Let's Get Dirty: GAN Based Data Augmentation for Camera Lens Soiling Detection in Autonomous Driving Supplementary Material

Michal Uřičář[1*], Ganesh Sistu[2], Hazem Rashed[2], Antonín Vobecký[3,2],
Varun Ravi Kumar[2], Pavel Křížek[1*], Fabian Bürger[2] and Senthil Yogamani[2]
[1]Independent Researcher [2]Valeo [3]CTU in Prague
uricar.michal@gmail.com antonin.vobecky@cvut.cz firstname.lastname@valeo.com

1. Generated Data Influence on Soiling Semantic Segmentation

In Figure 1, we depict the problems, we are facing when using the *WoodScape Dataset*. Because only coarse polygonal annotation is available (which is moreover prone to errors), the classical fully supervised training of the semantic segmentation for the soiling is affected by the annotation quality. If we, however, add the generated images along with the precisely generated annotations, the same fully supervised training exploits these annotations, with a desired effect on the semantic segmentation output quality. Thus, in addition to providing novel patterns for augmentation in training, the proposed method also generates precise annotation compared to coarse manual annotation.

2. Network Architectures

In this section, we provide details about the network architectures we used in our experiments.

2.1. Baseline Networks Architecture

Generator and Discriminator of CycleGAN In the CycleGAN generator network (see Figure 2), We use the residual block depicted in Figure 4. Note, that we do not use any kind of normalization, like BatchNorm or InstanceNorm. In Discriminator (see Figure 3, we use Leaky ReLU and a fully convolutional classification layer at the end.

Soiling Mask Segmentation In Figure 5, we show the semantic segmentation network for the soiling mask detection. For the soiling mask segmentation network, we use Instance Normalization in convolutional blocks and BatchNorm in the residual block.

Variational AutoEncoder We depict the encoder part of the VAE in Figure 6. The $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are combined into $\boldsymbol{z}$ via reparametrization trick. The VAE's decoder is shown in Figure 7.

2.2. DirtyGAN Architecture

DirtyGAN is composed of the building blocks used in the baseline algorithm, therefore the networks architecture is mostly the same, only with some minor tweaks. The DirtyGAN scheme is depicted in Figure 10.

3. Generated Data

In Figure 8 and 9, we depict random frame from generated videos, which we enclose to the supplementary. To fit the size limit, we resized the images by $1/4$ for both width and height. The first image in the row is the original frame from the video sequence as it was recorded. The second image in the row is the artificially generated soiled version of the first image and the third image in the row is the soiling mask, which can serve as the automatic annotation for the soiled image. Although it is possible to use our algorithm to generate dynamic soiling effects, we present here only the static soiling. Please, see the video files for the best experience. We have shared a subset of 11 videos with different soiling patterns.

*Most of this work was done while Michal Uřičář and Pavel Křížek were employed by Valeo.

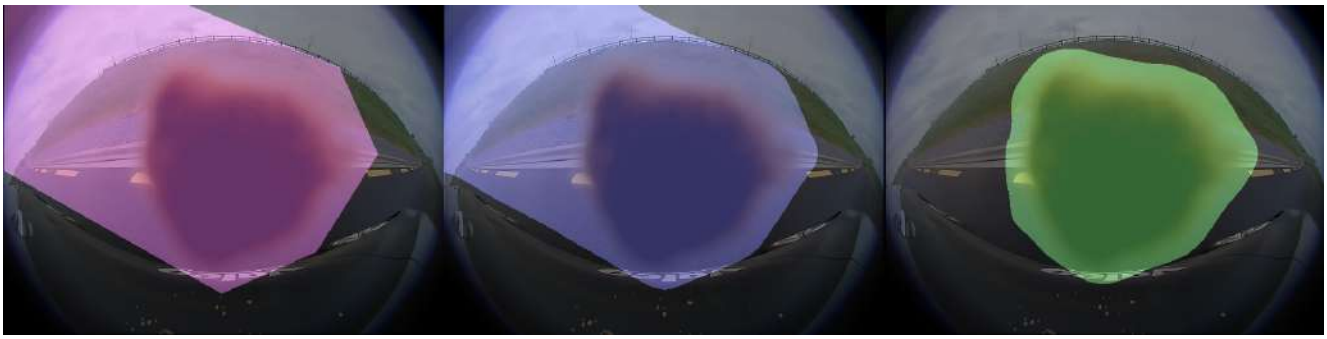

Figure 1: The problem of weak annotation labels and how generated data can help, presented on a testing image, which was not used during the training. The images come from the *WoodScape Dataset*. Left: original weak (polygonal) annotation; Note, that not only it is very coarse, but also prone to errors. Middle: semantic segmentation output, when trained on the weak annotation labels in a fully supervised manner. Right: the same semantic segmentation network output, however, in this case also the generated data were used. Note, how having precise annotation labels for the generated data help in refining the segmentation output.

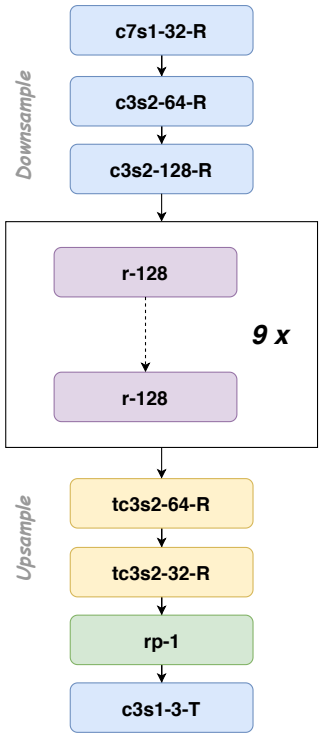

Figure 2: The Generator's architecture, used in the CycleGAN. "c7s1-32-R" reads as 2D convolutional layer with kernel size 7×7 pixels, stride 1, 32 output channels, followed by a ReLU activation layer. T stands for the $\tanh$ activation; "tc" is a shortcut for the transposed convolution; "rp" means reflection padding; "r-128" is a shorthand for residual block with 128 channels.

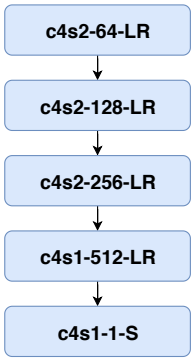

Figure 3: The Discriminator's architecture, used in the CycleGAN. "c4s2-64-LR" reads as 2D convolutional layer with kernel size 4×4 pixels, stride 2, 64 output channels, followed by a Leaky ReLU activation layer. S stands for the sigmoid activation layer.

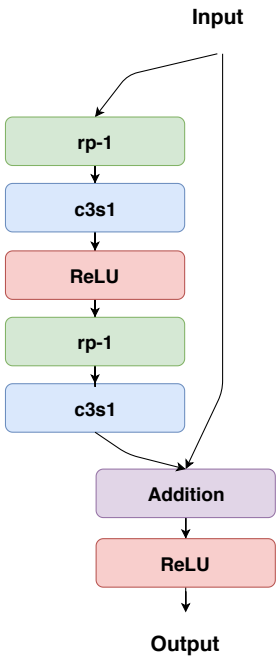

Figure 4: The residual block, which is used in the Generator's architecture. Note, that convolutional blocks "c3s1" might be followed by a BatchNorm layer (used only in the soiling mask segmentation network).

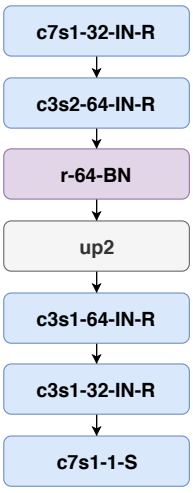

Figure 5: The soiling mask segmentation network. The same type of shorthands as in previous figures are used with an addition of "IN" for Instance Normalization and "BN" for BatchNorm layers. "up2" is a simple nearest neighbor upsampling with scale factor 2.

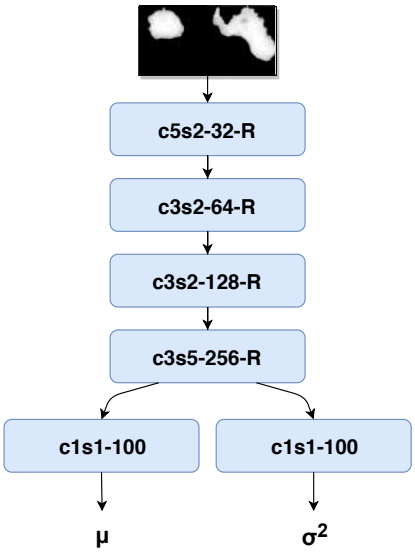

Figure 6: The architecture of the VAE encoder used in our experiments. The notation "c3s2-64-R" corresponds to a 2D convolutional layer, with kernel size 3×3 pixels, stride of 2, 64 output channels, followed by a ReLU activation layer.

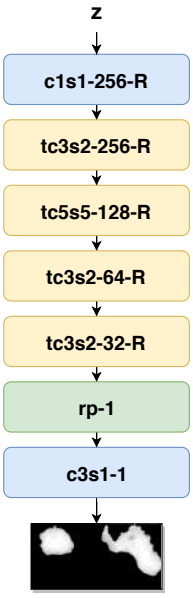

Figure 7: The architecture of the VAE decoder used in our experiments. The notation "c3s2-64-R" corresponds to a 2D convolutional layer, with kernel size 3×3 pixels, stride of 2, 64 output channels, followed by a ReLU activation layer. "tc3s2-256-R" represents a transposed convolution, with kernel 3×3 pixels, stride 2, 256 output channels, followed by ReLU activation. "rp-1" means a reflection padding of size 1.

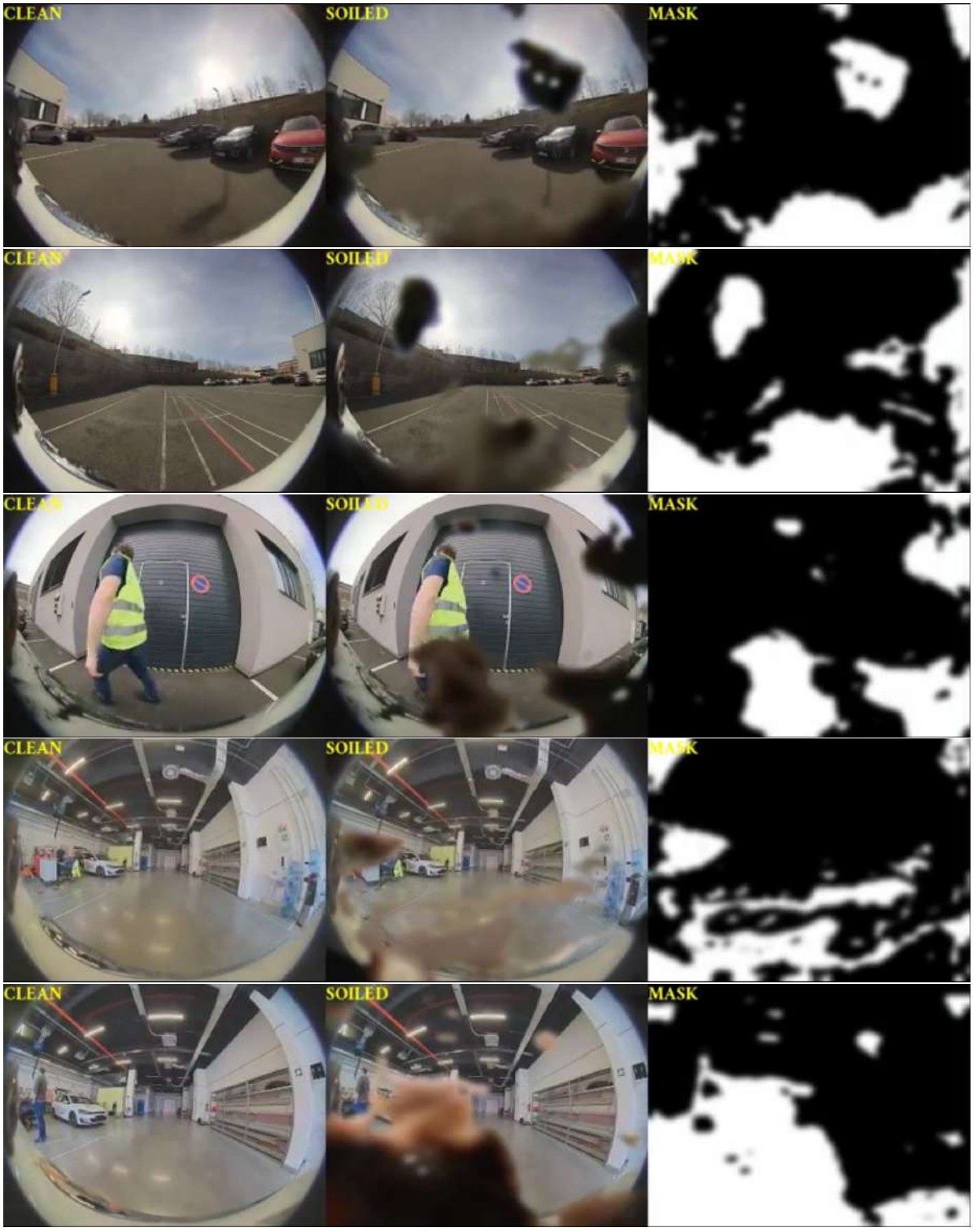

Figure 8: Several frames extracted from the supplementary video files. Please, see the original video files for the best experience.

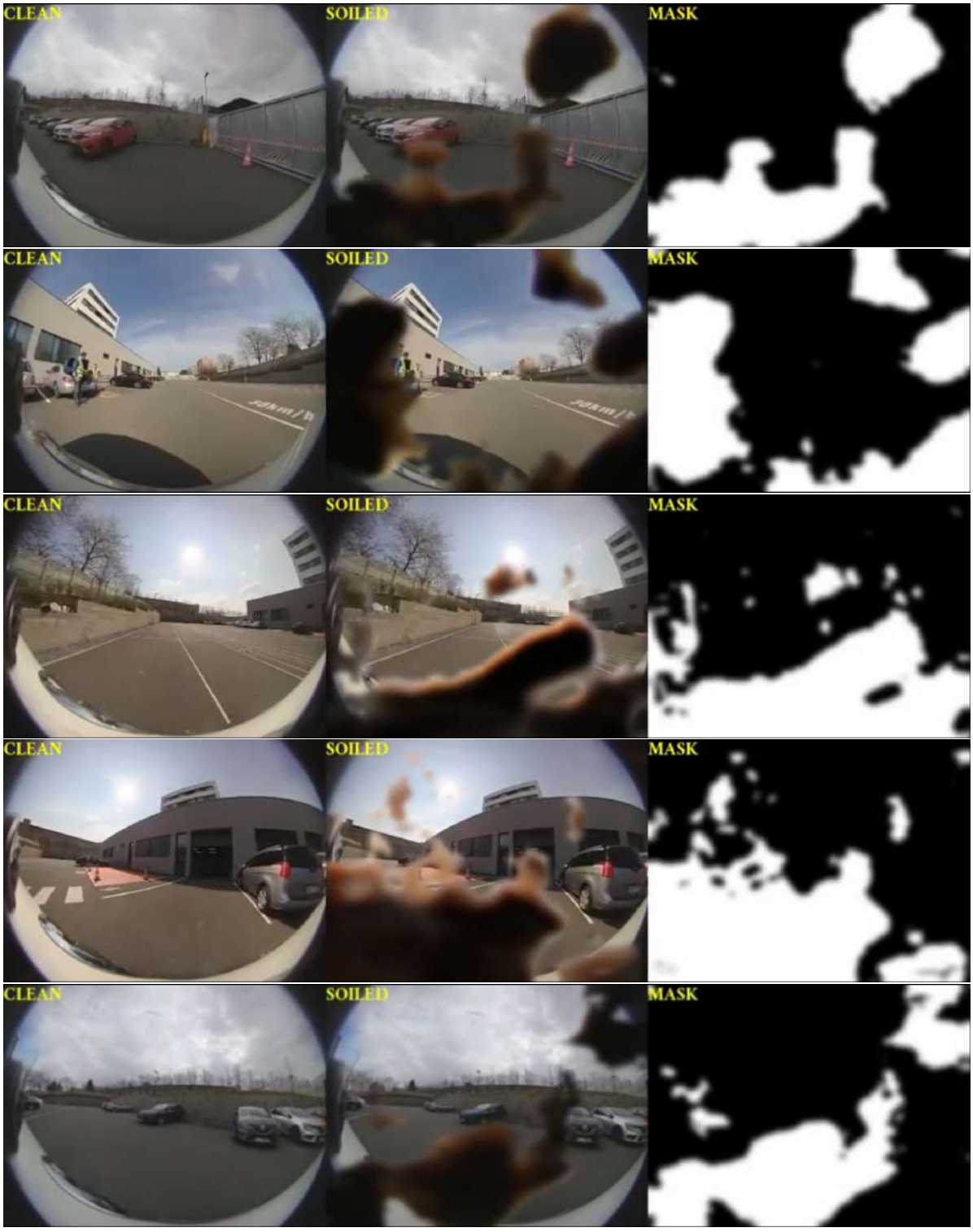

Figure 9: Several frames extracted from the supplementary video files. Please, see the original video files for the best experience.

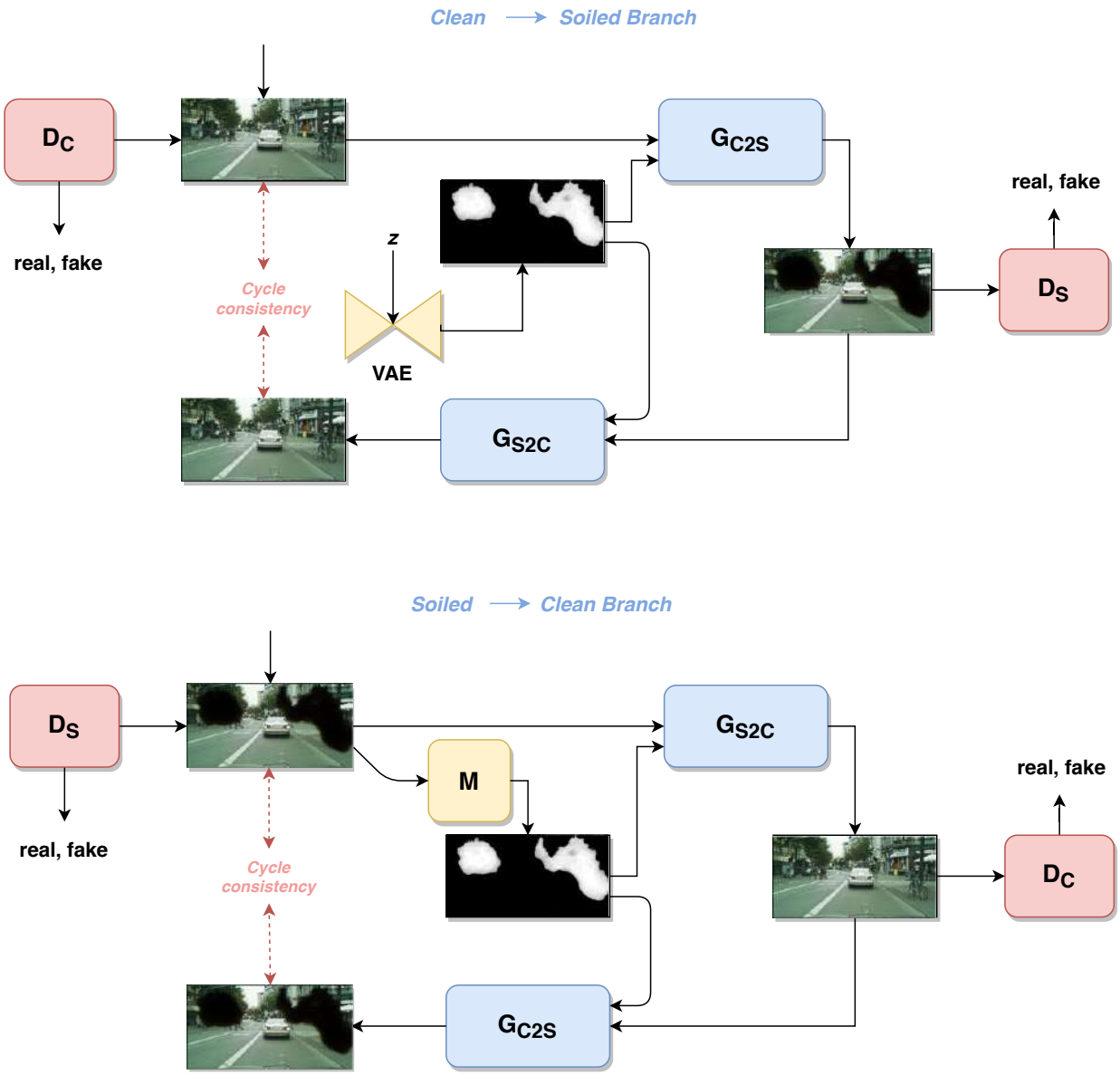

Figure 10: DirtyGAN scheme.