REST: Performance Improvement of a Black Box Model via RL-Based Spatial Transformation

Jae Myung Kim,[*] **Hyungjin Kim,**[*] **Chanwoo Park,**[*] **Jungwoo Lee**

Department of Electrical and Computer Engineering, Seoul National University, Seoul, Korea

{goldkim92, hjkim, cpark}@cml.snu.ac.kr, junglee@snu.ac.kr

Abstract

In recent years, deep neural networks (DNN) have become a highly active area of research, and shown remarkable achievements on a variety of computer vision tasks. DNNs, however, are known to often make overconfident yet incorrect predictions on out-of-distribution samples, which can be a major obstacle to real-world deployments because the training dataset is always limited compared to diverse real-world samples. Thus, it is fundamental to provide guarantees of robustness to the distribution shift between training and test time when we construct DNN models in practice. Moreover, in many cases, the deep learning models are deployed as black boxes and the performance has been already optimized for a training dataset, thus changing the black box itself can lead to performance degradation. We here study the robustness to the geometric transformations in a specific condition where the black-box image classifier is given. We propose an additional learner, *REinforcement Spatial Transform learner (REST)*, that transforms the warped input data into samples regarded as in-distribution by the black-box models. Our work aims to improve the robustness by adding a REST module in front of any black boxes and training only the REST module without retraining the original black box model in an end-to-end manner, i.e. we try to convert the real-world data into training distribution which the performance of the black-box model is best suited for. We use a confidence score that is obtained from the black-box model to determine whether the transformed input is drawn from in-distribution. We empirically show that our method has an advantage in generalization to geometric transformations and sample efficiency.

Introduction

In the last few years, convolutional neural networks(CNNs) have made great progress in visual recognition tasks such as image classification, object detection, and semantic segmentation(He et al. 2016; Redmon et al. 2016; Long, Shelhamer, and Darrell 2015). Although diverse CNN-based architectures have shown state-of-the-art performance in many computer vision tasks, this result is based on the assumption that training and test data are drawn from the same

distribution. However, when the deep learning applications are actually deployed in real-world settings, it is inevitable to deal with real data generated from a different distribution from the training phase, and the performance cannot be guaranteed. As is the case with geometric variations caused by different poses, deformations, and viewpoints, it can be one of the major challenges that deep learning systems face in reality. To address this issue, several studies have been reported in the literature(Xu et al. 2014; Marcos, Volpi, and Tuia 2016; Follmann and Bottger 2018; You et al. 2018; Worrall et al. 2017).

Apart from the issue of model performance on real-world data, automated decision-making systems are usually opaque, which means we are not fully accessible to them. The only thing we can observe is the input and the predicted output of black box model. This raises a natural question about the systems. How much should we trust the predictions of the black box system with the given input? Even if we assume there is an indicator that tells us how much we can trust, what actions should we take to obtain reliable predictions if it tells us that the predicted result is highly uncertain?

For example, let us consider a car number recognition system. The embedded system outputs the predicted number given a license plate. The predictions of the system are usually very accurate given the numbers with typical shapes. However, a crumpled license plate by a car accident (e.g. tilted number 7 written in the plate) would prevent the current system from recognizing the numbers accurately (e.g. predicted as 1). The system would also have high uncertainty in its prediction.

Motivated by the question above, this paper aims to spatially transform the input data to obtain the most reliable predictions by applying the sequence of modified inputs to the black-box model. If test data is not sampled from the same distribution as the data used for training the black-box model, the probability of false prediction increases. To improve accuracy and reliability of predictions, the test input data should be transformed to follow the training data distribution of the black-box model. We refer to the transformation model as a *transform learner*. After the test data is modified by the transform learner, the black-box model maps the
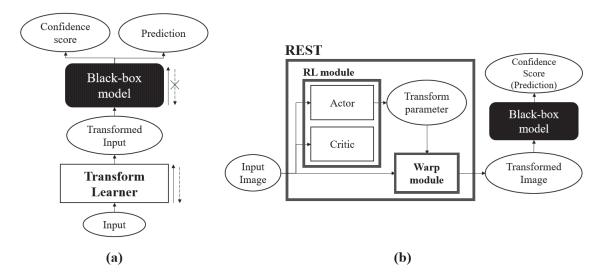
---

[*]Equal Contribution

Figure 1: The schematic description. Figure (a) shows the framework of our approach. The white rectangle demonstrates a trainable model and the black represents a black-box model. Ellipse is an input or an output of the models. Solid lines are forward propagation and dashed lines are backpropagation. Figure (b) shows a more detailed framework where the black-box is an image classifier. The REST learner consists of two modules which are the RL module and the warp module.

transformed data to the output which tells us not only the prediction but also the confidence of the prediction, which is named *confidence score* here. The schematic description is given in Figure 1(a).

The main challenge in our setting is that we do not know what data distribution the black box system has trained from. The prediction and the confidence score are provided by the black-box model only when a particular input is given. We cannot directly train the transform learner by a gradient-based method because the black-box model prevents the gradient flow from reaching to the transform learner. To deal with the problem, we introduce a black-box optimization technique which approximates the gradient of the output by reinforcement learning(Jacovi et al. 2019).

Specifically, in this paper, we take a close look at image classification as the black-box problem. We use a pre-trained image classifier for the black-box model. Input images are transformed by a transform learner, which we call *REinforcement Spatial Transform learner (REST)*. REST is trained to perform geometric transformation on the input images to increase the confidence score. At inference time, an input image is transformed by REST, followed by predicting the class label via the black-box model.

Related Work

Invariance to geometric transformations

There have been many studies to achieve geometric invariance or equivariance in computer vision problems such as classification and segmentation, and we here review four methods: Data augmentation, Spatial Transformer network, Deformable Convolutional Networks, and Capsule network.

Data augmentation Data augmentation (DA)(Simard et al. 2003) is a ubiquitous technique largely used to improves

generalization. Although DA is a good starting point for transformation invariance, it is extremely expensive to augment training images with every possible combination of random rotations, shifts, and scales and it is often observed that the models learned with DA are only invariant to known transformations rather than arbitrary changes.

Spatial Transformer Network Spatial transformer networks (STN)(Jaderberg et al. 2015) is the first work which learns to apply spatial transformation to warp feature maps in an end-to-end fashion. STN consists of three components - localization network, grid generator and sampler. Localization network learns input-dependent transformations and allows the entire network to increase classification accuracy. The drawback of the STN is that it has to be trained with various transformations in the training phase, and we find that STN failed to generalize when unknown, rare transformations are applied to input images.

Deformable Convolutional Networks In (Dai et al. 2017) the authors argue that it is inherently difficult to handle objects with different scales and shapes in regular CNN because their operations, e.g. convolution kernels, max-pooling, have geometrically fixed patterns. Deformable ConvNets tries to model the dense spatial transformations by learning 2D offsets to the regular sampling grid from target tasks instead of parametric transformations. Deformable ConvNets is then applied to semantic segmentation and object detection to demonstrate its efficiency.

Capsule network Capsules(Sabour, Frosst, and Hinton 2017) are represented by a vector which contains the features of an object and its likelihood. Higher capsules are activated only when the group of objects below them is consistent in their orientation and size with each other. The authors tested CapsNet on affNIST dataset to show its robustness to

affine transformations. CapsNet was trained on MNIST digits only with random translations and achieved 79% accuracy on affNIST which confirms it generalizes well to small affine transformations.

Deep RL for computer vision

Deep RL methods have been making steady progress in games (Mnih et al. 2013), robotics (Levine et al. 2016), finance (Deng et al. 2016), etc., and have recently been expanded to a wide range of computer vision tasks. (Caicedo and Lazebnik 2015) addresses the problem of detection by learning localization policy to find region proposals which best focus on the object while (Kong et al. 2017) suggests joint agent detection to reduce the iterations compared to a single agent.

For visual tracking, (Yun et al. 2017) proposes to dynamically track the object with increased efficiency in search space through selecting sequential actions on candidate bounding boxes. (Ren et al. 2018) extends to multi-object tracking by modeling each object as an agent and exploiting the collaborative interactions between agents.

Video recognition can be computationally expensive if performing exhaustive search in every frame. For the first time, (Rao, Lu, and Zhou 2017) authors produce temporal-spatial representations then find the most relevant information in video pairs as a Markov decision process (MDP) for face recognition. In action recognition, (Tang et al. 2018) propose to reduce the computational burden by selecting only the key frames in skeleton-based videos, and generating the reward with the graph-based convolutional NN.

Deep RL has also been applied to image editing and color enhancement. (Li et al. 2018) deploys a RL-based method for automatic image cropping where the agent makes sequential decisions on where to crop the original image to maximize the aesthetics score. (Yu et al. 2018) progressively restores a corrupted image by selecting a specific tool from the toolbox at each step while (Park et al. 2018) focuses on automatic color enhancement where the agent takes an interpretable action sequence to produce a retouched image like a human expert. Here, we present a novel RL-based strategy for geometric invariance and this is, to the best of our knowledge, the first work to apply deep RL to its kind.

Method

In this section, we describe the procedure of REST. The purpose of REST is to warp the input image for reliable prediction of the black-box model. We first define the black-box model which performs a classification task and provides the confidence scores. Our method, REST, is composed of two modules. The first part is a trainable RL module which outputs the parameters of the transformation when geometrically deformed images are given as states. The other is a warp module which takes an input image and performs warping with the transformation parameters.

Black Box Model

We view the black-box function as a probabilistic model $P(y|x)$ where the model assigns a probability of the class y for an input x. For the black-box image classifier B, $x \in \mathbb{R}^{H \times W \times C}$ is an input image, where H, W, C are the height, width, and channels of an input image, and $y \in \mathcal{Y}$ is the class with $\mathcal{Y}$ a set of classes. We formulate the function of black-box as y and $c = B(x)$, where c is the confidence score obtained from the black box B. We discuss the definition of confidence score and which method we choose to measure it in the next section. The classifier B is trained on the dataset $D = \{(x_i, y_i) | 1 \le i \le n, \ i \in \mathbb{N}\}$, and x_i is drawn from the data distribution $\mathcal{P}(x)$ which is regarded as in-distribution. When training the REST learner, we are not allowed to access the original dataset D to ensure black-box assumption.

Confidence Score

It is important for neural networks to understand how uncertain they are in their predictions. While there are methods to quantify the predictive uncertainty by training neural networks where the structure is different from that of prevailing deterministic models (Graves 2011; Blundell et al. 2015), there are also other approaches to obtain the uncertainty without modifying the existing classifier structure or training procedure (Gal 2016; Gal and Ghahramani 2016; Ritter, Botev, and Barber 2018). We choose one of the latter methods as we are dealing with the black-box model. We here set a confidence score c to be inversely proportional to the predictive uncertainty obtained by the black-box model.

One of the most widely used approaches to uncertainty estimation is Monte-Carlo (MC) Dropout due to its simplicity (Gal 2016; Gal and Ghahramani 2016). When the black-box classifier has a dropout layer, the uncertainty is calculated by performing dropout at the inference time. Another simple method for measuring uncertainty is using a value proportional to the inverse of the prediction probability $\mathcal{P}(y|x)$ (which means the prediction probability is a confidence score).

While there exists an extensive literature that softmax outputs are sometimes overly confident and thus not sufficient to consider it as confidence scores (Gal 2016; Gal and Ghahramani 2016), in our experiment, the output of the softmax itself was a proper indicator for confidence level of predictions. Experimental results of replacing a confidence score with predictive uncertainty by MC dropout are given in Appendix B.

Reinforcement Spatial Transform Learner

REST learner consists of two modules: the RL module and the warp module. Given a distorted input image x', the RL module provides θ which is used for the warp module T_θ to transform a distorted image x' into x'_{tf}. As in-distribution $\mathcal{P}(x)$ and the dataset D are unknown, we indirectly estimate the possibility of x'_{tf} by the confidence score from in-distribution sample.

we train the RL module with a new dataset $D' = \{(x'_i, y'_i) | 1 \le i \le n, \ i \in \mathbb{N}\}$ to find a transform parameter θ for each data x'_i that makes a higher confidence score for the transformed image $T_\theta(x'_i)$. Let us clarify the state, action, policy, and reward of the RL module. State S_t is an
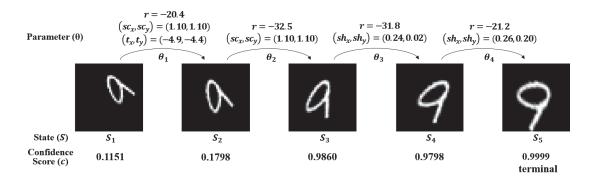
Figure 2: The sequence of transformation of an input image. A distorted input image gets a sequence of parameter to transform it into a canonical style. The sequence terminates when the transformed image gives a high confidence score. It is interesting to see that the final image resembles a style of typical MNIST data.

input data in time step t with $S_1 = x'_i$, and action A_t is a parameter θ used in the warp module T_θ. By mapping S_t by T_θ, we get a next state $S_{t+1} = T_\theta(S_t)$. Policy $\pi(A_t|S_t)$ is a probability for each parameter θ to be selected given input state, where the purpose of training the RL module is to find the optimal policy.

As the agent is trained to maximize a cumulative reward, we formulate a Reward R_t to increase as the confidence score c_{t+1} of the image S_{t+1} is bigger than c_t.

$$R_t = \log(c_{t+1}) - \log(c_t) \tag{1}$$

In training mode, we set the confidence score to be the the prediction probability of the target class label, $c_t = \mathcal{P}(y'_i|S_t)$ when $S_1 = x'_i$. Then the equation (1) can be interpreted as a difference between log-likelihood of the target class.

To get higher reward rate when the confidence score gets closer to 1, and to give penalty per step for a shorter length of episode, we modify the reward function as

$$R_t = (-\log(1 - c_{t+1})) - (-\log(1 - c_t)) - 1 \tag{2}$$

We also perform an ablation study that compares reward function (1) and (2) in Appendix C. In training mode, episode terminates when the confidence score c_{t+1} exceeds a threshold value in time step t or the number of steps reaches max number. We set a max number to be 10 for all the experiments. In inference mode, we use maximum likelihood, $max_{y \in \mathcal{Y}}\mathcal{P}(y|S_{t+1})$, instead of the prediction probability of the target class, $\mathcal{P}(y'_i|S_t)$, for the confidence score to determine the termination, and the other parts are the same.

Training

We use PPO algorithm (Schulman et al. 2017) for the RL module. To make it operate in continuous action space, we modify the actor network where the output is a mean of the Normal distribution with standard deviation to be 1. The re-parameterization trick is used to enable backpropagation.

Method	R	RSc	RSh	RSS	RSST
BB	69.28	65.52	56.52	53.95	17.99
REST+BB	**97.63**	**96.30**	**94.81**	**93.20**	**85.05**

Table 1: The accuracy of a distorted MNIST dataset. BB refers to a black-box model. While it presents a low performance when using only a black-box model, by applying the REST, the accuracy increases significantly.

After setting the activation function of actor network to the hyperbolic tangent function, we change the output bounds of $(-1, 1)$ to match action bounds so that the parameter θ exists within action bounds, making stable learning. In the test phase, we reduce the standard deviation of a normal distribution to decrease the randomness of actor network. We train the model using Adam optimization (Kingma and Ba 2014) with the learning rate 0.0001 and batch size 256.

Experiment

In this section we demonstrate the effectiveness of our approach in generalization by attaching the REST learner to the black-box model. To train the RL module, we generate a dataset D' by applying a random affine transformation to the expected canonical style of the dataset D used for training the black box classifier. For example, if we know that a black-box model B classifies a gray-scale numeric image, it is plausible to assume that the model B is trained by the MNIST dataset, and therefore we generate a dataset D' by random affine transformation of MNIST images.

We also use the affine transformation for the warp module T_θ. Although 6 parameters are typically used for affine matrix, we factorize the affine matrix into 4 matrices to have 7 parameters, $\theta = (\theta_r, \theta_{sc1}, \theta_{sc2}, \theta_{sh1}, \theta_{sh2}, \theta_{t1}, \theta_{t2})$.

$$T_\theta = \begin{bmatrix} \cos\theta_r & -\sin\theta_r & 0 \\ \sin\theta_r & \cos\theta_r & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \theta_{sc1} & 0 & 0 \\ 0 & \theta_{sc2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$\cdot \begin{bmatrix} 1 & \theta_{sh1} & 0 \\ \theta_{sh2} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & \theta_{t1} \\ 0 & 1 & \theta_{t2} \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Dataset	Method	base	R	RSc	RSh	RSS	RSST
SVHN	BB	96.03	59.96	56.46	57.05	53.54	26.09
	REST + BB	-	**89.38**	**88.58**	**89.00**	**85.92**	**83.82**
CIFAR10	BB	93.78	51.58	49.14	50.69	48.16	32.09
	REST + BB	-	**74.33**	**72.15**	**70.73**	**69.46**	**60.27**
STL10	BB	77.59	41.57	38.46	41.26	37.97	30.90
	REST + BB	-	**62.24**	**59.18**	**59.94**	**56.55**	**53.20**

Table 2: The accuracy of a distorted real-world dataset. We show that applying our model also performs better in real-world datasets such as SVHN, CIFAR10, and STL10.
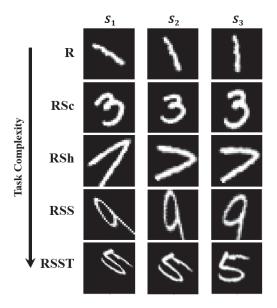

Figure 3: Various transformations of different complexity. S_1 is the initial state(a deformed image) and sequence of best actions are taken given the states.

where θ_r, θ_{sc}, θ_{sh}, and θ_t are parameters for rotation, scaling, shearing, and translation, respectively. Details of generating affine-warped dataset and settings of action bounds are described in Appendix A.

As we use the image dataset for experiments, the actor and the critic networks in RL module are constructed by two convolution layers followed by two fully connected layers. We choose 5-layer CNN and STN(Jaderberg et al. 2015) as baselines and test the transformation invariance on affine-warped MNIST, SVHN, CIFAR10, and STL10 datasets. We train the baseline models using Adam optimization(Kingma and Ba 2014) with the learning rate 0.0001 and batch size 128.

We start with the distorted MNIST dataset to show the improvement of classification performance in a shifted data distribution. We further test our model in a more challenging real-world dataset such as SVHN, CIFAR10, and STL10. Then, we demonstrate the ability of generalization of our model in a more shifted, arduous setting where disjointing subsets are selected for training and testing the REST

learner. Finally, we show that even with training a REST learner with a small number of training data, the performance does not drop significantly, which results in good sample efficiency.
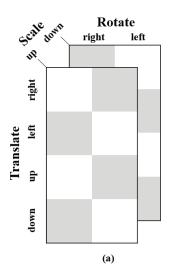
Improvement of Classification Performance

To evaluate the improvement of black-box's performance by using REST learner, we begin with the black-box gray-scale numeric image classifier. We use the CNN pre-trained with MNIST dataset as a black-box classifier which performs 99% accuracy in MNIST test data. However, when applying randomly rotated MNIST test images (R), the accuracy decreases to 69.28%. By attaching the REST in front of the black-box model, the accuracy increases back to 97.63%.

We also experiment with more difficult tasks by generating the data with rotation and scaling (RSc), rotation and shearing (RSh), rotation, scaling, and shearing (RSS), and rotation, scaling, shearing, and translation (RSST) of the MNIST test digits.

Examples of the sequence of transformation for each warping method are shown in Figure 3. All the affinely warped input images in the first column are transformed two times to get the final output. The quantitative results are shown in Table 1. As task complexity increases from simple rotation to full affine transformation, the black-box classifier gets worse for predicting the label. For the RSST method, the accuracy of the black box model is 17.99 percent, which is almost random choice. However, by adding the REST model in front of the black-box model, it highly improves the performance. The difference in accuracy between applying the REST model and only using black-box model gets larger as the task becomes more difficult.

Figure 2. shows the sequence of how the distorted input image is transformed by REST. When affinely extended image of number 9 is taken for the first state S_1, the RL module outputs transform parameter $\theta = (20.4, 1.1, 1.1, 0, 0, -4.9, -4.4)$. Then, the warp module receives the state S_1 and the parameter θ to produce the next state S_2, which is rotated clockwise, scaled up, and translated to the upper-left direction. The black-box model maps the transformed image S_2 to a confidence score c_2. As the confidence score does not exceed a terminal threshold, the process iterates until the terminal condition is satisfied. An interesting observation is that the final image S_5 seems to resemble a style of data sampled from in-distribution D, which is the MNIST dataset in this case. The number 9 in the im-
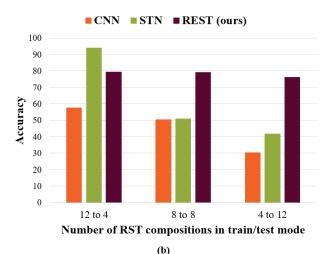
Figure 4: The first figure is an example of separating the composition of RST transformation into two disjoint subsets. The gray parts are used for generating distorted training data while the white parts are for deformed test data. The second figure shows test results on three different decomposition setting. Our models shows the best generalization effect as the task gets harder.

age S_5 is standing upright and locating at the center of an image with its size similar to typical MNIST data.

We further evaluate our model in real-world datasets. We use three datasets which are SVHN, CIFAR10, and STL10. For the black-box models, we use CNN models that perform 96.03%, 93.78%, and 77.59% for accuracy, respectively. Table 2 shows the test accuracy of black-box classifier before and after attaching REST. The performance improves when the REST model is applied at the front-end of the black-box model.

Generalization

In the previous section, we have shown that the REST works well by interacting with the black-box model. However, the limitation of the previous experiment is that the training and test data are generated in the same transformation format. In this section, we demonstrate the effect of generalization by generating the training data and test data in a different condition.

We first set the black box model as a general MNIST classifier. We then generate a distorted MNIST dataset by rotation, scaling, and translation (RST) to train the REST learner. We constrain the behavior of transformation by choosing one in (right, left), (up, down), (right, left, up, down) for each transformation R, S, and T, respectively. Therefore, there are 16 possible transformations of RST. We split 16 transformations of RST into two disjoint subsets, and apply each of the subset to MNIST for generating the training data and the test data. Figure 4(a) shows an example of separating 16 transformations into two subsets, each containing 8 transformations.

We compared the REST model with CNN and STN, and the results are shown in Figure 4(b). When the training dataset is generated by 12 RST transformations, STN performs better than ours. However, as the number of RST com-

Method	R	RSc	RSh	RSS	RSST
CNN	87.24	82.91	76.67	70.97	38.16
CNN++	88.69	85.86	80.48	76.58	39.92
STN	55.95	38.60	34.87	27.83	14.94
STN++	93.63	92.13	89.37	85.39	80.80
REST (ours)	**97.02**	**95.94**	**93.45**	**92.79**	**83.18**

Table 3: The accuracy of models trained by a small number of data. We trained CNN, STN, and REST by 1000 affine-warped training data. CNN++ and STN++ are trained by 56000 data where 55000 is a typical MNIST training data and the rest 1000 is affine-warped training data. The result shows that our model performs best on all different kinds of transformation.

positions used for transforming the training data is decreasing, CNN and STN suffer from predicting the correct labels. On the contrary, our model has a slight decrease in accuracy. It means that our model has better generalization in data shift conditions. It is assumed that our model shows a generalization effect because our model takes exploration in the training process, which generates samples that are not present in the training data distribution.

Sample Efficiency

In this section, we examine the effect of sample efficiency of our model. While 55,000 randomly affine-warped data has been generated from 55,000 training MNIST data to train the REST learner, in this section, we only generate 1000 randomly affine-warped data from 1000 training MNIST data. Then we perform the same process for training and test the model. We compare our model with CNN and STN. Both networks are trained by 1000 data. We also trained CNN and STN with 56,000 data, where 55,000 data is MNIST training data and the other 1000 data is randomly affine-warped data.

We call it CNN++ and STN++, respectively. All models are tested with 10,000 affine-warped test data and the results are shown in Table 3.

As a few data is used for training CNN and STN, they result in low accuracy for all different types of transformations. Also, although many data is used for training CNN++ and STN++, they also show a low performance because only a small number of training data is in the same distribution with that of test data. On the contrary, our model has a best performance. It is considered that our model shows a sample efficiency because of the exploration process in the training step. In the process of creating a state to train the model, new images are created for each episode. These images can be considered as training data which is implicitly augmented.

Conclusion

Robustness to real-world data is essential for deep learning systems to be successfully deployed in reality. Most studies so far have focused on improving generalization performance on test datasets when a single whole dataset is randomly split into training and test dataset. In other words, the training and test dataset are drawn from the same distribution and have similar sample statistics. Under this experimental assumption, the performance on the real-world data is not necessarily guaranteed even if the system generalizes well on the test dataset.

In this paper, we addressed geometric invariance using deep RL by transforming out-of-distribution samples into training distribution of the pretrained black box classifier in the system. We showed that the proposed method, REST, can improve the robustness of deep learning systems to various image warping. Specifically, as the complexity of the task gradually increased from simple rotation to full affine transformation, i.e. from one to six degrees of freedom, the relative performance of REST over the black-box model also increased accordingly.

We analyzed the generalization performance on unknown transformations by defining 16 disjoint subsets of affine transformations. REST generalized better as we trained it with a fewer number of transformation combinations while more of novel and unseen transformations were given at test time. Lastly, we experimented baseline methods with only 1000 affine-warped training data and showed REST is efficient in learning with a small number of samples as well. The action space of our method is focused on geometric transformation in this work but can be extended to other image processing techniques such as auto exposure, white balancing, edge enhancement, noise reduction to fill the gap between controlled experimental settings and real-world scenarios in the future works.

Acknowledgments

References

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural network. In *International Conference on Machine Learning*, 1613–1622.

Caicedo, J. C., and Lazebnik, S. 2015. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2488–2496.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, 764–773.

Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; and Dai, Q. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems* 28(3):653–664.

Follmann, P., and Bottger, T. 2018. A rotationally-invariant convolution module by feature map back-rotation. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 784–792. IEEE.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

Gal, Y. 2016. Uncertainty in deep learning. *University of Cambridge*.

Graves, A. 2011. Practical variational inference for neural networks. In *Advances in neural information processing systems*, 2348–2356.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Jacovi, A.; Hadash, G.; Kermany, E.; Carmeli, B.; Lavi, O.; Kour, G.; and Berant, J. 2019. Neural network gradient-based learning of black-box function interfaces. In *International Conference on Learning Representations*.

Jaderberg, M.; Simonyan, K.; Zisserman, A.; et al. 2015. Spatial transformer networks. In *Advances in neural information processing systems*, 2017–2025.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Kong, X.; Xin, B.; Wang, Y.; and Hua, G. 2017. Collaborative deep reinforcement learning for joint object search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1695–1704.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17(1):1334–1373.

Li, D.; Wu, H.; Zhang, J.; and Huang, K. 2018. A2-rl: Aesthetics aware reinforcement learning for image cropping. In

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 8193–8201.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Marcos, D.; Volpi, M.; and Tuia, D. 2016. Learning rotation invariant convolutional filters for texture classification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, 2012–2017. IEEE.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Park, J.; Lee, J.-Y.; Yoo, D.; and So Kweon, I. 2018. Distort-and-recover: Color enhancement using deep reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5928–5936.

Rao, Y.; Lu, J.; and Zhou, J. 2017. Attention-aware deep reinforcement learning for video face recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, 3931–3940.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 779–788.

Ren, L.; Lu, J.; Wang, Z.; Tian, Q.; and Zhou, J. 2018. Collaborative deep reinforcement learning for multi-object tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 586–602.

Ritter, H.; Botev, A.; and Barber, D. 2018. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*.

Sabour, S.; Frosst, N.; and Hinton, G. E. 2017. Dynamic routing between capsules. In *Advances in neural information processing systems*, 3856–3866.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Simard, P. Y.; Steinkraus, D.; Platt, J. C.; et al. 2003. Best practices for convolutional neural networks applied to visual document analysis. In *Icdar*, volume 3.

Tang, Y.; Tian, Y.; Lu, J.; Li, P.; and Zhou, J. 2018. Deep progressive reinforcement learning for skeleton-based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5323–5332.

Worrall, D. E.; Garbin, S. J.; Turmukhambetov, D.; and Brostow, G. J. 2017. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5028–5037.

Xu, Y.; Xiao, T.; Zhang, J.; Yang, K.; and Zhang, Z. 2014. Scale-invariant convolutional neural networks. *arXiv preprint arXiv:1411.6369*.

You, Y.; Lou, Y.; Liu, Q.; Ma, L.; Wang, W.; Tai, Y.; and Lu,

C. 2018. Prin: Pointwise rotation-invariant network. *arXiv preprint arXiv:1811.09361*.

Yu, K.; Dong, C.; Lin, L.; and Change Loy, C. 2018. Crafting a toolchain for image restoration by deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2443–2452.

Yun, S.; Choi, J.; Yoo, Y.; Yun, K.; and Young Choi, J. 2017. Action-decision networks for visual tracking with deep reinforcement learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2711–2720.