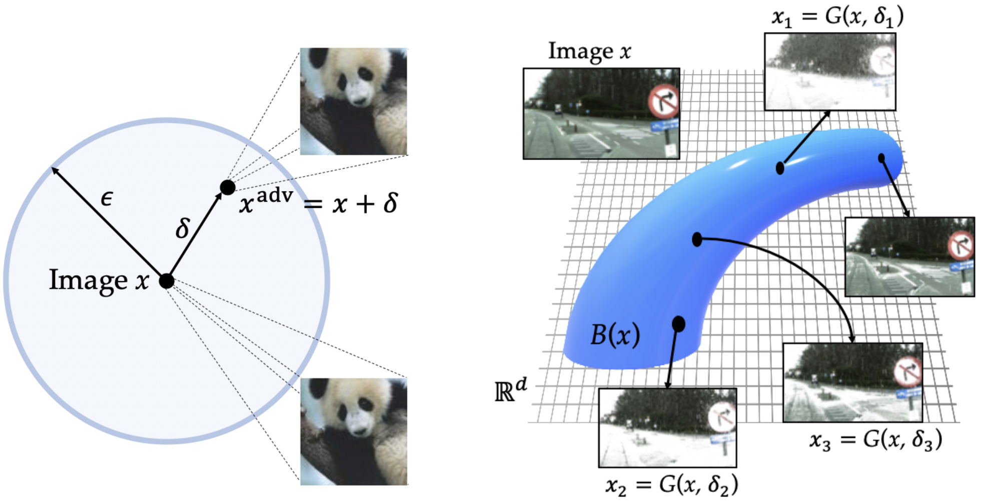
CONTENTS



(a) **Perturbation-based robustness.** In perturbation-based adversarial robustness, an adversary can perturb a datum x into a perceptually similar datum $x^{\text{adv}} := x + \delta$. When δ is constrained to lie in a set $\Delta := \{\delta \in \mathbb{R}^d : ||\delta||_p \leq \epsilon\}$, the underlying geometry of the problem can be used to find worst-case additive perturbations.

(b) **Model-based robustness.** When data can varying with respect to a nonlinear nuisance transformation such as the weather conditions in an image, defenses cannot easily exploit the linearity or geometry of the underlying problem. Indeed, there may be no analytic form for the transformation $G(x, \delta)$ for the transformation from sunny to snowy weather.

Figure 3: **Geometry of adversarial and model-based robustness.** When a form of natural variation in data can be described by a simple analytic expression, it is often possible to take advantage of this form to derive adversarial training algorithms. However, when data can vary according to nonlinear natural or physical phenomena, one must devise different schemes for providing robustness.

A ON THE QUALITY AND REPRESENTATIONAL POWER OF (LEARNED) MODELS OF NATURAL VARIATION

A.1 A GEOMETRIC INTERPRETATION OF MODELS OF NATURAL VARIATION

To provide geometric intuition for the model-based robust training formulation, consider Figure 3. The geometry of the classical perturbation-based adversarial training is captured in Figure 3a, wherein each datum x can be perturbed to any other datum x^{adv} contained in a small ϵ-neighborhood around x. That is, the data can be additively perturbed via $x \mapsto x^{\text{adv}} := x + \delta$ where δ is constrained to lie in a set $\Delta := \{\delta \in \mathbb{R}^d : ||\delta||_p \leq \epsilon\}$. On the other hand, Figure 3b shows the geometry of the model-based robust training paradigm. Let us consider a task in which our goal is to correctly classify images of street signs in varying weather conditions. In our model-based paradigm, we are equipped with a model $G(x, \delta)$ of natural variation that can naturally vary an image x by changing the nuisance parameter $\delta \in \Delta$. For example, if our data contains images x in sunny weather, the model $G(x, \delta)$ may be designed to continuously vary the weather conditions in the image without changing the scene or the street sign. More generally, such model-based variations around x have a manifold-like structure and belong to $B(x) := \{x' \in \mathbb{R}^d : x' = G(x, \delta) \text{ for some } \delta \in \Delta\}$. Note that in many models of natural variation, the dimension of model parameter $\delta \in \Delta$, and therefore the dimension of manifold $B(x)$, will be significantly lower than the dimension of data $x \in \mathbb{R}^d$. In other words, $B(x)$ will be comprised of submanifolds around x in the data space $\mathbb{R}^d$.

One subtle underlying assumption in the classical adversarial robustness formulation for classification tasks is that the additive perturbation $x + \delta$ must preserve the label y of the original datum x. For instance, in Figure 3, it is essential that the mapping $x \mapsto x + \delta$ where $||\delta||_p \leq \epsilon$ produces an example $x^{\text{adv}} = x + \delta$ which has the same label as x. Similarly, in this paper we restrict our attention to models $G(x, \delta)$ that preserve the semantic label of the input datum x for any $\delta \in \Delta$. In other words, we focus on models $G(x, \delta)$ that can naturally vary data x using nuisance parameter δ

(e.g. weather conditions, contrast, background color) while leaving the label of the original datum unchanged. In Figure 3b, this corresponds to all points $x' \in B(x)$ with varying snowy weather having the same label y as the original input datum x.

A.2 A STATISTICAL INTERPRETATION OF MODELS OF NATURAL VARIATION

Our approach toward formalizing the idea of learning $G(x, \delta)$ from data is to view G as a mechanism that transforms the distribution of data in the source domain A so that it resembles the distribution of data in the target domain B. More formally, let $\mathbb{P}_A$ and $\mathbb{P}_B$ be the data distributions corresponding to domains A and B respectively. Our objective is to find a mapping G that takes as input a datum $x \sim \mathbb{P}_A$ and a nuisance parameter $\delta \in \Delta$ and then produces a new datum $x' \sim \mathbb{P}_B$. Statistically speaking, the nuisance parameter δ represents the extra randomness or variation required to generate x' from x. For example, when considering images with varying weather conditions, the randomness in the nuisance might control whether an image of a sunny scene is mapped to a corresponding image with a dusting of snow or to an image in an all-out blizzard. In this way, we without loss of generality we assume that the nuisance parameter is independently generated from a simple distribution $\mathbb{P}_\Delta$ (e.g. uniform or Gaussian) to represent the extra randomness required to generate x' from x.[1] Using this formalism, we can view $G(\cdot, \cdot)$ as a mapping that transforms the distribution $\mathbb{P}_A \times \mathbb{P}_\Delta$ into the distribution $\mathbb{P}_B$. More specifically, G pushes forward the measure $\mathbb{P}_A \times \mathbb{P}_\Delta$, which is defined over $A \times \Delta$, to $\mathbb{P}_B$, which is defined over B. That is, $\mathbb{P}_B = G \,\#\, (\mathbb{P}_A \times \mathbb{P}_\Delta)$, where $\#$ denotes the push-forward measure. Now in order to learn a model of natural variation G, we consider a parametric family of models $\mathcal{G} := \{G_\theta : \theta \in \Theta\}$ defined over a parameter space $\Theta \subset \mathbb{R}^m$. We can express the problem of learning a model of natural variation G_{θ^*} parameterized by $\theta^* \in \Theta$ that best fits the above formalism in the following way:

$$\theta^* = \operatorname*{arg\,min}_{\theta \in \Theta} d\left(\mathbb{P}_B, G_\theta \,\#\, (\mathbb{P}_A \times \mathbb{P}_\Delta)\right). \tag{4}$$

Here $d(\cdot, \cdot)$ is an appropriately-chosen distance metric that measures the distance between two probability distributions (e.g. the KL-divergence, total variation, etc.). This formulation has received broad interest in the machine learning community thanks to the recent advances in generative modeling. In particular, in the fields of image-to-image translation and style-transfer, learning mappings between unpaired image domains is a well-studied problem (Huang et al., 2018; Zhu et al., 2017b; Yi et al., 2017). In the next section, we will show how the breakthroughs in these fields can be used to learn a model of natural variation G that approximates underlying natural phenomena.

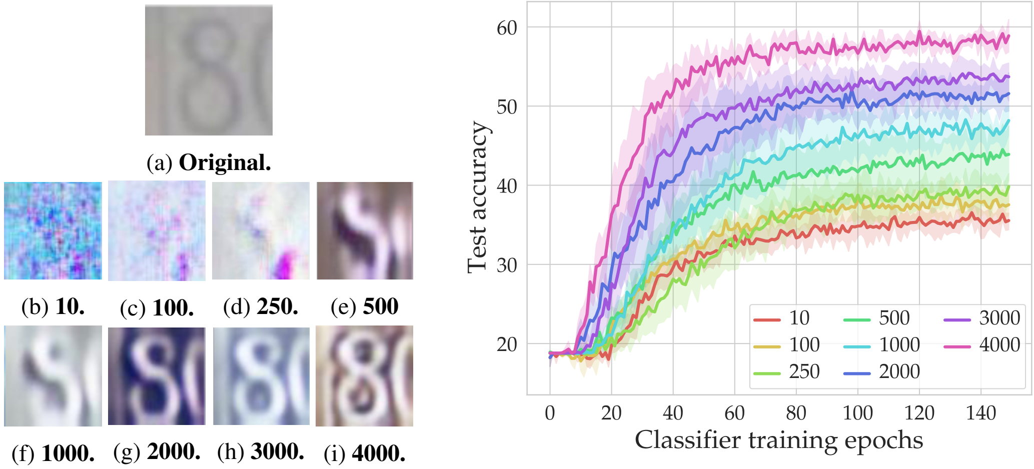
A.3 ON THE CHOICE OF THE MUNIT ARCHITECTURE IN THIS PAPER

Importantly, a number of methods have been designed toward achieving the goal detailed in Section A.2. In the fields of *unsupervised, non-conditional* image-to-image translation, such methods include CycleGAN (Zhu et al., 2017a), DualGAN (Yi et al., 2017), Augmented CycleGAN (Almahairi et al., 2018), BicycleGAN (Zhu et al., 2017b), CSVAE (Klys et al., 2018), UNIT (Liu et al., 2017), MUNIT (Huang et al., 2018), DRIT (Lee et al., 2018), MSGAN (Mao et al., 2019), StarGAN (Choi et al., 2018), StarGAN v2 (Choi et al., 2020). Among these methods, CSVAE, BicycleGAN, Augmented CycleGAN, and MUNIT, DRIT, MSGAN, and StarGAN seek to learn *multimodal* mappings that disentangle the *semantic content* of a datum (e.g. its label and characterizing features) from the *nuisance content* (e.g. background color, weather conditions) by solving the statistical problem of equation 4. We highlight these methods because learning a multimodal mapping is a concomitant property toward learning models that can produce images with *varying* nuisance content.

While performing the experiments for this paper, we tested three different architectures for G in our MRDEL framework: CSVAE, MUNIT, and StarGAN v2. However, we could not get CSVAE to produce high-quality images corresponding to different shifts in natural variation. Similarly, the performance of StarGAN was also poor; StarGAN v2 suffered mode collapse on nearly every dataset we ran it on. On the other hand, we found that the MUNIT framework consistently produced realistic, naturally-varying images, which motivated our decision to exclusively use the MUNIT framework in our final experiments.

---

[1] The role of the nuisance parameter is similar to the role of the noise variable in generative adversarial networks (Goodfellow et al., 2014a).



(a) **Original.**

(b) **10.** (c) **100.** (d) **250.** (e) **500**

(f) **1000.** (g) **2000.** (h) **3000.** (i) **4000.**

Output images from models in $\mathcal{G}$. We show an example image from domain A in (a), and subsequently show the corresponding output images for each $G \in \mathcal{G}$ for a randomly chosen $\delta \in \Delta$ in (b)-(i).

(j) **MRT using models from $\mathcal{G}$.** For each model in $\mathcal{G}$, we run MRT for five trials and show the resulting test accuracy on samples from the test set from Domain B. Note that the robustness of the trained classifier increases as the number of training steps used to train the model increases.

Figure 4: **A better model implies more robustness.** By learning a family of models $\mathcal{G}$ that are trained for different numbers of steps, we show empirically that models that can more accurately model distribution shifts engender classifiers with higher levels of robustness.

However, we do not claim that MUNIT is the optimal framework for model-based robust deep learning. Indeed, in future work we plan to compare the efficacy of using different architectures for G. To this end, another interesting direction for future work is to leverage class-conditional information in G. That is, by extending the definition of a model of natural variation to $G : \mathbb{R}^d \times \mathbb{R}^{[k]} \times \mathbb{R}^q \to \mathbb{R}^d$ so that $G(x, y, \delta)$ maps a datum, its label, and a nuisance parameter to a new image, we could leverage the literature surrounding *conditional* image-to-image translation, including BasisGAN (Wang et al., 2019) and MAD-GAN (Li et al., 2019).

A.4 ON THE CHOICE TO LEARN $G(x, \delta)$ OFFLINE

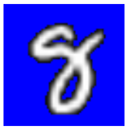
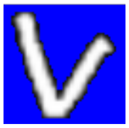
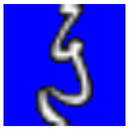
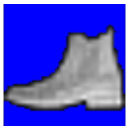
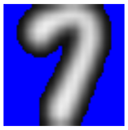
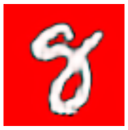
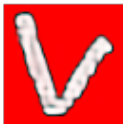
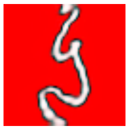
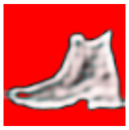
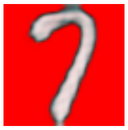
In this paper, the paradigm that we developed assumes that a model G is learned offline before training a classifier. Indeed, in our experiments, we show that by decoupling the process of training the model of natural variation and the classifier, models of natural variation can be reused on new datasets and can be composed to provide robustness against multiple sources of natural variation. However, this decision to learn G offline is not essential to the success of our paradigm. Indeed, an interesting direction for future work is design an algorithm that learns a suitable model of natural variation and a classifier simultaneously. Similar ideas have been used frequently in the domain adaptation literature (Tzeng et al., 2017; Wang & Deng, 2018).

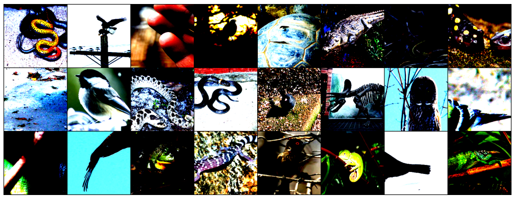
A.5 QUANTIFYING THE ABILITY OF LEARNED MODELS OF NATURAL VARIATION TO GENERATE REALISTIC IMAGES

An essential yet so far undiscussed piece of the efficacy of the model-based paradigm is the impact of the model quality on the robustness we are ultimately able to provide. In scenarios where we don't have access to a known model, the ability to provide any sort of meaningful robustness relies on learned models that can accurately render realistic looking data with varying nuisances. To this end, it is reasonable to expect that models that can more effectively render realistic yet challenging data should result in classifiers that are more robust to shifts in natural variation.

To examine the impact of models in our paradigm, we consider the task of Section 5.5 on SVHN, in which we learned a model that mapped low-contrast samples, which comprised domain A, to high-contrast samples, which comprised domain B. While learning this model, we saved snapshots

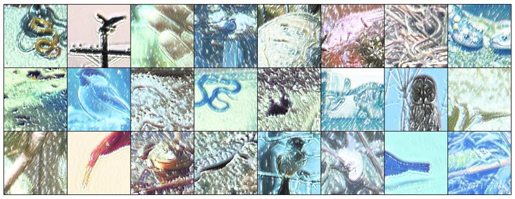
Table 6: **Passing samples from other datasets through a model learned on MNIST.** The first row of images in this table are samples taken from colorized versions of Q-MNIST, E-MNIST, K-MNIST, Fashion-MNIST, and USPS. The second row of images shows samples passed through a model trained on the original MNIST dataset to change the background color from blue to red.

	QMNIST	EMNIST	KMNIST	Fashion-MNIST	USPS
Domain A					
Domain B					

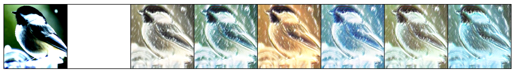


(a) Domain A images.

(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.



(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 5: **ImageNet-c brightness (no→yes).** Used in: Table 3, row 3; Table 2, row 1.

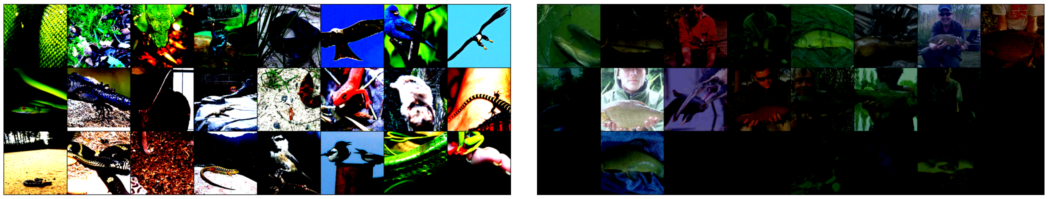
of the model at various points during the training procedure. In particular, we collected a family of intermediate models

$$\mathcal{G} = \{G_{10}, G_{100}, G_{250}, G_{500}, G_{1000}, G_{2000}, G_{3000}, G_{4000}\},$$

where the index denotes the MUNIT iteration number. In Figure 4j, we show the result of training classifiers with MRT using each model $G \in \mathcal{G}$. Note that the models that are trained for more training steps engender classifiers that provide higher levels of robustness against the shift in nuisance variation. In other words, better models provide improved test accuracy for classifiers using model-based robust training.

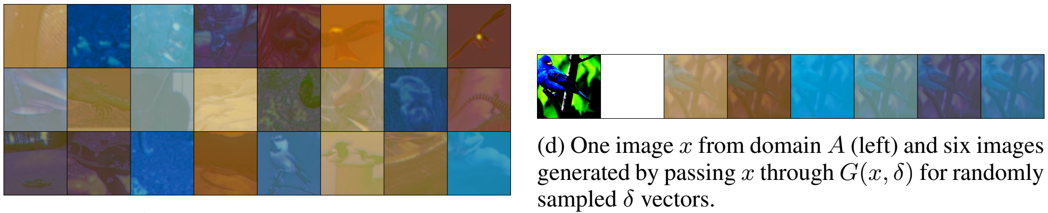
A.6 A GALLERY OF MODELS OF NATURAL VARIATION

We conclude this section by showing images corresponding to the many distributional shifts used in the experiments section. Furthermore, we show images generated by passing domain A images through learned models of natural variation.



(a) Domain A images.

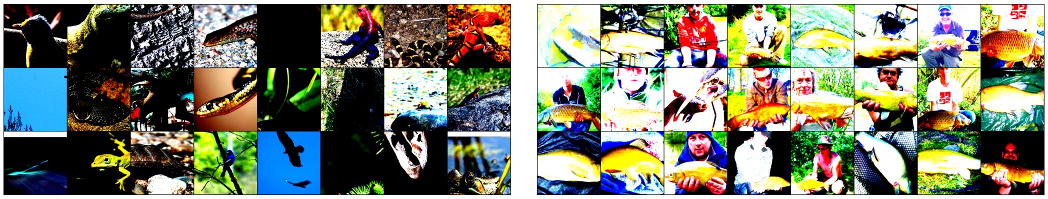
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

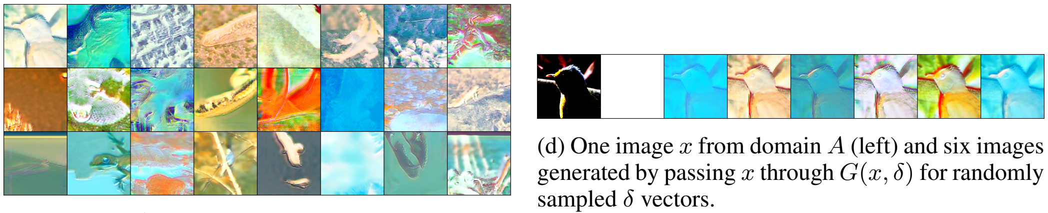
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 6: **ImageNet-c contrast (no→yes).** Used in: Table 3, rows 2, 5; Table 2, row 2.



(a) Domain A images.

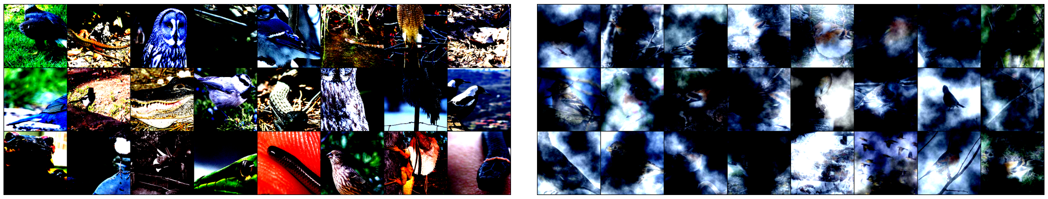
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

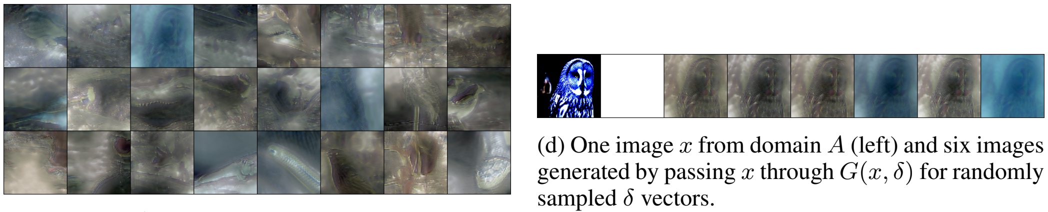
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 7: **ImageNet-c brightness (low→high).** Used in: Table 3, rows 2-4; Table 2, row 3.



(a) Domain A images.

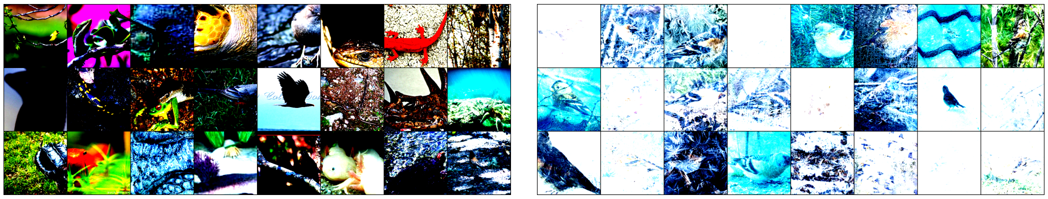
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

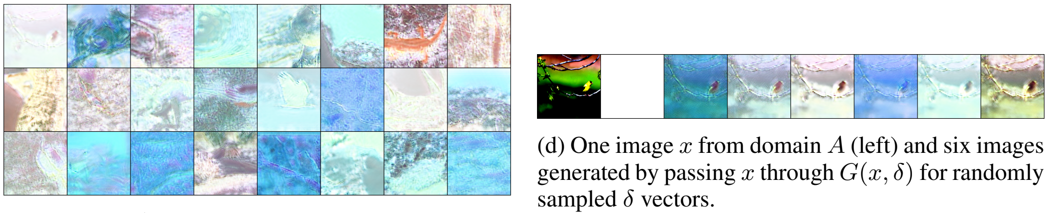
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 8: **ImageNet-c fog (no→yes).** Used in: Table 3, rows 4-5; Table 2, row 4.



(a) Domain A images.

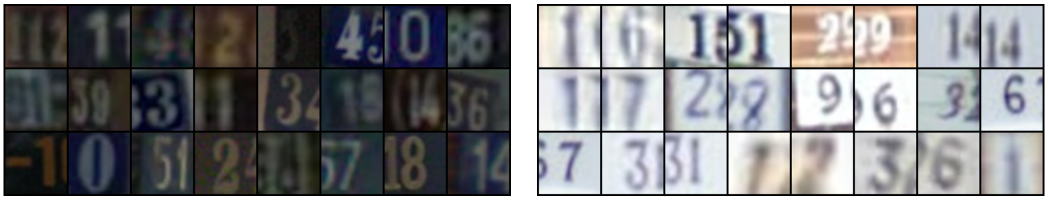
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

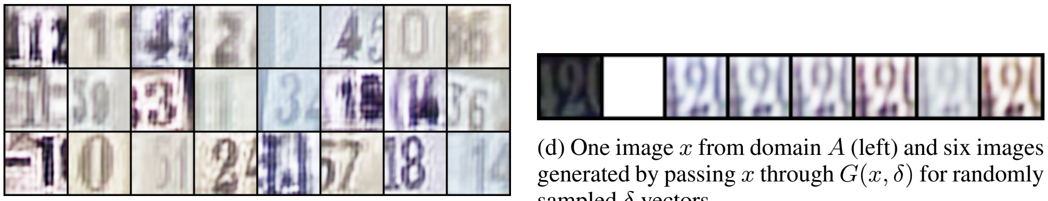
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 9: **ImageNet-c frost (no→yes).** Used in: Table 2, row 5.



(a) Domain A images.

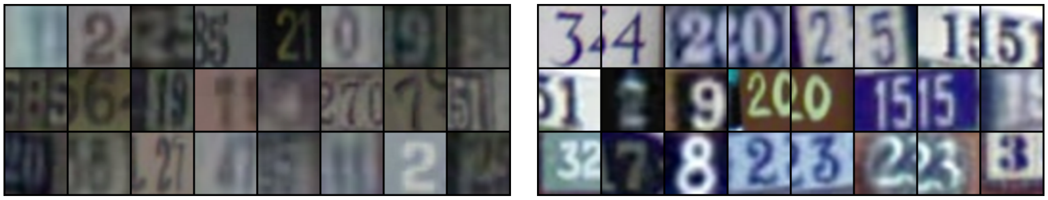
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

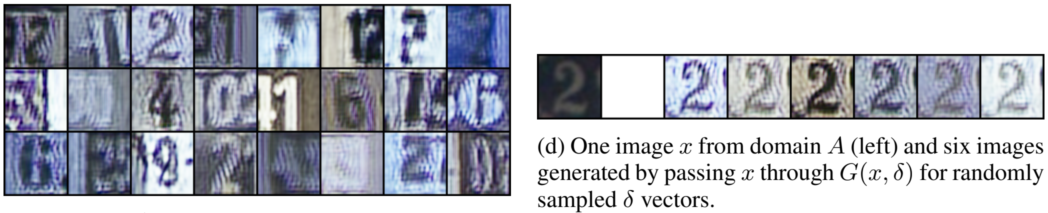
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 10: **SVHN brightness (low→high).** Used in: Table 3, row 1; Table 5, row 1.



(a) Domain A images.

(b) Domain B images.



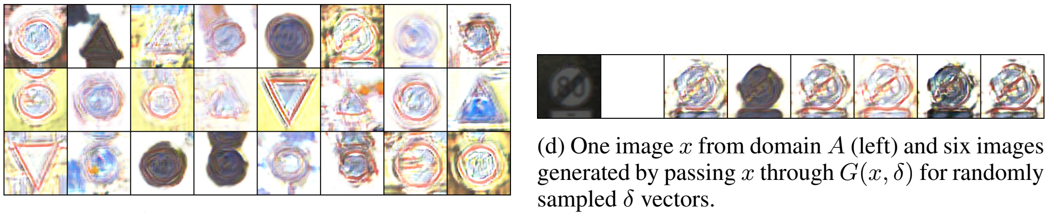
(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 11: **SVHN contrast (low→right).** Used in: Table 3, row 1; Table 5, row 2.

(a) Domain A images.

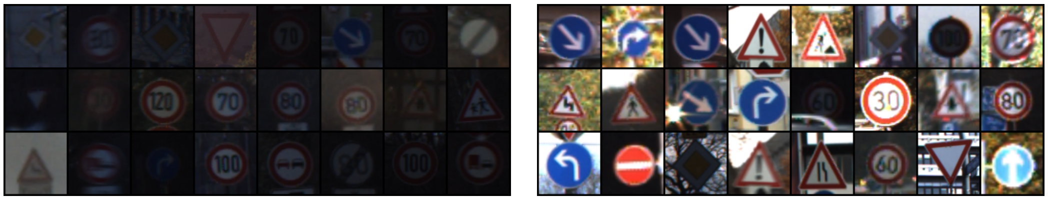
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

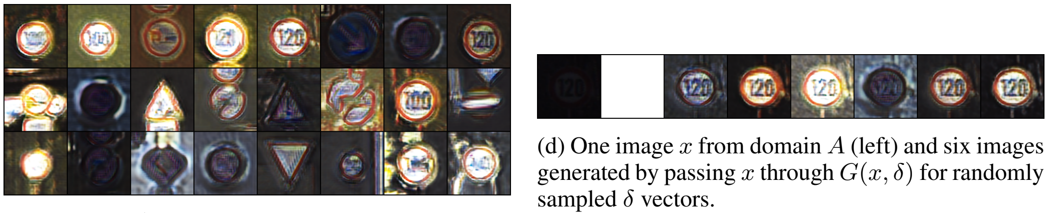
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 12: **GTSRB brightness (low→high).** Used in Table 5, row 3.



(a) Domain A images.

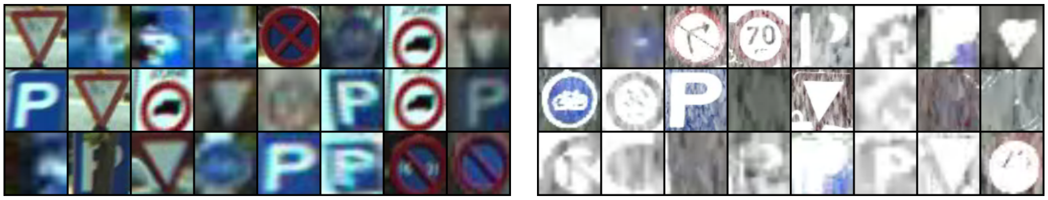
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

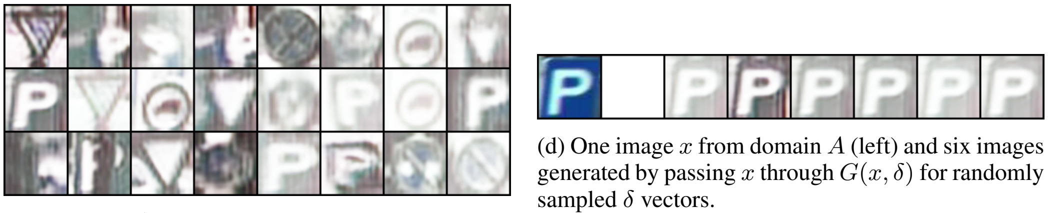
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 13: **GTSRB contrast (low→high).** Used in: Table 5, row 4.



(a) Domain A images.

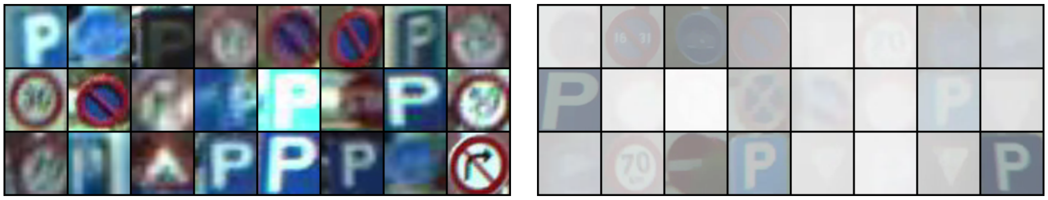
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

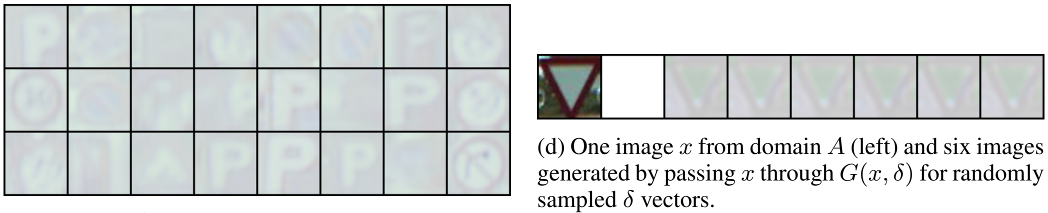
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 14: **CURE-TSR snow (no→yes).** Used in: Table 3, row 7; Table 5, row 5.



(a) Domain A images.

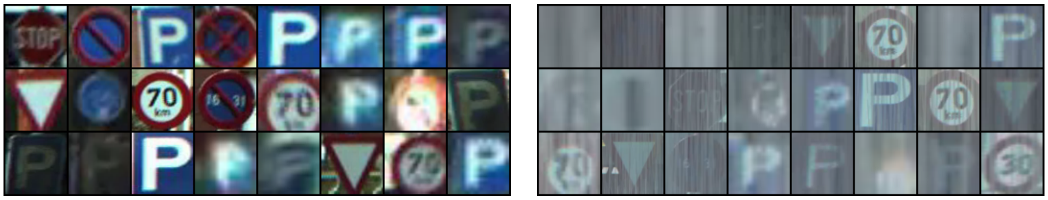
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

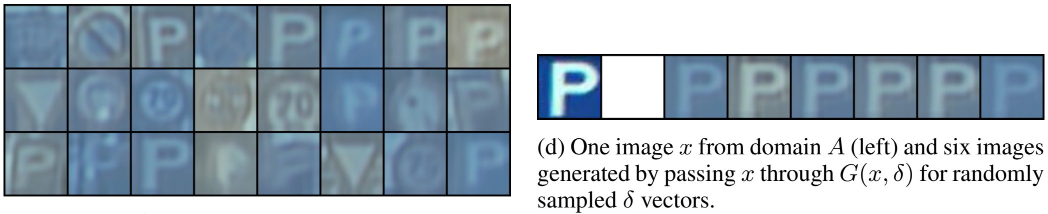
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 15: **CURE-TSR haze (no→yes).** Used in: Table 1, row 2; Table 5, row 6.



(a) Domain A images.

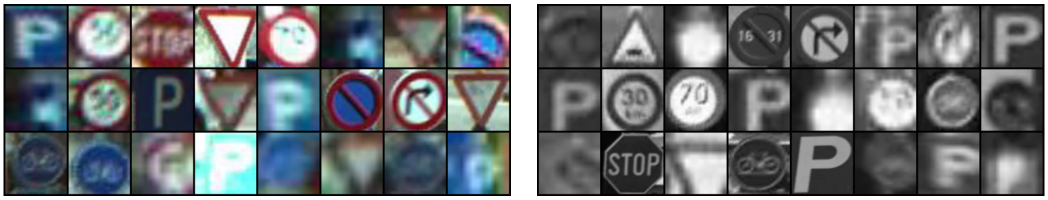
(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

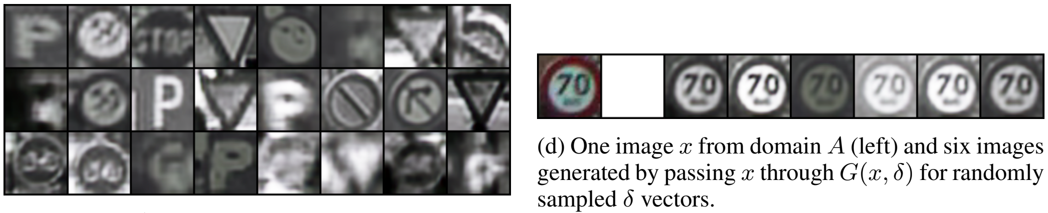
(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 16: **CURE-TSR rain (no→yes).** Used in: Table 5, row 7.



(a) Domain A images.

(b) Domain B images.



(c) Domain A images from (a) passed through a learned model of natural variation $G(x, \delta)$.

(d) One image x from domain A (left) and six images generated by passing x through $G(x, \delta)$ for randomly sampled δ vectors.

Figure 17: **CURE-TSR decolorization (no→yes).** Used in: Table 1, row 3.

B FURTHER DISCUSSION OF MAT, MRT, AND MDA

B.1 FURTHER DESCRIPTIONS OF MAT, MRT, AND MDA

To supplement Section 4, we provide pseudocode for MAT and MDA in Algorithms 2 and 3. In the remainder of this section, we discuss each of the algorithms in further detail.

Model-based Adversarial Training (MAT). At first look, the sampling-based approach used by MRT may not seem as powerful as a first-order (i.e. gradient-based) approach, which has been shown to be effective at improving perturbation-based robustness (Athalye et al., 2018). Indeed, under the assumption that our model $G(x, \delta)$ is differentiable, it is natural to extend these first order methods to our model-based paradigm. In particular, by replacing lines 3-11 in Algorithm 2 with a subroutine that performs k rounds of projected gradient ascent on the empirical batched loss $\sum_{j=1}^{m} \ell(G(x^{(j)}, \delta), y^{(j)}; w)$ with respect to $\delta \in \Delta$, we seek to more accurately solve the inner maximization problem to find a loss-maximizing nuisance parameter δ^{adv}. We then augment $\mathcal{D}_n$ with instance-labels pairs $(G(x^{(j)}, \delta^{\text{adv}}), y^{(j)})$ and solve the outer problem in the same way as in MRT. To emphasize the role of the number of gradient steps k used to find $\delta^{\text{adv}} \in \Delta$, we will often refer to this algorithm as MAT-k.

Model-based Robust Training (MRT). The idea behind the MRT algorithm is to sample k different nuisance parameters $\delta \in \Delta$ for each instance-label pair $(x^{(j)}, y^{(j)})$ and among those sampled values, find the nuisance parameter δ^{adv} that gives the highest empirical loss for $G(x^{(j)}, \delta^{\text{adv}})$ under ℓ. Indeed, this approach is not designed to find an exact solution to the inner maximization problem; rather it aims to find a difficult example by sampling in the nuisance space of the model. Once we approximately solve the inner problem by obtaining δ^{adv}, this nuisance parameter is used to perform data augmentation. That is, we treat $(G(x^{(j)}, \delta^{\text{adv}}), y^{(j)})$ as a new instance-label pair that we use to supplement $\mathcal{D}_n$. These training data can be used together with first-order optimization methods (e.g. SGD, Adam (Kingma & Ba, 2014), Adadelta (Zeiler, 2012), etc.) to solve the outer minimization problem to a locally optimal solution w^*. Algorithm 2 shows the pseudocode for this approach. To make clear the role of the number k of nuisance parameters sampled per instance, we will often refer to Algorithm 2 as MRT-k.

Model-based Data Augmentation (MDA). Both MRT and MAT adhere to the philosophy of augmenting $\mathcal{D}_n$ with loss-maximizing, model-generated data. One alternative to this adversarial approach is to expose classifiers to a *diversity* of model-generated data during training. More specifically, in MDA we perform data augmentation by choosing k randomly sampled nuisance parameters from Δ and creating new instance-label pairs $(G(x^{(j)}, \delta), y^{(j)})$ with each of these k values of δ. Thus each classifier will be exposed to k copies of the same image, where each copy has a different level of the same source of natural variation. To make this explicit, we will refer to this algorithm as MDA-k.

B.2 SAMPLING VS. ADVERSARIAL MINDSET

From an optimization perspective, we can group our model-based algorithms into two categories: sampling (zeroth-order) methods and adversarial (first-order) methods. Sampling-based methods refer to those that seek to solve the inner maximization term in (2) by querying the model. This is particularly important for models that are not differentiable. Both MRT and MDA are sampling (zeroth-order) methods in that we obtain new data by sampling different nuisance parameters $\delta \in \Delta$ for each batch in the training set. On the other hand, the technique used to obtain new data in the MAT algorithm is an adversarial (first-order) method, as we statistically approximate the gradient of the model $\nabla_\delta G(x, \delta)$ to perform the optimization–i.e. search for the worst-case nuisance parameter. If the model G is differentiable (which is not required in our framework), then one can directly compute the gradient of the model $\nabla_\delta G(x, \delta)$.

Throughout the experiments, in general we see that the sampling algorithms presented in this paper achieve higher levels of robustness against almost all sources of natural variation. This finding stands in contrast to field of perturbation-based robustness, in which adversarial methods have been shown to be the most effective in improving the robustness against small, norm-bounded perturbations (Athalye et al., 2018). Going forward, an interesting research direction is not only to consider new

Algorithm 2 Model-based Robust Training (MRT)

```
Input: data sample $\mathcal{D}_n = \{(x^{(j)}, y^{(j)})\}_{j=1}^n$, number of steps $k$
Output: learned weight $w$
repeat
    for minibatch $B_m := \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\} \subset \mathcal{D}_n$ do
        Initialize $max\_loss \leftarrow 0$ and $\delta^{\text{adv}} := (\delta_1^{\text{adv}}, \delta_2^{\text{adv}}, \dots, \delta_m^{\text{adv}}) \leftarrow (0_q, 0_q, \dots, 0_q)$
        for $k$ steps do
            Sample $\delta_j$ randomly from $\Delta$ for $j = 1, \dots, m$
            $current\_loss \leftarrow \sum_{j=1}^m \ell(G(x_j, \delta_j), y_j; w)$
            if $current\_loss > max\_loss$ then
                $max\_loss \leftarrow current\_loss$
                $\delta_j^{\text{adv}} \leftarrow \delta_j$ for $j = 1, \dots, m$
            end if
        end for
        $g \leftarrow \nabla_w \sum_{j=1}^m [\ell(G(x_j, \delta_j^{\text{adv}}), y_j; w) + \lambda \cdot \ell(x_j, y_j; w)]$
        $w \leftarrow \text{Update}(g, w)$    # Update function could be SGD, Adam, Adadelta, etc.
    end for
until convergence
```

Algorithm 3 Model-Based Data Augmentation (MDA)

```
Input: data sample $\mathcal{D}_n = \{(x_j, y_j)\}_{j=1}^n$, number of steps $k$
Output: learned weight $w$
repeat
    for minibatch $B_m := \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\} \subset \mathcal{D}_n$ do
        Initialize $x_i^{(j)} \leftarrow 0_d$ for $i = 1, \dots, k$ and for $j = 1, \dots, m$
        for $k$ steps do
            Sample $\delta_j$ randomly from $\Delta$ for $j = 1, \dots, m$
            $x_i^{(j)} \leftarrow G(x_j, \delta_j)$ for $j = 1, \dots, m$
        end for
        $g \leftarrow \nabla_w \sum_{j=1}^{m} \left[ \sum_{i=1}^{k} \ell\left(x_i^{(j)}, y_j; w\right) + \lambda \cdot \ell(x_j, y_j; w) \right]$
        $w \leftarrow \text{Update}(g, w)$    # Update function could be SGD, Adam, Adadelta, etc.
    end for
until convergence
```

algorithms but also to understand whether sampling-based or adversarial techniques provide more robustness with respect to a given model.

Table 7: **Impact of varying k in the model-based algorithms.** We study the impact of varying k for each of the model-based algorithms on the brightness (low→high) shift on SVHN. Throughout, we use a trade-off parameter of $\lambda = 1$.

k	Test accuracy (top-1)		
	MAT-k	MRT-k	MDA-k
1	81.9	81.2	81.7
5	75.6	81.7	81.1
10	75.7	82.6	81.1
20	79.9	83.2	80.3
50	76.3	82.5	79.7

B.3 VARYING k IN THE MODEL-BASED ALGORITHMS

As we show in our experiments, each of the three model-based algorithms can be used to provide significant out-of-distribution robustness against various sources of natural variation. In this subsection, we focus on the impact of varying the parameter k in each of these algorithms. In particular, in Table 7, we see that varying k has a different impact for each of the three algorithms. For MAT, we see that increasing k decreases the accuracy of the trained classifier; one interpretation of this phenomenon is that larger values of k allow MAT to find more challenging forms of natural variation. On the other hand, the test accuracy of MRT improves slightly as k increases. Recall that while both MRT and MAT seek to find "worst-case" natural variation, MRT employs a sampling-based approach to solving the inner maximization problem as opposed to the more precise, gradient-based procedure used by MAT. Thus the differences in the impact of varying k between MAT and MRT may be due to the fact that MRT only approximately solves the inner problem at each iteration. Finally, we see that increasing k slightly decreases the test accuracy of classifiers trained with MDA.

This study can also be used as an algorithm selection criteria. Indeed, when data presents many modes corresponding to different levels of natural variation, it may be more efficacious to use MRT or MDA, which will observe a more diverse set of natural conditions due to their sampling-based approaches. On the other hand, when facing a single challenging source of natural variation, it may be more useful to use MAT, which seeks to find "worst-case," natural, out-of-distribution data.

C TRAINING DETAILS

C.1 SELECTING THE DIMENSION OF THE NUISANCE PARAMETER SPACE Δ

Recall that in the inner maximization problem of (2), we optimize over the space Δ of so-called nuisance parameters. For convenience, this inner maximization problem is reproduced below:

$$\delta^\star \in \arg\max_{\delta \in \Delta} \ell(G(x, \delta), y; w) \tag{5}$$

Given a fixed instance $x \in \mathbb{R}^d$, Δ characterizes the set of instances that can be obtained under the mapping of a model of natural variation $G(x, \delta)$. Therefore, the dimension of Δ, henceforth denoted as $\dim(\Delta)$, should be small enough so that Δ can be efficiently optimized over and large enough so that it can accurately capture the underlying source of natural variation that the model $G(x, \delta)$ seeks to describe. In this sense, $\dim(\Delta)$ should reflect the complexity of both the source of natural variation and indeed of the data itself.

To make this precise, for fixed x, we first define the *learned image manifold*

$$B(x) := \{x' \in \mathbb{R}^d | x' = G(x, \delta) \text{ for some } \delta \in \Delta\}$$

which, more formally, is a paramterized $\dim(\Delta)$-manifold sitting in $\mathbb{R}^d$. Using this notation, we can rewrite (5) in the following way:

$$\delta^\star \in \arg\max_{x' \in B(x)} \ell(x', y; w) \tag{6}$$

This representation of the inner maximization problem elucidates the fact that Δ must be rich enough to be able to produce representative images on the learned image manifold. However, in the extreme case when $\dim(\Delta) = d$, as is the case in much of the adversarial robustness literature, it is well known that δ is difficult to efficiently optimize over (Madry et al., 2017).

To this end, throughout our experiments, we generally scale $\dim(\Delta)$ with d. For low-dimensional data (e.g. MNIST, SVHN, etc.), we found that $\dim(\Delta) = 2$ sufficed toward capturing the underlying source of natural variation effectively. However, we found that on datasets such as GTSRB, for which we rescaled instances into $64 \times 64 \times 3$ arrays, we found that $\dim(\Delta) = 8$ was more appropriate for capturing the full range of natural variation. Indeed, on ImageNet, which contains instances of size $224 \times 224 \times 3$, we found that $\dim(\Delta) = 8$ still produced images that captured the essence of the underlying source of natural variation.

C.2 SELECTING THE RADIUS OF THE NUISANCE PARAMETER SPACE Δ

Given the preceeding discussion concerning the dimension of the nuisance space Δ, a concomitant question is how to pick the radius of the set Δ. In every experiment described in this paper, we let $\Delta := \{x \in \mathbb{R}^q | -1 \preceq x \preceq 1\}$ where $q = \dim(\Delta)$. This choice is not fundamental, and indeed we plan to explore varying this radius in future work.

C.3 CLASSIFIER ARCHITECTURE AND HYPERPARAMETER SELECTION

Here we use the following conventions for describing architectures. `c32-3` refers to a $2D$ convolutional operator with 32 kernels, each of which has shape 3×3. `p2` refers to a max-pooling layer with kernel size 2. `d0.25` refers to a dropout layer, which drops an activation with probability 0.25. `flat` refers to a flattening layer. `fc-128` refers to a fully-connected layer mapping into $\mathbb{R}^{128}$.

When training classifiers for MNIST, Q-MNIST, E-MNIST, K-MNIST, Fashion-MNIST, USPS, SVHN, GTSRB, and CURE-TSR we use a simple CNN architecture with two convolutional layers and two feed forward layers. More specifically, the architecture can be described in the following way:

```
c32-3, c64-3, p2, c128-3, p2, d0.25, flat, fc128, d0.5, fc10.
```

For each of these experiments, we use the Adadelta (Zeiler, 2012) optimizer with a learning rate of 1.0. We also use a batch size of 64. Images from MNIST, Q-MNIST, E-MNIST, K-MNIST, Fashion-MNIST, USPS, SVHN, and CURE-TSR are resized to $32 \times 32 \times 3$; for grayscale datasets such as MNIST, we repeat the channels three times. Images from GTSRB are resized to $64 \times 64 \times 3$. We train each classifier for 100 epochs.

When training on ImageNet, we use the ResNet-50 (He et al., 2016) architecture. We note that architectural choices are possible and will be explored in future work. For each of the experiments on ImageNet, we use SGD with an initial learning rate of 0.05; we decay the learning rate linearly to 0.001 over 100 epochs. We use a batch size of 64.

When training with PGD (Madry et al., 2017), we use a step size of 0.01, we set $\epsilon = 8/255$, and we allow the adversary to take 10 steps per iteration. When training with the model-based algorithms, we use fixed choices for k throughout the experiments. For MDA, we use $k = 1$; for MRT, we use $k = 10$; for MAT, we use $k = 10$. We discuss the impact of k in Appendix B. We use a trade-off parameter of $\lambda = 1$ throughout.

C.4 MUNIT FRAMEWORK OVERVIEW

For completeness, we give a brief overview of the MUNIT framework (Huang et al., 2018) and described the architecture we used for MUNIT in this paper.

To begin, let $x_A \in A$ and $x_B \in B$ be images from two unpaired image domains A and B; in the notation of the previous section, we assume that these images are drawn from two marginal distributions $\mathbb{P}_A$ and $\mathbb{P}_B$. Further, the MUNIT model assumes that each image from either domain can be decomposed into two components: a style code s that contains information about factors of natural or nuisance variation, and a content code c that contains information about higher level features such as the label of the image. Further, it is assumed that the content codes for images in either domain are drawn from a common set $\mathcal{C}$, but that the style codes are drawn from domain specific sets $\mathcal{S}_A$ and $\mathcal{S}_B$. In this way, a pair of corresponding images (x_A, x_B) are of the form $x_A = \text{Dec}_A(c, s_A)$ and $x_B = \text{Dec}_B(c, s_B)$, where $c \in \mathcal{C}$, $s_A \in \mathcal{S}_A$, $s_B \in \mathcal{S}_B$, and where Dec_A and Dec_B are unknown decoding networks corresponding to domains A and B respectively. The authors of (Huang et al., 2018) call this setting a partially shared latent space assumption.

The MUNIT model consists of an encoder-decoder pair $(\text{Enc}_A, \text{Dec}_A)$ and $(\text{Enc}_B, \text{Dec}_B)$ for each image domain A and B. These encoder-decoder pairs are trained to learn a mapping that reconstructs its input. That is, $x_A \approx \text{Dec}_A(\text{Enc}_A(x_A))$ and $x_B \approx \text{Dec}_B(\text{Enc}_B(x_B))$. More specifically, $\text{Enc}_A : \mathcal{A} \to \mathcal{C} \times \mathcal{S}_A$ is trained to encode x_A into a content code $c \in \mathcal{C}$ and a style code $s_A \in \mathcal{S}_A$. Similarly, $\text{Enc}_B : \mathcal{B} \to \mathcal{C} \times \mathcal{S}_B$ is trained to encode x_B into $c \in \mathcal{C}$ and $s_B \in \mathcal{S}_B$. Then the decoding networks $\text{Dec}_A : \mathcal{C} \times \mathcal{S}_A \to A$ and $\text{Dec}_B : \mathcal{C} \times \mathcal{S}_B \to B$ are trained to reconstruct the encoded pairs (c, s_A) and (c, s_B) into the respective images x_A and x_B.

Inter-domain image translation is performed by swapping the decoders. In this way, to map an image x_A from A to B, x_A is first encoded into $\text{Enc}_A(x_A) = (c, s_A)$. Then, a new style vector s_B is sampled from $\mathcal{S}_B$ from a prior distribution π_B on the set $\mathcal{S}_B$ and the translated image $x_{A \to B}$ is equal to $\text{Dec}_B(c, s_B)$. The translation of x_B from B to A can be described via a similar procedure with Enc_B, Dec_A, and a prior π_A supported on $\mathcal{S}_A$. In this paper, we follow the convention used in (Huang et al., 2018) as use a Gaussian distribution for both π_A and π_B with zero mean and an identity covariance matrix.

Training an MUNIT model involves considering four loss terms. First, the encoder-decoder pairs $(\text{Enc}_A, \text{Dec}_A)$ and $(\text{Enc}_B, \text{Dec}_B)$ are trained to reconstruct their inputs my minimizing the following loss:

$$\ell_{\text{recon}} = \mathbb{E}_{x_A \sim \mathbb{P}_A} \left|\left|\text{Dec}_A(\text{Enc}_A(x_A)) - x_A\right|\right|_1 + \mathbb{E}_{x_B \sim P_B} \left|\left|\text{Dec}_B(\text{Enc}_B(x_B)) - x_B\right|\right|_1$$

Further, when translating an image from one domain to another, the authors of (Huang et al., 2018) argue that we should be able to reconstruct the style and content codes. By rewriting the encoding networks as $\text{Enc}_A(x_A) = (\text{Enc}^c_A(x_A), \text{Enc}^s_A(x_A))$ and $\text{Enc}_B(x_B) = (\text{Enc}^c_B(x_B), \text{Enc}^s_B(x_B))$, the constraint on the content codes can be expressed in the following way:

$$\ell^c_{\text{recon}} = \mathbb{E}_{\substack{c_A \sim \mathbb{P}(c_A) \\ s_B \sim \pi_B}} \left|\left|\text{Enc}^c_B(\text{Dec}_B(c_A, s_B)) - c_A\right|\right|_1 + \mathbb{E}_{\substack{c_B \sim \mathbb{P}(c_B) \\ s_A \sim \pi_A}} \left|\left|\text{Enc}^c_A(\text{Dec}_A(c_B, s_A)) - c_B\right|\right|_1$$

where $\mathbb{P}(c_A)$ is the distribution given by $c_A = \text{Enc}^c_A(x_A)$ where $x_A \sim \mathbb{P}_A$ and $\mathbb{P}(c_B)$ is the distribution given by $c_B = \text{Enc}^c_B(x_B)$ where $x_B \sim \mathbb{P}_B$. Similar, the constraint on the style codes can be

Table 8: **MUNIT hyperparameters.**

Name	**Value**
Number of iterations	10000
Batch size	1
Weight decay	0.0001
Weight initialization	Kaiming
Learning rate	0.0001
Learning rate policy	Step
γ (learning rate decay amount)	0.5
λ_x	10
λ_c	1
λ_s	1

written as

$$\ell^s_{\text{recon}} = \mathbb{E}_{\substack{c_A \sim \mathbb{P}(c_A) \\ s_B \sim \pi_B}} ||\text{Enc}^s_B(\text{Dec}_B(c_A, s_B)) - s_B||_1 + \mathbb{E}_{\substack{c_B \sim \mathbb{P}(c_B) \\ s_A \sim \pi_A}} ||\text{Enc}^s_A(\text{Dec}_A(c_B, s_A)) - s_A||_1 .$$

Finally, two GANs corresponding to the two domains A and B are used to form an adversarial loss term. The GANs use the decoders Dec_A and Dec_B as the respective generators for domains A and B. By denoting the discriminators for these domains by D_A and D_B, we can write the GANs as (Dec_A, D_A) and (Dec_B, D_B). In this way, the final loss term takes the following form:

$$\begin{aligned}\ell_{\text{GAN}} = &\mathbb{E}_{\substack{c_A \sim \mathbb{P}(c_A) \\ s_B \sim \pi_B}} [\log(1 - D_B(\text{Dec}_B(c_A, s_B)))] + \mathbb{E}_{x_B \sim \mathbb{P}_B}[\log D_B(x_B)] \\ &+ \mathbb{E}_{\substack{c_B \sim \mathbb{P}(c_B) \\ s_A \sim \pi_A}} [\log(1 - D_A(\text{Dec}_A(c_B, s_A)))] + \mathbb{E}_{x_A \sim \mathbb{P}_A}[\log D_A(x_A)]\end{aligned}$$

Using the four loss terms we have described, the MUNIT framework uses first-order methods to solve the following nonconvex optimization problem:

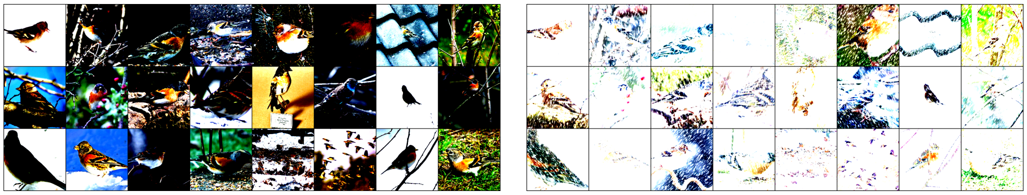
$$\min_{\substack{\text{Enc}_A, \text{Enc}_B \\ \text{Dec}_A, \text{Dec}_B}} \max_{D_1, D_2} \quad \ell_{\text{GAN}} + \lambda_x \ell_{\text{recon}} + \lambda_c \ell^c_{\text{recon}} + \lambda_s \ell^s_{\text{recon}}$$

C.5 HYPERPARAMETERS AND IMPLEMENTATION OF MUNIT

In this subsection, we discuss hyperparameters and implementation details for MUNIT. In particular, in Table 8 we record the hyperparameters we used for training MUNIT models of natural variation. The hyperparameters we selected are generally in line with those suggessted in (Huang et al., 2018). We use the same architetures for the encoder, decoder, and discriminative networks as are described in Appendix B.2 of (Huang et al., 2018).

Table 9: **Dataset descriptions.** We provide a brief description of the datasets used in this paper.

Dataset	Num. of classes	Format	Image size
MNIST (LeCun et al., 2010)	10	Grayscale	28
SVHN (Netzer et al., 2011)	10	RGB	32
GTSRB (Stallkamp et al., 2011)	43	RGB	15 – 250
CURE-TSR (Temel et al., 2019)	14	RGB	3 – 277
MNIST-m (Ganin et al., 2016)	10	RGB	28
Fashion-MNIST (Xiao et al., 2017)	10	Grayscale	28
E-MNIST (Cohen et al., 2017)	26	Grayscale	28
K-MNIST (Clanuwat et al., 2018)	10	Grayscale	28
Q-MNIST (Yadav & Bottou, 2019)	10	Grayscale	28
USPS (Hull, 1994)	10	Grayscale	16
ImageNet-1K (Deng et al., 2009)	1000	RGB	224
ImageNet-C (Hendrycks & Dietterich, 2019)	1000	RGB	224



(a) Test images from ImageNet.

(b) Corresponding images from new dataset.

Figure 18: **Brightness and snow.** We use the challenge-level 1 transforms from ImageNet-c to generate an ImageNet test set with shifts in both brightness and snow.

D DETAILS CONCERNING DATASETS AND DOMAINS

D.1 AN OVERVIEW OF THE DATASETS USED IN THIS PAPER

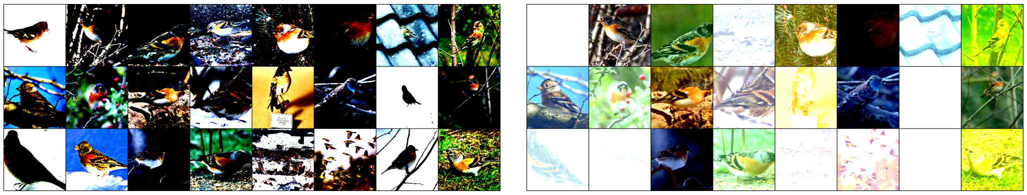
In Table 9, we give an overview of the datasets used in this work. Specifically, we use twelve distinct datasets over the course of our experiments, including the recently curated ImageNet-c test set, which contains copies of the original ImageNet test set corrupted by common sources of natural variation, such as snow, fog, and frost (Hendrycks & Dietterich, 2019).

D.2 DATASETS INTRODUCED IN THIS PAPER

In this paper, we introduced several new datasets which contain multiple simultaneous corruptions, including show, brightness, contrast, and fog. In particular, we used the transforms used to create ImageNet-c to add multiple corruptions to the ImageNet test set. To do so, we used the open-source code from Hendrycks & Dietterich (2019)[2]. Images of these datasets are shown in Figures 18-21.

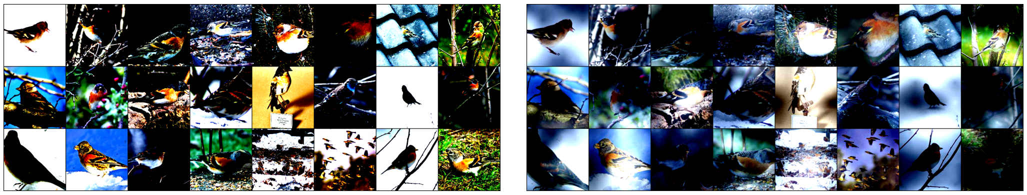
---

[2] `https://github.com/hendrycks/robustness`



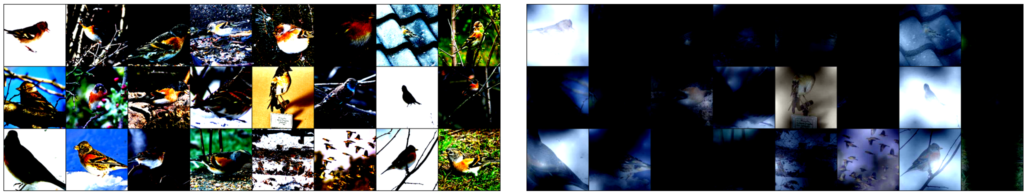
(a) Test images from ImageNet. (b) Corresponding images from new dataset.

Figure 19: **Brightness and contrast.** We use the challenge-level 2 transforms from ImageNet-c to generate an ImageNet test set with shifts in both brightness and contrast.



(a) Test images from ImageNet. (b) Corresponding images from new dataset.

Figure 20: **Brightness and fog.** We use the challenge-level 1 transforms from ImageNet-c to generate an ImageNet test set with shifts in both brightness and fog.



(a) Test images from ImageNet. (b) Corresponding images from new dataset.

Figure 21: **Contrast and fog.** We use the challenge-level 1 transforms from ImageNet-c to generate an ImageNet test set with shifts in both contrast and fog.

	SVHN			**GTSRB**		
	Low	Medium	High	Low	Medium	High
Brightness	$\mathcal{B} < 60$	$160 < \mathcal{B} < 170$	$\mathcal{B} > 180$	$\mathcal{B} < 40$	$85 < \mathcal{B} < 125$	$\mathcal{B} > 170$
Contrast	$\mathcal{C} < 80$	$90 < \mathcal{C} < 100$	$\mathcal{C} > 190$	$\mathcal{C} < 80$	$140 < \mathcal{C} < 200$	$\mathcal{C} > 230$

Table 10: **Brightness and contrast thresholds.** This table shows the thresholds we chose to represent low, medium, and high values of contrast and brightness for SVHN and GTSRB.

D.3 NATURAL VS. SYNTHETIC VARIATION IN DATA

Throughout our experiments, we demonstrate that our methods are able to provide robustness against many challenging sources of natural variation. Furthermore, our experiments contain domains with both *naturally-occurring* and *artificially-generated* variation. Notably, every experiment involving data from SVHN or GTSRB used naturally-occurring variation. In what follows, we discuss both of these categories.

D.3.1 NATURALLY-OCCURRING VARIATION

Throughout the experiments, we used data from SVHN and GTSRB to train neural networks to be robust against contrast and brightness variation. To extract naturally-occurring variation from these datasets, we used simple metrics to threshold the data into subsets corresponding to different levels of natural variation. Specifically, we defined the brightness $\mathcal{B}(x)$ of an RGB image x to be the mean pixel value of x, and we define the contrast $\mathcal{C}(x)$ to be the difference between the largest and smallest pixel values. Table 10 show the thresholds we chose for contrast and brightness on SVHN and GTSRB. Note that these thresholds were chosen somewhat subjectively to reflect our perception of low, medium and high values of brightness and contrast. We intend to experiment with different thresholds in future work.

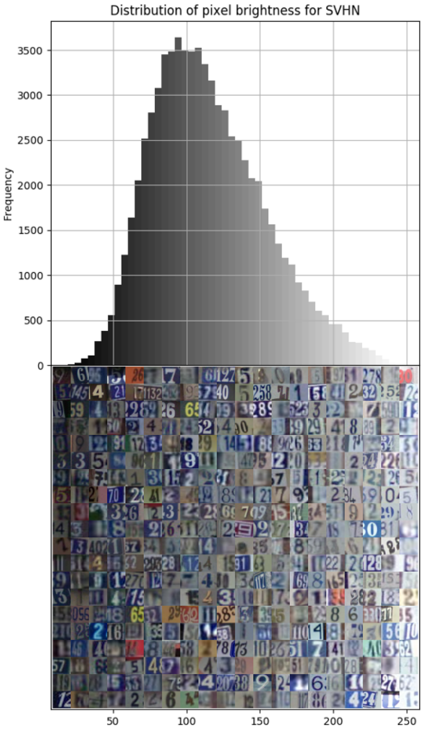
Figure 22 shows a summary of the subsets of SVHN that we compiled corresponding to brightness. In particular, Figure 22a shows a histogram of the brightnesses of images in SVHN. We used this histogram to set thresholds for low, medium, and high brightness, which are given in Table 10. The images below the histogram correspond to the bins of the histogram; that is, images further to the left in Figure 22a have lower brightness, whereas images further to the right have high brightness. In Figures 22b, 22c, and 22d, we show samples from the subsets of low, medium and high contrast subsets of SVHN that we compiled. Figure 23 tells the same story as 22 for the contrast nuisances in SVHN. Again, Figure 23a shows a histogram and accompanying images corresponding to different values of contrast. Figures 23b, 23c, and 23d show samples from the subsets of low, medium, and high contrast images we compiled.

We repeat this analysis for the brightness and contrast thresholding operations for GTSRB in Figures 24 and 25. Again, the difference between high- and low-brightness samples is remarkable, as is the difference in the samples corresponding to high- and low-contrast. However, an interesting difference between the distributions of brightness and contrast on GTSRB vis-a-vis SVHN is that the distributions for GTSRB are skewed, whereas the distributions for SVHN are close to being symmetric.

D.3.2 ARTIFICIALLY-GENERATED VARIATION

The remainder of the experiments, including those on MNIST, CURE-TSR, and ImageNet, use artifically-generated variation. Indeed, one challenge in addressing deep learning's lack of robustness to natural variation is that relatively few datasets contain labeled forms of naturally-occurring sources of variation. To this end, an important research challenge is to curate datasets with naturally-occurring variation; we plan to pursue this goal in future work.

When data with naturally-occurring variation is not available, artificially-generated variation can be used as an effective proxy for testing the robustness of deep learning against different forms of variation (Hendrycks & Dietterich, 2019; Hendrycks et al., 2020). Indeed, the recently curated CURE-TSR (Temel et al., 2019) and ImageNet-c (Hendrycks & Dietterich, 2019) were created using



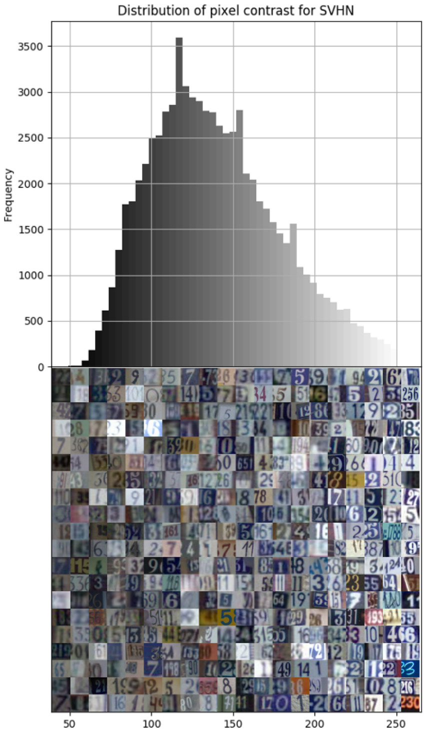
(a) **SVHN brightness histogram.** The histogram shows the distribution of pixel brightness for SVHN. The images below the histogram correspond to the bins of the histogram, meaning samples to the left have low brightness whereas samples further to the right have higher brightness.

(b) **Low brightness samples.**

(c) **Medium brightness samples.**

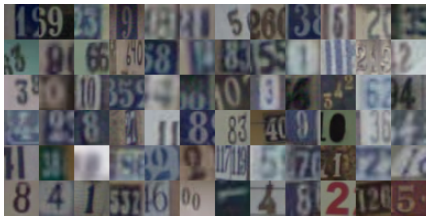
(d) **High brightness samples.**

Figure 22: **SVHN brightness thresholding overview.**

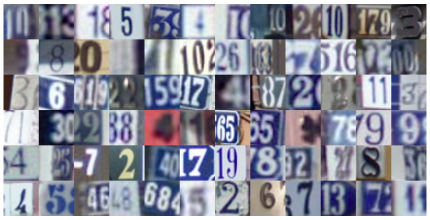


(a) **SVHN contrast histogram.** The histogram shows the distribution of pixel contrast for SVHN. The images below the histogram correspond to the bins of the histogram, meaning samples to the left have low contrast whereas samples further to the right have higher contrast.

(b) **Low contrast samples.**

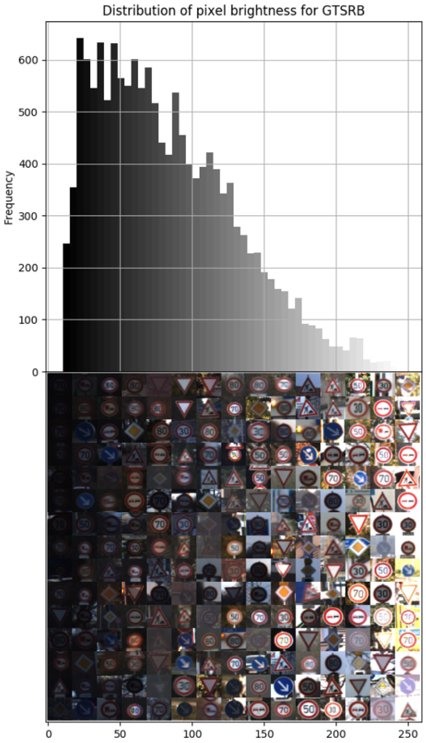


(c) **Medium contrast samples.**



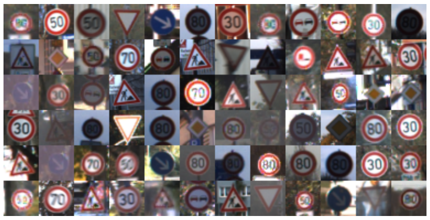
(d) **High contrast samples.**

Figure 23: **SVHN contrast thresholding overview.**

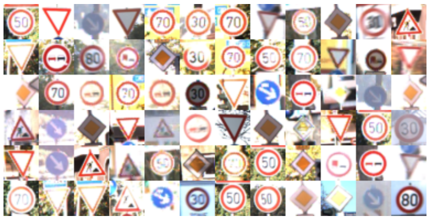


(a) **GTSRB brightness histogram.** The histogram shows the distribution of pixel brightness for GTSRB. The images below the histogram correspond to the bins of the histogram, meaning samples to the left have low brightness whereas samples further to the right have higher brightness.

(b) **Low brightness samples.**

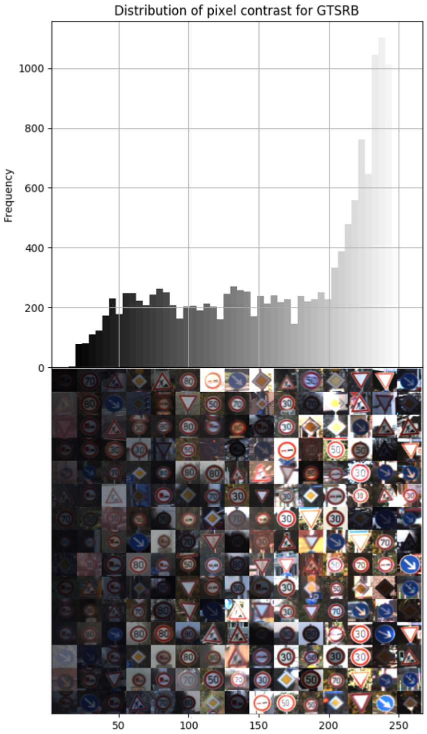


(c) **Medium brightness samples.**

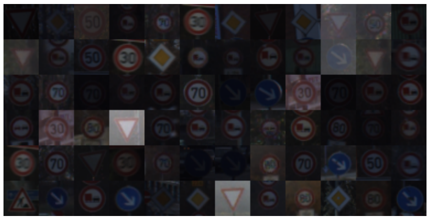


(d) **High brightness samples.**

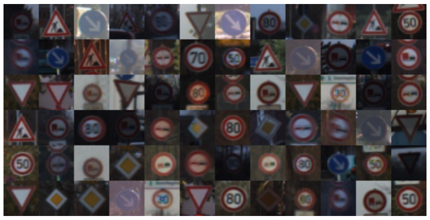
Figure 24: **GTSRB brightness thresholding overview.**



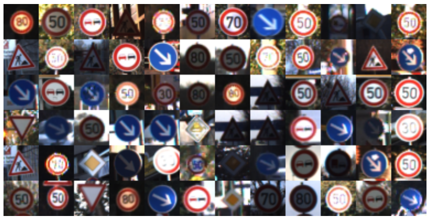
(a) **SVHN contrast histogram.** The histogram shows the distribution of pixel contrast for SVHN. The images below the histogram correspond to the bins of the histogram, mening samples to the left have low contrast whereas samples further to the right have higher contrast.



(b) **Low contrast samples.**



(c) **Medium contrast samples.**



(d) **High contrast samples.**

Figure 25: **SVHN contrast thresholding overview.**

pre-defined, artifical transformations of data. While these transformations are synthetic, the images in Appendix A show that they are indeed quite realistic.