A Dataset

A.1 The Surface Defect Inspection Dataset

The Surface Defect Inspection (SDI) dataset consists of 20,414 images at various resolutions. It contains three kinds of products—product A, product B and product C. Each can be further classified into three foreground defect domains—`Normal`, `Scratches` and `Spots`. Fig. 6 shows example images of the SDI dataset. It is worth mentioning that the dataset is highly imbalanced not only between normal and defective samples but also between different products as shown in Table 6. This sets a more challenging task when training deep neural networks like GANs and downstream classifiers.

For each foreground defect and background product, we randomly select 50 images, 30% of which are used for validation and 70% as the test set. All remaining images are used as training sets for GAN and classifier training. We present the distribution of the training set when training DT-GAN in Table 7. Note that the normal samples used in GAN training are only a subset of all available samples in `Normal` and we keep the rest of them for generating defective samples at test time. For classifier training, we show the statistics in Table 8, where the number of normal samples involved in classifier training increases incrementally. The validation set is used to select the best model during classifier training while the test set is left untouched until the final evaluation. Both the validation and test set are inaccessible by DT-GAN.

A.2 The MVTec Anomaly Detection Dataset

The MVTec Anomaly Detection (MVTec AD) dataset [1] contains 15 different object and texture categories for anomaly detection. The dataset is formed of non-defective images for training and both non-defective and defective images with various kinds of defects for testing. The pixel-level annotations of all defective images are also provided. It is worth noting that the MVTec AD dataset is relatively small in terms of the number of images, where the number of training images is ranging from 60 to 391. Moreover, the number of defective images for each defect category in the test set is varying only from 8 to 30, which is relatively limited considering the sophisticated pattern of defects.

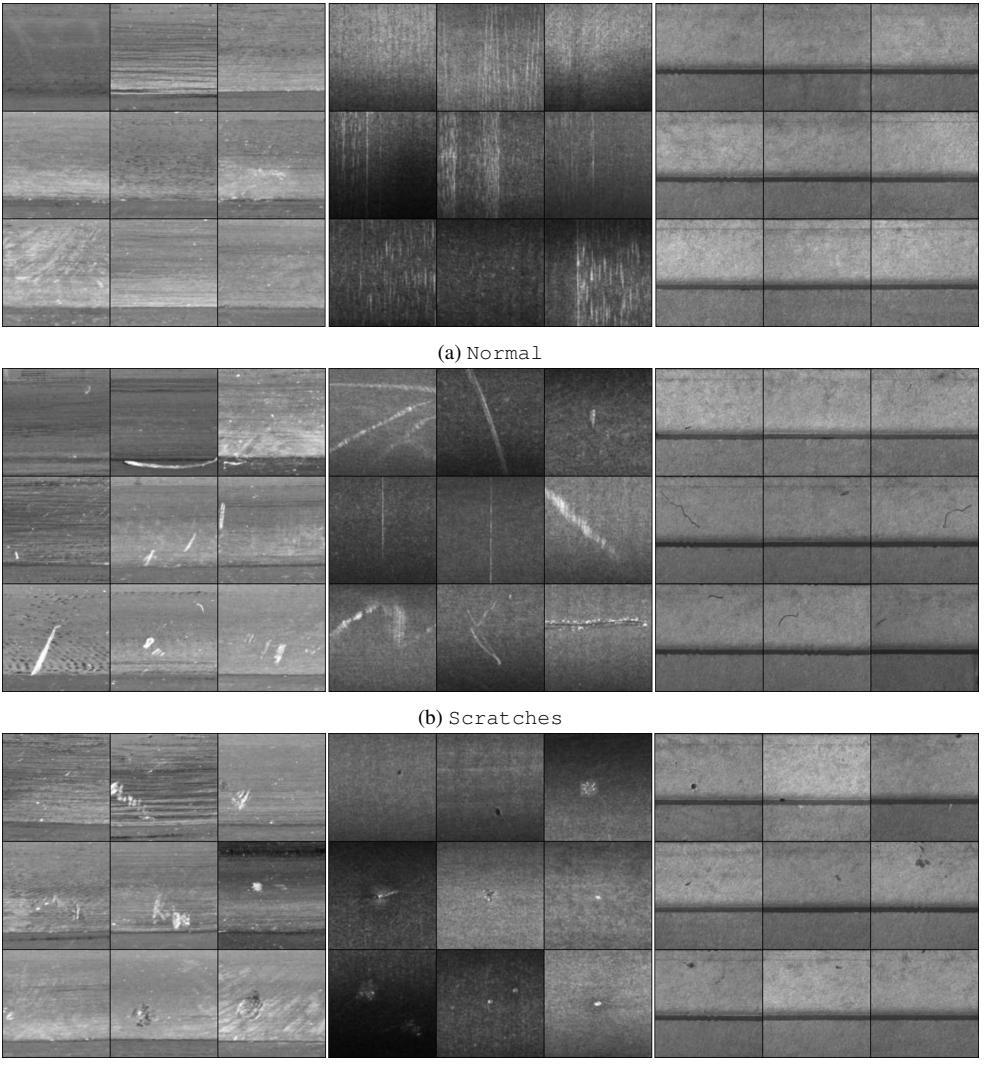
We conducted image synthesis experiments on a subset of the MVTec AD dataset, where we selected four texture categories: CARPET, LEATHER, WOOD and TILE for our targeted scenario, i.e. surface defects. Furthermore, we aggregated some of the original defect types defined in the MVTec AD dataset into `Scratches`-like and `Spots`-like defects according to their visual appearance. Details of the resulting dataset are shown in Table 9.

Table 6: Distribution of the full SDI dataset.

	Overview		
	A	**B**	**C**
`Normal`	6,250	6,250	6,250
`Scratches`	340	167	121
`Spots`	108	670	258

Table 7: The training set for DT-GAN and the baseline image synthesis methods.

	Overview		
	A	**B**	**C**
`Normal`	700	700	700
`Scratches`	290	117	71
`Spots`	58	620	208



(a) Normal

(b) Scratches

(c) Spots

Figure 6: Overview of the SDI dataset. (Best viewed in color.)

Table 8: The training, validation, and test set for classifier training, where N increases incrementally—1,500, 2,200, 4,000, and 6,200.

	Train			Validation			Test		
	A	**B**	**C**	**A**	**B**	**C**	**A**	**B**	**C**
Normal	N	N	N	12	18	15	38	32	35
Scratches	290	117	71	14	16	15	36	34	35
Spots	58	620	208	14	16	15	36	34	35

Table 9: Overview of our formation of the MVTec AD sub-dataset. The first column represents the original defect types in the MVTec AD dataset while the first row stands for the defect types in our targeted scenario. We list the ID of samples we took from the MVTec AD dataset and show the number of samples in row **Sum**.

(a) CARPET

	Scratches	Spots
Color	011, 012, 014, 016, 017	000, 003, 004, 007, 015, 018
Thread	000-018	-
Hole	-	000-016
Sum	24	23

(b) LEATHER

	Scratches	Spots
Color	001, 003, 005, 007, 009, 011, 013, 015, 018	000, 002, 006, 008, 010, 012, 014
Cut	000-018	-
Fold	000-006, 009-016	-
Glue	003, 009, 010, 016, 017	000-002, 005-009, 011-015, 018
Poke	-	000-017
Sum	48	39

(c) TILE

	Scratches	Spots
Crack	000-016	-
Sum	17	0

(d) WOOD

	Scratches	Spots
Color	003, 005	-
Scratch	001-006, 008-010, 013-016, 018-020	000-016
Hole	-	000-004, 006-009
Combined	008	001, 002, 009
Sum	19	12

A.3 The Magnetic Tile Defects Dataset

The Magnetic Tile Defects (MTD) dataset contains 925 defect-free and 392 anomalous magnetic tile images from five different defect classes—CRACK, BLOWHOLE, BREAK, FRAY and UNEVEN. All of the images are gray-scale with varied illumination levels and image sizes. Among them, we selected CRACK as the `Scratches`-like defect and BLOWHOLE as the `Spots`-like defect for our demonstration.

B Training Details

B.1 DT-GAN

Full Objective. Our full objective functions can be summarized as

$$\begin{aligned}\min_{G,M,E}\max_{D} \quad & \lambda_{\text{adv}} \cdot \mathcal{L}_{\text{adv}} + \lambda_{\text{sd_rec}} \cdot \mathcal{L}_{\text{sd_rec}} + \lambda_{\text{d_rec}} \cdot \mathcal{L}_{\text{d_rec}} \\ & - \lambda_{\text{ds}} \cdot \mathcal{L}_{\text{ds}} + \lambda_{\text{cyc}} \cdot \mathcal{L}_{\text{cyc}} + \lambda_{\text{FG}} \cdot \mathcal{L}_{\text{FG}} + \lambda_{\text{BG}} \cdot \mathcal{L}_{\text{BG}} \ ,\end{aligned} \tag{8}$$

where λ_{adv}, $\lambda_{\text{sd_rec}}$, $\lambda_{\text{d_rec}}$, λ_{ds}, λ_{cyc}, λ_{FG}, and λ_{BG} are the positive hyperparameters for each term.

We follow the training scheme as described in StarGAN v2 [5] with the following modifications. To fit the model on a single Nvidia GTX TITAN X, the batch size is reduced to four while the model is still trained for 100,000 iterations. The training time is about three and a half days on the dedicated GPU with the modified network architecture[4] and loss functions mentioned in Section 3 in PyTorch [32]. We set $\lambda_{\text{adv}} = \lambda_{\text{sd_rec}} = \lambda_{\text{d_rec}} = \lambda_{\text{ds}} = \lambda_{\text{cyc}} = \lambda_{\text{FG}} = \lambda_{\text{BG}} = 1$ for all three datasets.

B.2 Classifiers

We train all the classifiers that use ResNet-50 as the backbone for 100 epochs with the SGD optimizer [35] and batch size 64. The initial learning rate is 0.001, momentum is 0.9 and weight decay is 1e-4. A learning rate scheduler is set to reduce the learning rate by a factor of 0.1 when the validation loss stops decreasing for 5 epochs. The same setting also applies to EfficientNet-b4, except the batch size is set to 128. Although DT-GAN can synthesize realistic defective samples, we notice that there still exists a domain gap between the generated samples and the real samples. To explore the full potential of the generated samples, we attach an auxiliary source classifier to distinguish between synthetic and real samples. Then, this classifier is connected to the backbone (e.g., ResNet-50) through a Gradient Reversal Layer. With the help of the Gradient Reversal Layer, the backbone is forced to extract the shared features between synthetic and real samples, which ensures all training samples are effectively learned.

We design a two-layer perceptron that connects to the average pooling layer in ResNet-50 as shown in Fig. 7. Note that the usual fully connected layer after the average pooling in ResNet-50 remains the same and is not affected by the extra branch we added. Inspired by [4], a three-layer perceptron is used for EfficientNet-b4 instead as shown in Fig. 8. Its layers are initialized with a random normal distribution, where the standard deviation is set to 0.01 for the first two layers and 0.05 for the output layer. The biases for all layers are set to 0.

[4] We based our implementation on the source code from StarGAN v2: https://github.com/clovaai/stargan-v2

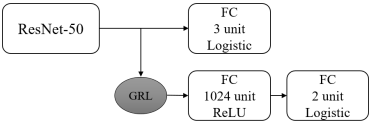


Figure 7: ResNet-50 with GRL.

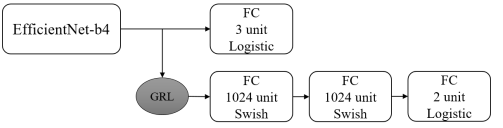


Figure 8: EfficientNet-b4 with GRL.

C Evaluation setup

C.1 Generated Samples from DT-GAN.

DT-GAN requires images as input for generating synthetic data. At test time, we translated each `Normal` image in all datasets into four defective images: two with `Scratches` and two with `Spots`. The translations were performed by two subnetworks: by the mapping network M using random noise ('latent-guided') and by the style-defect encoder E using a reference image ('reference-guided'). We first randomly sampled one latent code for each foreground defect domain. Similarly, we also randomly sampled one reference image from the training set for each foreground defect domain. The corresponding defects and styles were then produced by the two subnetworks respectively and fed to the generator G for target image generation.

C.2 Setup for Classifier Training

We generated synthetic images following the procedure in Appendix C.1 and conducted classification experiments separately on images generated from each subnetwork and a set of mixed samples from both subnetworks. We refer to the latter case as the mixture set, where we randomly sampled 50% of the generated images from each subnetwork. We collected additional 6,200 normal images for each product to generate defective samples and evaluated the performance of classifiers when the varying amount of normal samples is available — 1,500, 2,200, 4,000, and 6,200. This results in increasing dataset sizes from 4,500 to 18,600. Table 10 shows consistent gains of using synthetic images generated from DT-GAN. We observe that the mixture set further boosts the performance of the classifiers in most cases. Presumably, the mixture set benefits from the combination of samples from reference-guided synthesis, which are well aligned with the original defect distribution, and the samples from latent-guided synthesis (i.e. from random noise), which adds novel but plausible defects to the dataset. In the main paper, we report the results of the mixture set at a data scale of 12,000 for all experiments, including the quantitative evaluation of DT-GAN.

C.3 Frechét Inception Distance (FID)

We used the feature vectors from the last average pooling layer of the ImageNet pretrained Inception-V3 [37] to calculate the score. For each test image from the `Normal` domain, we translated it into a synthetic defective image of each defect domain. The defects and styles for the translations were acquired in two ways: by randomly sampling from the standard normal distribution and by randomly sampling a reference image from the train set of a defect domain. To calculate the FID score, we generated 4,000 defective samples per product per defect domain for each way of guidance and formed the mixture set by randomly sampling 2,000 images per product per defect domain from each way. The reported FID scores

Table 10: Classification results with regard to the synthetic images generated from the two subnetworks and the mixture set.

Dataset Size	A			
	Trad-Aug	Latent	Reference	Mix
4500	16.00±1.04	13.81±1.63	15.27±3.49	**10.18±1.75**
6600	14.90±1.38	12.36±0.81	16.00±2.85	**10.54±1.22**
12000	13.81±2.36	10.90±3.34	11.09±3.49	**6.72±1.65**
18600	13.62±2.22	**10.18±1.18**	13.81±8.56	10.54±2.45

Dataset Size	B			
	Trad-Aug	Latent	Reference	Mix
4500	8.8±0.45	6.2±2.17	6.8±1.30	**5.6±1.51**
6600	7.6±1.51	8.0±0.70	**6.0±1.41**	6.8±3.11
12000	6.8±1.64	5.8±1.30	5.6±1.34	**4.6±0.89**
18600	6.8±1.79	7.0±1.00	6.0±1.58	**5.0±1.87**

Dataset Size	C			
	Trad-Aug	Latent	Reference	Mix
4500	17.14±4.62	14.66±2.74	**12.00±1.59**	14.09±2.27
6600	15.23±2.33	13.14±1.95	12.57±1.57	**11.42±0.00**
12000	16.57±3.20	12.95±4.34	**11.42±0.96**	12.76±1.97
18600	15.62±0.85	12.57±2.46	12.95±2.98	**11.64±1.24**

were then computed between the defective images in the training set and the mixture set of synthetic defective images.

D Network Architecture

In this section, we provide the architectural details of all four modules in DT-GAN.

Generator (Table 11)**.** The encoder part of the generator consists of three downsampling blocks and two intermediate blocks (Table 11 (a)), all of which are pre-activation residual units [13]. Then the encoded feature map is split depth-wise into background (Table 11 (b)) and foreground (Table 11 (c)). Both of them are then carried through separate decoders. We use instance normalization (IN) and adaptive instance normalization (AdaIN) as indicated in the activation column. The style is injected into all AdaIN layers to modulate the affine transformations. Note that AdaIN is only used in the foreground decoder. The outputs of both decoders are only merged in the end (Table 11 (d)).

Mapping Network (Table 12)**.** The mapping network consists of four shared linear layers (Table 12 (a)) and two separate branches: one for generating styles (Table 12(b)) and one for defects (Table 12(c)). Each of them is further divided into K output branches, where K denotes the number of foreground defect domains. The dimension of the input, the output style, and the output defect are set to 16, 64, and $16 \times 16 \times 64$, respectively. The latent code is sampled from the standard normal distribution. Note that we apply per-pixel noise after each convolution in the defect branch, which we have observed to increase the diversity of generated defects significantly (cf. Table 14 (e)).

Style-Defect Encoder (Table 13)**.** The style-defect encoder consists of a CNN (Table 13 (a)) with two branches (Table 13 (b) and (c)) as in the mapping network. Each branch has K

Table 11: Generator architecture.

(a) Encoder

Layer	Resample	Norm	Output Shape
Image **x**	-	-	$128 \times 128 \times 3$
Conv 1×1	-	-	$128 \times 128 \times 128$
ResBlk	AvgPool	IN	$64 \times 64 \times 256$
ResBlk	AvgPool	IN	$32 \times 32 \times 512$
ResBlk	AvgPool	IN	$16 \times 16 \times 512$
ResBlk	-	IN	$16 \times 16 \times 512$
ResBlk	-	IN	$16 \times 16 \times 512$

(b) Background Decoder

Layer	Resample	Norm	Output Shape
Input	-	-	$16 \times 16 \times 448$
ResBlk	-	IN	$16 \times 16 \times 448$
ResBlk	-	IN	$16 \times 16 \times 512$
ResBlk	-	IN	$16 \times 16 \times 512$
ResBlk	Upsample	IN	$32 \times 32 \times 512$
ResBlk	Upsample	IN	$64 \times 64 \times 256$
ResBlk	Upsample	IN	$128 \times 128 \times 448$

(c) Foreground Decoder

Layer	Resample	Norm	Output Shape
Input	-	-	$16 \times 16 \times 64$
ResBlk	-	AdaIN	$16 \times 16 \times 64$
ResBlk	-	AdaIN	$16 \times 16 \times 256$
ResBlk	-	AdaIN	$16 \times 16 \times 256$
ResBlk	Upsample	AdaIN	$32 \times 32 \times 256$
ResBlk	Upsample	AdaIN	$64 \times 64 \times 128$
ResBlk	Upsample	AdaIN	$128 \times 128 \times 64$

(d) Fusion

Layer	Resample	Norm	Output Shape
Input	-	-	$128 \times 128 \times (448 + 64)$
Conv 1×1	-	-	$128 \times 128 \times 3$

Table 12: Mapping network architecture.

(a) Shared Layers

Layer	Activation	Output Shape
Latent **z**	-	16
Linear	ReLU	512
Linear	ReLU	512
Linear	ReLU	512
Linear	ReLU	512

(b) Style

Layer	Activation	Output Shape
Input	-	512
Linear	ReLU	512
Linear	ReLU	512
Linear	ReLU	512
Linear	-	64

(c) Defect

Layer	Resample	Activation	Noise	Output Shape
Input	-	-	-	512
Reshape	-	-	-	$1 \times 1 \times 512$
ResBlk	Upsample	IN	True	$2 \times 2 \times 512$
ResBlk	Upsample	IN	True	$4 \times 4 \times 512$
ResBlk	Upsample	IN	True	$8 \times 8 \times 256$
ResBlk	Upsample	IN	True	$16 \times 16 \times 128$
Conv 1×1	-	IN	True	$16 \times 16 \times 64$

Table 13: Style-defect encoder and discriminator architectures.

(a) Shared Layers

Layer	Resample	Norm	Output Shape
Input $\mathbf{x}$	-	-	$128 \times 128 \times 3$
Conv 1×1	-	-	$128 \times 128 \times 64$
ResBlk	AvgPool	-	$64 \times 64 \times 256$
ResBlk	AvgPool	-	$32 \times 32 \times 512$
ResBlk	AvgPool	-	$16 \times 16 \times 512$

(b) Style / Discriminator and BG Classifier

Layer	Resample	Norm	Output Shape
Input	-	-	$16 \times 16 \times 512$
ResBlk	AvgPool	-	$8 \times 8 \times 512$
ResBlk	AvgPool	-	$4 \times 4 \times 512$
LReLU	-	-	$4 \times 4 \times 512$
Conv 4×4	-	-	$1 \times 1 \times 512$
LReLU	-	-	$1 \times 1 \times 512$
Reshape	-	-	512
Linear$*K$	-	-	$D * K$

(c) Defect / FG Classifier

Layer	Norm	Output Shape
Input	-	$16 \times 16 \times 512$
LReLU	-	$16 \times 16 \times 512$
Conv $1\times1*K$	-	$16 \times 16 \times 64*K$

Table 14: Ablation study in FID over all datasets. 'Latent', 'Reference' and 'Mix' indicate the subnetworks and a set of mixed samples from the two subnetworks.

	FID↓		
	Latent	Reference	Mix
(a) Baseline StarGAN v2 [8]	226.26	232.89	227.67
(b) + Style-defect branches	292.76	292.12	287.35
(c) + FG and BG classifier	**95.33**	99.84	**95.81**
(d) + Separately decoding FG and BG in G	115.89	99.61	101.02
(e) + Anchor domain and Noise injection	97.37	**98.97**	96.27

outputs, where K is the number of domains. Three pre-activation residual blocks are shared among two branches, followed by a specific structure for each branch. The output dimension D in Table 13 is set to 64, which denotes the dimension of the style.

Discriminator (Table 13)**.** The discriminator is a multi-task discriminator with two auxiliary classifiers for the foreground defect and the background. The structure is almost identical to the style-defect encoder, except the output dimension D is set to 1 for real/fake classification. The background classifier acts in parallel to the final linear layer in Table 13 (b) and provides the logits for background classification. The foreground classifier instead acts on top of the output in Table 13 (c) and four more pre-activation residual layers are applied to encode the defect into logits for foreground defect classification.

E Additional Results

E.1 Ablation Study in FID

We report the average FID over all three datasets for each configuration in Table 14.

Table 15: Quantitative comparison of the baseline methods on the defect classification task. The reported values are the achieved error rates (%) and standard deviation over five runs.

Aug. Method	Syn. Data	EfficientNet-b4		
		A	B	C
None	None	16.54±0.76	9.6±3.05	26.29±1.73
Trad	None	12.55±1.49	6.8±0.84	12.76±0.52
Trad	Mokady et al. [28]	14.18±1.52	8.0±1.58	17.52±1.59
Trad	StarGAN v2 [5]	14.73±2.17	10.5±1.34	16.76±1.44
Trad	StyleGAN2 [21]	14.55±1.11	6.4±1.14	15.81±0.85
Trad	BigGAN* [3]	13.82±1.34	9.2±1.30	17.74±1.09
Trad	Ours	**12.00±0.76**	**5.0±1.58**	**11.43±1.78**
CutMix [42]	None	14.00±4.06	5.6±1.67	12.00±1.86
CutOut [8]	None	10.73±0.41	7.2±2.17	13.90±2.09
MixUp [44]	None	12.18±1.22	5.8±1.64	12.95±3.96
CutMix [42]	Ours	14.55±1.44	**3.4±1.34**	17.90±2.89
CutOut [8]	Ours	**10.36±0.81**	4.2±0.45	13.14±0.80
MixUp [44]	Ours	12.00±1.00	3.6±1.52	**11.81±0.85**

E.2 Additional Reference-guided Image Synthesis Results on the SDI Dataset

We provide additional reference-guided image synthesis results on the SDI dataset in Fig. 9–Fig. 11. We demonstrate all the possible transfers among all foreground defect domains on the purposed DT-GAN and StarGAN v2 [5]. For [28], we trained one model for each translation and omitted some cases like `Normal`-to-`Normal` due to the lack of practical usefulness. Both defect and style are extracted from the reference image. Note that DT-GAN is the only model that can faithfully append and remove foreground defects onto both `Normal` and defective samples. For example, in the fifth column of Fig. 9, the original scratch in the source image is removed and only the defects from the reference images are presented in the output images.

E.3 Additional Reference-guided Image Synthesis Results on the SDI Dataset

Effectiveness of Synthetic Data. To demonstrate that the performance gains provided by our synthetic data are not confined to a specific network, we also show the experiment results with EfficientNet-b4 [38] in Table 15.

Besides, we present additional evaluations showing the effectiveness of our synthetic data when incrementally adding more of them to the training set. Each dataset in Table 16 is formed following the same procedure: first, the images in the `Normal` class are always real, and the defective classes are either balanced by oversampling (Trad-Aug) or by synthetic data from DT-GAN (Ours). We various the number of available `Normal` training images from 1,500 to 6,200, which results in increasing dataset sizes from 4,500 to 18,600. As seen in Table 16, the synthetic images from DT-GAN boost the performance in all scales. We report the performance on a scale of 12,000 in the main paper.

As discussed in Section 4.2, we assumed that the data-insufficiency problem can be mitigated by transferring defects across multiple background products. To examine if this assumption holds, we compared the performance of classifiers trained on synthetic images with defects from a specific source (v-A, v-B, v-C) to classifiers trained on images with defects

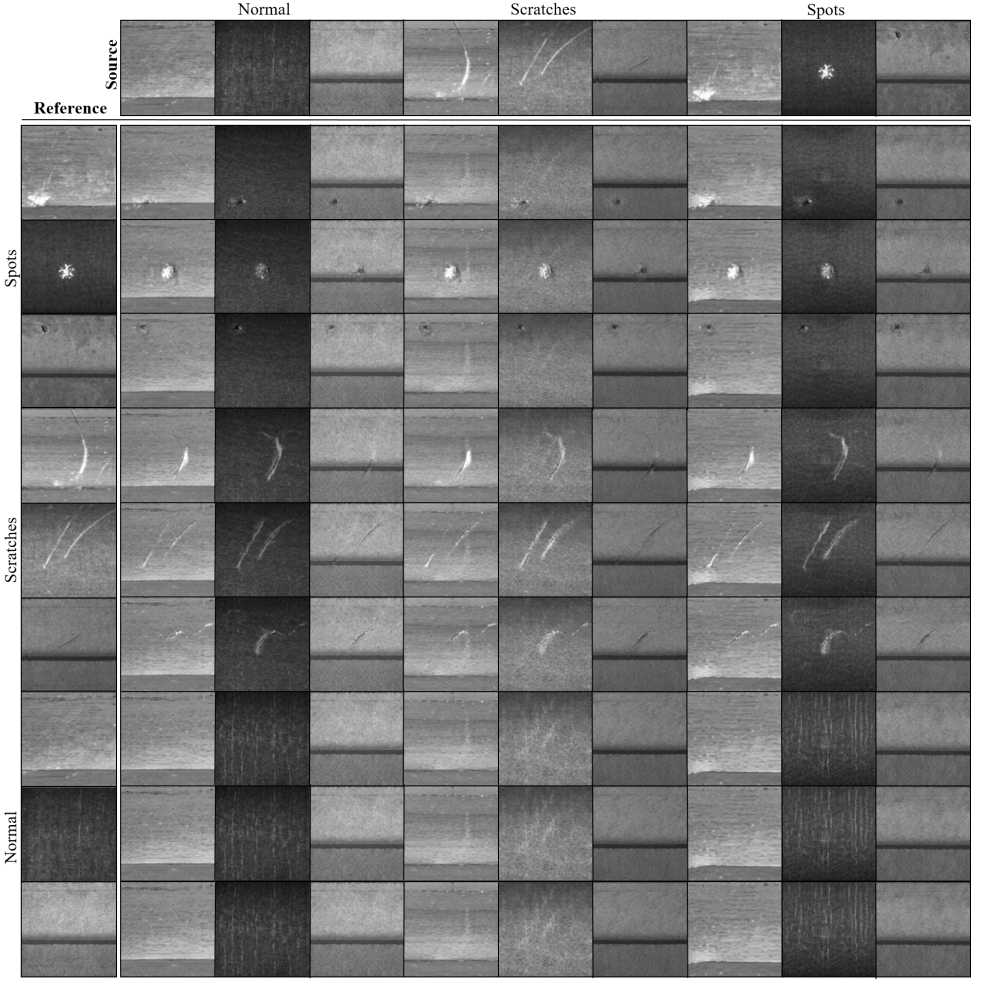


Figure 9: Reference-guided image synthesis results on the SDI dataset. The first row and the first column are the real images sampled from the dataset, while the rest are synthetic images generated by the proposed DT-GAN. Note that DT-GAN provides translations between different foreground domains (`Normal`, `Scratches` and `Spots`) with defects and styles extracted from reference images while the backgrounds from source images are well preserved. (Best viewed in color and zoom in.)

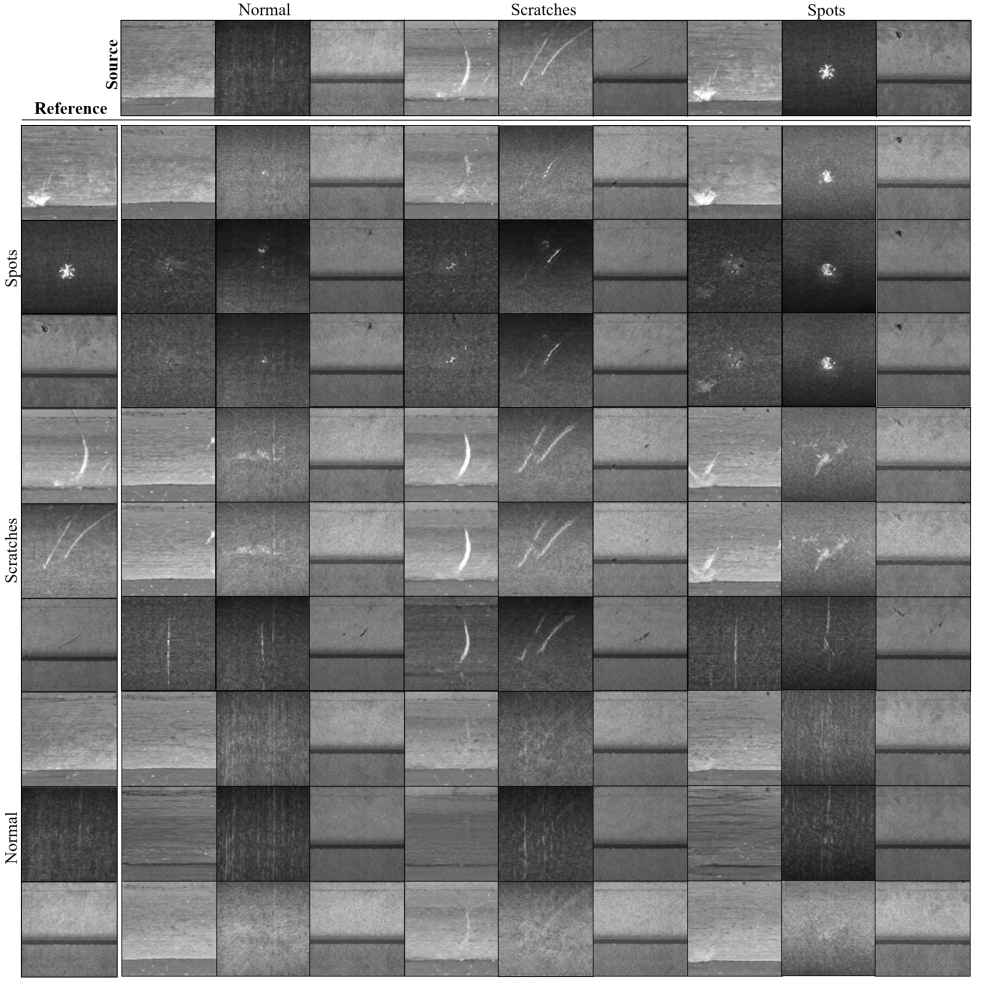


Figure 10: Reference-guided image synthesis results of StarGAN v2 on the SDI dataset. The first row and the first column are the real images sampled from the dataset, while the rest are synthetic images generated by StarGAN v2. (Best viewed in color and zoom in.)

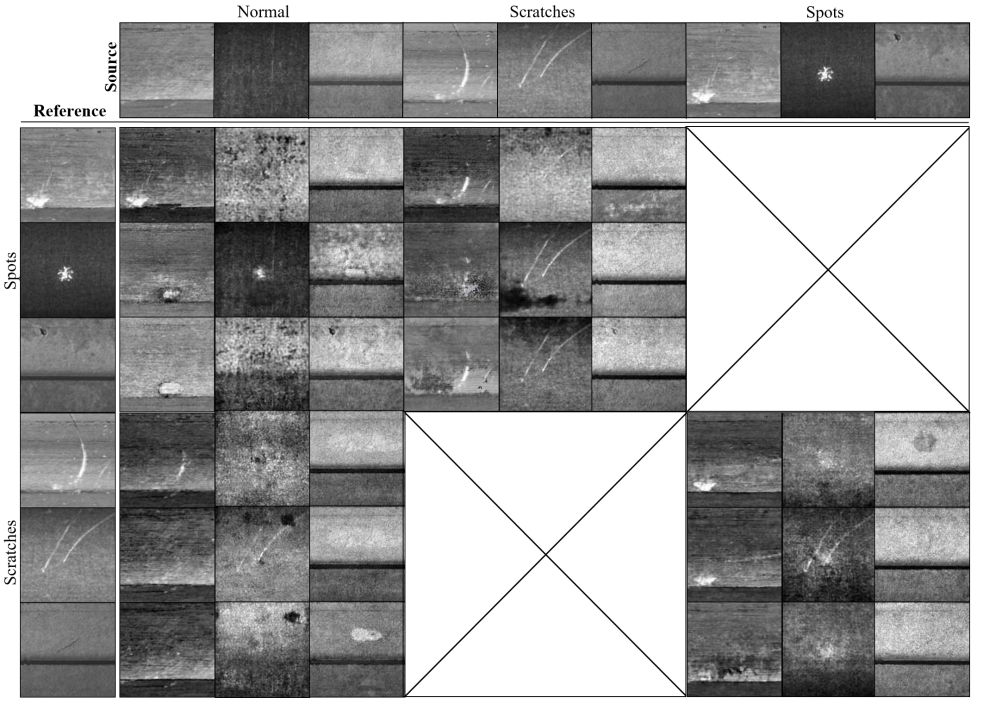


Figure 11: Reference-guided image synthesis results of Mokady et al. on the SDI dataset. The first row and the first column are the real images sampled from the dataset, while the rest are synthetic images generated by Mokady et al.. (Best viewed in color and zoom in.)

Table 16: Quantitative results for DT-GAN as a data augmentation method with an incremental number of synthetic data. The leftmost column indicates the size of the dataset, including all images from the training set plus increasing amounts of synthetic images. In the first row, **20A** refers to the case of 20 real defective samples for product A, while **All** refers to the full training set.

Dataset Size	**20A**					
	A		B		C	
	Trad-Aug	Ours	Trad-Aug	Ours	Trad-Aug	Ours
4500	35.09±2.62	**27.64±3.12**	7.8±1.48	**5.6±1.67**	15.24±1.90	**13.14±1.70**
6600	39.64±2.28	**27.64±1.65**	8.8±1.64	**6.2±1.64**	15.81±1.73	**12.38±1.65**
12000	34.18±4.39	**28.55±7.32**	5.8±0.45	**5.6±1.14**	16.19±1.17	**10.86±1.28**
18600	39.45±7.06	**32.55±5.04**	7.2±0.84	**5.2±1.10**	14.86±0.85	**13.14±2.06**

Dataset Size	**All**					
	A		B		C	
	Trad-Aug	Ours	Trad-Aug	Ours	Trad-Aug	Ours
4500	16.00±1.04	**10.18±1.75**	8.8±0.45	**5.6±1.51**	17.13±6.62	**14.09±2.27**
6600	14.90±1.38	**10.54±1.22**	7.6±1.51	**6.8±3.11**	15.23±2.33	**11.42±0.00**
12000	13.81±2.36	**6.72±1.65**	6.8±1.64	**4.6±0.00**	16.57±3.20	**13.90±2.57**
18600	13.63±2.22	**10.54±2.45**	6.8±1.79	**5.0±1.87**	15.62±0.85	**11.61±1.24**

Table 17: Cross-product defect transfer on classifiers trained with reference-guided synthetic images at different scales. Note that here A, B and C stand for 3 products in the SDI dataset while v-A, v-B, v-C, and v-ABC indicate the defects are copied from which reference set.

Dataset Size	A				
	Trad-Aug	v-A	v-B	v-C	v-ABC
4500	16.00±1.04	**12.90±2.61**	13.08±1.65	14.90±2.46	15.27±3.49
6600	14.90±1.38	13.99±1.89	**11.26±1.04**	14.36±4.04	16.00±2.85
12000	13.81±2.36	11.81±2.65	12.72±2.87	11.99±1.63	**11.09±3.49**
18600	13.63±2.22	**12.72±5.22**	14.36±3.83	14.18±5.05	13.81±8.56

Dataset Size	B				
	Trad-Aug	v-A	v-B	v-C	v-ABC
4500	8.8±0.45	7.8±2.15	**5.6±1.14**	10.19±0.84	6.8±1.30
6600	7.6±1.51	6.8±1.65	7.8±1.10	8.0±2.34	**6.0±1.41**
12000	6.8±1.64	6.4±1.34	6.6±1.52	6.6±1.34	**5.6±1.34**
18600	6.8±1.79	6.2±1.78	**4.4±1.14**	6.6±1.95	6.0±1.58

Dataset Size	C				
	Trad-Aug	v-A	v-B	v-C	v-ABC
4500	17.14±4.62	14.85±0.52	16.76±2.58	13.90±1.98	**12.00±1.59**
6600	15.23±2.33	13.14±1.24	13.90±2.29	14.28±1.34	**12.57±1.57**
12000	16.57±3.20	13.14±2.81	**11.23±0.80**	14.85±1.73	11.42±0.96
18600	15.62±0.85	13.71±1.73	15.99±6.75	**12.57±3.26**	12.95±2.98

from all products (v-ABC). The results of the cross-product defect transfer with regard to different sizes of the training set are shown in Table 17. We again notice that using our synthetic data is beneficial. Moreover, in most cases, the performance is further improved when the shared characteristics are used (i.e. by transferring defects from other products). We interpret this as support for our assumption and the practical usefulness of our method in a real-world scenario.

E.4 Data Augmentation in GANs

We provide additional results in the case of training GANs with augmentation methods in Table 18. Augmentation methods like ADA [70] and DiffAug [46] are proposed to adapt GAN training to limited data. We applied these two augmentation methods to StyleGAN v2 [71] and BigGAN [7] respectively, because these state-of-the-art image synthesis methods are not optimized for small datasets. However, incorporating the augmentation methods in training GANs is not always beneficial. The performance of StyleGAN v2 is not always improved when using ADA, potentially due to the conflict between augmentation methods and the decentralized location of defects—in the real-world scenario, defects can occur anywhere on the surface. This is in contrast to datasets that were used to evaluate the aforementioned augmentation methods in GANs, where the objects are centralized (e.g., ImageNet [6], Cifar [72]) and their attributes (e.g. beard, eyeglasses in CelebA [74]) only occur in specific images parts. Thus, we only report the best setting with an average lower FID the in the main paper.

Table 18: Quantitative comparison of DT-GAN with baseline image synthesis methods using FID. Note that the reported values are not comparable between columns, because they are calculated on different training sets. * indicates that the model is trained with augmentation methods.

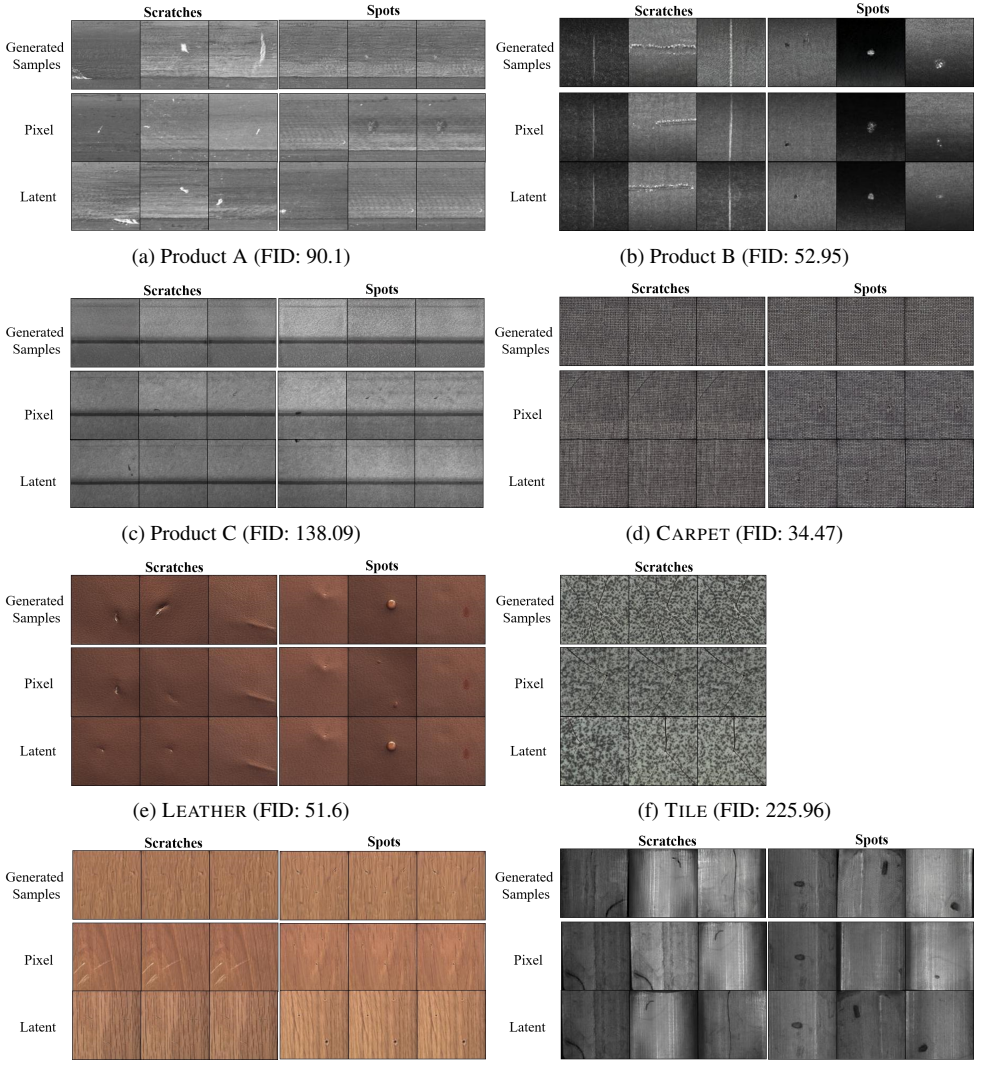
Method	FID↓								
	A	B	C	CARPET	LEATHER	TILE	WOOD	MTD	All
Mokady et al. [28]	68.69	66.9	**36.21**	41.87	60.26	275.12	81.71	68.30	**87.38**
StarGAN v2 [6]	96.85	58.28	50.95	354.31	336.63	434.77	411.37	84.49	228.46
StyleGAN2 [21]	90.1	**52.95**	138.09	51.37	**51.6**	225.96	140.01	**51.39**	100.18
StyleGAN2* [21]	149.66	42.75	135.69	30.63	217.83	**109.06**	**54.09**	226.68	120.80
BigGAN [1]	235.66	192.89	193.61	64.84	143.64	408.49	142.50	277.62	207.41
BigGAN* [1]	218.74	134.41	270.89	34.47	101.7	391.54	113.32	67.91	166.62
Ours	**65.62**	53.62	37.94	**27.33**	78.01	352.15	77.11	78.41	96.27

E.5 Nearest Neighbor Study

To provide an insight into the overfitting phenomenon, we performed a class-wise nearest neighbor search among the training images in pixel space and the feature space of the pre-trained Inception V3[37] network. We provide the nearest neighbor analysis with the full set of products and baseline methods in Fig. 12–Fig. 15. Note that we omit the results on StarGAN v2 due to its identity-shift problem. As mentioned in the main paper, all the baseline methods show signs of overfitting and suffering from mode collapsing, especially [28] and [21]. We believe this explains the better ID reported in Table 1. In contrast, our method produces visually distinct defects from its nearest neighbor. Together with the FIDs reported in quantitative evaluation, we believe our method achieves better performance due to the fact that it allows free combinations of foreground defects and backgrounds, making the generated images more diverse whereas restraining model overfitting even with a small number of training samples.

E.6 More Qualitative Examples on 'Latent-guided' and 'Reference-guided' Image Synthesis

We present additional 'latent-guided' image synthesis results of StyleGAN v2 in Fig. 16 and Fig. 17 and BigGAN in Fig. 18 and Fig. 19. The results are acquired by training one model for each product and then generating 16 images from randomly sampled latent codes from each of them. As pointed out in Section 4.1, both methods cannot adapt well to small datasets. They suffer from mode collapsing and show signs of overfitting by generating images similar to the training data. For example, StyleGAN v2 generates images either with no clear defect or identical to the training set (e.g. LEATHER in Fig. 16 and Product B in Fig. 17). The overfitting we observe here also explains the better FID in Table 1. For TILE, we can see clear signs of mode collapsing in the generated TILE images of StyleGAN v2. Similarly, BigGAN produces images with a single mode and abnormal patterns (e.g. grid structure and gray edges). Unlike StyleGAN v2 and BigGAN, StarGAN v2 and our method both require images as input (i.e. **Source**). Therefore, we randomly sampled two `Normal` images and applied eight defects, which are generated from randomly sampled latent codes to each of them. As seen in Fig. 20 and Fig. 21, StarGAN v2 fails to preserve the background



(a) Product A (FID: 90.1)

(b) Product B (FID: 52.95)

(c) Product C (FID: 138.09)

(d) CARPET (FID: 34.47)

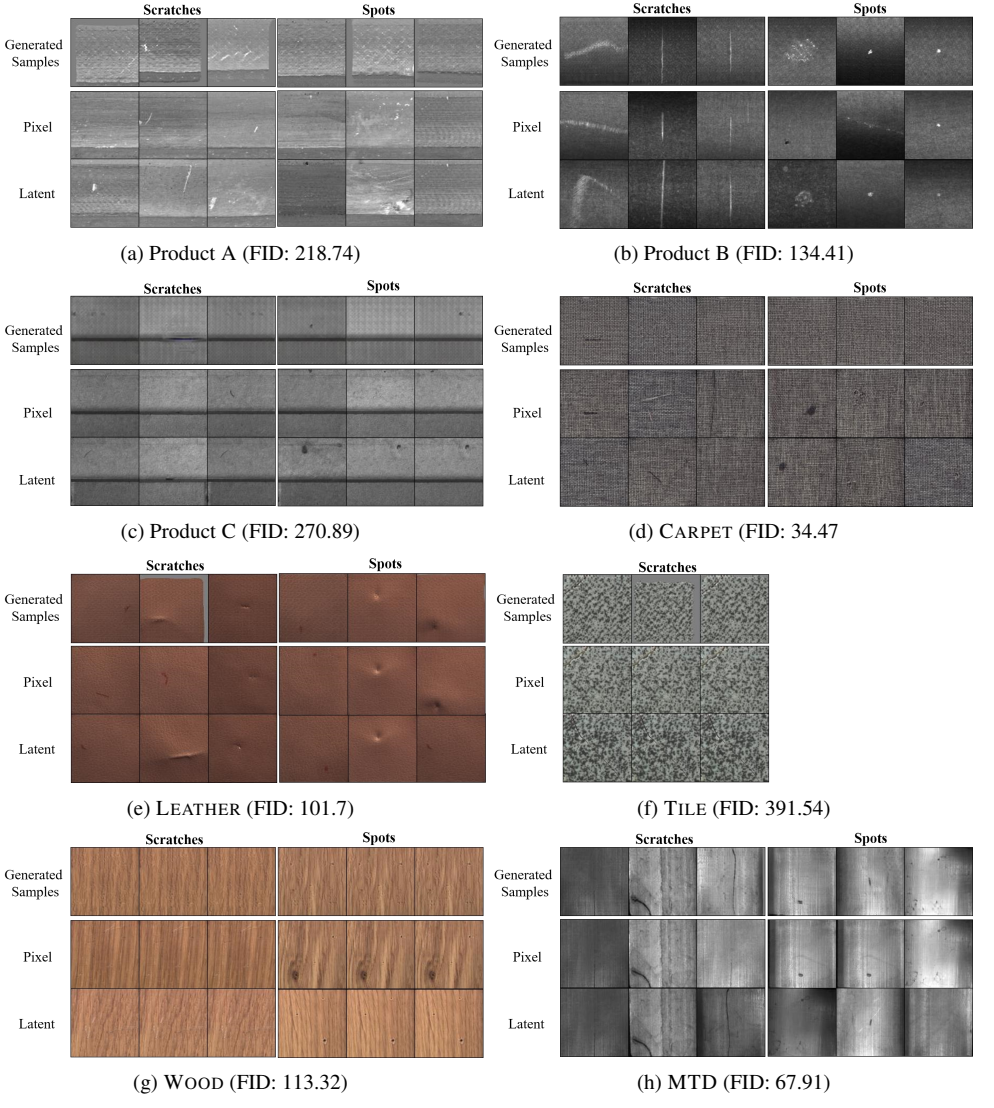
(e) LEATHER (FID: 51.6)

(f) TILE (FID: 225.96)

(g) WOOD (FID: 140.01)

(h) MTD (FID: 51.39)

Figure 12: Nearest neighbor analysis on generated images from StyleGAN2. The first row in each subgraph presents the generated samples and the following two columns show their nearest neighbor from the training set in pixel space and latent space respectively. (Best viewed in color and zoom in.)



(a) Product A (FID: 218.74)

(b) Product B (FID: 134.41)

(c) Product C (FID: 270.89)

(d) CARPET (FID: 34.47

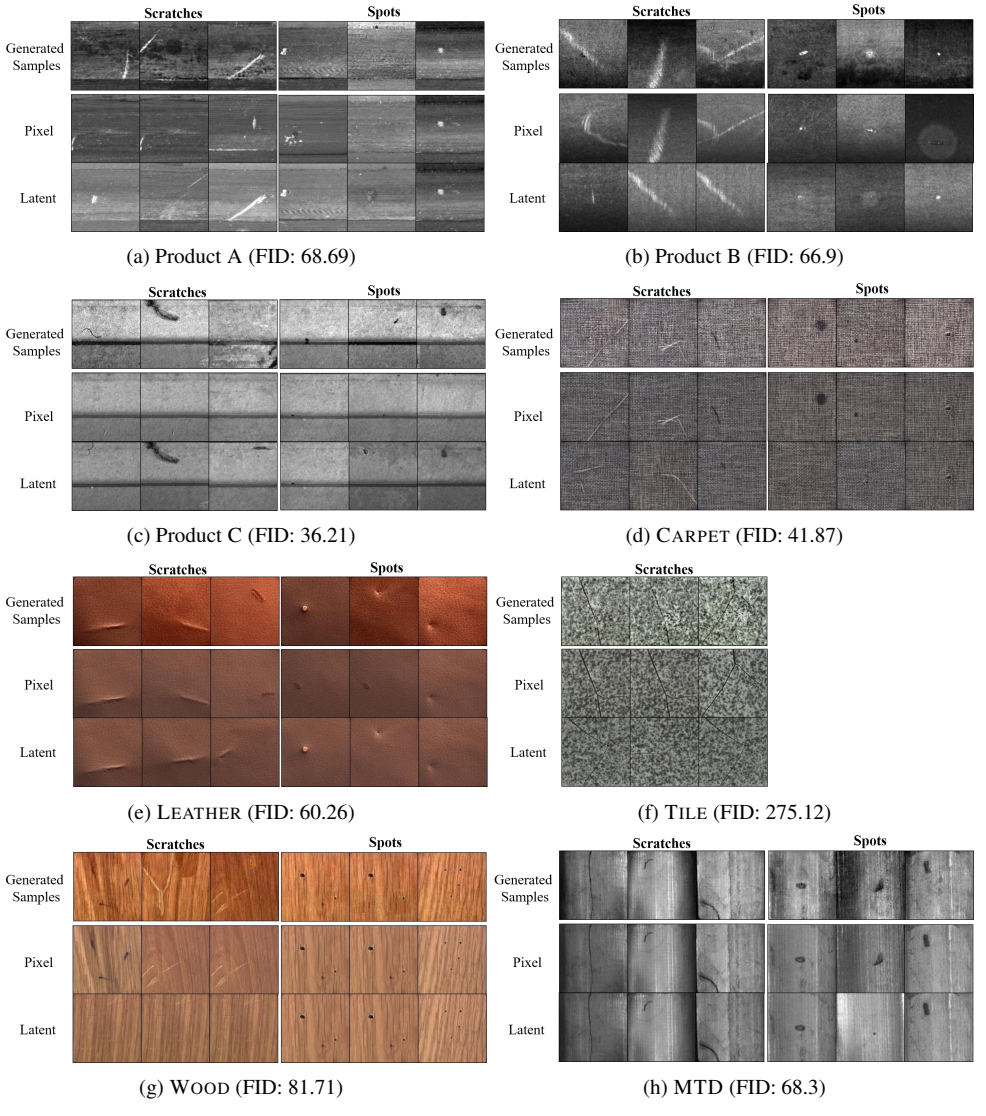
(e) LEATHER (FID: 101.7)

(f) TILE (FID: 391.54)

(g) WOOD (FID: 113.32)

(h) MTD (FID: 67.91)

Figure 13: Nearest neighbor analysis on generated images from BigGAN with DiffAug. The first row in each subgraph presents the generated samples and the following two columns show their nearest neighbor from the training set in pixel space and latent space respectively. (Best viewed in color and zoom in.)



(a) Product A (FID: 68.69)

(b) Product B (FID: 66.9)

(c) Product C (FID: 36.21)

(d) CARPET (FID: 41.87)

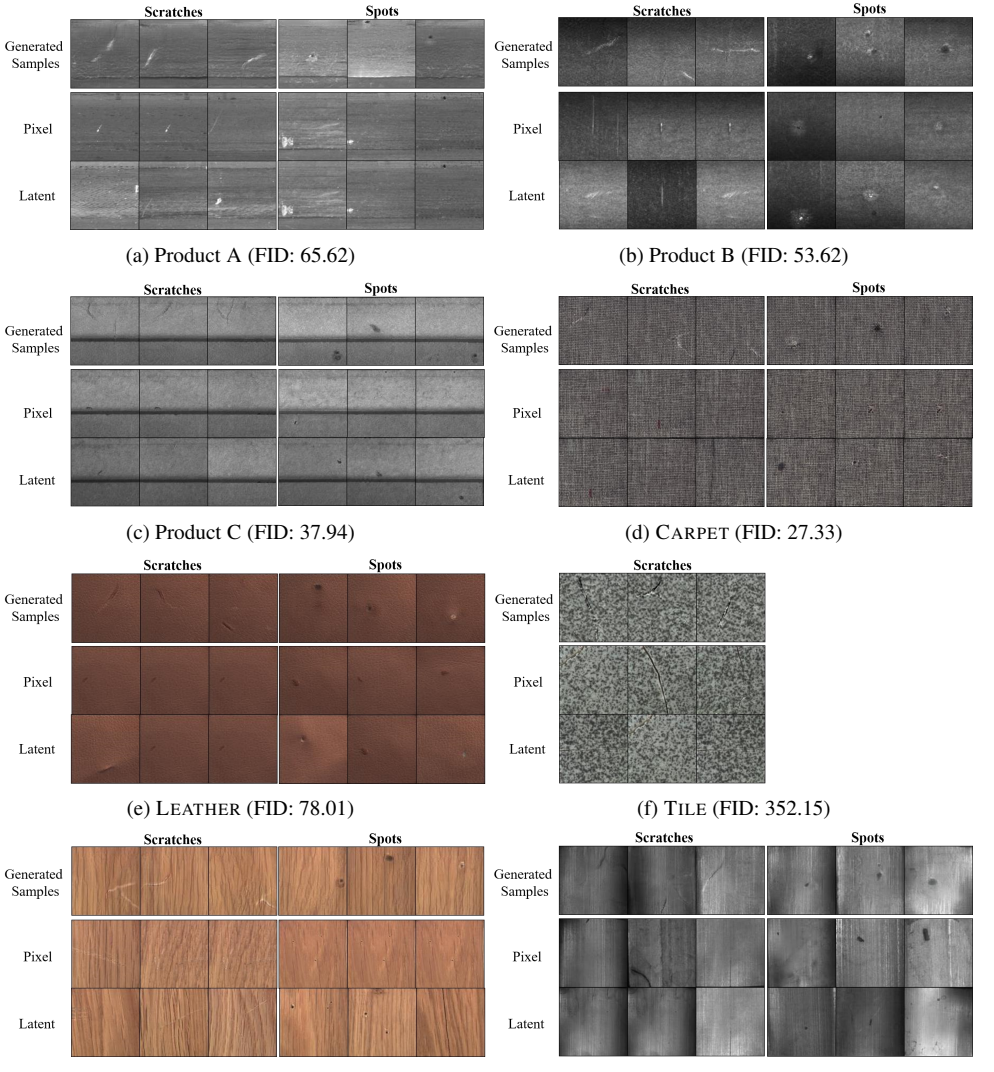
(e) LEATHER (FID: 60.26)

(f) TILE (FID: 275.12)

(g) WOOD (FID: 81.71)

(h) MTD (FID: 68.3)

Figure 14: Nearest neighbor analysis on generated images from Mokady et al.[28]. The first row in each subgraph presents the generated samples and the following two columns show their nearest neighbor from the training set in pixel space and latent space respectively. (Best viewed in color and zoom in.)



(a) Product A (FID: 65.62)

(b) Product B (FID: 53.62)

(c) Product C (FID: 37.94)

(d) CARPET (FID: 27.33)

(e) LEATHER (FID: 78.01)

(f) TILE (FID: 352.15)

(g) WOOD (FID: 77.11)

(h) MTD (FID: 78.41)

Figure 15: Nearest neighbor analysis on generated images from DT-GAN. The first row in each subgraph presents the generated samples and the following two columns show their nearest neighbor from the training set in pixel space and latent space respectively. (Best viewed in color and zoom in.)

from the given input images due to the highly entangled foreground and background. Also, it fails to generate legit and diverse defects without separately modeling the defect and the style. In contrast to the aforementioned methods, our DT-GAN produces images with higher fidelity and more diversity in defect patterns as shown in Fig. 22 and Fig. 23. We believe this again proves the importance of FG/BG disentanglement and style-defect separation, which we introduce in Section 3.1.

For 'reference-guided' image synthesis, the additional results of [28] are shown in Fig. 24 and Fig. 25 while the results of StarGAN v2 are in Fig. 26 and Fig. 27. We can observe clear color shifts in all the outputs from [28]. Moreover, [28] can only transfer defect between *two* domains. In order to perform translation from a non-defective sample to a defective one, we trained a model for each type of defect and for each product. This sums up to be 15 models (`Scratches` and `Spots` for 7 categories and `Scratches` only for TILE). The results from the intended translation within a single product can be found on the diagonal and are marked in red in both Fig. 24 and Fig. 25. We still show the generated images that we obtain for reference images from other products. As expected, the model then fails to preserve the background of given source images and introduces artifacts to the outputs.

Similarly, StarGAN v2 does not preserve the background from the input images. We observe that StarGAN v2 encodes the background characteristics together with the foreground defect of the reference images due to the lack of style-defect separation and FG/BG disentanglement. This results in identity shits in its output images. Moreover, the output images either show no clear defect or contain abnormal patterns which sabotage the fidelity. On the contrary, our method can faithfully transfer the foreground defect of the reference images across the given background of different products as shown in Fig. 28 and Fig. 29, which demonstrates the effectiveness of the style-defect separation and the FG/BG disentanglement we introduced in Section 3.1.

It is also worth noting that our method can perform image synthesis even if the desired combination is not presented in the training set. We demonstrate this on product TILE, which only has training images with `Scratches` but no `Spots`. As shown in Fig. 23 and Fig. 29, DT-GAN can generate spots on given TILE images. This kind of transformation is most useful when the new combination is reasonable for downstream applications.

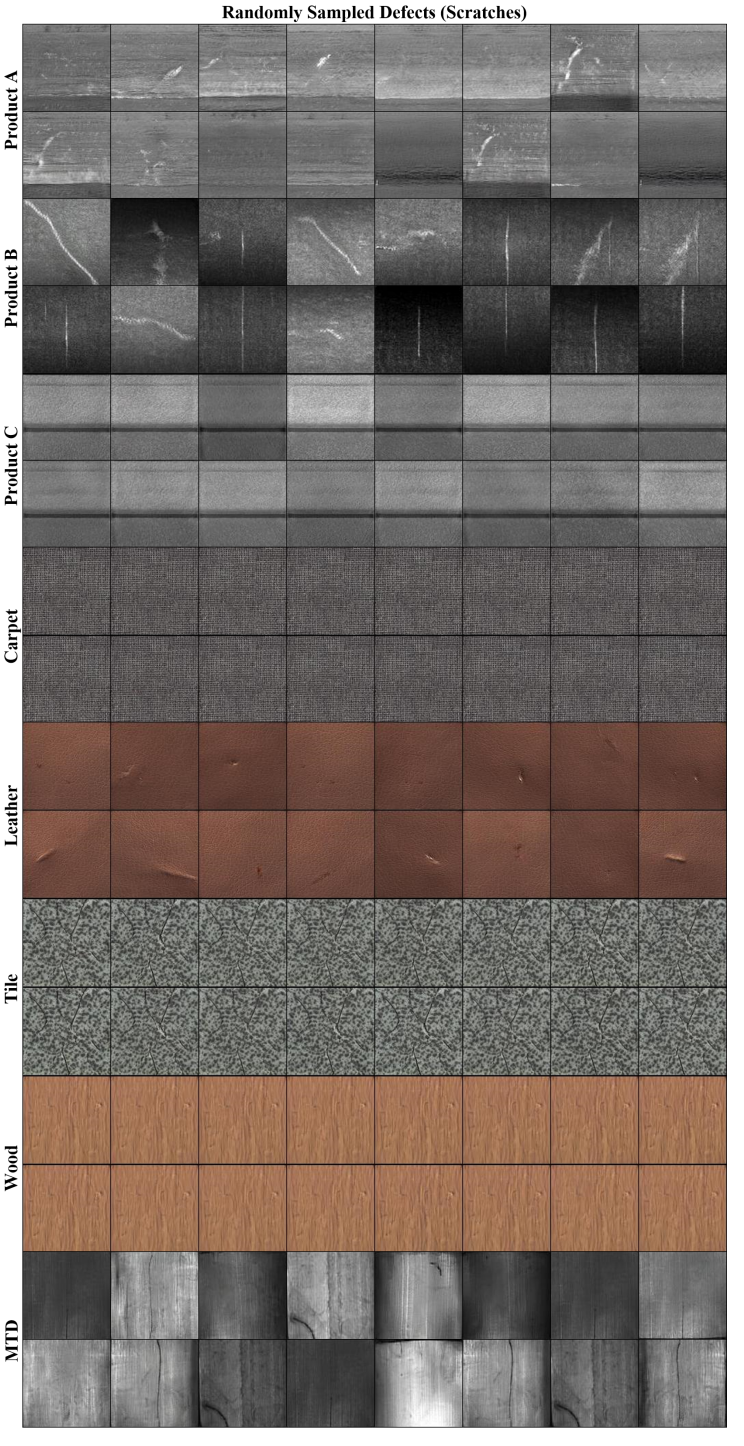


Figure 16: Latent-guided image synthesis results of StyleGAN v2 on all three datasets. We train a model for each product and generate 16 `Scratches` images from randomly sampled latent codes. (Best viewed in color and zoom in.)

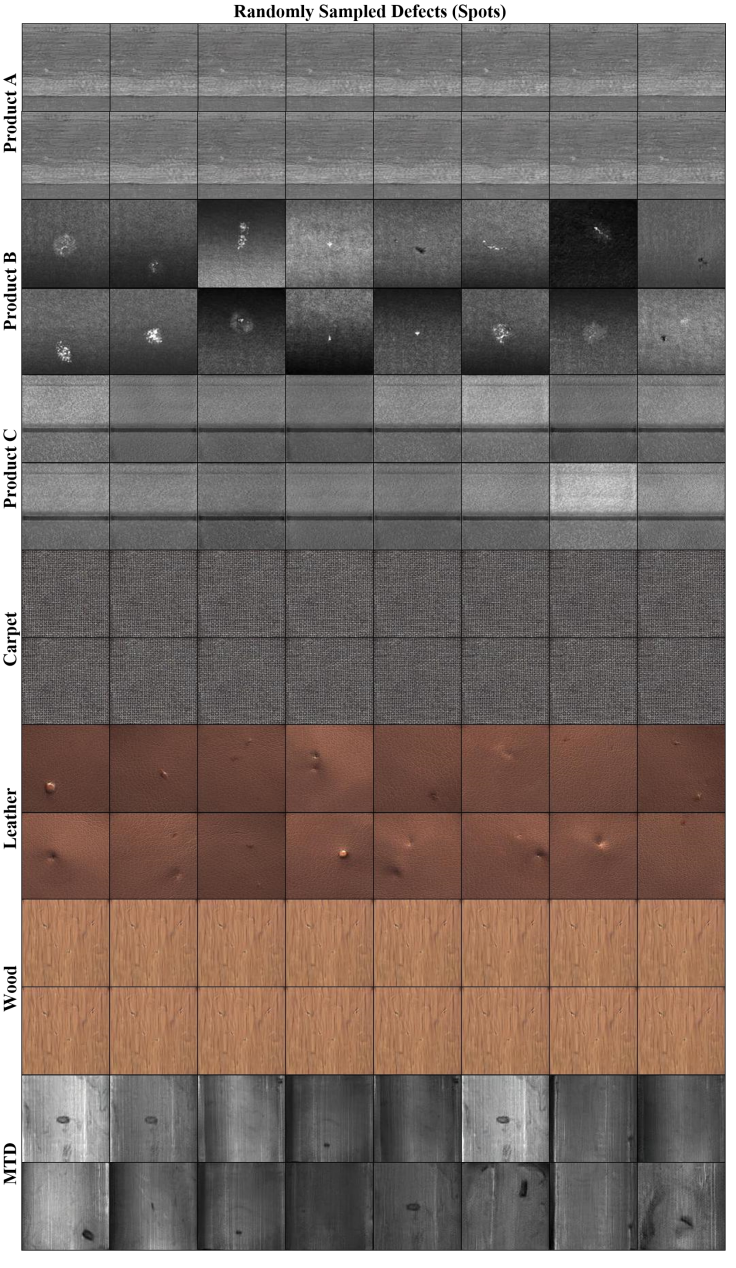


Figure 17: Latent-guided image synthesis results of StyleGAN v2 on all three datasets. We train a model for each product and generate 16 `Spots` images from randomly sampled latent codes. (Best viewed in color and zoom in.)

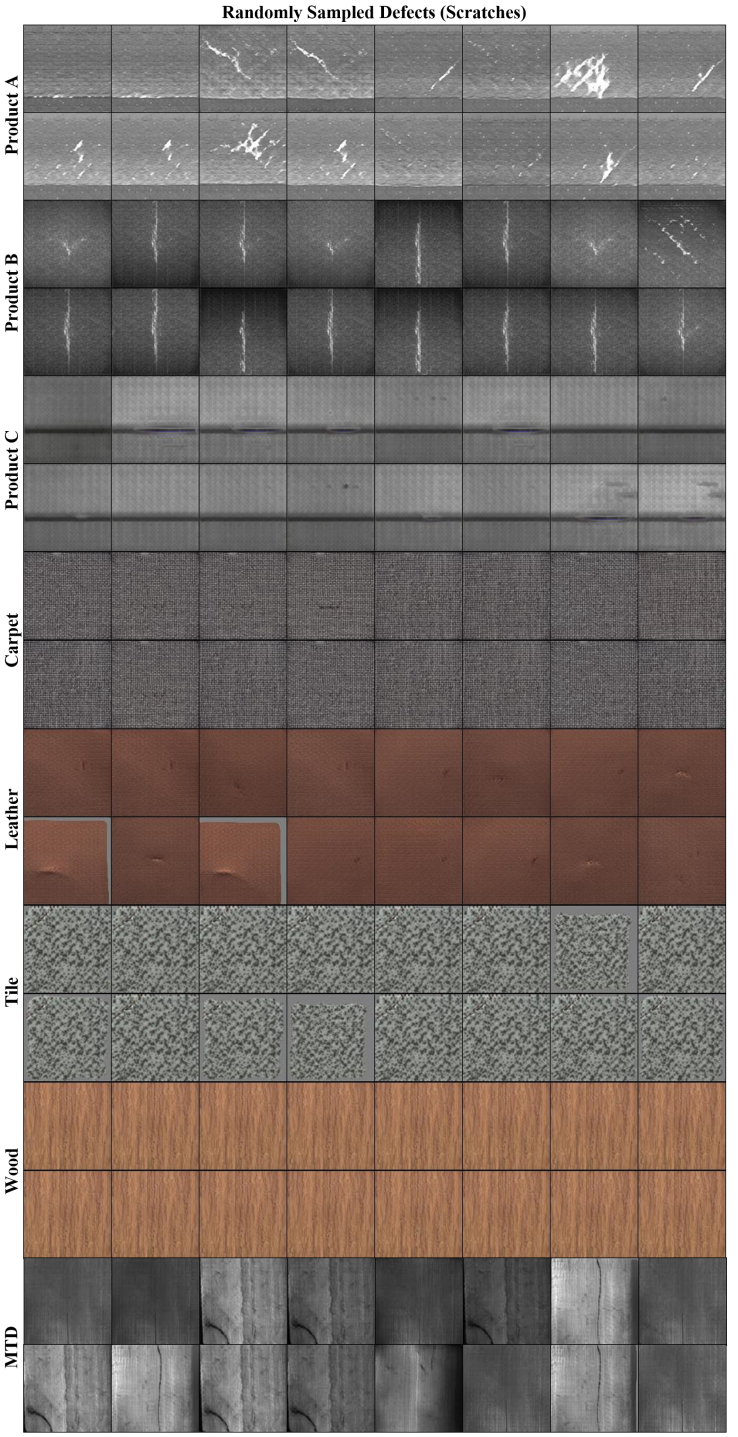


Figure 18: Latent-guided image synthesis results of BigGAN with DiffAug on all three datasets. We train a model for each product and generate 16 `Scratches` images from randomly sampled latent codes. (Best viewed in color and zoom in.)

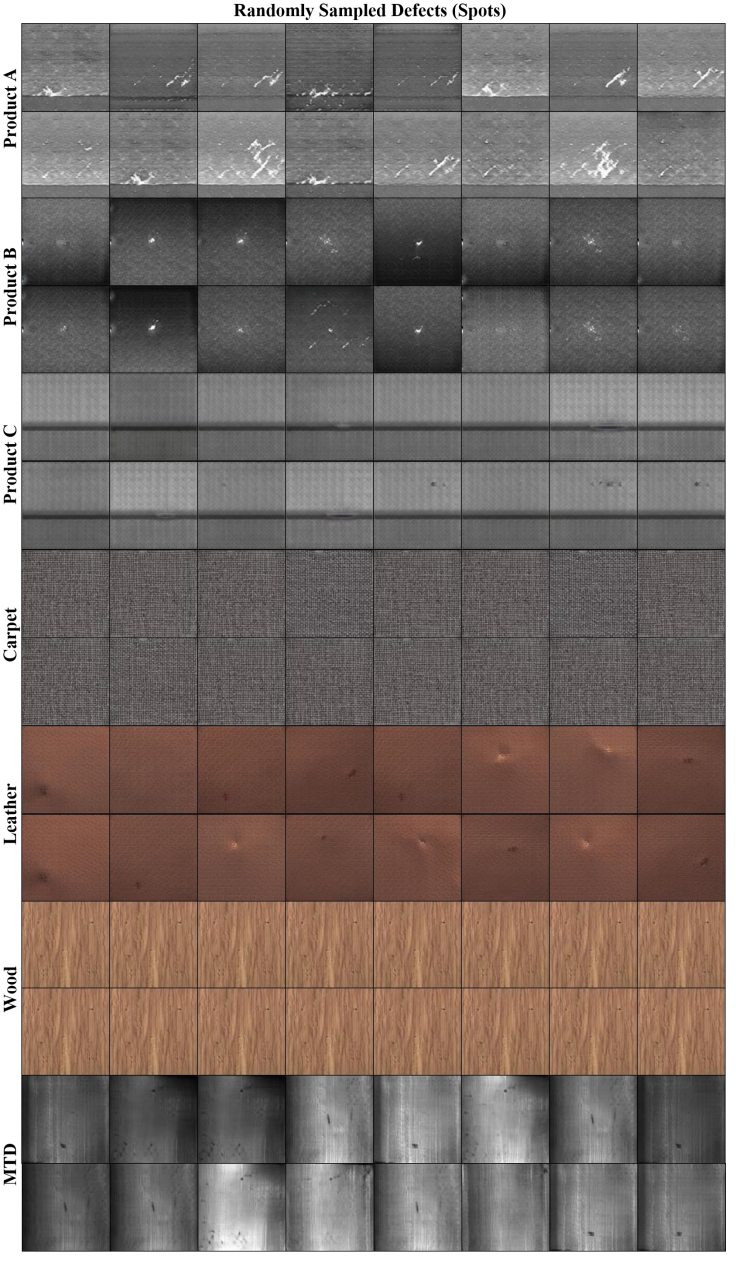


Figure 19: Latent-guided image synthesis results of BigGAN with DiffAug on all three datasets. We train a model for each product and generate 16 `Spots` images from randomly sampled latent codes. (Best viewed in color and zoom in.)

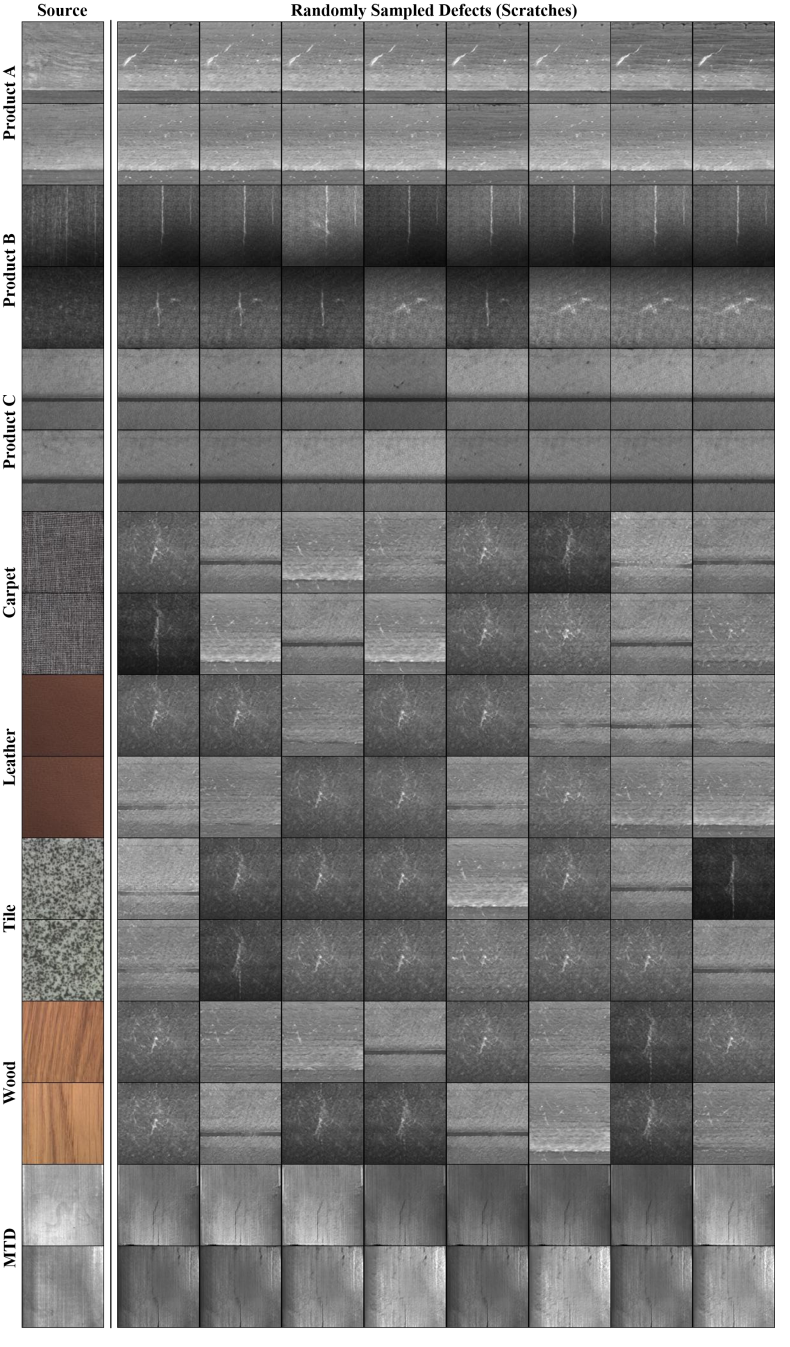


Figure 20: Latent-guided image synthesis results of StarGAN v2 on all three datasets. The model performs translation from `Normal` to `Scratches`. Note that without the style-defect separation and the FG/BG disentanglement, StarGAN v2 not only fails to preserve the background from the given **Source** image but also fails to generate legit defects. (Best viewed in color and zoom in.)

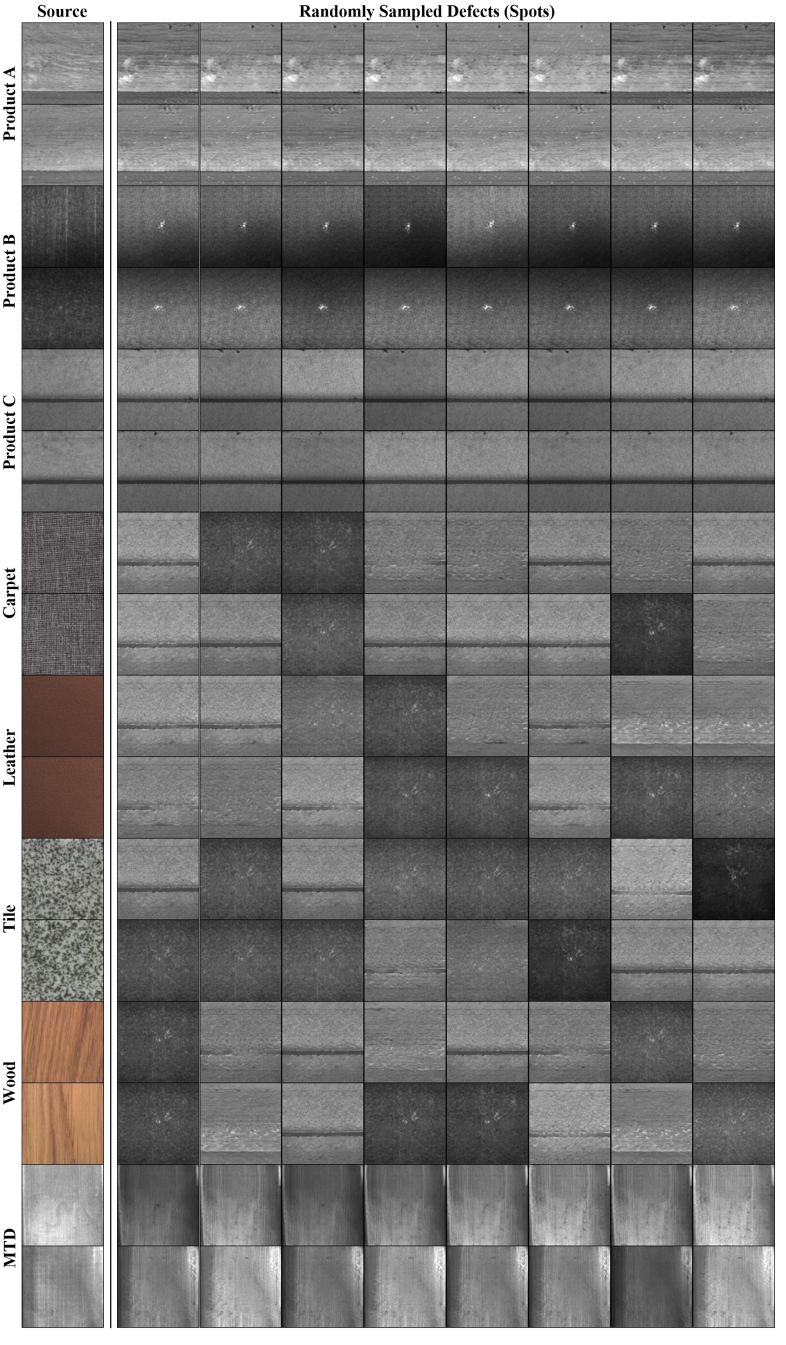


Figure 21: Latent-guided image synthesis results of StarGAN v2 on all three datasets. The model performs translation from `Normal` to `Spots`. Note that without the style-defect separation and the FG/BG disentanglement, StarGAN v2 not only fails to preserve the background from the given **Source** image but also fails to generate legit defects. (Best viewed in color and zoom in.)

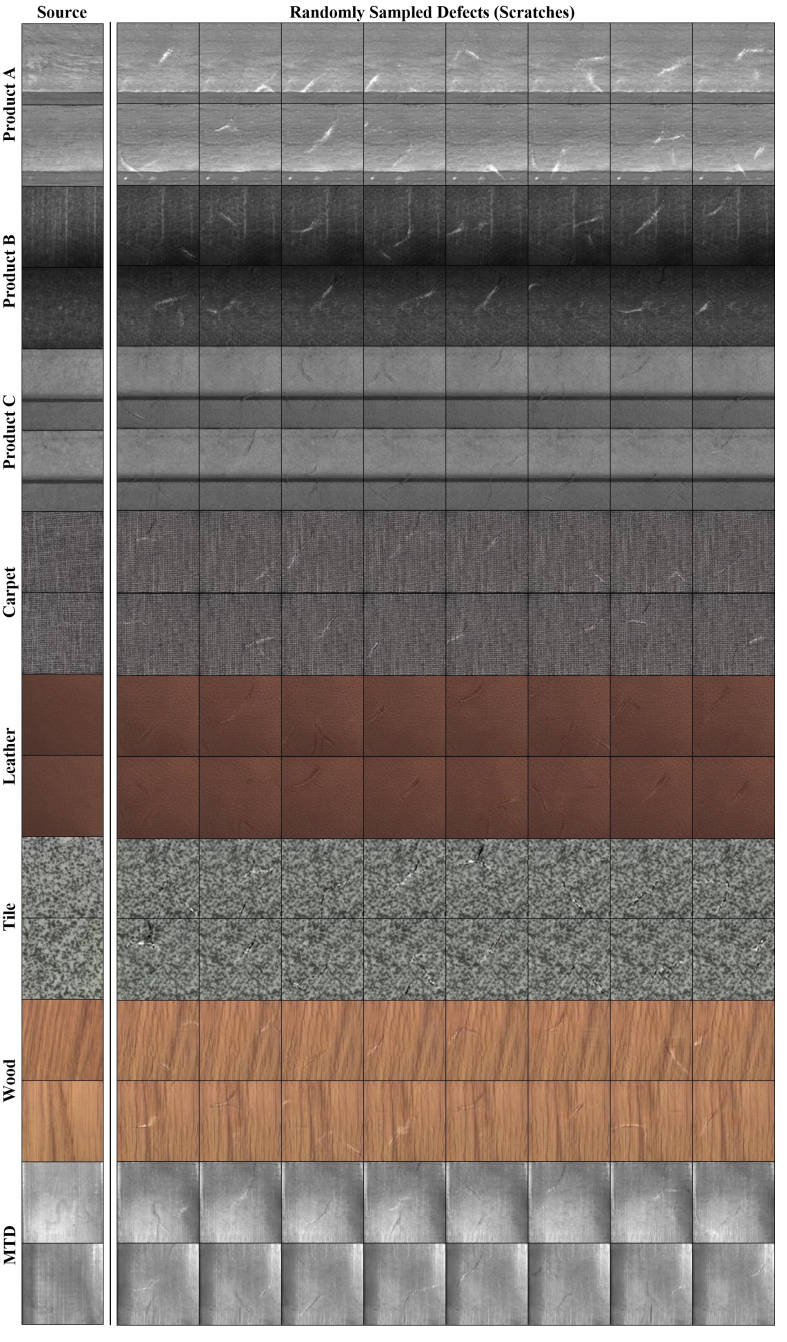


Figure 22: Latent-guided image synthesis results of DT-GAN on all three datasets. The model performs translation from `Normal` to `Scratches`. Note that our model takes input **Source** images as background and only synthesizes the foreground defects from randomly sampled latent code compared to StyleGAN v2 and BigGAN. (Best viewed in color and zoom in.))

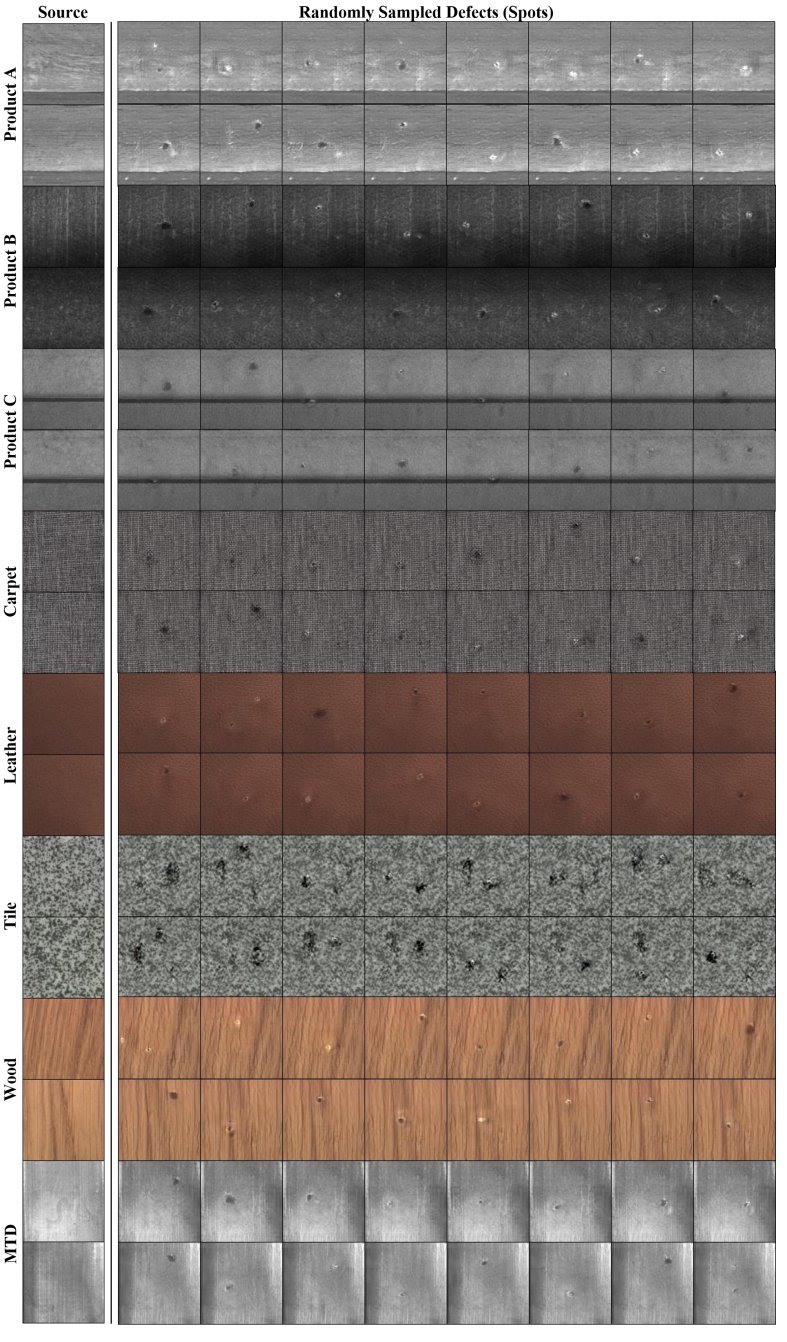


Figure 23: Latent-guided image synthesis results of DT-GAN on all three datasets. The model performs translation from `Normal` to `Spots`. Note that our model takes input **Source** images as background and only synthesizes the foreground defects from randomly sampled latent code compared to StyleGAN v2 and BigGAN. (Best viewed in color and zoom in.)

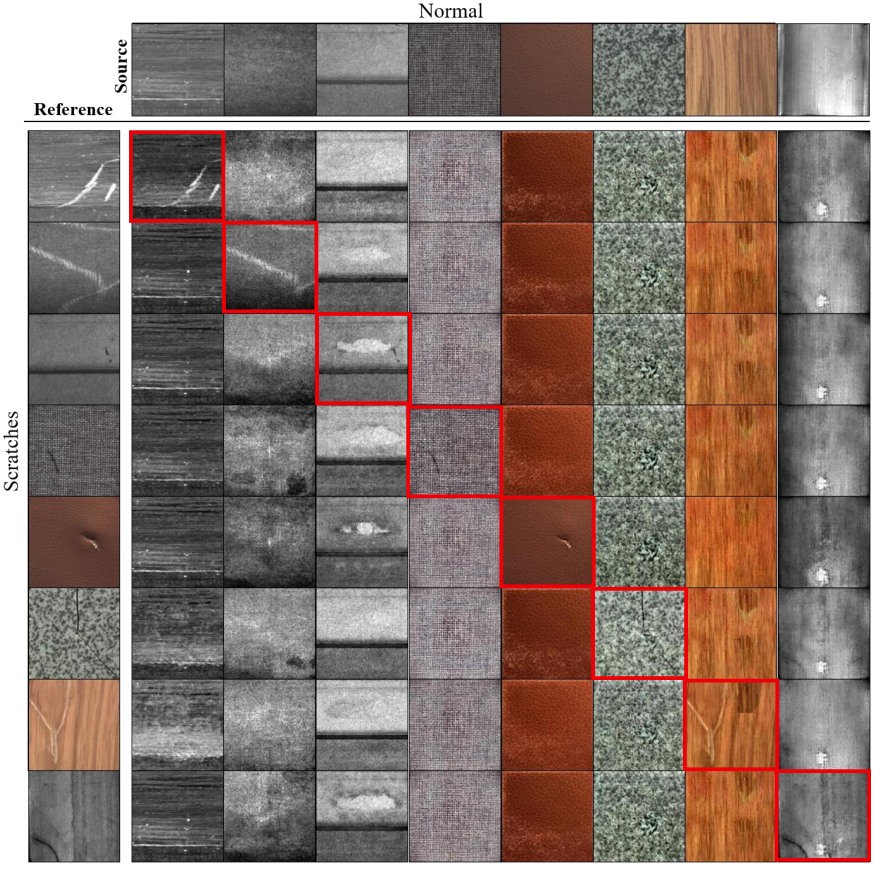


Figure 24: Reference-guided image synthesis results of Mokady et al. on all three datasets. We train a model for each product and each defect type. Then we translate `Normal` images to `Scratches` by taking the **Source** as background and applying the foreground defect from **Reference** to it. Note that the results from the intended translation within a single product can be found on the diagonal and are marked in red. (Best viewed in color and zoom in.)

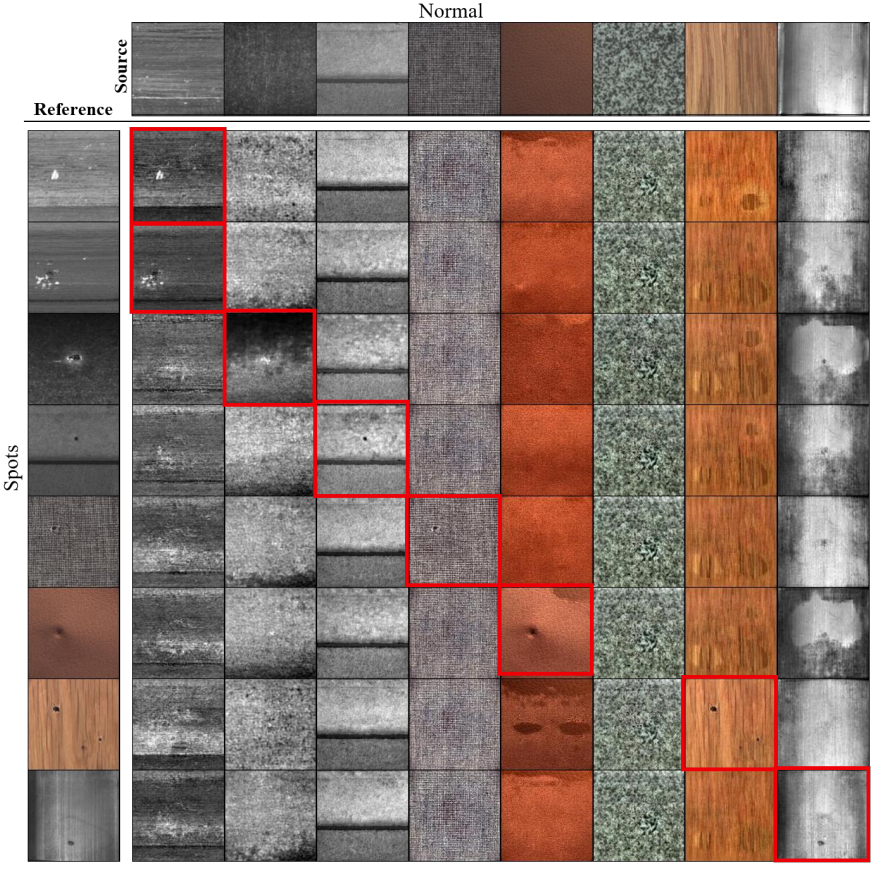


Figure 25: Reference-guided image synthesis results of Mokady et al. on all three datasets. We train a model for each product and each defect type. Then we translate `Normal` images to `Spots` by taking the **Source** as background and applying the foreground defect from **Reference** to it. Note that the results from the intended translation within a single product can be found on the diagonal and are marked in red. (Best viewed in color and zoom in.)

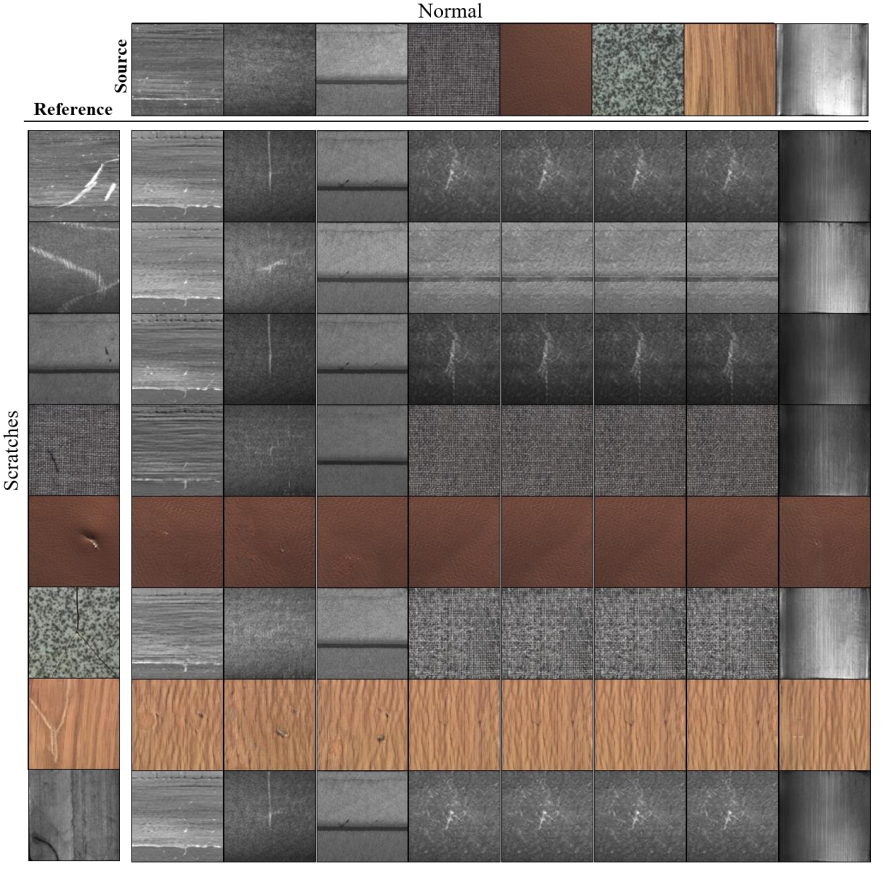


Figure 26: Reference-guided image synthesis results of StarGAN v2 on all three datasets. The model performs translation from `Normal` to `Scratches` by taking the **Source** as background and applying the foreground defect from **Reference** to it. Note that without the style-defect separation and the FG/BG disentanglement, StarGAN v2 not only fails to preserve the background from the given **Source** image but also fails to generate legit defects. (Best viewed in color and zoom in.)

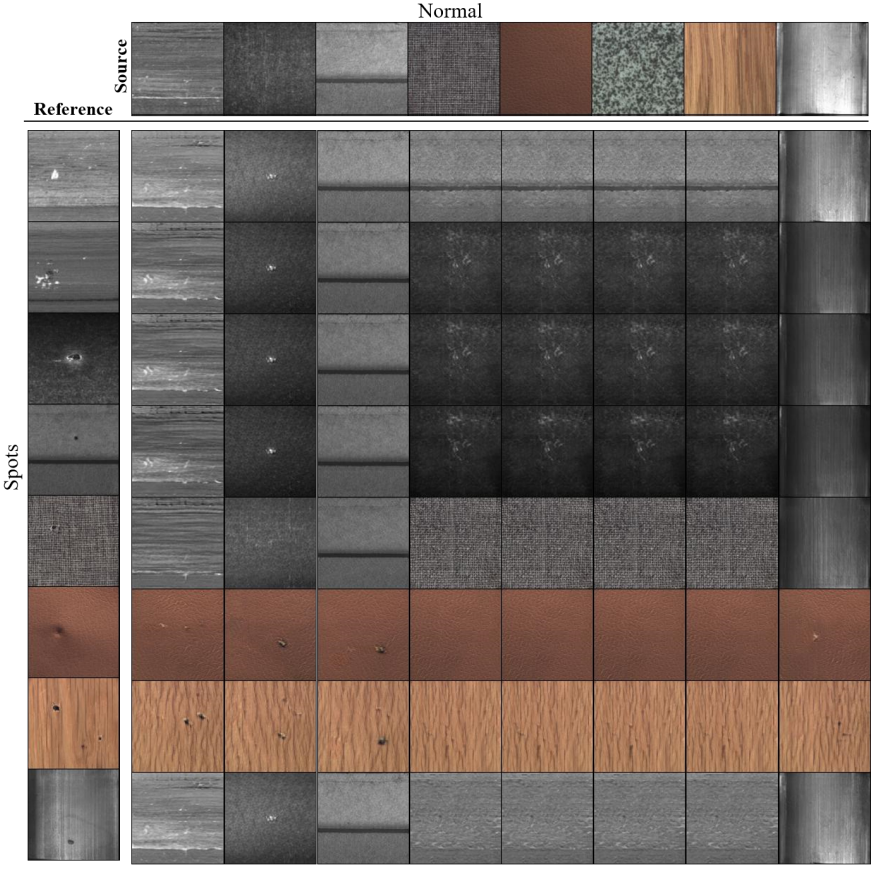


Figure 27: Reference-guided image synthesis results of StarGAN v2 on all three datasets. The model performs translation from `Normal` to `Spots` by taking the **Source** as background and applying the foreground defect from **Reference** to it. Note that without the style-defect separation and the FG/BG disentanglement, StarGAN v2 not only fails to preserve the background from the given **Source** image but also fails to generate legit defects. (Best viewed in color and zoom in.)

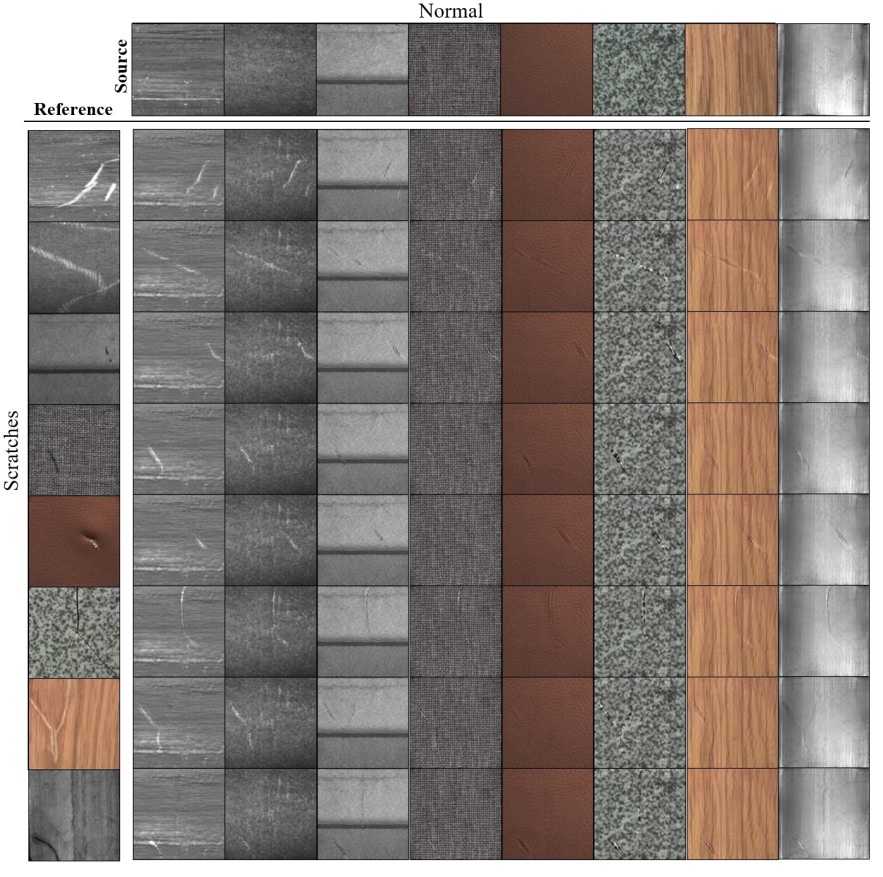


Figure 28: Reference-guided image synthesis results DT-GAN on all three datasets. The model performs translation from `Normal` to `Scratches` by taking the **Source** as background and applying the foreground defect from **Reference** to it. (Best viewed in color and zoom in.)

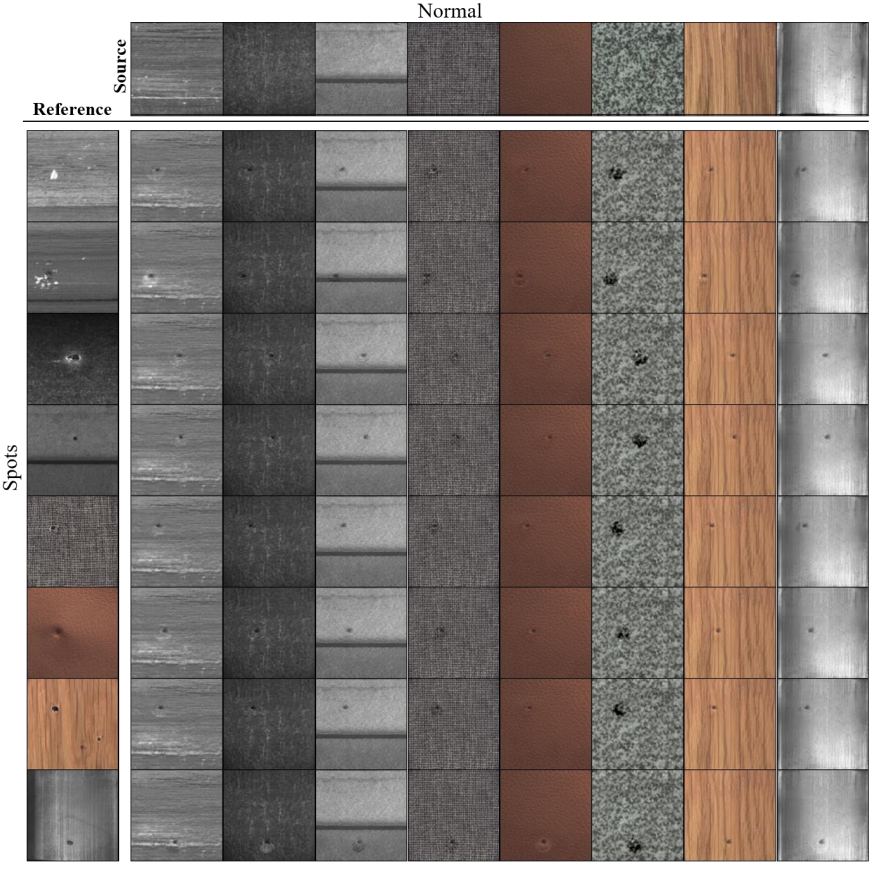


Figure 29: Reference-guided image synthesis results of DT-GAN on all three datasets. The model performs translation from `Normal` to `Spots` by taking the **Source** as background and applying the foreground defect from **Reference** to it. (Best viewed in color and zoom in.)