Defending Object Detection Models against Image Distortions

Mark Ofori-Oduro Maria Amer
Department of Electrical and Computer Engineering, Concordia University, Montréal, Canada

Abstract

Image distortions pose a significant challenge to object detection. To address this issue, our paper introduces a novel data augmentation method that generates new samples resembling the original training images. The new sample exhibits randomly altered pixels based on a pixel distribution obtained from multiple image distortions using kernel density estimation (KDE). The main steps of our method, GSES, are generating distorted versions of each pixel of an original training image, selecting a set of pixels in each version, and then, for each selected pixel, estimating its distribution using KDE and then sampling one pixel from this distribution. By employing this approach, the new samples possess distorted pixels while maintaining a certain degree of similarity to the original image. This degree of similarity is essential to balance the accuracy of object detection models under distorted and clean images. Our approach improves the accuracy of different object detection models under 15 image distortions, such as motion blur, fog, and noise. For example, the average accuracy of YOLOv4 improves by 9.19% and 9.54 % across all 15 distortions added to the COCO and PASCAL datasets, respectively. Our method surpasses other defence methods to combat image distortions. Our ablation and stability studies show why our method performs well. Moreover, we also show that our method can be well used to improve the accuracy of image classification under 15 distortions and cross-domains. Our code is available at https://github.com/moforio/GSES/.

1. Introduction

Developing models that can perform well even in the presence of image distortions is essential to ensure accurate object detection in real-world applications, such as self-driving, activity recognition, and security [5, 11, 12, 26, 27, 33, 39]. Image distortions include Gaussian noise, impulse noise, shot noise, jpeg compression, pixelated, elastic transform, motion blur, glass blur, zoom blur, defocus blur, contrast, snow, fog, frost, and brightness. Such distortions cause the accuracy of the object detection model to significantly decrease (for example, YOLOv4 by 26.55% under snow using PASCAL).

Simply adding noise (or any other distortion) to the training images can increase a model's robustness. However, as shown in [9, 13, 18], it has two main drawbacks: reduced performance on clean images and poor generalisation to other distortions not used in training. Our experiments on the PASCAL dataset with object detector YOLOv4 mirrored these findings: when trained with 100% clean and 50% distorted samples, ensuring that 15 image distortions were equally represented, we noticed a boost in average accuracy under the 15 distorted conditions by +10.11% during validation. However, this was offset by a dip in accuracy for clean images by -1.32%.

Data augmentation can help improve accuracy under distortions. However, it needs to be developed carefully to avoid loss of accuracy under clean samples.

This paper presents a novel data augmentation approach to generate new samples to improve object detection performance under distortions. In our approach, a pixel of a clean (original) image is replaced by a value sampled from a set of 15 estimated distributions, which are estimated using KDE. Not all pixels are replaced to maintain a certain similarity to the original samples. We show the superiority of our approach to related works using different detection models under 15 image distortions. We also show that our approach generalises to unknown distortions and to image classification models.

2. Literature Review

Related works are categorised into three groups: conventional data augmentation methods [2, 4, 16, 36–38], image distortion defence methods [1, 20, 24, 32], and image enhancer-based defence methods [23, 30, 31]. These methods can be implemented either during the training process (online) or before training (offline), with the capability to operate in the image domain or the network domain, potentially altering the image classification or object detection model architecture. This section first reviews these related works, followed by Section 2.4, where we elucidate the distinctions between our and previously mentioned methods.

2.1. Conventional data augmentation methods

MixUp [37], CutMix [36], AugMix [16], and RandE [38] are data augmentation methods designed to enhance the performance image classification models. MixUp [37] linearly interpolates between pairs of images and their labels online; CutMix [36] takes this MixUp idea spatially by cutting and pasting patches between image pairs, adjusting labels based on the area of the patches online; AugMix [16] blends multiple diverse augmentations of an image online and employs a distribution-matching loss to ensure the augmented images align closely with the original data manifold; RandE [38] improves performance by randomly selecting and occluding portions of the image with random values online.

The state-of-the-art in data augmentation for object detection is epitomized by two methods: Mosaic [4] and GridMask [2]. Mosaic, introduced by Dadboud et al. [4], ingeniously combines four images into a mosaic formation online, infusing the training process with diverse images that enhance model performance; GridMask, proposed by Chen *et al.* [2], employs a novel approach of overlaying a grid-like mask on the input online. This unique technique stimulates spatial diversity and fosters the model's ability to extract features from various regions of the input effectively.

2.2. Image distortion defences methods

Image distortion defence methods utilize various methodologies, including data augmentation [3, 20, 24, 25] and detection model modification [1, 32]. Network modifications have been proposed as effective strategies for improving model performance against specific image distortions. Borkar and Karam [1] proposed DeepCorrect, a correction network to improve classification models against image noise and blur, while Sun et al. [32] proposed a feature quantization method for defence against image noise, blur, and compression in image classification and object detection models. The SmoothMix data augmentation technique [20], proposed by Lee *et al.*, enhances the robustness of image classification models by augmenting their training images with new samples generated by blending two images online; Michaelis *et al.* [24] show that applying the style transfer (superpose a style of an image on another image [8]) offline as a data augmentation technique (termed Stylize) can improve the robustness of object detection models (such as Mask R-CNN [15] and RetinaNet [22]) on their provided benchmark of 15 image distortions; Det-AdvProp [3] generates additional data by dynamically generating augmented images online by adding a weighted signed gradient derived from the classification and location loss functions; In [25], the authors propose an algorithm to generate distorted images using artificial immune systems; it was evaluated only for Gaussian noise. Compared to the model modification approach, the augmentation-based defence approach has a significant advantage because it can be easily extended to many models.

2.3. Image enhancer-based defence methods

Image enhancers URIE [30], DeepN [23], and OWAN [31] propose front-end CNN models to enhance or denoise corrupted images before being fed into classification or object detection models. Overall, these defence methods contribute to developing robust models.

2.4. Distinguishing our method from existing

Different from the enhancers URIE [30], DeepN [23], and OWAN [31], our method does not focus on enhancing or denoising distorted inputs. Our approach has been validated under 15 distortions and clean samples on object detection and image classification models, unlike the defence methods MixUp [37], CutMix [36], AugMix [16], RandE [38], SmoothMix [20], and DeepCorrect [1], which are primarily employed in image classification and , except SmoothMix, not evaluated under distortions.

As witnessed in RandE [38], Mosaic [4], and Stylize [24], data augmentation methods often encounter problems with overfitting, leading to decreased generalization. They also struggle to maintain a balance in class distributions and preserve label integrity, as shown in MixUp [37], CutMix [36], SmoothMix [20], and Mosaic [4]. Such challenges often result in a reduction in overall model performance. Our approach is designed to mitigate these challenges.

Our method retains some of the original pixels in the newly generated image rather than completely transforming the style, as seen in Stylize [24], or distorting all pixels, as in Det-AdvProp [3]. This distortion mode reduces overfitting, enabling our method to provide consistent results across small and large datasets, unlike Stylize [24]. Additionally, our method does not require supplementary CNN models or data, an advantage over the Stylize method [24].

Unlike in MixUp [37], CutMix [36], and SmoothMix [20], which require new labels for distorted images, our method utilizes original image labels to generate corresponding distorted images. Moreover, our method only generates a single new sample from an original, avoiding the class imbalance that may arise when merging two original images randomly, as in MixUp [37], CutMix [36], and SmoothMix [20]. Importantly, our method leaves the architecture of the CNN model untouched, unlike the approach in DeepCorrect [1] and [32].

3. Proposed method

3.1. Overview of our method

Fig. 1 overviews our proposed method to generate new samples resembling the original training images. We randomly select a pixel location for a given image x and change that pixel based on a pixel distribution obtained from multiple pixel distortions using kernel density estimation. We repeat this not for all input image pixels but for a selected set. By employing this approach, the generated new sample possesses distorted pixels while maintaining a certain degree of similarity to the clean image x.

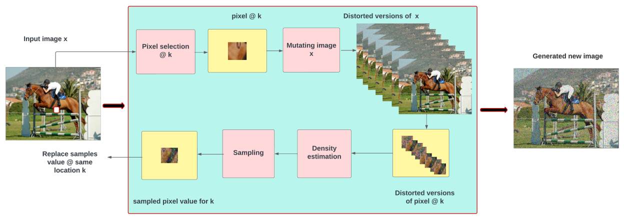

Figure 1. A simplified overview of our method (best viewed by zooming).

To ensure that our method maintains a balance between object detection performance under clean and distorted images, we generate new samples for a subset of the original training images and maintain a certain level of similarity between these new and their corresponding original images. Overall, our data augmentation approach can be divided into 3 main steps as detailed in the next 3 subsections and Algorithm 1: 1) Data pre-processing, where we randomly select a portion of the original images to use for generating new samples; 2) New sample generation, where we generate the new samples by creating distorted versions of the selected original images while maintaining the desired level of similarity; 3) Data augmentation, where we use these generated new samples to augment the original images for training.

The effectiveness of our method comes from two main aspects: 1) randomly, at the pixel level, balancing the similarity of the new and the original samples (i.e., the use of τ in Algorithm 1) and 2) diversity (that is, we use $D = 15$ distortions for pixel distribution estimation and pixel sampling in Algorithm 1). A detection model trained with our data augmentation learns, hence, more robust features from clean and diversely distorted pixels. We show in the result Section 4.1 how decreasing similarity (larger τ) causes accuracy loss under clean samples and does not significantly improve the accuracy under distortion. We also show that pixel diversity helps the detection model to generalize to unknown distortions (Section 4.5) and that decreasing diversity decreases accuracy under distortions (Section 4.7).

3.2. Data pre-processing

Let the training dataset $\{x^i, y^i\}_{i=1}^M$ consist of M training images, where each input image x^i is associated with a corresponding label y^i. We denote these images as Ag. We aim to create distorted versions of x^i, collectively denoted as Ab, to train object detection models and consequently enhance their accuracy under various image distortions.

Algorithm 1: Proposed new sample generation by distribution estimation and sampling.

```
Input: Original images $Ag = \{x^i, y^i\}_{i=1}^M$, $D$
       distortions types, augmentation ratio $p$,
       affinity threshold $\tau$
Output: new samples $Ab$
1  Number of $Ab$: $N = p \cdot M$ ;
2  Number of $Ab_d$: $T = N/D$ ;
3  $Ab \longleftarrow$ Select $N$ images randomly from $Ag$ ;
4  for distortion $d = 1$ to $D$ do
5      $Ab \longleftarrow$ Randomly shuffle $Ab$;
6      Select $T$ images from $Ab$ to get
         $Ab_d = \{x^i, y^i\}_{i=((d-1)\cdot T)+1}^{d\cdot T}$ ;
7      for each image $x^i \in Ab_d$ do
8          $\{\tilde{x}_i\}^S \longleftarrow$ Mutate: generate $S$ distorted
             versions of $x^i$;
9          $K \longleftarrow$ Select a fraction $\tau$ of pixels from $x^i$,
             uniformly at random ;
10         for each pixel $k$ in $K$ do
11             $\tilde{X} \longleftarrow$ Copy the distorted pixels at
                 position $k$ from all images in $\{\tilde{x}_i\}^S$ ;
12             $E_{\tilde{X}} \longleftarrow$ Estimate the distribution of $\tilde{X}$
                 using KDE ;
13             $\tilde{x}_{new} \longleftarrow$ Sample one pixel uniformly at
                 random from pixel distribution $E_{\tilde{X}}$ ;
14             $x^i \longleftarrow$ Mutate $x^i$ by replacing the value
                 at $k$ with $\tilde{x}_{new}$ ;
15         end
16     end
17 end
```

In the data pre-processing step, as outlined in Algorithm 1 lines 1 to 6, we randomly select N from the input M images $\{x^i, y^i\}_{i=1}^M$ to get $\{x^i, y^i\}_{i=1}^N$ Ab, where N=$p \cdot M$ and $0 < p \leq 1$ is an augmentation ratio p, a hyper-parameter of our method (see Algorithm 1 input). Before this random selection, it is always important to check the class distribution of the training images Ag. When dealing with a skewed class distribution, one should select images class-awarely to prevent class imbalance in the final training set. To consider various distortions, we select d different distortion types, and hence we divide these Ab into D groups Ab_d. Each Ab_d for a distortion type d contains $T = N/D$ images (see Algorithm 1 line 6). Given $d = 1$ to D, when $d = 1$, i is from 1 to T, when $d = 2$, i is from $T + 1$ to $2T$ and so on. And since the datasets we use (PASCAL [6], and COCO [21]) have uniformly distributed classes, we can do the aforementioned division randomly.

3.3. New sample generation

To generate new samples, Algorithm 1 *selects* an image, *estimates* its distribution, and *mutates* an image. We aim to mutate (change) selected pixels of an image x^i with the desired distortion. We perform the image mutation using two steps: 1) distortion of the original images using image processing tasks taking distortion D into account; 2) mutation (changing) of selected pixels by sampling from the distorted pixel distribution.

In the first mutation step, given an image x^i and a distortion type D (e.g., Gaussian noise, jpeg compression, and snow), we generate S distorted images $\{\tilde{x}_i\}^S$ using image processing tasks filtering, clipping, scaling, and shuffling taking D into account. The image processing tasks cause all pixels to be mutated in the distorted image $\tilde{x}_i$, resulting in not only distortion D but also other changes (see [24] for details). Note that for each of the input T images selected for a distorted type (see Algorithm 1 line 6), we generate S distorted images, where $S = 100$ (see Algorithm 1 line 8); higher S unnecessarily increases computations.

In the second mutation step, we focus on how many pixels and not how much the pixels are changed. In each image x^i, we mutate only a fraction τ of its pixels to keep a level of similarity with its clean version. We uniformly select K pixels (see Algorithm 1 line 9), where $K = \tau \cdot \text{pixels}\{x_i\}$ of the pixels we want to mutate from x^i uniformly.

Our method requires the hyper-parameter τ, while the distortion types D are fixed to 15. For example, if $\tau = 0.75$, 75% of the pixels of x^i are selected for mutation by estimating and sampling from the distortion type under consideration. We consider $D = 15$ distortion types, i.e., Gaussian noise, impulse noise, shot noise, jpeg compression, pixelated, elastic transform, motion blur, glass blur, zoom blur, defocus blur, contrast, snow, fog, frost, and brightness. τ controls how many pixels are mutated and how many are left unchanged (clean). Thus, it compromises clean and distorted pixels, improving object detection under distorted images without deterioration on clean images. During training, the object detection model learns a mixture of "distorted" and "clean" features.

To effectively mutate pixels in x^i, we need to estimate distortion distributions. We can achieve this in two main ways, namely parametric and non-parametric approaches. We opt for the standard non-parametric approach of kernel density estimation (KDE) because it makes no assumptions about the distributions of the distortion, allowing us to apply it to different distortion types. KDE has been adapted for different applications in computer vision, such as for spatio-temporal background modelling [29].

For KDE, for a selected pixel location k in x^i, we form the distorted vector $\tilde{X}$. Recall $\tilde{x}$ is a distorted version x^i and there are S such images for each x^i; we form $\tilde{X} = [\tilde{x}_i{}^1, \tilde{x}_i{}^2, ..., \tilde{x}_i{}^S]^\top$ for pixel location k. In KDE, we make use of Gaussian kernels to estimate the distribution $\tilde{X}$. More specifically, for each pixel location, $\tilde{X}$ is expressed as a linear combination of equal-width Gaussians centred around each point $\tilde{X}$, where the width (or standard deviation) of the Gaussians is a hyper-parameter. As KDE is a general non-parametric method, we may utilize different kernels, including uniform or triangular kernels. However, we limit ourselves to the Gaussian kernel because this fits a mixture of Gaussians (a universal density approximator [10]) to the pixel distribution. Once the estimation of $\tilde{X}$ is obtained and denoted as $E_{\tilde{X}}$, we sample one pixel value from $E_{\tilde{X}}$ to get $\tilde{x}_{new}$ and use $\tilde{x}_{new}$ to mutate x^i by replacing pixel value of x^i at index k. For example, in Fig. 2, we show four new samples generated by mutating pixels with new pixel values sampled from an estimated distribution of Gaussian noise (top-left), Gaussian blur (top-right), jpeg compression (bottom-left), and snow (bottom-right).

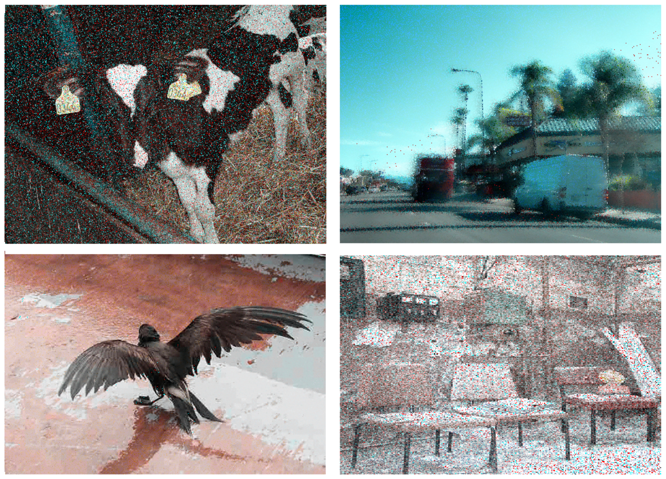
Figure 2. Examples of generated new samples. The model estimates and samples from the distribution of Gaussian noise (top-left), motion blur (top-right), jpeg compression (bottom-left), and snow (bottom-right).

3.4. Data augmentation

In the data augmentation step, we augment the new samples Ab to the original training images Ag to train object detection models. Before the training, we shuffle the Ab and Ag to prevent bias to either of them.

4. Outcome of experiments and analysis

We benchmark our augmentation-based defence technique against three categories: 1) augmentation-based defence methods, 2) image-enhancer-based defence methods, and 3) conventional augmentation techniques. Further, we demonstrate our method's performance under unknown distortions and its applicability to image classification and autonomous driving.

In the subsequent tables of results, entries highlighted in red represent the top performance, while those in blue indicate the second best. The notation +() represents the gain, signifying the performance difference of an object detector with and without a defence method.

4.1. Configuration for experiments

We measure the accuracy of object detection using the mean average precision (mAP) calculated based on the intersection over union (IoU) of the ground truth and estimated bounding box (BB) of the object. When evaluating the PASCAL dataset [21], we follow standard practice and set the IoU to 0.5 for all reported mAPs. For the COCO dataset, unless stated otherwise, the mAP is an average of 10 IoU thresholds ranging from 0.50 to 0.95. We evaluate performance under corruption using ”mean performance under corruption” (mPC) defined in [24] as

$$mPC = \frac{1}{N_a}\sum_{a=1}^{N_a}\frac{1}{N_b}\sum_{b=1}^{N_b} mAP_{a,b}, \tag{1}$$

where N_a is the total number of corruptions (a), N_b is the severity (b) level for each corruption type, and $mAP_{a,b}$ is mAP for a at a specific b. We also use ”relative performance under corruption” (rPC) [24] defined as

$$rPC = \frac{mPC}{mAP_{org}}, \tag{2}$$

where mAP_{org} is the mAP under original images.

For Algorithm 1 (line 4), we set $D = 15$, i.e., we estimate the distribution from 15 distortions: Gaussian noise, impulse noise, shot noise, jpeg compression, pixelated, elastic transform, motion blur, glass blur, zoom blur, defocus blur, contrast, snow, fog, frost, and brightness. Unless indicated for all result tables, we use five different severity levels (1 to 5) for each distortion.

We apply our method to two datasets, the PASCAL dataset [6] and the COCO dataset [21]. For the PASCAL dataset, following standard practice, we utilize the combined training and validation sets of 2007 and 2012 as our training data and the test set of 2007 as our validation data. As for the COCO dataset, we use the training and validation sets of 2017 as our training and validation data, respectively. In our experiments, we matched the ground truth label of a clean image to its corresponding new sample. To demonstrate the effectiveness of the proposed data augmentation, we trained YOLOv4 (CSPDarknet-53, one-stage detector) [34] and Faster RCNN (ResNet-101, two-stage detector) [28] models with and without our data augmentation.

The hyper-parameters of our method are the augmentation ratio p and the affinity threshold τ, which are set to 0.50 and 0.75, respectively, for all object detection models. As summarized in Table 1, these values provide the best compromise for accuracy under clean and distorted images using the PASCAL dataset on YOLOv4.

$p = 1$			
τ	0.65	0.75	0.85
mAP_{org}	83.04	83.00	82.86
mPC	62.82	64.66	65.50
$\tau = 0.75$			
p	0.25	0.50	1.00
mAP_{org}	83.58	83.32	83.00
mPC	60.39	62.02	64.66

Table 1. PASCAL and YOLOv4: Tuning the hyper-parameters (p and τ) of our method.

4.2. Validation outcomes

For the comparison of distortion defence methods, we compare our method with the data augmentation methods, Stylize [24] and SmoothMix (SMix) [20], and with image enhancement methods URIE [30], OWAN [31], and DeepN [23]. (We do not compare to Det-AdvProb [3] and [32] because the code for training is not publicly available). As seen in Table 2 for the PASCAL dataset, our method is the best to provide a balance between performance under clean and distorted images among all related works. Among the data augmentation methods, it outperforms (+9.54% and +5.77%) the second-best method Stylize (+7.96% and 5.57%) in both YOLOv4 and Faster RCNN under distorted images, respectively. Under clean images, our method does not result in any deterioration, which is significant in data augmentation methods [19]. For the image enhancers, as demonstrated in Table 2, only URIE showed improvement under distortions, while OWAN failed to provide any improvement. The OWAN model did not undergo training on the 15 distortions tested, unlike the URIE model. DeepN only improved (4.96%) under noise (Gaussian, impulse, and shot) but resulted in significant deterioration in other distortion, hence overall negative (-10.47%) impact on accuracy.

Model	mAP_{org}	mPC	rPC
YOLOv4	83.31	52.48	0.6299
+ Our	83.32 (+0.01)	62.02 (+9.54)	0.7444
+ Stylize	83.90 (+0.59)	60.44 (+7.96)	0.7254
+ SMix	83.65 (+0.34)	53.04 (+0.56)	0.6367
+ URIE	-	58.05 (+5.57)	0.6968
+ DeepN	-	41.45 (-11.02)	0.4976
+ OWAN	-	40.10 (-12.38)	0.4813
Faster RCNN	79.25	54.63	0.6893
+ Our	79.30 (+0.05)	60.40 (+5.77)	0.7621
+ Stylize	78.52 (-0.73)	60.20 (+5.57)	0.7596
+ SMix	79.65 (+0.40)	54.82 (+0.20)	0.6918
+ URIE	-	60.08 (+5.45)	0.7581
+ DeepN	-	44.72 (-9.91)	0.5643
+ OWAN	-	41.72 (-12.91)	0.5264

Table 2. PASCAL validation set for YOLOv4 and Faster RCNN: Comparison of our method with image distortion defence and image enhancer methods.

We evaluate the performance of our method against conventional data augmentation methods: MixUp [37], CutMix [36], AugMix [16], RandE [38], and Mosaic [4] using

YOLOv4 under the 15 distortions on the PASCAL validation dataset. As shown in Table 3, while all methods improve accuracy (mAP_{org}) under clean images, our method significantly outperforms these conventional augmentation methods under the 15 distortions.

Model	mAP_{org}	mPC	rPC
YOLOv4*	82.77	50.45	0.6095
+ Our	83.22 (+0.45)	60.82 (+10.37)	0.7348
+ MixUp	82.90 (+0.13)	50.76 (+0.31)	0.6133
+ CutMix	83.31 (+0.54)	52.48 (+2.03)	0.6340
+ AugMix	81.27 (-1.50)	51.55 (+1.10)	0.6228
+ RandE	83.43 (+0.66)	49.91 (-0.54)	0.6030
+ Mosaic	83.61 (+0.84)	51.05 (+0.60)	0.6168

Table 3. YOLOv4 and PASCAL validation set: Comparison of our method with conventional data augmentation methods. YOLOv4* is YOLOv4 without its baseline augmentation CutMix.

For the COCO validation dataset, we compare our method with related augmentation-based and enhancer-based defence methods, which showed promising results for PASCAL in Table 2. As for the detection model, we only use YOLOv4 because it has a better accuracy under clean samples than Faster RCNN and is also more prone to distortions as shown by their mAP_{org} and mPC under PASCAL. As shown in Table 4, our method significantly outperforms Stylize under clean and distorted images. URIE also improves, but to a lesser extent, than our method.

Model	mAP_{org}	mPC	rPC
YOLOv4	40.67	23.6134	0.5806
+ Our	40.81 (+0.14)	32.81 (+9.19)	0.8067
+ Stylize	38.95 (-1.73)	26.59 (+2.97)	0.6537
+ URIE	-	24.52 (+0.91)	0.6029

Table 4. COCO validation set and YOLOv4: Comparison of our method with Stylize and URIE.

4.3. Test outcomes

We conducted a performance evaluation of our method, Stylize, and URIE on the COCO test dataset using YOLOv4. Due to the limitations imposed on submitting results to the COCO server, we assessed performance under four distortions with severity 3, representing main types: noise, blur, artefacts, and weather conditions. Specifically, we selected impulse noise, zoom blur, jpeg compression, and snow as representative distortions that are known to have a significant impact on YOLOv4 for each distortion type, respectively. In Table 5, it is evident that our method (+6.95) outperforms URIE (+4.90) and Stylize (+4.03), showcasing its superiority in handling these distortions.

4.4. Cross-domain analysis

To show if features learnt by a model using our approach are transferable to unseen object detection datasets, we investigate the effect of training YOLOv4 on COCO but validating it on PASCAL without fine-tuning. As shown in Table 6, our method added to YOLOv4 still improves the accuracy under the clean samples (on average by 0.60%) and the 15 distortions (on average by 7.94% under severity level 3). Comparing the cross-dataset (COCO to PASCAL) and in-dataset (PASCAL to PASCAL) in Table 6, we observe two things. First, the cross-dataset approach improves YOLOv4 under distortions by +3.69% in terms of mPC (56.25 versus 52.56). This could be due to the model using features (learnt from COCO) which are fundamental to the detection of objects in PASCAL. These COCO features appear to be more robust to distortions. Second, YOLOv4 performs lower by -2.61% in terms of mAP_{org} (80.70) under cross-datasets compared to in-dataset (83.31). This observation could be attributed to the difference in the resolution and object distribution of images in both datasets.

Model	mAP_{org}	mPC	rPC
YOLOv4	41.60	23.68	0.5691
+ Our	41.90 (+0.30)	30.63 (+6.95)	0.7363
+ Stylize	38.90 (-1.80)	27.70 (+4.03)	0.6659
+ URIE	-	28.58 (+4.90)	0.6869

Table 5. COCO test set: Comparison of our method with Stylize and URIE.

	YOLOv4	+ Our	YOLOV4	+ Our
	COCO to PASCAL	COCO to PASCAL	PASCAL to PASCAL	PASCAL to PASCAL
clean	80.70	81.30 (+0.60)	83.31	83.50 (+0.19)
Gaussian noise	56.07	74.17 (+18.11)	52.83	72.49 (+19.66)
shot noise	58.54	75.99 (+17.45)	56.36	73.98 (+17.62)
impulse noise	50.96	80.41 (+29.45)	48.48	73.56 (+25.08)
motion blur	48.47	54.09 (+5.62)	37.72	42.62 (+4.90)
zoom blur	42.12	45.90 (+3.78)	37.72	38.23 (+0.51)
glass blur	27.24	38.37 (+11.13)	20.12	25.14 (+5.02)
defocus blur	57.69	63.86 (+6.17)	47.84	51.12 (+3.28)
contrast	77.31	78.47 (+1.16)	73.69	74.08 (+0.39)
jpeg compression	59.17	64.85 (+5.68)	55.76	69.38 (+13.62)
pixelate	32.69	37.88 (+5.20)	31.76	43.94 (+12.18)
elastic transform	50.64	56.49 (+3.22)	53.11	53.96 (0.85)
frost	64.72	67.94 (+1.19)	60.64	59.99 (-0.65)
fog	78.08	79.26 (+3.88)	75.76	76.02 (+0.26)
snow	60.40	64.28 (+1.19)	57.37	56.88 (-0.49)
brightness	79.58	80.77 (+7.94)	79.27	79.70 (+0.43)
mPC	56.25	64.18 (+7.94)	52.56	59.41 (+6.84)
rPC	0.6970	0.7953	0.6309	0.7131 (+0.0822)

Table 6. Cross-dataset: Gain introduced by YOLOv4+Our on training on COCO and validating on PASCAL. Here, we used severity level 3. (Results for PASCAL to PASCAL are provided for comparison.)

We also evaluate the performance of our method trained on COCO and validated on the autonomous driving datasets KITTI [7] and BDD100K [35] without fine-tuning. They include images with real-world distortions. Hence, we do not add any distortions to them during validation. In Table 7, we report results for the class of objects that overlap with the COCO dataset. As seen, our method improves YOLOv4 by 1.69% and 1.15%, respectively.

	YOLOv4 COCO to KITTI	+ Our COCO to KITTI	YOLOv4 COCO to BDD100K	+ Our COCO to BDD100K
Person	47.62	47.90 (+0.28)	45.60	46.24 (+0.64)
bicycle	4.12	9.30 (+5.18)	34.27	34.35 (+0.08)
car	74.70	74.80 +(+0.10)	57.79	58.09 (+0.30)
bus	5.00	5.74 (+0.74)	39.62	41.63 (+2.10)
truck	18.79	20.97 (+2.18)	33.36	34.66 (+1.30)
motorbike	-	-	32.03	32.65 (+0.62)
traffic light	-	-	21.63	24.70 (+3.07)
mAP_{org}	30.05	31.74 (+1.69)	37.76	38.90 (+1.15)

Table 7. Cross-domain: Gain introduced by YOLOv4+Our on training on COCO and validating on KITTI and BDD100K.

4.5. Generalisation to unknown distortions

Our method and URIE both have prior knowledge of the 15 image distortions used in the validation phase of our experiments. This section checks how both methods perform under unknown distortions, here Gaussian blur, speckle noise, spatter, and saturate. Like the 15 known distortions, each of these unknowns is added to the clean image with five severity levels. As shown in Table 8, for the PASCAL validation set and YOLOv4, our method is effective and can be generalized for unknown distortions. Also, we see our method significantly outperforms URIE for the unknown distortions. We provide the results for Stylize, which does not have prior knowledge of the distortions, to show our method outperforms it under these additional distortions.

Model	mAP_{org}	mPC	rPC
YOLOv4	83.31	63.13	0.7578
+ Ours	83.32 (+0.01)	68.75 (+5.62)	0.8252
+ URIE	-	53.89 (-9.24)	0.6468
+ Stylize	83.90 (+0.59)	67.90 (+4.77)	0.8151

Table 8. PASCAL validation set and YOLOv4: Comparison of our method with URIE under unknown distortions.

4.6. Generalisation to image classification

This section highlights the versatility of our method by extending it to image classification tasks. For this purpose, we employ SpinalNet [17], a recent classification model, and evaluate its performance on three well-known classification datasets: CIFAR-10, CIFAR-100, and Caltech-101. Similar to the testing conducted on the COCO dataset, we utilize distorted versions of the respective classification datasets, incorporating impulse noise, motion blur, jpeg compression, and snow with severity level 3. The results presented in Table 9 demonstrate the efficacy of our approach in enhancing the accuracy of SpinalNet across all tested distortions.

4.7. Ablation study

The first ablation study is about the effect of the distortion diversity, that is, the number of distortions D in Algorithm 1. We aim to maintain an equilibrium among different distortions within the total number used to maintain a balanced diversity. For instance, when we tested for 4 distortions, we used one distortion from noise, blur, artefacts, and weather condition distortion. Table 10 seems to suggest that increasing the number of distortions enhances performance in the presence of distortions. However, this increase may cause the performance under clean to reduce, although not worse than the base model.

	CIFAR-10		CIFAR-100		Caltech-101	
	SpinalNet	+ Our	SpinalNet	+ Our	SpinalNet	+ Our
clean	97.50	97.71	86.79	87.24	97.07	97.35
impulse noise	33.55	94.47	11.83	79.44	94.34	95.88
motion blur	42.44	42.44	23.75	67.09	92.39	95.51
jpeg compression	54.80	54.80	26.09	59.67	95.28	96.58
snow	79.85	79.85	48.80	74.49	89.26	92.74

Table 9. CIFAR-10, CIFAR-100, and Caltech-101 test set: *Accuracy* comparison of our method on SpinalNet. *Accuracy* is the ratio of correct predictions to total predictions. Here, we used severity level 3.

No. of distortions	mAP_{org}	mPC
4	83.71(+0.40)	59.41(+6.93))
8	83.87(+0.56)	60.32(+7.84)
12	83.32(+0.01)	61.31 (+8.83)
15	83.32(+0.01)	62.02(+9.54)

Table 10. Effect of the number of distortions used in new sample generation for YOLOv4 and PASCAL.

The second ablation study concerns model complexity versus accuracy. This relationship can be understood through the bias-variance trade-off, which considers the total number of model parameters [14]. For instance, when analyzing Table 11, it becomes evident that Faster RCNN achieves the highest accuracy with ResNet-101, which consists of around 44 million parameters, whereas VGG-16, with 138 million parameters, performs relatively worse.

Model	mAP_{org}		mPC	
Faster RCNN	-	+ our	-	+ our
w/ ResNet-101	79.25	79.30	54.63	60.40
w/ VGG-16	75.51	75.55	45.89	49.00

Table 11. Effect of the complexity.

For a stability study, we conducted five experiments (i.e., we trained and validated each model five times). We used our method and Stylize (the second-best method) to calculate their mean and standard deviation. As shown in Table 12, YOLOv4 appears to be the most stable under clean images. However, under distortion, YOLOv4+our appears to be the most stable. This observation is consistently confirmed under mPC and rPC. Using our seems to provide the best stability under both clean and distorted images.

Model	YOLOv4	+ our	+ Stylize
mAP_{org}	83.39($\pm$0.1559)	83.29($\pm$0.1696)	83.80($\pm$0.1753)
mPC	52.20($\pm$0.4739)	62.28($\pm$0.4469)	61.11($\pm$0.5129)
rPC	0.6259($\pm$0.0058)	0.7476($\pm$0.0054)	0.7328($\pm$0.0065)

Table 12. Stability analysis using mean and standard deviation.

4.8. Computational speed

The new samples from our method were generated on a PC with 260GB RAM and an Intel Xeon processor with 36 cores, clocked at 2.30 GHz. In Table 13, we present the average time required to generate a new sample for the PASCAL and COCO datasets with an affinity threshold of $\tau = 0.75$, factoring in the estimation and sampling of a distortion type. Notably, 70% of the time allocated for the new sample generation is consumed in creating S distorted images (see Algorithm 1 line 8) during the estimation and sampling phases (see Algorithm 1 lines 10 to 15). Regarding time consumption, blur distortions (defocus, glass, motion, and zoom) demand the most time, whereas artefacts (contrast, elastic transform, pixelation, and jpeg compression) require the least.

Distortion	Average time (s)	Distortion	Average time (s)
Gaussian noise	2.96	snow	5.16
shot noise	5.16	frost	2.90
impulse noise	2.69	fog	2.82
-	-	brightness	7.54
defocus blur	3.10	contrast	2.56
glass blur	8.57	elastic transform	7.61
motion blur	4.61	pixelate	1.90
zoom blur	28.21	jpeg compression	1.91

Table 13. Cost of generating a new sample.

5. Conclusion

This paper proposed a data augmentation method to create variations of the original training clean images by randomly replacing pixels with new values sampled from an estimated distribution while maintaining a level of similarity to the clean images. To estimate the distribution, we use kernel density estimation (KDE), and we use a diverse 15 set of distortions for this, i.e., Gaussian noise, impulse noise, shot noise, jpeg compression, pixelated, elastic transform, motion blur, glass blur, zoom blur, defocus blur, contrast, snow, fog, frost, and brightness. We showed that the proposed method outperforms related conventional data augmentation methods, image distortion defence methods, and image enhancer-based defence methods; our method is more consistent across different CNN object detection architectures (YOLOv4 and Faster RCNN) and datasets (PASCAL and COCO). Furthermore, unlike conventional data augmentation methods, our method enhances the accuracy of CNN models in the presence of distortions and clean images. Our method generalises to unknown distortions different from the 15 distortions used in KDE and can be applied to image classification using the SpinalNet model on the CIFAR-10, CIFAR-100, and Caltech-101 datasets.

A. Distortion-centric evaluation

To show which of the 15 distortions significantly impact the robustness of YOLOv4 on the PASCAL and COCO validation datasets, we compared the two datasets using an $IOU = 0.5$ to calculate mAP_{org} and mPC. As seen in Table 14, YOLOv4 is most vulnerable to blur distortions (such as glass, zoom, motion, and defocus) and noise (impulse, Gaussian, and shot) and more resilient to weather-related distortions like brightness, fog, frost, and snow. As shown in Table 15, our method improves the model's performance under all 15 distortions. However, it has the most significant impact on noise (by an average of +18.77%), compression (by an average of 9.35%), and blur (by an average of +6.76 %).

PASCAL ($mAP_{org} = 83.31$)		COCO ($mAP_{org} = 62.1$)	
Distortion	mPC	Distortion	mPC
glass blur	34.36 (-48.95)	zoom blur	18.43 (-43.64)
zoom blur	38.09 (-45.22)	glass blur	27.19 (-34.88)
pixelate	38.90 (-44.41)	impulse noise	29.25 (-32.82)
motion blur	41.09 (-42.22)	motion blur	31.75 (-30.32)
impulse noise	42.15 (-41.16)	pixelate	33.26 (-28.81)
Gaussian noise	48.44 (-34.87)	Gaussian noise	34.80 (-27.27)
defocus blur	48.57 (-34.74)	shot noise	35.48 (-26.59)
jpeg compression	49.96 (-33.35)	snow	36.05 (-26.02)
shot noise	51.25 (-32.06)	defocus blur	36.72 (-25.35)
elastic transform	53.03 (-30.28)	jpeg compression	37.16 (-24.91)
snow	56.76 (-26.55)	elastic transform	39.42 (-22.65)
frost	63.11 (-20.20)	frost	44.57 (-17.50)
contrast	66.92 (-16.39)	contrast	48.70 (-13.37)
fog	75.54 (-7.77)	fog	56.43 (-5.64)
brightness	78.98 (-4.33)	brightness	59.29 (-2.78)

Table 14. Susceptibility of YOLOv4 to distortions arranged from most to least using PASCAL and COCO validation set.

	YOLOv4 $mAP_{org} = 62.1$	Our $mAP_{org} = 62.1$
Distortion	mPC: YOLOv4	mPC: YOLOv4+Our
impulse noise	29.25	53.85 (+24.60)
pixelate	33.26	53.11 (+19.85)
Gaussian noise	34.80	50.73 (+15.93)
shot noise	35.48	51.25 (+15.77)
glass blur	27.19	41.04 (+13.85)
jpeg compression	37.16	50.32 (+13.16)
snow	36.05	41.08 (+5.03)
defocus blur	36.72	41.41 (+4.68)
zoom blur	18.43	22.69 (+4.26)
motion blur	31.75	35.98 (+4.23)
elastic transform	39.42	43.03 (+3.61)
frost	44.57	46.62 (+2.05)
contrast	48.70	49.48 (+0.79)
fog	56.43	56.97 (+0.54)
brightness	59.29	59.56 (+0.27)

Table 15. Gain introduced by YOLOv4+Our on type of distortion arranged from most to least using COCO.

References

[1] Tejas S Borkar and Lina J Karam. DeepCorrect: Correcting DNN Models against Image Distortions. *IEEE Trans. Image Process.*, 28:6022–6034, 2019. 1, 2

[2] Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Gridmask data augmentation. *arXiv preprint arXiv:2001.04086*, 2020. 1, 2

[3] Xiangning Chen et al. Robust and Accurate Object Detection via Adversarial Learning. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 16622–16631, 2021. 2, 5

[4] Fardad Dadboud, Patel, et al. Single-stage UAV detection and classification with YOLOv5: Mosaic data augmentation and panet. In *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–8. IEEE, 2021. 1, 2, 5

[5] Samuel Dodge and Lina Karam. Understanding how Image Quality affects Deep Neural Networks. In *International conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2016. 1

[6] Mark Everingham et al. The PASCAL Visual Object Classes (VOC) challenge. *IJCV*, 88:303–338, 2010. 3, 5

[7] Jannik Fritsch, Tobias Kuehnl, and Andreas Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *International Conference on Intelligent Transportation Systems (ITSC)*, 2013. 6

[8] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image Style Transfer using Convolutional Neural Networks. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 2414–2423, 2016. 2

[9] Robert Geirhos et al. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*. OpenReview.net, 2019. 1

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016. 4

[11] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations, ICLR*. OpenReview.net, 2015. 1

[12] Klemen Grm et al. Strengths and Weaknesses of Deep Learning Models for Face Recognition against Image Degradations. *IET Biometrics*, 7:81–89, 2017. 1

[13] K. Grm et al. Strengths and weaknesses of Deep Learning Models for Face Recognition against Image Degradations. *IET Biometrics*, 7:81–89, 2018. 1

[14] Trevor Hastie, Robert Tibshirani, and Jerome H Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, volume 2. Springer, 2009. 7

[15] Kaiming He et al. Mask R-CNN. In *Proc. IEEE Int. Conf. Computer Vision*, pages 2961–2969, 2017. 2

[16] Dan Hendrycks* et al. AugMix: A simple method to improve robustness and uncertainty under data shift. In *ICLR*, 2020. 1, 2, 5

[17] HM Dipu Kabir et al. SpinalNet: Deep neural network with gradual input. *IEEE Transactions on Artificial Intelligence*, 2022. 7

[18] Michał Koziarski and Bogusław Cyganek. Image recognition with Deep Neural Networks in Presence of Noise-Dealing with and Taking Advantage of Distortions. *Integrated Computer-Aided Engineering*, 24:337–349, 2017. 1

[19] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. 5

[20] Jin-Ha Lee et al. SmoothMix: A Simple Yet Effective Data Augmentation to Train Robust Classifiers. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 756–757, 2020. 1, 2, 5

[21] Tsung-Yi Lin et al. Microsoft COCO: Common Objects in Context. In *ECCV*, pages 740–755. Springer, 2014. 3, 5

[22] Tsung-Yi Lin et al. Focal Loss for Dense Object Detection. In *Proc. IEEE Int. Conf. Computer Vision*, pages 2980–2988, 2017. 2

[23] Ding Liu et al. When Image Denoising Meets High-Level Vision Tasks: A Deep Learning Approach. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018. 1, 2, 5

[24] Claudio Michaelis et al. Benchmarking Robustness in Object Detection: Autonomous Driving when Winter is Coming. *arXiv preprint arXiv:1907.07484*, 2019. 1, 2, 4, 5

[25] Mark Ofori-Oduro and Maria A Amer. Data Augmentation Using Artificial Immune Systems For Noise-Robust CNN Models. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*, pages 833–837, 2020. 2

[26] Zhaoqing Pan et al. DACNN: Blind Image Quality Assessment via a Distortion-Aware Convolutional Neural Network. *IEEE Trans. Circuits Syst. Video Techn.*, 32(11):7518–7531, 2022. 1

[27] Zhenyu Peng et al. Lggd+: Image retargeting quality assessment by measuring local and global geometric distortions. *IEEE-T-CSVT*, 2021. 1

[28] Shaoqing Ren et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 5

[29] Subhaluxmi Sahoo and Pradipta Kumar Nanda. Adaptive feature fusion and spatio-temporal background modeling in KDE framework for object detection and shadow removal. *IEEE Transactions on Circuits and Systems for Video Technology*, 32(3):1103–1118, 2021. 4

[30] Taeyoung Son et al. URIE: Universal Image Enhancement for Visual Recognition in the Wild. In *ECCV*, pages 749–765. Springer, 2020. 1, 2, 5

[31] Masanori Suganuma, Xing Liu, and Takayuki Okatani. Attention-Based Adaptive Selection of Operations for Image Restoration in the Presence of Unknown Combined Distortions. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 9039–9048, 2019. 1, 2, 5

[32] Zhun Sun et al. Feature Quantization for Defending against Distortion of Images. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 7957–7966, 2018. 1, 2, 5

[33] Florian Tramèr et al. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations, ICLR*. OpenReview.net, 2018. 1

[34] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling Cross Stage Partial Network. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 13029–13038, 2021. 5

[35] Fisher Yu et al. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 2636–2645, 2020. 6

[36] Sangdoo Yun et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 6023–6032, 2019. 1, 2, 5

[37] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. MixUp: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 2, 5

[38] Zhun Zhong et al. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 1, 2, 5

[39] Yu Zhou et al. Omnidirectional Image Quality Assessment by Distortion Discrimination Assisted Multi-Stream Network. *IEEE Trans. Circuits Syst. Video Techn.*, 32(4):1767–1777, 2021. 1