Robust and Generalizable Visual Representation Learning via Random Convolutions

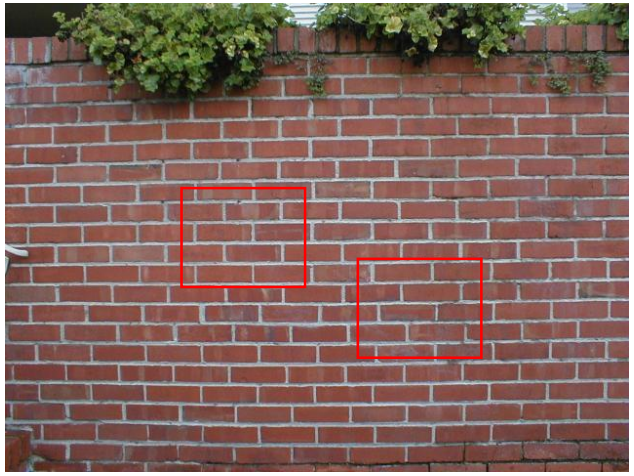
Zhenlin Xu[1], **Deyi Liu**[1], **Junlin Yang**[2], **Colin Raffel**[1], and **Marc Niethammer**[1]

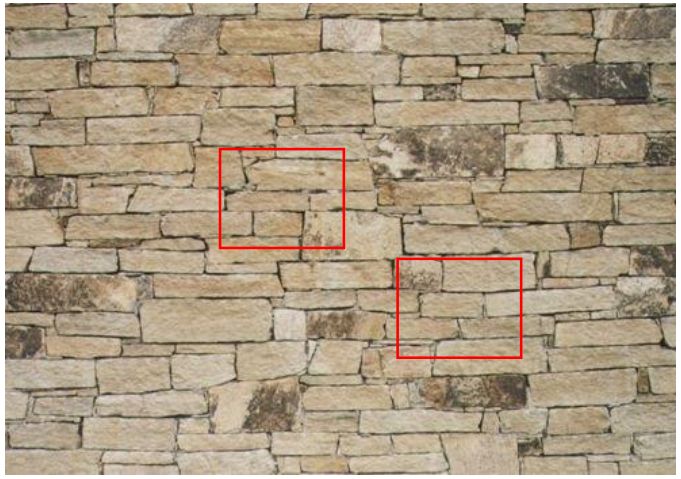
[1] University of North Carolina at Chapel Hill
[2] Yale University

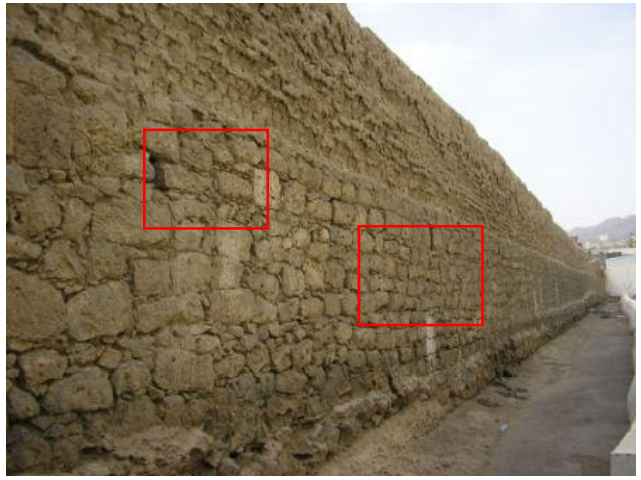
[1]{zhenlinx, mn, craffel}@cs.unc.edu, deyi@live.unc.edu
[2]junlin.yang@yale.edu

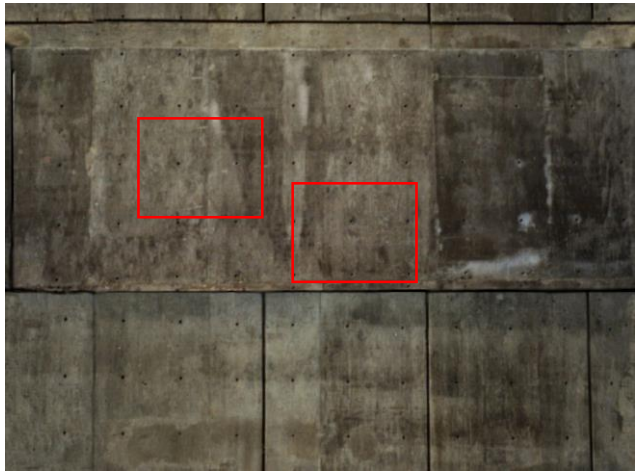
ICLR 2021

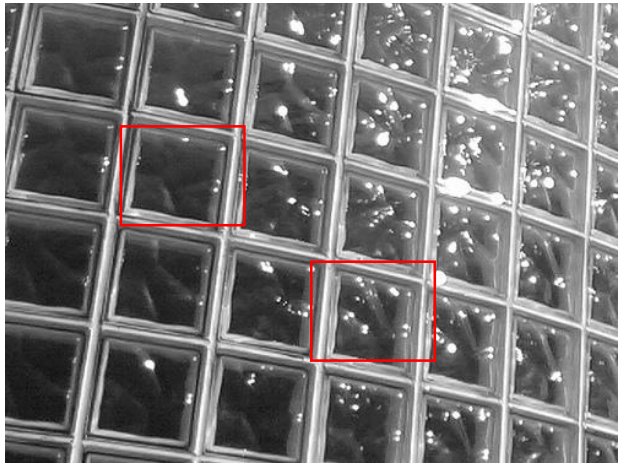
Texture



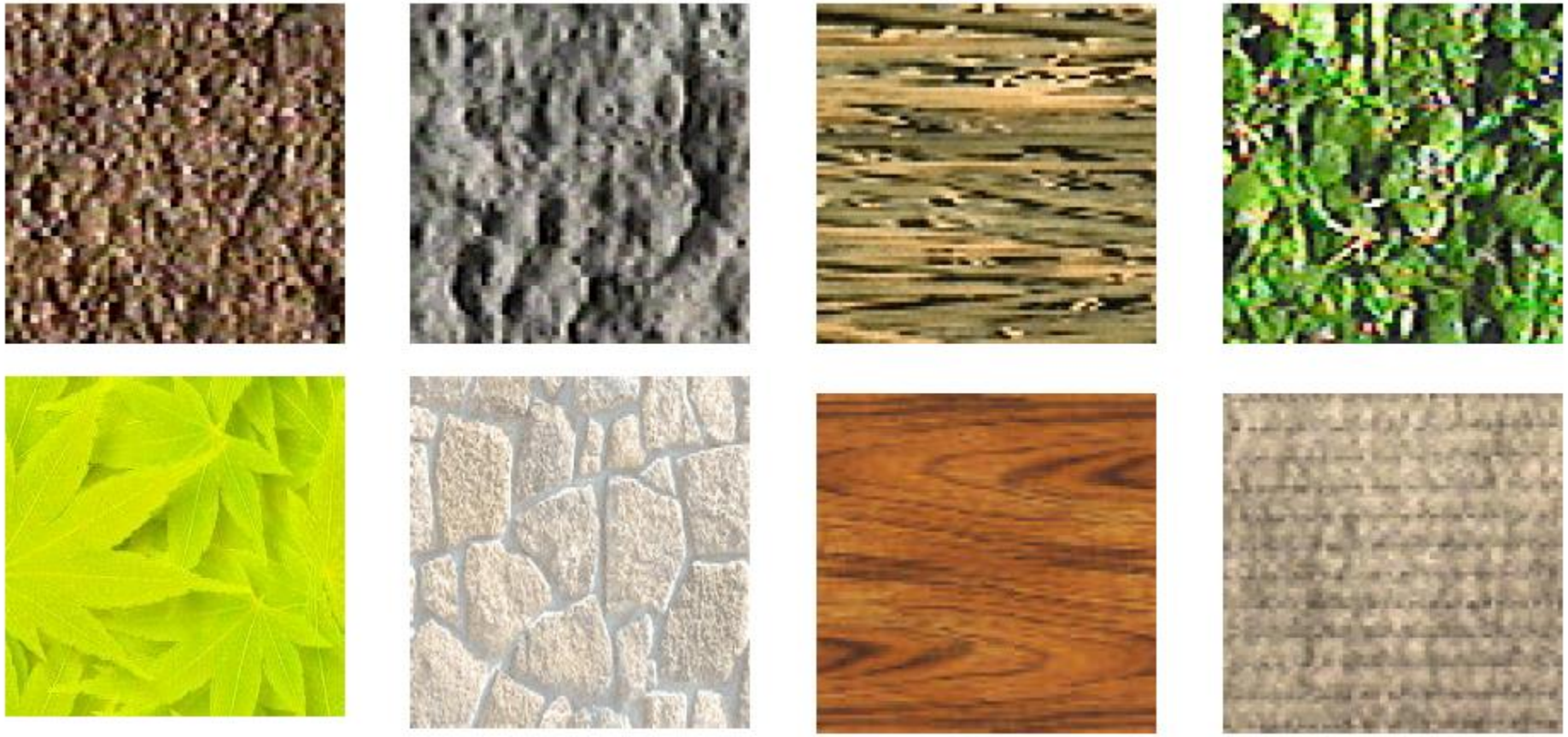






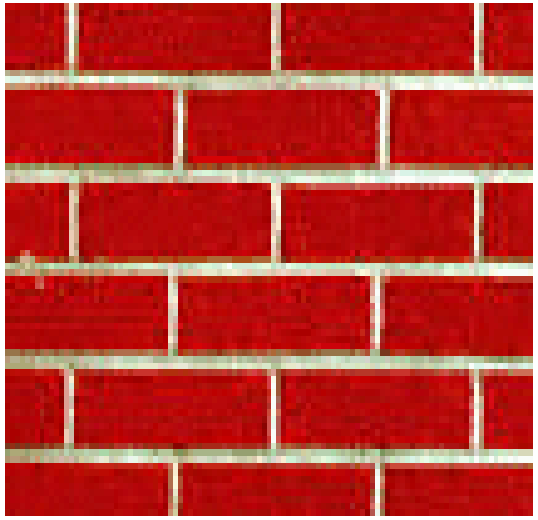
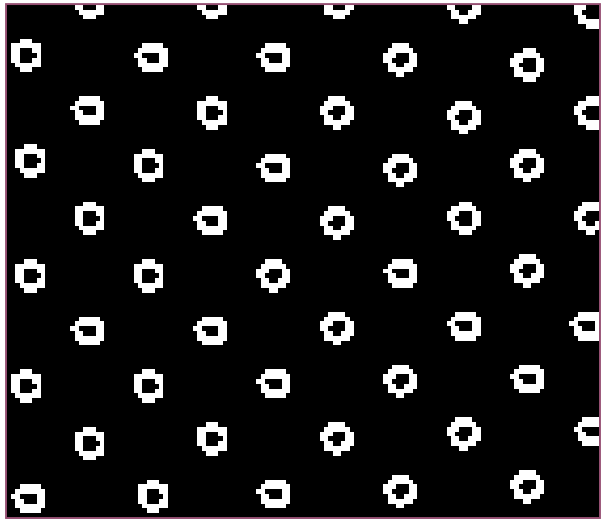
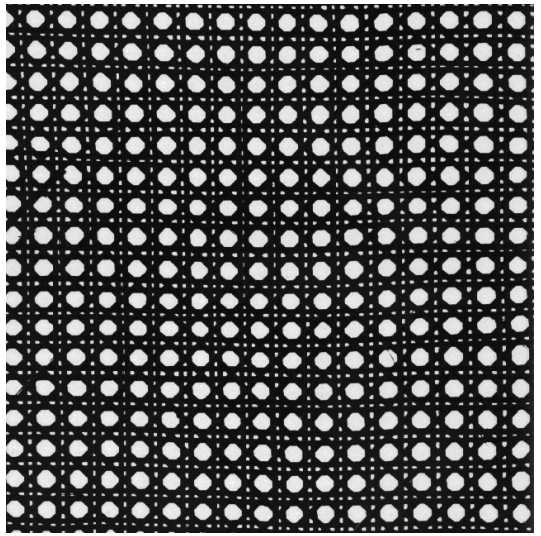


Texture



Texture

- Regular Patterns

Global Shape VS Texture

Deep neural networks have the tendency to utilize the superficial features like color and local texture rather than global shapes[1]

(a) Texture image

81.4%	**Indian elephant**
10.3%	indri
8.2%	black swan

(b) Content image

71.1%	**tabby cat**
17.3%	grey fox
3.3%	Siamese cat

(c) Texture-shape cue conflict

63.9%	**Indian elephant**
26.4%	indri
9.6%	black swan

Figure 1: Classification of a standard ResNet-50 of **(a)** a texture image (elephant skin: only texture cues); **(b)** a normal image of a cat (with both shape and texture cues), and **(c)** an image with a texture-shape cue conflict, generated by style transfer between the first two images.

This bias can make neural networks vulnerable to domain shift and small perturbations.

[1] ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness

Global Shape VS Texture

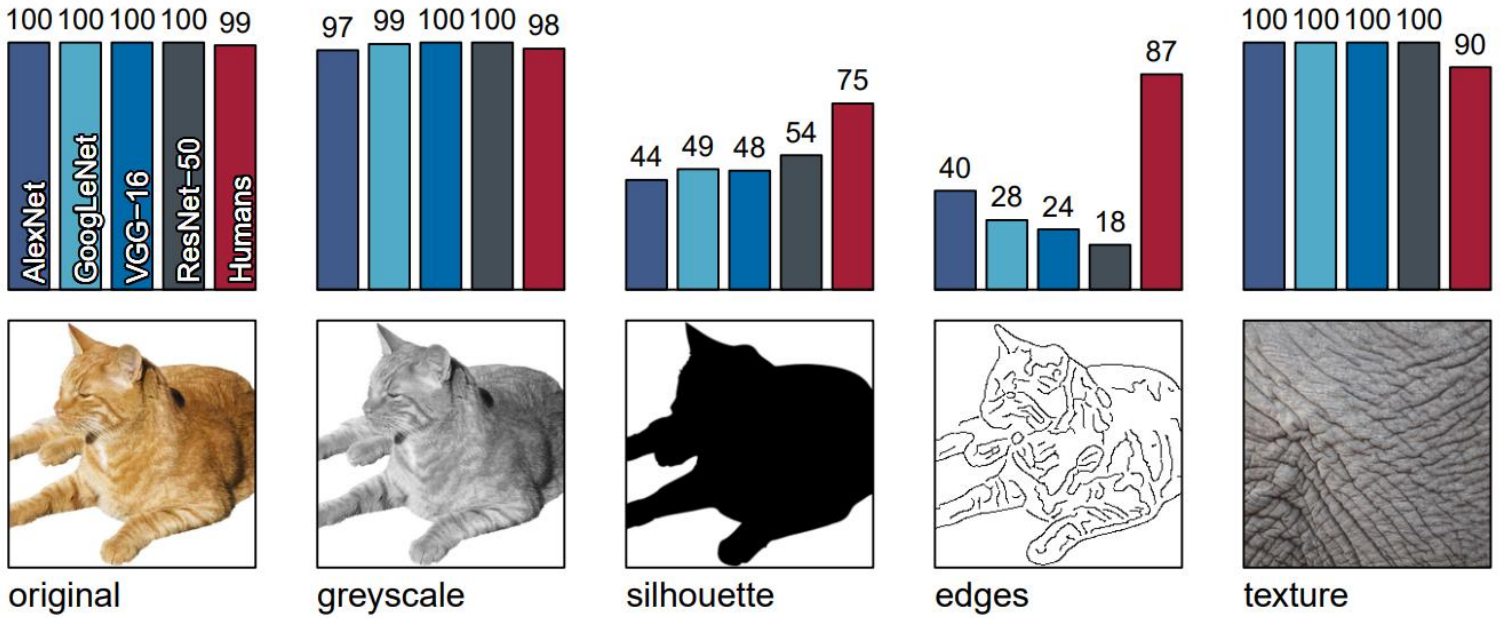
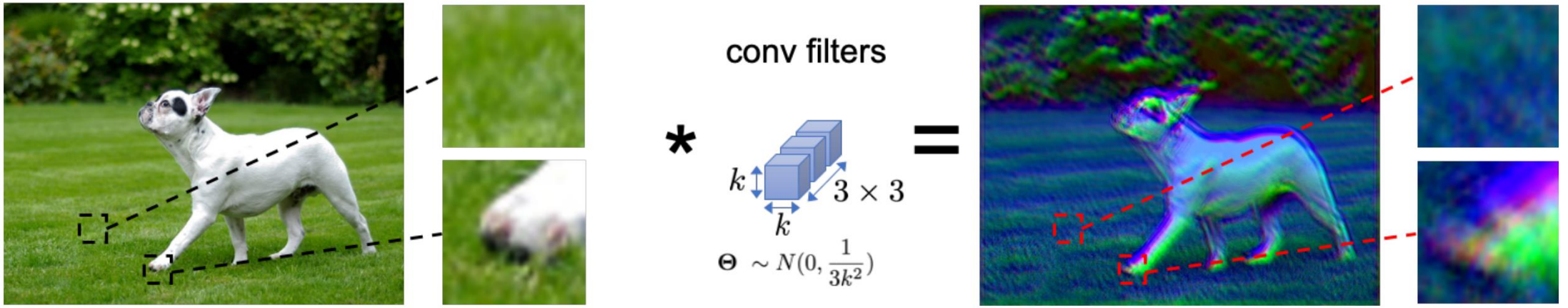


Figure 2: Accuracies and example stimuli for five different experiments without cue conflict.

Rand Conv

- A convolution layer with random sampled weights generates images with random texture but consistent shapes



Multi-scale & Mix

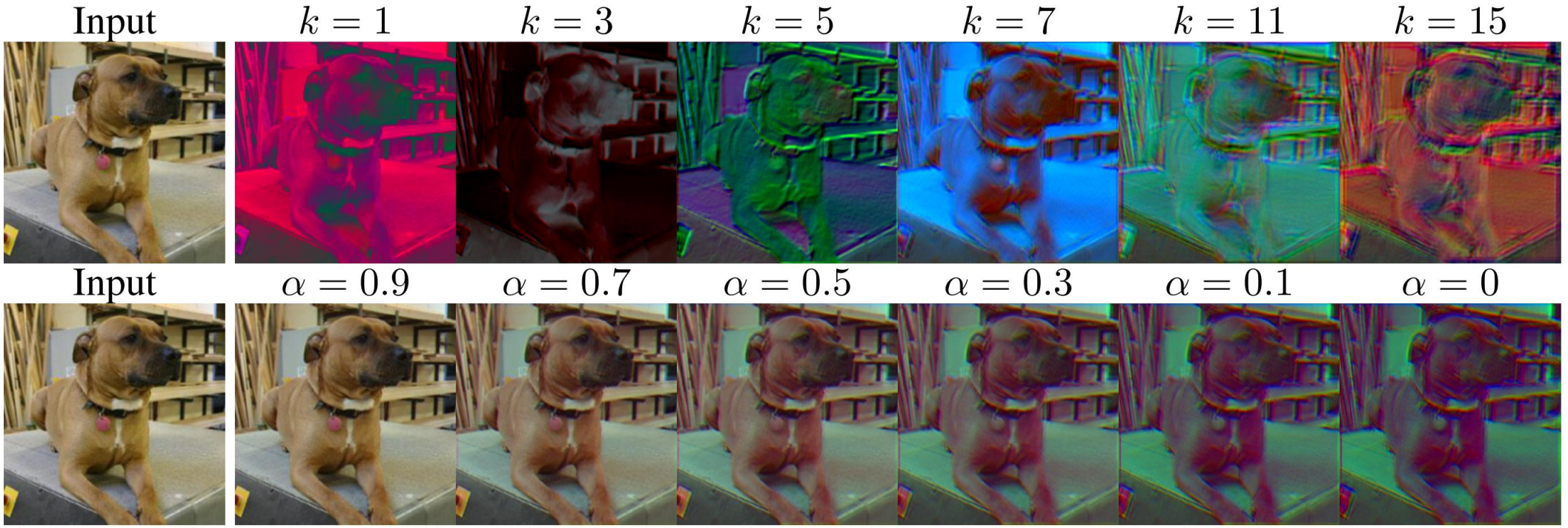


Figure 1: **Top**: Illustration that `RandConv` randomize local texture but preserve shapes in the image. **Middle:** First column is the input image of size 224^2; following columns are convolutions results using random filters of different sizes k. **Bottom:** Mixing results between an image and one of its random convolution results with different mixing coefficients α.

Consistency Loss

- 3 augmented variants of the same image

$$G_j = \mathrm{RandConv}^j(I)$$

- Use a model to predict for the 3 variants and the average predictions

$$y^j = \Phi\left(G^j\right) \quad \bar{y} = \sum_{j=1}^{3} y^j/3$$

- Penalize the KL divergence between the average prediction and individual predictions.

$$\sum_{j=1}^{3} \mathrm{KL}\left(y^j \| \bar{y}\right)$$

Algorithm

Algorithm 1 Learning with Data Augmentation by Random Convolutions

Input: Model Φ, task loss $\mathcal{L}_{task}$, training images $\{I_i\}_{i=1}^N$ and their labels $\{y_i\}_{i=1}^N$, pool of filter sizes $\mathcal{K} = \{1, ..., n\}$, fraction of original data p, whether to `mix` with original images, consistency loss weight λ
function RandConv(I, $\mathcal{K}$, `mix`, p)
    Sample $p_0 \sim U(0, 1)$
    if $p_0 < p$ and `mix` is False **then**
        return I ▷ When not in `mix` mode, use the original image with probability p
    else
        Sample scale $k \sim \mathcal{K}$
        Sample convolution weights $\mathbf{\Theta} \in \mathbb{R}^{k \times k \times 3 \times 3} \sim N(0, \frac{1}{3k^2})$
        $I_{rc} = I * \mathbf{\Theta}$ ▷ Apply convolution on I
        if `mix` is True **then**
            Sample $\alpha \sim U(0, 1)$
            return $\alpha I + (1 - \alpha) I_{rc}$ ▷ Mix with original images
        else
            return I_{rc}
Learning Objective:
for $i = 1 \rightarrow N$ **do**
    for $j = 1 \rightarrow 3$ **do**
        $\hat{y}_i^j = \Phi(\texttt{RandConv}(I_i))$ ▷ Predict labels for three augmented variants of the same image
    $\mathcal{L}_{cons} = \lambda \sum_{j=1}^3 \text{KL}(\hat{y}_i^j || \bar{y}_i)$ where $\bar{y}_i = \sum_{j=1}^3 \hat{y}_i^j / 3$ ▷ Consistency Loss
    $\mathcal{L} = \mathcal{L}_{task}(\hat{y}_i^1, y_i) + \lambda \mathcal{L}_{cons}$ ▷ Learning with the task loss and the consistency loss

Domain Generalization

Table 1: Average accuracy and 5-run standard deviation (in parenthesis) of MNIST10K model on MNIST-M, SVHN, SYNTH, USPS and their average (DG-avg); and average accuracy of 15 types of corruptions in MNIST-C. Both `RandConv` variants significantly outperform all other methods.

	MNIST	MNIST-M	SVHN	USPS	SYNTH	DG-Avg	MNIST-C
Data Augmentation							
Baseline	98.40(0.84)	58.87(3.73)	33.41(5.28)	79.27(2.70)	42.43(5.46)	53.50(4.23)	88.20(2.10)
GreyScale	98.82(0.02)	58.41(0.99)	36.06(1.48)	80.45(1.00)	45.00(0.80)	54.98(0.86)	89.15(0.44)
ColorJitter	98.72(0.05)	62.72(0.66)	39.61(0.88)	79.18(0.60)	46.40(0.34)	56.98(0.39)	89.48(0.18)
BandPass	98.65(0.11)	70.22(2.73)	48.34(2.56)	78.60(0.82)	57.17(2.01)	63.58(1.89)	87.89(0.68)
MultiAug	98.80(0.05)	62.32(0.66)	39.07(0.68)	79.31(1.02)	46.48(0.80)	56.79(0.34)	89.54(0.11)
Adversial							
PAR (our imp)	98.79(0.05)	61.16(0.21)	36.08(1.27)	79.95(1.18)	45.48(0.35)	55.67(0.33)	89.34(0.45)
GUD	-	60.41	35.51	77.26	45.32	54.62	-
M-ADA	-	67.94	42.55	78.53	48.95	59.49	-
$RC_{img1\text{-}7,\ p=0.5,\ \lambda=5}$	98.86(0.05)	87.67(0.37)	54.95(1.90)	82.08(1.46)	63.37(1.58)	72.02(1.15)	90.94(0.51)
$RC_{mix1\text{-}7,\lambda=10}$	98.85(0.04)	87.76(0.83)	57.52(2.09)	83.36(0.96)	62.88(0.78)	72.88(0.58)	**91.62(0.77)**
$RC_{mix1\text{-}7,\lambda=10}$ + MultiAug	98.82(0.06)	**87.89(0.29)**	**62.07(0.62)**	**84.39(1.02)**	**63.90(0.63)**	**74.56(0.46)**	91.40(0.93)

Robustness

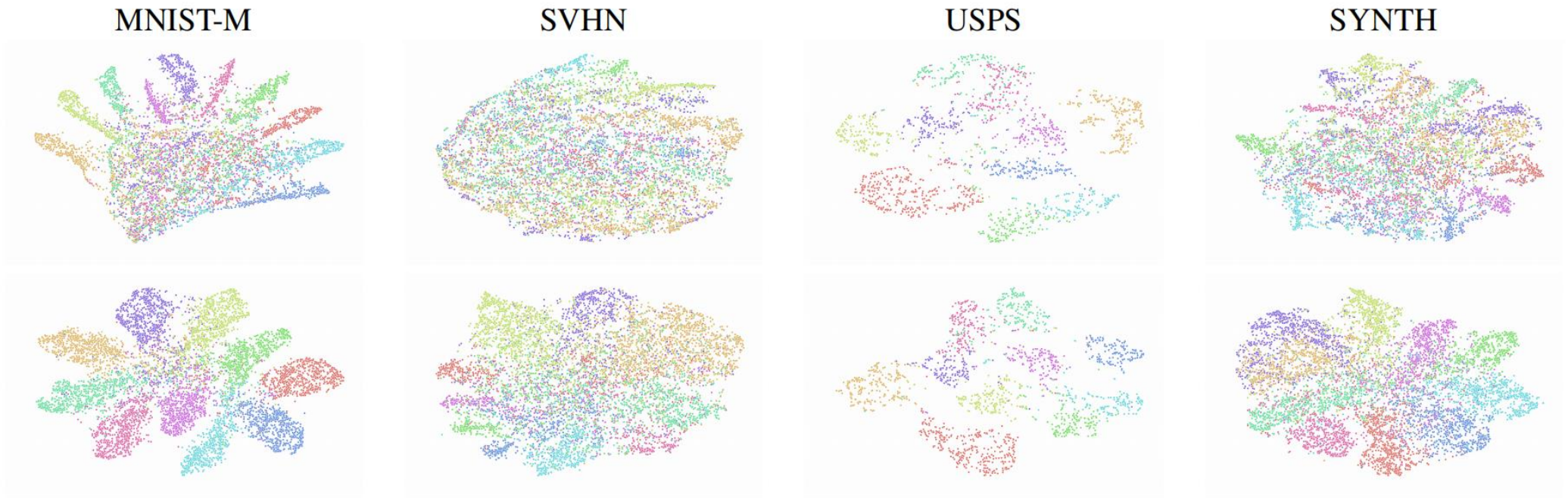


Figure 3: t-SNE feature embedding visualization for digit datasets for models trained on MNIST without (top) and with our $\text{RC}_{\text{mix1-7},\lambda=10}$ approach (bottom). Different colors denote different classes.

PAC Dataset

Base	Method	Photo	Art	Cartoon	Sketch	Average
Ours	Deep-All	86.77(0.42)	60.11(1.33)	64.12(0.32)	55.28(4.71)	66.57(1.36)
	GreyScale	83.93(1.47)	61.60(1.18)	62.12(0.61)	60.07(2.47)	66.93(0.83)
	ColorJitter	84.61(0.83)	59.01(0.24)	61.43(0.68)	62.44(1.68)	66.88(0.33)
	BandPass	87.08(0.57)	59.46(0.27)	**64.39(0.51)**	55.39(2.95)	66.58(0.73)
	MultiAug	85.21(0.47)	59.51(0.38)	62.88(1.01)	61.67(0.76)	67.32(0.23)
	PAR (our imp.)	**87.21(0.42)**	60.17(0.95)	63.63(0.88)	55.83(2.57)	66.71(0.58)
	$RC_{img1\text{-}7}$, p=0.5	86.50(0.72)	61.10(0.38)	64.24(0.62)	68.50(1.83)	70.09(0.43)
	$RC_{mix1\text{-}7}$	86.60(0.67)	61.74(0.90)	64.05(0.66)	69.74(0.66)	**70.53(0.25)**
	$RC_{mix1\text{-}7}$ + MultiAug	86.23(0.74)	**61.91(0.76)**	62.69(0.76)	67.74(1.21)	69.64(0.49)
	$RC_{img1\text{-}7}$, p=0.5, λ=10	81.15(0.76)	59.56(0.79)	62.42(0.59)	71.74(0.43)	68.72(0.58)
	$RC_{mix1\text{-}7}$,λ=10	81.78(1.11)	61.14(0.51)	63.57(0.29)	**71.97(0.38)**	69.62(0.24)

Transfer Pretrained Model

PACS	ImageNet	Photo	Art	Cartoon	Sketch	Avg
Deep-All	Baseline	**86.77 (0.42)**	60.11 (1.33)	64.12 (0.32)	55.28 (4.71)	66.57 (1.36)
	$RC_{img1\text{-}7,p=0.5,\lambda=10}$	84.48 (0.52)	62.61 (1.23)	66.13 (0.80)	69.24 (0.80)	70.61 (0.53)
	$RC_{mix1\text{-}7,\lambda=10}$	85.59 (0.40)	63.30 (0.99)	63.83 (0.85)	68.29 (1.27)	70.25 (0.45)
	SIN	85.33 (0.66)	**65.85 (0.87)**	65.39 (0.62)	65.75 (0.59)	70.58 (0.21)
$RC_{img1\text{-}7,p=0.5,\lambda=10}$	Baseline	81.15 (0.76)	59.56 (0.79)	62.42 (0.59)	71.74 (0.43)	68.72 (0.58)
	$RC_{img1\text{-}7,p=0.5,\lambda=10}$	84.36 (0.36)	63.73 (0.91)	**68.07 (0.55)**	75.41 (0.57)	72.89 (0.33)
	$RC_{mix1\text{-}7,\lambda=10}$	84.63 (0.97)	63.41 (1.22)	66.36 (0.43)	74.59 (0.84)	72.25 (0.54)
$RC_{mix1\text{-}7,\lambda=10}$	Baseline	81.78 (1.11)	61.14 (0.51)	63.57 (0.29)	71.97 (0.38)	69.62 (0.24)
	$RC_{img1\text{-}7,p=0.5,\lambda=10}$	85.16 (1.03)	63.17 (0.38)	67.68 (0.60)	**76.11 (0.43)**	**73.03 (0.46)**
	$RC_{mix1\text{-}7,\lambda=10}$	86.17 (0.56)	65.33 (1.05)	65.52 (1.13)	73.21 (1.03)	72.56 (0.50)

Ablation Study

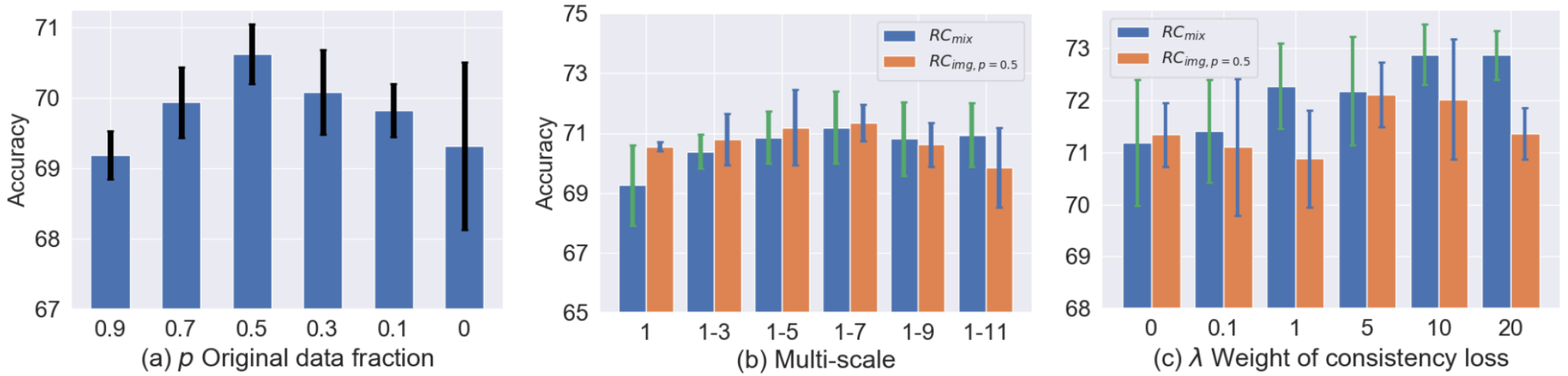


Figure 2: Average accuracy and 5-run variance of MNIST model on MNIST-M, SVHN, SYNTH and USPS. Studies for: (a) original data fraction p for RC_{img}; (b) multiscale design (1-n refers to using scales 1,3,..,n) for $\text{RC}_{\text{img},p=0.5}$ (orange) and RC_{mix} (blue); (c) consistency loss weight λ for $\text{RC}_{\text{img}1-7,p=0.5}$ (orange) and $\text{RC}_{\text{mix}1-7}$ (blue).