Repaint: Improving the Generalization of Down-Stream Visual Tasks by Generating Multiple Instances of Training Examples

Amin Banitalebi-Dehkordi
Yong Zhang
{amin.banitalebi,yong.zhang3}@huawei.com

Huawei Technologies Canada Co., Ltd.
Vancouver, Canada

Figure 1: The proposed method augments a training dataset by generating (repainting) an arbitrary number of similar instances to the training examples. Shapes are preserved but texture is diversified. In each case, the top-left corner image is the original example, and the other three demonstrate repainted versions.

Abstract

Convolutional Neural Networks (CNNs) for visual tasks are believed to learn both the low-level textures and high-level object attributes, throughout the network depth. This paper further investigates the 'texture bias' in CNNs. To this end, we regenerate multiple instances of training examples from each original image, through a process we call '*repainting*'. The repainted examples preserve the shape and structure of the regions and objects within the scenes, but diversify their texture and color. Our method can regenerate a same image at different daylight, season, or weather conditions, can have colorization or de-colorization effects, or even bring back some texture information from blacked-out areas. The in-place repaint allows us to further use these repainted examples for improving the generalization of CNNs. Through an extensive set of experiments, we demonstrate the usefulness of the repainted examples in training, for the tasks of image classification (ImageNet) and object detection (COCO), over several state-of-the-art network architectures at different capacities, and across different data availability regimes. Code is released as supplementary [9].

1 Introduction

Overfitting is a fundamental problem in training deep neural networks (DNNs) [8]. To overcome the overfitting, there has been a tremendous amount of research which led to many

successful approaches including: data augmentation, designing efficient architectures, regularization, dropouts [15, 36], early stopping, ensembling, etc. Collectively, these techniques have resulted in achieving a remarkable performance across many applications. That being said, the underlying cause of this problem has also been of interest for a long time. In this paper, we look at this phenomenon through the lens of texture and shape bias [14].

In the case of visual tasks such as object recognition or detection, a common intuition is that deep models such as DCNNs (Deep Convolutional Neural Networks) learn both low-level image features such as edges or texture patterns (within the earlier layers) as well as high-level attributes such as presence and shape of objects (in the deeper layers) [14, 22, 25]. Some works argue (and sometimes provide empirical results) that like in humans, shape is the single most important factor in CNNs for learning visual tasks [23, 33]. Others argue otherwise, that texture has a more significant role in CNNs [1, 9, 12, 13]. Authors in [14] designed a comprehensive study (texture-shape conflict stimuli) to understand this phenomenon. They concluded that CNNs are generally biased towards easier-to-learn texture features (shortcuts) at the expense of shape attributes (texture bias). [14] further proposed Shape-ResNet, in which they trained a ResNet model with stylized images and were subsequently able to improve the generalization and robustness of the network.

Other than the style transfer method used in [14], there has been a family of style transfer algorithms employed for different applications [8, 20, 27, 47]. However, these approaches generally produce artistic effects on images and diverge from natural-looking images. Moreover, the stylized transfer used in [14] is not trained to minimize a down-stream task loss, but rather is an off-the-shelf one [19].

In this paper, we propose a method to augment the training set, by generating multiple instances from each training example. To this end, we make use of a generative semantic synthesis model to generate new instances (in a variational manner), and tie this model to a down-stream task. In other words, we generate examples that adhere to the objects shapes of the original image while modifying the texture in a way that helps the down-stream task (e.g. image classification or object detection). Therefore, our method 'repaints' the original images, by changing their texture/color but preserving the shapes and locations of objects. In-place repaint makes it suitable for non-classification down-stream tasks such as object detection. Figure 1 demonstrates example images resulted from our method. We verify our approach with an extensive set of experiments for the tasks of image classification and object detection, on several network architectures and dataset sizes.

The main contributions of this paper can be summarized as:

- We propose a method of augmenting training datasets by repainting the examples. Repainted examples are diverse in texture and color in that they substitute regions, objects, or backgrounds with randomly drawn new instances learned from the dataset. Sometimes this results in interesting outcomes such as adding/removing colors, uncovering new information in blacked-out areas, or shifting day/night time or seasons.
- We utilize the repainted training examples to improve the generalization of CNNs. Due to the nature of this method, it can be applied to various visual tasks. We demonstrate results on image classification and object detection as two common use-cases.
- We present a comprehensive set of experiments over several state-of-the-art network architectures at different capacities, and across different data availability regimes. Results show a consistent improvement in the generalization of the CNNs.

2 Related works

In this section, we review the related areas to our work, and draw connections between them.

Image generation: A number of works such as [3] explore directly augmenting the training data using generative adversarial networks (GANs). These methods train an off-the-shelf GAN with the available images, and later use it to generate more. In this setting, no strong supervision between the GAN and the down-stream task is enforced, and these methods are more useful for situations like medical imaging tasks where data itself is scarce. Other works including [49] propose class-aware conditioned GANs to balance a dataset for an improved generalization. Our method conditions the generation to be shape preserving while also assisting a general-purpose down-stream task.

Semantic image synthesis: These methods generate synthetic images given semantic segmentation masks [30, 31]. The idea is to design a GAN (Generative Adversarial Network [16]) to generate images that can adhere to semantics. In our method, we make use of the spatially adaptive normalization [31] in order to preserve the shape structure of objects.

Stylization: It was argued in [14] that ImageNet CNNs are biased towards texture features since they are easier to learn than shape attributes. This shortcut then resulted in a less accurate generalization to unseen test data. To address this issue, authors in [14] proposed Shape-ResNet, in which multiple stylized versions of each training image are generated and used for training a CNN (e.g. ResNet). The model is then fine-tuned with the original train set. Since this method is based on style transfer, it preserves the shapes and structures but applies texture modification according to another image's style. In a way, it is also increasing the train set size by augmenting it with the stylized examples. Our findings are in agreement with the observations of [14] in that reducing the texture bias can improve the generalization. However, since the stylized examples generated for Shape-ResNet do not look like typical natural images found in standard datasets such as ImageNet [9] or Microsoft COCO [26], this can reduce the potential gains (as we see in Section 4). In addition, the style transfer step is detached from the down-stream task (image classification in case of [14]), and thus provides no guarantee that the stylized examples can confidently boost the generalization. Nonetheless, there are a variety of style transfer methods proposed in the literature [8, 20, 27, 47] which may be used similarly.

Learning to modify input examples: Related to our approach is a line work where the input training images are updated according to some loss term that is related to a down-stream task [37, 46, 50]. For example, authors in [37] propose to learn to resize input data, in such a way that can help with down-stream tasks of image classification or quality assessment. Our method is in some sense similar since we also learn to update the input data, however, it is different in the sense that we keep the original image size but instead learn to regenerate and replace (repaint) objects in the image. Therefore, the two methods are orthogonal and can be combined with each other.

Image augmentation: Image augmentation has a rich literature. Traditionally, global image-scale operations such as rotate, flip, blur, contrast stretch, etc. were used within augmentation pipelines. Over the past several years, many new augmentation techniques were proposed. These techniques include MixUp [18], CutOut [10], CutMix [48], AutoAugment [7], Thumbnail [45], ClassMix [29], etc. Our method is orthogonal to these kinds of image augmentations and in fact these augmentations can be applied on top of our method. We provide some results in this regard in Section 4.

3 The proposed repainting method

In this section we first introduce some basic setup, then explain our solution followed by several remarks and discussions.

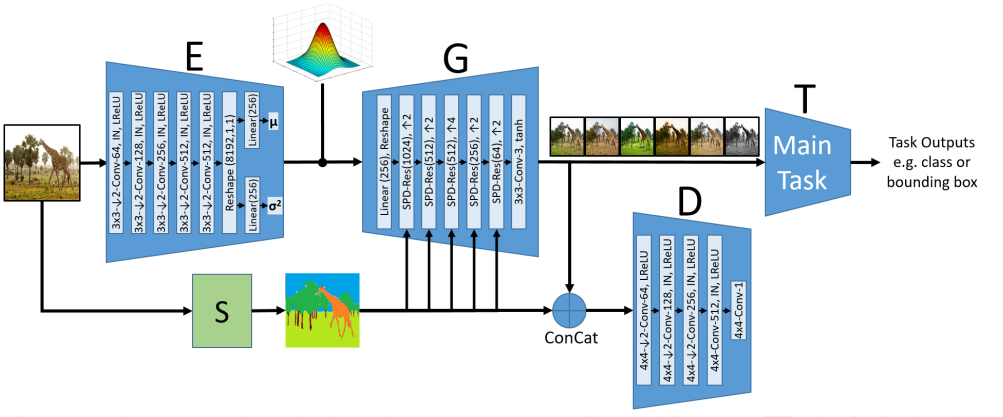

Figure 2: The proposed method employs a VAE-GAN-like architecture [24] that is tied to the down-stream task. E, G, D, and T denote the encoder, generator, discriminator, and the main task. S is a module that generates approximate segmentation masks, such as a DeepLab model [6], or the image-based Felzenszwalb-Huttenlocher (FH) [11] algorithm. There is no restriction on what the down-stream task can be, except that its input is an image. We examine image classification and object detection tasks in this paper.

3.1 Some basic setup

Let E, D, G, and T denote the encoder, discriminator, generator, and down-stream task in our setup, respectively. Also, let S denote a module that generates some form of semantic segmentation mask (details in the next subsection). We incorporate the aforementioned modules in our design, as shown in Figure 2.

Moreover, we make use of the SPatially Adaptive DEnormalization (SPADE) modules introduced in [31]. The SPADE module and its corresponding residual block denoted by SPD-Res enforce the consistency of shapes and structures, and can be formulated as:

$$f_{out}{}^{i}_{c} = \gamma^{i}_{c} \frac{f_{in}{}^{i}_{c} - \mu^{i}_{c}}{\sigma^{i}_{c}} + \beta^{i}_{c}, \tag{1}$$

where f_{in} and f_{out} are the input and output feature maps of shape $B \times N_c \times H \times W$, B is the mini-batch size, N_c is the number of channels, H and W denote the height and width of the activations tensor in (1), μ^{i}_{c} and σ^{i}_{c} are mean and standard deviations of input features, i and c denote the layer and channel indices, and γ and β are learned scale and bias modulation tensors (with spatial dimensions) that are multiplied and added element-wise to output of a sync-BN layer to create the output features. It can be observed in (1) that this kind of normalization is in some ways similar to regular batch normalization, but it has spatial dimensions that are learned, which in turn helps with enforcing shapes and structures.

Figure 3 illustrates the inner architecture of SPADE and its residual block. Note that γ and β in (1) and Figure 3, for each layer, are of shape $N_c \times H \times W$. Unlike the standard BatchNorm scale and bias parameters they depend on the spatial mask values. In some sense, (1) is like applying segmentations to the activation maps, thereby conditioning the shapes and structures.

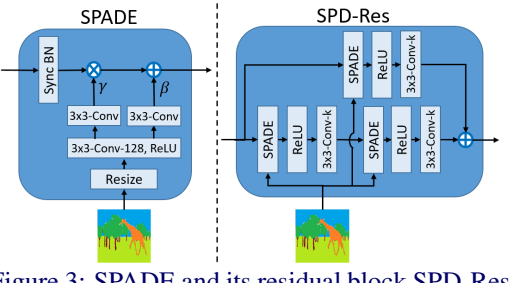

Figure 3: SPADE and its residual block SPD-Res.

3.2 Architecture design and loss

Figure 2 shows the flow-diagram of our method. As observed from this figure, there is a similarity (in terms of the overall architecture) to traditional image generation or semantic image synthesis architectures. The encoder and generator together generate batches of repainted images. The discriminator is responsible for pushing the generated examples to look 'real'. Module S outputs segmentation masks from the input images. Note that the segmentation masks from this module don't necessarily have to be very accurate. In our experiments in Section 4, we provide results based on using masks generated by a DeepLab-v2 [6] model, as well as masks generated by a completely unsupervised image-based operator of [11], and show that in both cases we can achieve generalization gains (the visualizations of repainted images are based on using the DeepLab-v2 masks). Finally, the T module refers to the task network. For a classification task, it could be any classification CNN e.g. ResNet [17] or EfficientNet [38] with a softmax layer at the end. Or, for an object detection task, it could be any detection network such as a YOLO [32] or EfficientDet [39] model. During training, each image is seen once per epoch, but is repainted slightly differently every time.

The training objective and loss terms follow those of [31] (and thus also pix2pixHD [40]), however we add a new loss term for training the down-stream task. The overall objective therefore contains three terms: $\mathcal{L}_{Generator-Encoder}$ to account for the encoder and generator losses, $\mathcal{L}_{Discriminator}$ to compute the discriminator loss, and finally $\mathcal{L}_{Task}$ to denote the task loss. The $\mathcal{L}_{Generator-Encoder}$ itself accounts for the generator loss ($\mathcal{L}_{Generator}$), feature matching loss ($\mathcal{L}_{Feat.}$ used in [40]), and a KLD loss to account for the variational sampling of the encoder's output ($\mathcal{L}_{KLD}$). (2) and (3) summarize the above. We refer the readers to [40] for further details on the generator and discriminator loss terms.

$$\mathcal{L} = \mathcal{L}_G + \mathcal{L}_D + \mathcal{L}_{Task} = \mathcal{L}_{GAN} + \mathcal{L}_{Feat.} + \mathcal{L}_{KLD} + \mathcal{L}_{Task}. \tag{2}$$

$$\mathcal{L}_{GAN}(G,D) = \mathbb{E}[logD(.)] + \mathbb{E}[log(1 - D(G(.)))]. \tag{3}$$

The task loss $\mathcal{L}_{Task}$ accounts for task-specific losses. For classification task it will be a cross-entropy loss, and for object detection it will be a detection loss according to specialized detection architectures (usually a regression loss to account for bounding boxes, a cross-entropy loss for objects category assignment, and a confidence score loss).

Training is done iteratively similar to GANs [16], however, we train in iterations for the terms in (2). To this end, on one iteration the model is run through, real images are run through D, and then D and T are updated using the discriminator loss and task loss. In the other iteration the model is run through, and E and G are updated based on the generator loss and task loss. As a result, the task loss would also be supervising E and G. We observed in our experiments that the best results were achieved when optimizing two iterations of discriminator and down-stream task, and one iteration for the generator. More details are given in Section 4.

3.3 Remarks and discussions

Here we discuss some remarks about our approach.

Consistent shapes, diversified texture: The variational sampling from the encoder's latent space and the SPADE blocks together result in images generated with a similar structure and objects shapes to the input, but with a new texture and color style that is randomly sampled from what the the model has learned from the texture and colors of previous examples. It is therefore like replacing/repainting each object/region with a new instance learned from the same distribution. Due to the texture bias phenomenon [14], this can improve the network's generalization. Figure 1 and 4 show examples of the generator's output.

Figure 4: Repaint examples. The top row shows the original samples, and the bottom row shows the repainted versions. Notice the texture changes such as: tower lights, boat reflection and design, beach waves pattern, animal bodies or land coverage, and mountain snow.

Colorization, time shift, uncovering new texture, etc.: We observe an interesting effect in the repainted images where sometimes they demonstrate effects such as: colorization, decolorization, changing the time of the day or seasons, or even bringing back textures which were blacked out in the original images. Figure 5 demonstrates examples of such effects.

Reuse of bounding boxes: Since the labels for the down-stream tasks (e.g. classes or bounding boxes) do not change, a repainted image may contain substitute instances of the same object categories present in the scene. The in-place repaint allows to perform tasks such as object detection since the locations of ground truth bounding boxes do not change. Hence, the same set of ground truth labels can be used for the augmented images.

Image generation: The goal of our method is not necessarily to generate visually pleasant or normal-looking images, but rather is to generate images that are suitable for the down-stream task. That being said, by including a discriminator loss, and carefully balancing the iterative training of the generator, discriminator, and task modules, we achieve an acceptable look on the generated examples.

Architecture: The general architecture and layers of the E, D, and G blocks are inspired by [31, 40], but customized for our purpose to be attached to down-stream tasks such as classification or detection. In particular, we have designed these blocks to be somewhat light-weight, as the main task can have a large burden on the GPU memory during training. For example, in Section 4 we tested our method with down-stream task of EfficientNet-D3 object detection that has 25B FLOPs (roughly 64× more operations and 2× more parameters than EfficientNet-B0 classification model). That being said, the GAN modules can be replaced with different architectures used in the generative models literature, as long as they enforce the shape consistency like we do.

Orthogonality with augmentations or regularizations: When training the main task, repainting is orthogonal to other kinds of image augmentations, and thus they can be applied at the same time. In fact, we show in Section 4 that an improved generalization can be achieved by applying augmentations such as CutMix [48] or regularizations such as DropBlock [15] together with repainting.

Limitations: An observation we made during our experiments with natural image datasets such as ImageNet and COCO is that our method is very good at repainting scenes in general, however, it sometimes has a hard time with finer details such as facial features. Examples of such failure cases are provided in the supplementary materials [2]. It is also worth noting that specialized GANs such as the ones used for faces, are trained with face datasets, whereas

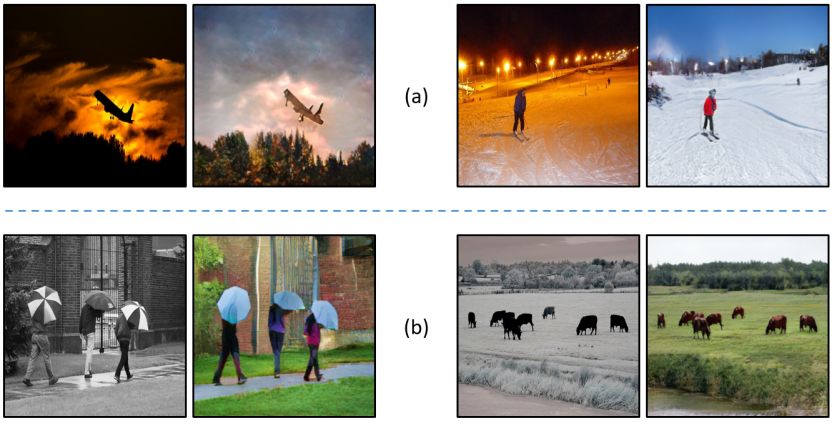

Figure 5: Examples of repainted images where new information is uncovered. a: uncovering from the dark things such as trees texture, mountain trees, or a building afar. b: colorization. For each pair, the left-side image is the original, and the right-side one is a repaint.

we used general purpose datasets such as ImageNet or COCO that contain a wide range of scenes and objects, and may not be very suitable for specialized tasks. That being said, we expect the image generation to perform well when trained on specialized and controlled datasets. Moreover, the goal is not to necessarily generate good looking images, but it is to generate images that help the down-stream task (reflected in the task performance results). In addition, as mentioned in Section 4.2, after training with repainted images for a while, at the end we fine-tune with the original dataset. This ensures the task performance will be protected from infrequent shortfalls of image generation.

4 Experiments

We discuss the experiment results in this section. To this end, we first explain the datasets and metrics used, followed by the training details and baselines. Then we discuss our results and ablation studies.

4.1 Datasets and metrics

Our experiments include two down-stream tasks of image classification and object detection. For classification, we use the ImageNet dataset [9] with 1.28M training and 50K validation examples. The main metric of performance is top-1 or top-5 classification accuracy (%).

For object detection, we use the Microsoft COCO dataset [26] with 118K training and 5K validation examples. Methods are assessed based on mean Average Precision (mAP) metric either at a certain IoU (Intersection over Union) threshold such as 0.5, or averaged over various IoUs e.g. @0.5:0.95.

4.2 Training details

Baselines include training various architecture at different model capacities. For classification, we use MobileNet-v2 [35], ResNet50 [17], and the EfficinetNet [38] family of B0, B1, B2, and B3. For object detection, we include EfficientDet-D0, D1, D2, and D3 [39]. Implementations were in PyTorch v1.6 and included customized code from the following repositories: timm [43] commit 532e3b4, efficientdet [44] commit 1c9a3d3, and SPADE [28] commit 1a687ba.

There are a large number of training experiments done for the two tasks of classification and detection which increases diversity in the training procedures, but in general we followed a 200 epoch training strategy. We used the learning rates of $1e^{-4}$ and $4e^{-4}$ for the generator and discriminator, respectively, with an Adam optimizer [21] with $\beta_1 = 0$ and $\beta_2 = 0.999$. In addition, for the down-stream task of classification, we used a learning rate of 0.12, with decays of 90% every 3 epochs, and a rmsprop optimizer [34] with warm-up. Similarly for the task of object detection, we used a learning rate of 0.06, rmsprop with cosine decay rule, and warmup. Baseline models were trained for 200 epochs. For our method, we first trained for 150 epochs, and then performed a 50 epoch fine-tuning of only the task part of the model with the original dataset. Note that longer training may result in slightly better performance. In fact, some state-of-the-art (SOTA) ImageNet models are trained for 500 epochs. That being said, our comparisons are fair and produce accuracies close to those of SOTA models.

Also note that repainting happens on the fly for each training example. In that sense, all the training images are used in each epoch, and each time a random repaint is applied.

Moreover, each training job was run on a 8-GPU node with V100 GPUs of 32GB memory, and was repeated 5 times to ensure the consistency of the results.

It is also worth noting that in our experiments we never used any ground truth semantic segmentation masks. We used approximate masks generated by a DeepLab-v2 model [6], as well as rough masks generated by the Felzenszwalb-Huttenlocher (FH) [11] method. The FH algorithm is a classical image-based method and is unsupervised in nature.

4.3 Main results

Table 1 shows the results of image classification experiments. It is observed from Table 1 that, by increasing the diversity of examples during training, repainting consistently improves the classification accuracy across several models with different capacities. In Table 3, we compare the results of our method with another image-generation based method, ShapeResNet [12], as well as several augmentation-based and regularization methods. Note that in general our method is orthogonal to augmentation-based or regularization-based approaches, and therefore its performance is expected to improve when combined with such approaches. This is consistent with our observations in Table 3.

It is also worth noting that applying Repaint without the task-specific loss (separate optimizations, i.e. a trained/frozen generator) can still lead to improvements over the baseline as it increases data diversity. However, the incorporation of the task loss can further boost the performance since it also encourages the generation to assist with the down-stream task.

Furthermore, as mentioned in Section 3.3 the proposed method does not rely on a spe-

Architecture	# params	FLOPs	Baseline	Repaint w/o task-loss	Ours (Repaint)
MobileNetv2	3.4M	0.3B	74.63	74.94	75.15 (+0.52)
ResNet-50	26M	4.1B	77.06	77.64	78.05 (+0.99)
EffNet-B0	5.3M	0.39B	76.66	77	77.24 (+0.58)
EffNet-B1	7.8M	0.70B	78.59	78.92	79.15 (+0.56)
EffNet-B2	9.2M	1.0B	79.24	79.61	79.83 (+0.60)
EffNet-B3	12M	1.8B	80.87	81.2	81.45 (+0.58)

Table 1: Top-1 accuracy for image classification on ImageNet.

Architecture	# params	FLOPs	Baseline	Repaint w/o task-loss	Ours (Repaint)
EffDet-D0	3.9M	2.5B	33.87	34.36	35.10 (+1.23)
EffDet-D1	6.6M	6B	38.98	39.38	39.97 (+0.99)
EffDet-D2	8.1M	11B	42.25	42.69	43.35 (+1.10)
EffDet-D3	12.0M	25B	45.27	45.98	46.86 (+1.59)

Table 2: mAP performance for object detection on COCO.

Method	Strategy	Top-1 (%)
Baseline		77.06
MixUp [43]	Augmentation	77.9
CutOut [10]	Augmentation	77.1
CutMix [43]	Augmentation	78.6
AutoAugment [6]	Augmentation	77.6
DropBlock [15]	Regularization	78.1
ISDA [41, 40]	Latent Augmentation	78.1
Shape-ResNet [14] (rerun)	Image Generation	77.42
[31] + task-specific loss	Image Generation	78.02
Ours (Repaint)	Image Generation	78.05
CutMix+Repaint	Combo	79.11
CutMix+DropBlock+Repaint	Combo	79.19

Table 3: A comparison of ImageNet top-1 classification accuracy on ResNet-50. Repaint combined with other augmentations or regularizations can yield a high performance.

cific type of generative network. Different architectures employed in the generative models' literature can also be used, as long as they enforce the shape consistency. One such example is [31]. That being said, the original architecture of [31] is relatively large; when attached to a large task network such as a large detection model, it will consume a large amount of GPU memory. This enforces a very small batch size which makes the training on large datasets such as COCO or ImageNet impractical. For the sake of comparisons however, we added results of [31] plus task-loss on ResNet50 to Table 3. We observe that this benchmark achieves a comparable top-1 accuracy to Repaint, suggesting that the extra generation capacity did not necessarily directly translate to a considerably better down-stream top-1.

Table 2 shows the results of object detection experiments. We observe from Table 2 that object detection training also benefits from repainting by a considerable margin.

In conclusion, the results obtained by our main experiments are inline with the observations of [14] in that texture bias has an important role in the training of CNN models, and texture diversification leads to generalization improvements.

4.4 Ablation studies

Next, we perform ablation studies on the proposed method. We first investigate the effect of segmentation mask quality. To this end, we try DeepLab-v2 and FH masks to consider both high and low quality masks. Table 4 and Table 5 show the gains over baselines achieved by our method for classification and detection across many models, over 5 runs. Note that in Table 4 and 5, the min-max intervals from the 5 runs are computed by measuring the gap between the best/worst baseline runs and the worst/best repaint runs. We observe from these tables that improvements are consistent, although slightly lower for the FH masks.

In another ablation study, we investigate the impact of labels availability in lower data regimes. To this end, we report results when 1% or 10% of training data is used. As observed in Table 6 and 7, higher gains are achieved when lower portions of data are used. This is somewhat expected since ImageNet and COCO datasets contain a large number of examples, and achieving better generalization with 100% of examples is therefore more difficult.

All-in-all, repainting shows a robust and consistent improvement in the generalization capability of image classification and object detection tasks, and therefore can be considered as an add-on orthogonal candidate for inclusion in existing training pipelines.

Computational complexity Compared to training only a task network (e.g. a classification or detection network), our method requires an additional generative module to repaint the images. This can be thought of as a learned augmentation, and thus incurs an extra overhead. Since the GAN part is designed to be relatively light-weight, and the segmentation

Architecture	Gains with DeepLab-v2 (min,max) over 5 runs	Gains with FH [] (min,max) over 5 runs
MobileNetv2	+0.52 (0.38,0.69)	+0.44 (0.28,0.60)
ResNet-50	+0.99 (0.71,1.28)	+0.79 (0.51,1.03)
EffNet-B0	+0.58 (0.41,0.75)	+0.51 (0.34,0.68)
EffNet-B1	+0.56 (0.39,0.72)	+0.48 (0.31,0.65)
EffNet-B2	+0.60 (0.42,0.77)	+0.50 (0.33,0.67)
EffNet-B3	+0.58 (0.40,0.76)	+0.49 (0.29,0.66)
Average	+0.77 (0.54,0.99)	+0.64 (0.41,0.85)

Table 4: Ablation on magnitude and consistency of performance improvements across weakly-supervised or unsupervised masks; Results on ImageNet top-1 classification accuracy (%).

Architecture	Gains with DeepLab-v2 (min,max) over 5 runs	Gains with FH [] (min,max) over 5 runs
EffDet-D0	+1.23 (1.01,1.46)	+1.02 (0.83,1.21)
EffDet-D1	+0.99 (0.84,1.14)	+0.78 (0.60,0.96)
EffDet-D2	+1.10 (0.93,1.29)	+0.89 (0.71, 1.07)
EffDet-D3	+1.59 (1.31,1.87)	+1.17 (0.88,1.46)
Average	+1.22 (1.17,1.44)	+0.96 (0.75,1.18)

Table 5: Ablation on magnitude and consistency of performance improvements across weakly-supervised or unsupervised masks; Results on COCO object detection accuracy (mAP %).

Method	Top-1			Top-5		
	1 %	10 %	100 %	1 %	10 %	100 %
Baseline	14.63	57.13	77.06	31.61	79.60	93.57
Ours (Repaint)	18.72	59.55	78.05	35.55	81.13	93.97
Gains	+4.09	+2.42	+0.99	+3.94	+1.53	+0.40

Table 6: ResNet-50 classification accuracy vs portion of ImageNet dataset used for training.

Method	mAP @0.50:0.95			AP @0.50		
	1 %	10 %	100 %	1 %	10 %	100 %
Baseline	6.21	22.44	33.87	23.82	39.06	52.36
Ours (Repaint)	10.35	24.73	35.10	27.95	41.58	53.71
Gains	+4.14	+2.29	+1.23	+4.13	+2.52	+1.35

Table 7: EfficientDet-D0 object detection accuracy vs portion of COCO dataset used for training.

piece is not being trained, the overall computational complexity is still comfortably manageable. On average, the training time of the image classification and object detection tasks observed an increase of $\approx 66\%$ and $\approx 53\%$, respectively. The overhead can be reduced by further compressing the generative network module, by offline training and freezing it, or by applying the repainting only on a percentage of the training examples.

5 Conclusion

In this paper, we proposed a method of augmenting a training dataset by variationally repainting the training images. The images generated by our method were diverse in texture and color but all preserved the original shape and structure. We then leveraged the augmented dataset to train models with improved generalization on test data. We demonstrated the performance of our method on the tasks of image classification (ImageNet) and object detection (COCO), over several state-of-the-art network architectures at different capacities, and across different data availability regimes. We hope our work can help facilitate further research in this direction.

References

[1] Pedro Ballester and Ricardo Araujo. On the performance of googlenet and alexnet applied to sketches. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[2] Amin Banitalebi-Dehkordi and Yong Zhang. Repaint: Improving the generalization of downstream visual tasks by generating multiple instances of training examples. In *32nd British Machine Vision Association annual conference on machine vision, image processing, and pattern recognition, BMVC*, 2021. Supplied as additional material. `0068supp.pdf`.

[3] Christopher Bowles, Liang Chen, Ricardo Guerrero, et al. Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

[4] Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

[5] Rich Caruana, Steve Lawrence, and Lee Giles. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. *Advances in neural information processing systems*, 2001.

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, et al. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 113–123, 2019.

[8] Brian Davis, Chris Tensmeyer, Brian Price, et al. Text and style conditioned gan for generation of offline handwriting lines. In *30th British Machine Vision Association annual conference on machine vision, image processing, and pattern recognition, BMVC 2020*, 2020.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition, CVPR*, pages 248–255, 2009.

[10] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[11] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International journal of computer vision*, 59(2):167–181, 2004.

[12] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. *arXiv preprint arXiv:1505.07376*, 2015.

[13] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Texture and art with deep neural networks. *Current opinion in neurobiology*, 46:178–186, 2017.

[14] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, et al. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[15] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: a regularization method for convolutional networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 10750–10760, 2018.

[16] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[18] Yann N. Dauphin David Lopez-Paz Hongyi Zhang, Moustapha Cisse. mixup: Beyond empirical risk minimization. *International Conference on Learning Representations*, 2018.

[19] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017.

[20] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4369–4376, 2020.

[21] Diederik P Kingma and Jimmy Ba. Adam: A methodfor stochastic optimization. In *International Conference onLearning Representations (ICLR)*, 2015.

[22] Nikolaus Kriegeskorte. Deep neural networks: A new framework for modeling biological vision and brain information processing. *Annual Review of Vision Science*, 1(1):417–446, 2015.

[23] Jonas Kubilius, Stefania Bracci, and Hans P Op de Beeck. Deep neural networks as a computational model for human shape sensitivity. *PLoS computational biology*, 12(4):e1004896, 2016.

[24] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566. PMLR, 2016.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553), 2015.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, et al. Microsoft coco: Common objects in context. In *European conference on computer vision, ECCV*, pages 740–755. Springer, 2014.

[27] Sanghyeon Na, Seungjoo Yoo, and Jaegul Choo. Multimodal image translation with stochastic style representations and mutual information loss. In *30th British Machine Vision Association annual conference on machine vision, image processing, and pattern recognition, BMVC*, 2020.

[28] NVlabs. Semantic image synthesis with SPADE. *https://github.com/NVlabs/SPADE*.

[29] Viktor Olsson, Wilhelm Tranheden, Juliano Pinto, and Lennart Svensson. Classmix: Segmentation-based data augmentation for semi-supervised learning. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[30] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!* 2019.

[31] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[33] Samuel Ritter, David GT Barrett, Adam Santoro, and Matt M Botvinick. Cognitive psychology for deep neural networks: A shape bias case study. In *International conference on machine learning*, pages 2940–2949. PMLR, 2017.

[34] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR*, pages 4510–4520, 2018.

[36] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[37] Hossein Talebi and Peyman Milanfar. Learning to resize images for computer vision tasks. *arXiv preprint arXiv:2103.09950*, 2021.

[38] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[39] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[40] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[41] Yulin Wang, Xuran Pan, Shiji Song, et al. Implicit semantic data augmentation for deep networks. *Advances in Neural Information Processing Systems*, 32:12635–12644, 2019.

[42] Yulin Wang, Gao Huang, Shiji Song, et al. Regularizing deep networks with semantic data augmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[43] Ross Wightman. PyTorch image models. *https://github.com/rwightman/pytorch-image-models*, .

[44] Ross Wightman. EfficientDet repository. *https://github.com/rwightman/efficientdet-pytorch*, .

[45] Tianshu Xie, Xuan Cheng, Minghui Liu, et al. Thumbnail: A novel data augmentation for convolutional neural network. *arXiv preprint arXiv:2103.05342*, 2021.

[46] Shaokai Ye, Kailu Wu, Mu Zhou, et al. Light-weight calibrator: a separable component for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13736–13745, 2020.

[47] Jonghwa Yim, Jisung Yoo, Won-joon Do, Beomsu Kim, and Jihwan Choe. Filter style transfer between photos. In *European Conference on Computer Vision*, pages 103–119. Springer, 2020.

[48] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, et al. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[49] Xinyue Zhu, Yifan Liu, Jiahong Li, Tao Wan, and Zengchang Qin. Emotion classification with data augmentation using generative adversarial networks. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 349–360. Springer, 2018.

[50] Yezi Zhu, Marc Aoun, Marcel Krijn, Joaquin Vanschoren, and High Tech Campus. Data augmentation using conditional generative adversarial networks for leaf counting in arabidopsis plants. In *BMVC*, page 324, 2018.