The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization Supplementary Material

Dan Hendrycks[1] Steven Basart[2*] Norman Mu[1*] Saurav Kadavath[1] Frank Wang[3]
Evan Dorundo[3] Rahul Desai[1] Tyler Zhu[1] Samyak Parajuli[1] Mike Guo[1]
Dawn Song[1] Jacob Steinhardt[1] Justin Gilmer[3]

A. Additional Results

ImageNet-R. Expanded ImageNet-R results are in Table 4. WSL pretraining on Instagram images appears to yield dramatic improvements on ImageNet-R, but the authors note the prevalence of artistic renditions of object classes on the Instagram platform. While ImageNet's data collection process actively excluded renditions, we do not have reason to believe the Instagram dataset excluded renditions. On a ResNeXt-101 32×8d model, WSL pretraining improves ImageNet-R performance by a massive 37.5% from 57.5% top-1 error to 24.2%. Ultimately, without examining the training images we are unable to determine whether ImageNet-R represents an actual distribution shift to the Instagram WSL models. However, we also observe that with greater controls, that is with ImageNet-21K pre-training, pretraining hardly helped ImageNet-R performance, so it is not clear that more pretraining data improves ImageNet-R performance.

Increasing model size appears to automatically improve ImageNet-R performance, as shown in Figure 1. A ResNet-50 (25.5M parameters) has 63.9% error, while a ResNet-152 (60M) has 58.7% error. ResNeXt-50 32×4d (25.0M) attains 62.3% error and ResNeXt-101 32×8d (88M) attains 57.5% error.

ImageNet-C. Expanded ImageNet-C results are Table 3. We also tested whether model size improves performance on ImageNet-C for even larger models. With a different codebase, we trained ResNet-50, ResNet-152, and ResNet-500 models which achieved 80.6, 74.0, and 68.5 mCE respectively. Expanded comparisons between ImageNet-C and Real Blurry Images is in Table 1.

ImageNet-A. ImageNet-A [5] is an adversarially filtered test set and is constructed based on existing model weaknesses (see [9] for another robustness dataset algorithmically determined by model weaknesses). This dataset contains examples that are difficult for a ResNet-50 to classify, so examples solvable by simple spurious cues are are especially infrequent in this dataset. Results are in Table 5. Notice Res2Net architectures [2] can greatly improve accuracy.

*Equal contribution.

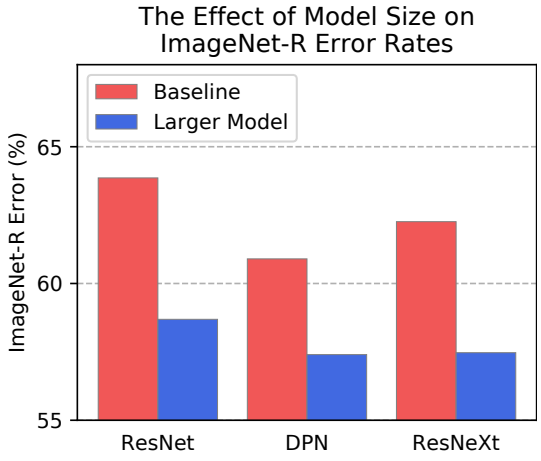

Figure 1: Larger models improve robustness on ImageNet-R. The baseline models are ResNet-50, DPN-68, and ResNeXt-50 (32 × 4d). The larger models are ResNet-152, DPN-98, and ResNeXt-101 (32 × 8d). The baseline ResNeXt has a 7.1% ImageNet error rate, while the large has a 6.2% error rate.

Results also show that *Larger Models*, *Self-Attention*, and *Pretraining* help, while *Diverse Data Augmentation* usually does not help substantially.

Implications for the Four Methods.
Larger Models help with ImageNet-C (+), ImageNet-A (+), ImageNet-R (+), yet does not markedly improve DFR (−) performance.
Self-Attention helps with ImageNet-C (+), ImageNet-A (+), yet does not help ImageNet-R (−) and DFR (−) performance.
Diverse Data Augmentation helps ImageNet-C (+), ImageNet-R (+), yet does not markedly improve ImageNet-A (−), DFR(−), nor SVSF (−) performance.
Pretraining helps with ImageNet-C (+), ImageNet-A (+), yet does not markedly improve DFR (−) nor ImageNet-R (−) performance.

Network	Defocus Blur	Glass Blur	Motion Blur	Zoom Blur	ImageNet-C Blur Mean	Real Blurry Images
ResNet-50	61	73	61	64	65	58.7
+ ImageNet-21K *Pretraining*	56	69	53	59	59	54.8
+ CBAM (*Self-Attention*)	60	69	56	61	62	56.5
+ ℓ_∞ Adversarial Training	80	71	72	71	74	71.6
+ Speckle Noise	57	68	60	64	62	56.9
+ Style Transfer	57	68	55	64	61	56.7
+ AugMix	52	65	46	51	54	54.4
+ DeepAugment	48	60	51	61	55	54.2
+ DeepAugment+AugMix	41	53	39	48	45	51.7
ResNet-152 (*Larger Models*)	67	81	66	74	58	54.3

Table 1: ImageNet-C Blurs (Defocus, Glass, Motion, Zoom) vs Real Blurry Images. All values are error rates and percentages. The rank orderings of the models on Real Blurry Images are similar to the rank orderings for "ImageNet-C Blur Mean," so ImageNet-C's simulated blurs track real-world blur performance.

Hypothesis	ImageNet-C	Real Blurry Images	ImageNet-A	ImageNet-R	DFR	SVSF
Larger Models	+	+	+	+	−	
Self-Attention	+	+	+	−	−	
Diverse Data Augmentation	+	+	−	+	−	−
Pretraining	+	+	+	−	−	

Table 2: A highly simplified account of each method when tested against different datasets. This table includes ImageNet-A results.

B. DeepAugment Details

Pseudocode. Below is Pythonic pseudocode for DeepAugment. The basic structure of DeepAugment is agnostic to the backbone network used, but specifics such as which layers are chosen for various transforms may vary as the backbone architecture varies. We do not need to train many different image-to-image models to get diverse distortions [10, 6]. We only use two existing models, the EDSR super-resolution model [7] and the CAE image compression model [8]. See full code for such details.

At a high level, DeepAugment processes each image with an image-to-image network. The image-to-image network's weights and feedforward activations are distorted with each pass. The distortion is made possible by, for example, negating the network's weights and applying dropout to the feedforward activations. These modifications were not carefully chosen and demonstrate the utility of mixing together diverse operations without tuning. The resulting image is distorted and saved. This process generates an augmented dataset.

Ablations. We run ablations on DeepAugment to understand the contributions from the EDSR and CAE models independently. Table 7 contains results of these experiments on ImageNet-R and Table 6 contains results of these experiments on ImageNet-C. In both tables, "DeepAugment (EDSR)" and "DeepAugment (CAE)" refer to experiments where we only use a single extra augmented training set (+ the standard training set), and train on those images.

Noise2Net. We show that untrained, randomly sampled neural networks can provide useful deep augmentations, highlighting the efficacy of the DeepAugment approach. While in the main paper we use EDSR and CAE to create DeepAugment augmentations, in this section we explore the use of randomly initialized image-to-image networks to generate diverse image augmentations. We propose a DeepAugment method, *Noise2Net*.

In Noise2Net, the architecture and weights are randomly sampled. Noise2Net is the composition of several residual blocks: $\text{Block}(x) = x + \varepsilon \cdot f_\Theta(x)$, where Θ is randomly initialized and ε is a parameter that controls the strength of the augmentation. For all our experiments, we use 4 Res2Net blocks [3] and $\varepsilon \sim U(0.375, 0.75)$. The weights of Noise2Net are resampled at every minibatch, and the dilation and kernel sizes of all the convolutions used in Noise2Net are randomly sampled every epoch. Hence Noise2Net augments an image to an augmented image by processing the image through a randomly sampled network with random weights.

Recall that in the case of EDSR and CAE, we used networks to generate a static dataset, and then we trained normally on that static dataset. This setup could not be done on-the-fly. That is because we fed in one example at a time with EDSR and CAE. If we pass the entire minibatch through

			Noise			Blur				Weather				Digital			
	Clean	*mCE*	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
ResNet-50	23.9	76.7	80	82	83	75	89	78	80	78	75	66	57	71	85	77	77
+ ImageNet-21K *Pretraining*	22.4	65.8	61	64	63	69	84	68	74	69	71	61	53	53	81	54	63
+ SE (*Self-Attention*)	22.4	68.2	63	66	66	71	82	67	74	74	72	64	55	71	73	60	67
+ CBAM (*Self-Attention*)	22.4	70.0	67	68	68	74	83	71	76	73	72	65	54	70	79	62	67
+ ℓ_∞ Adversarial Training	46.2	94.0	91	92	95	97	86	92	88	93	99	118	104	111	90	72	81
+ Speckle Noise	24.2	68.3	51	47	55	70	83	77	80	76	71	66	57	70	82	72	69
+ Style Transfer	25.4	69.3	66	67	68	70	82	69	80	68	71	65	58	66	78	62	70
+ AugMix	22.5	65.3	67	66	68	64	79	59	64	69	68	65	54	57	74	60	66
+ DeepAugment	23.3	60.4	49	50	47	59	73	65	76	64	60	58	51	61	76	48	67
+ DeepAugment + AugMix	24.2	53.6	46	45	44	50	64	50	61	58	57	54	52	48	71	43	61
ResNet-152 (*Larger Models*)	21.7	69.3	73	73	76	67	81	66	74	71	68	62	51	67	76	69	65
ResNeXt-101 32×8d (*Larger Models*)	20.7	66.7	68	69	71	65	79	66	71	69	66	60	50	66	74	61	64
+ WSL *Pretraining* (1000× data)	17.8	51.7	49	50	51	53	72	55	63	53	51	42	37	41	67	40	51
+ DeepAugment + AugMix	20.1	44.5	36	35	34	43	55	42	55	48	48	47	43	39	59	34	50

Table 3: Clean Error, Corruption Error (CE), and mean CE (mCE) values for various models, and training methods on ImageNet-C. The mCE value is computed by averaging across all 15 CE values. A CE value greater than 100 (e.g. adversarial training on contrast) denotes worse performance than AlexNet. DeepAugment+AugMix improves robustness by over 23 mCE.

	ImageNet-200 (%)	ImageNet-R (%)	Gap
ResNet-50 [4]	7.9	63.9	56.0
+ ImageNet-21K *Pretraining* (10× data)	7.0	62.8	55.8
+ CBAM (*Self-Attention*)	7.0	63.2	56.2
+ ℓ_∞ Adversarial Training	25.1	68.6	43.5
+ Speckle Noise	8.1	62.1	54.0
+ Style Transfer	8.9	58.5	49.6
+ AugMix	7.1	58.9	51.8
+ DeepAugment	7.5	57.8	50.3
+ DeepAugment + AugMix	8.0	53.2	45.2
ResNet-101 (*Larger Models*)	7.1	60.7	53.6
+ SE (*Self-Attention*)	6.7	61.0	54.3
ResNet-152 (*Larger Models*)	6.8	58.7	51.9
+ SE (*Self-Attention*)	6.6	60.0	53.4
ResNeXt-101 32×4d (*Larger Models*)	6.8	58.0	51.2
+ SE (*Self-Attention*)	5.9	59.6	53.7
ResNeXt-101 32×8d (*Larger Models*)	6.2	57.5	51.3
+ WSL *Pretraining* (1000× data)	4.1	24.2	20.1
+ DeepAugment + AugMix	6.1	47.9	41.8

Table 4: ImageNet-200 and ImageNet-Renditions error rates. ImageNet-21K and WSL Pretraining are *Pretraining* methods, and pretraining gives mixed benefits. CBAM and SE are forms of *Self-Attention*, and these *hurt* robustness. ResNet-152 and ResNeXt-101 32×8d test the impact of using *Larger Models*, and these help. Other methods augment data, and Style Transfer, AugMix, and DeepAugment provide support for the *Diverse Data Augmentation*.

EDSR or CAE, we will end up applying the same augmentation to all images in the minibatch, reducing stochasticity and augmentation diversity. In contrast, Noise2Net enables us to process batches of images on-the-fly and obviates the need for creating a static augmented dataset.

In Noise2Net, each example is processed differently in parallel, so we generate more diverse augmentations in real-time. To make this possible, we use grouped convolutions. A grouped convolution with number of groups = N will take a set of kN channels as input, and apply N independent convolutions on channels $\{1,\dots,k\},\{k+1,\dots,2k\},\dots,\{(N-1)k+1,\dots,Nk\}$. Given a minibatch of size B, we can apply a randomly initialized grouped convolution with $N = B$ groups in order to apply a different random convolutional fil-

	ImageNet-A (%)
ResNet-50	2.2
+ ImageNet-21K *Pretraining* (10× data)	11.4
+ Squeeze-and-Excitation (*Self-Attention*)	6.2
+ CBAM (*Self-Attention*)	6.9
+ ℓ_∞ Adversarial Training	1.7
+ Style Transfer	2.0
+ AugMix	3.8
+ DeepAugment	3.5
+ DeepAugment + AugMix	3.9
ResNet-152 (*Larger Models*)	6.1
ResNet-152+Squeeze-and-Excitation (*Self-Attention*)	9.4
Res2Net-50 v1b	14.6
Res2Net-152 v1b (*Larger Models*)	22.4
ResNeXt-101 (32 × 8d) (*Larger Models*)	10.2
+ WSL *Pretraining* (1000× data)	45.4
+ DeepAugment + AugMix	11.5

Table 5: ImageNet-A top-1 accuracy.

ter to each element in the batch in a single forward pass. By replacing all the convolutions in each Res2Net block with a grouped convolution and randomly initializing network weights, we arrive at Noise2Net, a variant of DeepAugment. See Figure 2 for a high-level overview of Noise2Net and Figure 3 for sample outputs.

We evaluate the Noise2Net variant of DeepAugment on ImageNet-R. Table 7 shows that it outperforms the EDSR and CAE variants of DeepAugment, even though the network architecture is randomly sampled, its weights are random, and the network is not trained. This demonstrates the flexibility of the DeepAugment approach. Below is Pythonic pseudocode for training a classifier using the Noise2Net variant of DeepAugment.

```
def main():
    net.apply_weights(deepAugment_getNetwork())  # EDSR, CAE, ...
    for image in dataset:  # May be the ImageNet training set
        if np.random.uniform() < 0.05:  # Arbitrary refresh prob
            net.apply_weights(deepAugment_getNetwork())
        new_image = net.deepAugment_forwardPass(image)

def deepAugment_getNetwork():
    weights = load_clean_weights()
    weight_distortions = sample_weight_distortions()
    for d in weight_distortions:
        weights = apply_distortion(d, weights)
    return weights


def sample_weight_distortions():
    distortions = [
        negate_weights,
        zero_weights,
        flip_transpose_weights,
        ...
    ]

    return random_subset(distortions)

def sample_signal_distortions():
    distortions = [
        gelu,
        negate_signal_random_mask,
        flip_signal,
        ...
    ]

    return random_subset(distortions)


class Network():
    def apply_weights(weights):
        ... # Apply given weight tensors to network

    # Clean forward pass. Compare to deepAugment_forwardPass()
    def clean_forwardPass(X):
        X = network.block1(X)
        X = network.block2(X)
        ...
        X = network.blockN(X)
        return X

    # Our forward pass. Compare to clean_forwardPass()
    def deepAugment_forwardPass(X):
```

```
        # Returns a list of distortions, each of which
        # will be applied at a different layer.
        signal_distortions = sample_signal_distortions()

        X = network.block1(X)
        apply_layer_1_distortions(X, signal_distortions)
        X = network.block2(X)
        apply_layer_2_distortions(X, signal_distortions)
        ...
        apply_layer_N-1_distortions(X, signal_distortions)
        X = network.blockN(X)
        apply_layer_N_distortions(X, signal_distortions)

    return X


def train_one_epoch(classifier, batch_size, dataloader):
    noise2net = Noise2Net(batch_size=batch_size)
    for batch, target in dataloader:
        noise2net.reload_weights()
        noise2net.set_epsilon(random.uniform(0.375, 0.75))
        logits = model(noise2net.forward(batch))
        ... # Calculate loss and backrop

def train():
    for epoch in range(epochs):
        train_one_epoch(classifier, batch_size, dataloader)

class DeepAugment_Noise2Net:
    def __init__(self, batch_size=5):
        self.block1 = Res2NetBlock(batch_size=batch_size)
        self.block2 = Res2NetBlock(batch_size=batch_size)
        self.block3 = Res2NetBlock(batch_size=batch_size)
        self.block4 = Res2NetBlock(batch_size=batch_size)

    def reload_weights(self):
        ... # Reload Network parameters

    def set_epsilon(self, new_eps):
        self.epsilon = new_eps

    def forward(self, x):
        x = x + self.block1(x) * self.epsilon
        x = x + self.block2(x) * self.epsilon
        x = x + self.block3(x) * self.epsilon
        x = x + self.block4(x) * self.epsilon
        return x
```

C. Further Dataset Descriptions

ImageNet-R Classes. The 200 ImageNet classes and their WordNet IDs in ImageNet-R are as follows.

Goldfish, great white shark, hammerhead, stingray, hen, ostrich, goldfinch, junco, bald eagle, vulture, newt, axolotl, tree frog, iguana, African chameleon, cobra, scorpion, tarantula, centipede, peacock, lorikeet, hummingbird, toucan, duck, goose, black swan, koala, jellyfish, snail, lobster, hermit crab, flamingo, american egret, pelican, king penguin, grey whale, killer whale, sea lion, chihuahua, shih tzu, afghan hound, basset hound, beagle, bloodhound, italian greyhound, whippet, weimaraner, yorkshire terrier, boston terrier, scottish terrier, west highland white terrier, golden retriever, labrador retriever, cocker spaniels, collie, border collie, rottweiler, german shepherd dog, boxer, french bulldog, saint bernard, husky, dalmatian, pug, pomeranian, chow chow, pembroke welsh corgi, toy poodle, standard poodle, timber wolf, hyena, red fox, tabby cat, leopard, snow leopard, lion, tiger, cheetah, polar bear, meerkat, ladybug, fly, bee, ant, grasshopper, cockroach, mantis, dragonfly, monarch butterfly, starfish, wood rabbit, porcupine, fox squirrel, beaver, guinea pig, zebra, pig, hippopotamus, bison, gazelle, llama, skunk, badger, orangutan, gorilla, chimpanzee, gibbon, baboon, panda, eel, clown fish, puffer fish, accordion, ambulance, assault rifle, backpack, barn, wheelbarrow, basketball, bathtub, lighthouse, beer glass, binoculars, birdhouse, bow tie, broom, bucket, cauldron, candle, cannon, canoe, carousel, castle, mobile phone, cowboy hat, electric guitar, fire engine, flute, gasmask, grand piano, guillotine, hammer, harmonica, harp, hatchet, jeep, joystick, lab coat, lawn mower, lipstick, mailbox, missile, mitten, parachute, pickup truck, pirate ship, revolver, rugby ball, sandal, saxophone, school bus, schooner, shield, soccer ball, space shuttle, spider web, steam locomotive, scarf, submarine, tank, tennis ball, tractor, trombone, vase, violin, military aircraft, wine bottle, ice cream, bagel, pretzel, cheeseburger, hotdog, cabbage, broccoli, cucumber, bell pepper, mushroom, Granny Smith, strawberry, lemon, pineapple, banana, pomegranate, pizza, burrito, espresso, volcano, baseball player, scuba diver, acorn.

n01443537, n01484850, n01494475, n01498041, n01514859, n01518878, n01531178, n01534433, n01614925, n01616318, n01630670, n01632777, n01644373, n01677366, n01694178, n01748264, n01770393, n01774750, n01784675, n01806143,

			Noise			Blur				Weather				Digital			
	Clean	*mCE*	Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Contrast	Elastic	Pixel	JPEG
ResNet-50	23.9	76.7	80	82	83	75	89	78	80	78	75	66	57	71	85	77	77
+ DeepAugment (EDSR)	23.5	64.0	56	57	54	64	77	71	78	68	64	64	55	64	78	46	67
+ DeepAugment (CAE)	23.2	67.0	58	60	62	62	75	73	77	68	66	60	52	66	80	63	78
+ DeepAugment (Both)	23.3	60.4	49	50	47	59	73	65	76	64	60	58	51	61	76	48	67

Table 6: Clean Error, Corruption Error (CE), and mean CE (mCE) values for DeepAugment ablations on ImageNet-C. The mCE value is computed by averaging across all 15 CE values.

	ImageNet-200 (%)	ImageNet-R (%)	Gap
ResNet-50	7.9	63.9	56.0
+ DeepAugment (EDSR)	7.9	60.3	55.1
+ DeepAugment (CAE)	7.6	58.5	50.9
+ DeepAugment (EDSR + CAE)	7.5	57.8	50.3
+ DeepAugment (Noise2Net)	7.2	57.6	50.4
+ DeepAugment (All 3)	7.4	56.0	48.6

Table 7: DeepAugment ablations on ImageNet-200 and ImageNet-Renditions.

n01820546, n01833805, n01843383, n01847000, n01855672, n01860187, n01882714, n01910747, n01944390, n01983481, n01986214, n02007558, n02009912, n02051845, n02056570, n02066245, n02071294, n02077923, n02085620, n02086240, n02088094, n02088238, n02088364, n02088466, n02091032, n02091134, n02092339, n02094433, n02096585, n02097298, n02098286, n02099601, n02099712, n02102318, n02106030, n02106166, n02106550, n02106662, n02108089, n02108915, n02109525, n02110185, n02110341, n02110958, n02112018, n02112137, n02113023, n02113624, n02113799, n02114367, n02117135, n02119022, n02123045, n02128385, n02128757, n02129165, n02129604, n02130308, n02134084, n02138441, n02165456, n02190166, n02206856, n02219486, n02226429, n02233338, n02236044, n02268443, n02279972, n02317335, n02325366, n02346627, n02356798, n02363005, n02364673, n02391049, n02395406, n02398521, n02410509, n02423022, n02437616, n02445715, n02447366, n02480495, n02480855, n02481823, n02483362, n02486410, n02510455, n02526121, n02607072, n02655020, n02672831, n02701002, n02749479, n02769748, n02793495, n02797295, n02802426, n02808440, n02814860, n02823750, n02841315, n02843684, n02883205, n02906734, n02909870, n02939185, n02948072, n02950826, n02951358, n02966193, n02980441, n02992529, n03124170, n03272010, n03345487, n03372029, n03424325, n03452741, n03467068, n03481172, n03494278, n03495258, n03498962, n03594945, n03602883, n03630383, n03649909, n03676483, n03710193, n03773504, n03775071, n03888257, n03930630, n03947888, n04086273, n04118538, n04133789, n04141076, n04146614, n04147183, n04192698, n04254680, n04266014, n04275548, n04310018, n04325704, n04347754, n04389033, n04409515, n04465501, n04487394, n04522168, n04536866, n04552348, n04591713, n07614500, n07693725, n07695742, n07697313, n07697537, n07714571, n07714990, n07718472, n07720875, n07734744, n07742313, n07745940, n07749582, n07753275, n07753592, n07768694, n07873807, n07880968, n07920052, n09472597, n09835506, n10565667, n12267677.

Street View StoreFronts. The classes are

- auto shop
- bakery
- bank
- beauty salon
- car dealer
- car wash
- cell phone store
- dentist
- discount store
- dry cleaner
- furniture store
- gas station
- gym
- hardware store
- hotel
- liquor store
- pharmacy
- religious institution
- storage facility
- veterinary care.

DeepFashion Remixed. The classes are

- short sleeve top
- long sleeve top
- short sleeve outerwear
- long sleeve outerwear
- vest
- sling
- shorts
- trousers
- skirt
- short sleeve dress
- long sleep dress
- vest dress
- sling dress.

Size (small, moderate, or large) defines how much of the image the article of clothing takes up. Occlusion (slight, medium, or heavy) defines the degree to which the object is occluded from the camera. Viewpoint (front, side/back, or not worn) defines the camera position relative to the article of clothing. Zoom (no zoom, medium, or large) defines how much camera zoom was used to take the picture.

References

[1] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *CVPR*, 2009. 8

[2] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xinyu Zhang, Ming-Hsuan Yang, and Philip H. S. Torr. Res2net: A new multi-scale backbone architecture. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 1

[3] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip H.S. Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. corr abs/1512.03385 (2015), 2015. 3

[5] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *ArXiv*, abs/1907.07174, 2019. 1

[6] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. In *ICLR*, 2020. 2

[7] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 136–144, 2017. 2

[8] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. *arXiv preprint arXiv:1703.00395*, 2017. 2

[9] Haotao Wang, Tianlong Chen, Zhangyang Wang, and Kede Ma. I am going mad: Maximum discrepancy competition for comparing classifiers adaptively. In *International Conference on Learning Representations*, 2020. 1

[10] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 2

	Represented Distribution Shifts
ImageNet-Renditions	artistic renditions (cartoons, graffiti, embroidery, graphics, origami, paintings, sculptures, sketches, tattoos, toys, ...)
DeepFashion Remixed	occlusion, size, viewpoint, zoom
StreetView StoreFronts	camera, capture year, country

Table 8: Various distribution shifts represented in our three new benchmarks. ImageNet-Renditions is a new test set for ImageNet trained models measuring robustness to various object renditions. DeepFashion Remixed and StreetView StoreFronts each contain a training set and multiple test sets capturing a variety of distribution shifts.

	Training set	Testing images
ImageNet-R	1281167	30000
DFR	48000	42640, 7440, 28160, 10360, 480, 11040, 10520, 10640
SVSF	200000	10000, 10000, 10000, 8195, 9788

Table 9: Number of images in each training and test set. ImageNet-R training set refers to the ILSVRC 2012 training set [1]. DeepFashion Remixed test sets are: in-distribution, occlusion - none/slight, occlusion - heavy, size - small, size - large, viewpoint - frontal, viewpoint - not-worn, zoom-in - medium, zoom-in - large. StreetView StoreFronts test sets are: in-distribution, capture year - 2018, capture year - 2017, camera system - new, country - France.

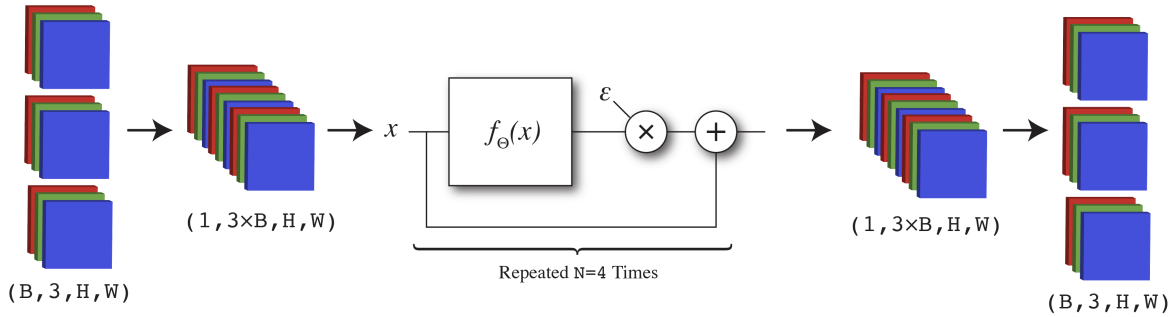

Figure 2: Parallel augmentation with Noise2Net. We collapse batches to the channel dimension to ensure that different transformations are applied to each image in the batch. Feeding images into the network in the standard way would result in the same augmentation being applied to each image, which is undesirable. The function $f_\Theta(x)$ is a Res2Net block with all convolutions replaced with grouped convolutions.

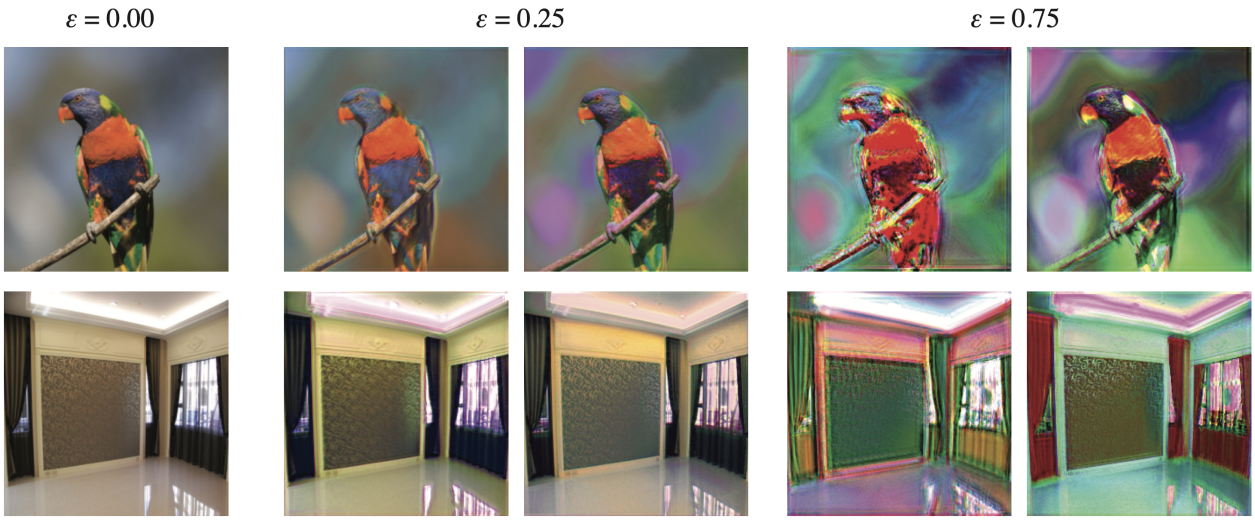

Figure 3: Example outputs of Noise2Net for different values of ε. Note $\varepsilon = 0$ is the original image.