes (*i.e.*, ResNet50 with RSC approaches the results of baseline ResNet101, and ResNet101 with RSC approaches the results of baseline ResNet151). The practical implication is that, RSC could induce faster performance saturation than increasing model sizes. Therefore one could scale down the size of networks to be deployed at comparable performance.

6 Conclusion

We introduced a simple training heuristic method that can be directly applied to almost any CNN architecture with no extra model architecture, and almost no increment of computing efforts. We name our method Representation Self-challenging (RSC). RSC iteratively forces a CNN to activate features that are less dominant in the training domain, but still correlated with labels. Theoretical and empirical analysis of RSC validate that it is a fundamental and effective way of expanding feature distribution of the training domain. RSC produced the state-of-the-art improvement over baseline CNNs under the standard DG settings of small networks and small datasets. Moreover, our work went beyond the standard DG settings, to illustrate effectiveness of RSC on more prevalent problem scales, *e.g.*, the ImageNet database and network sizes up-to ResNet152.

Acknowledgement

This work was partially supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/ Interior Business Center (DOI/IBC) contract number D17PC00340. In addition, Haohan Wang is supported by NIH R01GM114311, NIH P30DA035778, and NSF IIS1617583.

References

1. Balaji, Y., Sankaranarayanan, S., Chellappa, R.: Metareg: Towards domain generalization using meta-regularization. In: Advances in Neural Information Processing Systems. pp. 998–1008 (2018)
2. Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., Vaughan, J.W.: A theory of learning from different domains. Machine learning **79**(1), 151–175 (2010)
3. Bridle, J.S., Cox, S.J.: Recnorm: Simultaneous normalisation and classification applied to speech recognition. In: Advances in Neural Information Processing Systems. pp. 234–240 (1991)
4. Carlucci, F.M., D'Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2229–2238 (2019)
5. Csurka, G.: Domain adaptation for visual applications: A comprehensive survey. arXiv preprint arXiv:1702.05374 (2017)
6. DeVries, T., Taylor, G.W.: Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552 (2017)
7. Dou, Q., Castro, D.C., Kamnitsas, K., Glocker, B.: Domain generalization via model-agnostic learning of semantic features. arXiv preprint arXiv:1910.13580 (2019)
8. Gastaldi, X.: Shake-shake regularization. arXiv preprint arXiv:1705.07485 (2017)
9. Ghiasi, G., Lin, T.Y., Le, Q.V.: Dropblock: A regularization method for convolutional networks. In: Advances in Neural Information Processing Systems. pp. 10727–10737 (2018)
10. Ghifary, M., Bastiaan Kleijn, W., Zhang, M., Balduzzi, D.: Domain generalization for object recognition with multi-task autoencoders. In: Proceedings of the IEEE international conference on computer vision. pp. 2551–2559 (2015)
11. Larsson, G., Maire, M., Shakhnarovich, G.: Fractalnet: Ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648 (2016)
12. Lee, S., Kim, D., Kim, N., Jeong, S.G.: Drop to adapt: Learning discriminative features for unsupervised domain adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 91–100 (2019)
13. Li, D., Yang, Y., Song, Y.Z., Hospedales, T.M.: Deeper, broader and artier domain generalization. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5542–5550 (2017)
14. Li, D., Zhang, J., Yang, Y., Liu, C., Song, Y.Z., Hospedales, T.M.: Episodic training for domain generalization. arXiv preprint arXiv:1902.00113 (2019)
15. Li, H., Jialin Pan, S., Wang, S., Kot, A.C.: Domain generalization with adversarial feature learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5400–5409 (2018)
16. Mitchell, T.M., et al.: Machine learning. 1997. Burr Ridge, IL: McGraw Hill **45**(37), 870–877 (1997)
17. Morerio, P., Cavazza, J., Volpi, R., Vidal, R., Murino, V.: Curriculum dropout. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3544–3552 (2017)
18. Muandet, K., Balduzzi, D., Schölkopf, B.: Domain generalization via invariant feature representation. In: International Conference on Machine Learning. pp. 10–18 (2013)
19. Nowlan, S.J., Hinton, G.E.: Simplifying neural networks by soft weight-sharing. Neural computation **4**(4), 473–493 (1992)

20. Park, S., Kwak, N.: Analysis on the dropout effect in convolutional neural networks. In: Asian conference on computer vision. pp. 189–204. Springer (2016)
21. Park, S., Park, J., Shin, S.J., Moon, I.C.: Adversarial dropout for supervised and semi-supervised learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
22. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). https://doi.org/10.1007/s11263-015-0816-y
23. Shankar, S., Piratla, V., Chakrabarti, S., Chaudhuri, S., Jyothi, P., Sarawagi, S.: Generalizing across domains via cross-gradient training. arXiv preprint arXiv:1804.10745 (2018)
24. Singh, K.K., Lee, Y.J.: Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In: 2017 IEEE international conference on computer vision (ICCV). pp. 3544–3553. IEEE (2017)
25. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
26. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient object localization using convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 648–656 (2015)
27. Torralba, A., Efros, A.A., et al.: Unbiased look at dataset bias. In: CVPR. vol. 1, p. 7. Citeseer (2011)
28. Venkateswara, H., Eusebio, J., Chakraborty, S., Panchanathan, S.: Deep hashing network for unsupervised domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5018–5027 (2017)
29. Volpi, R., Namkoong, H., Sener, O., Duchi, J.C., Murino, V., Savarese, S.: Generalizing to unseen domains via adversarial data augmentation. In: Advances in Neural Information Processing Systems. pp. 5334–5344 (2018)
30. Wang, H., Ge, S., Xing, E.P., Lipton, Z.C.: Learning robust global representations by penalizing local predictive power. In: Advances in Neural Information Processing Systems (NeurIPS 2019) (2019)
31. Wang, H., He, Z., Lipton, Z.C., Xing, E.P.: Learning robust representations by projecting superficial statistics out. In: International Conference on Learning Representations (2019)
32. Wang, H., Wu, X., Huang, Z., Xing, E.P.: High-frequency component helps explain the generalization of convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8684–8694 (2020)
33. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312**, 135–153 (2018)