A. Filter Intensities and their Respective Parameters

In Section 4.1 we introduce abstract levels of intensities for each filter we apply to an image. We now map each intensity to actual parameters passed to editing libraries to achieve the given filter intensity.

	Distortion	Intensity	Actual Parameters
Technical	JPEG compression	[0, . . . +4, +5]	The editing library accepts the technical intensities as is.
	Defocus blur	[0, . . . +4, +5]	
	Motion blur	[0, . . . +4, +5]	
	Pixelate	[0, . . . +4, +5]	
	Gaussian noise	[0, . . . +4, +5]	
	Impulse noise	[0, . . . +4, +5]	
Style	Brightness	[-5, -4, . . . , +4, +5]	$[-1.0, -0.8, \ldots, 0.8, 1.0]$
	Contrast	[-5, -4, . . . , +4, +5]	$[-1.0, -0.8, \ldots, 0.8, 1.0]$
	Saturation	[-5, -4, . . . , +4, +5]	$[-1.0, -0.8, \ldots, 0.8, 1.0]$
	Exposure	[-5, -4, . . . , +4, +5]	$[-3.0, -2.4, \ldots, 2.4, 3.0]$
	Shadows	[-5, -4, . . . , +2, +3]	$[-100, -60, -20, 20, 40, 50, 60, 80, 100]$
	Highlights	[-3, -2, . . . , +4, +5]	$[-100, -80, -60, -50, -40, -20, 20, 60, 100]$
	Temperature	[-4, -3, . . . , +4, +5]	[2000, 3000, 5000, 6000, 6500, 7000, 8000, 10 000, 14 000, 18 000]
	Tint	[-5, -4, . . . , +4, +5]	$[0.75, 0.8, \ldots, 1.2, 1.25]$
	Vibrance	[-2, -1, . . . , +3, +4]	[0, 20, 25, 40, 60, 80, 100]
Composition	Rotation	[-5, -4, . . . , +4, +5]	[10° ↻, 8° ↻, . . . , 8° ↻, 10° ↻]
	Horizontal crop	[-5, -4, . . . , +4, +5]	We resize the image to 336px and then crop patches of size 224px from the resulting image. Intensity 0 is a centercrop, while a $\lvert intensity \rvert == 5$ results in a crop from the images' border.
	Vertical crop	[-5, -4, . . . , +4, +5]	
	Left Diagonal crop	[-5, -4, . . . , +4, +5]	
	Right Diagonal crop	[-5, -4, . . . , +4, +5]	
	Image Ratio	[-5, -4, . . . , +4, +5]	[stretch along y-axis 100 %, y80 %, . . . , x80 %, stretch along x-axis 100 %]

Table 3. Actual parameters and implementation specifics for each distortion and intensity level. Technical parameters are passed to imagenet-c [12] and style parameters to darktable [37] while compositional distortions are implemented by us.

B. Dataset Content Analysis

To show that the images of our dataset (Section 4.2) contain a large variety of contents, we apply a pretrained DenseNet121 [16] for image classification and RetinaNet [25] for object detection on our newly introduced dataset. We find that the images of our dataset spread across many different classes and contain a wide variety of objects and subjects.

most common classes		**most common objects**	
class	count	object	count
seashore	3554	person	44301
alp	2568	car	3186
lakeside	2446	cup	2880
fountain	2265	bird	2788
valley	2011	cell phone	1749
miniskirt	1455	boat	1618
gown	1430	dog	1581
bikini	1176	potted plant	1580
. . .	. . .	. . .	. . .
sloth_bear	2	refrigerator	31
affenpinscher	2	snowboard	25
patas	1	skis	23
Sealyham_terrier	1	hair dryer	7
Japanese_spaniel	1	toaster	7

Table 4. Most commonly detected classes and objects in the images of our dataset.
Full list: https://github.com/janpf/self-supervised-multi-task-aesthetic-pretraining

C. Baseline Implementation Details

In the following, we give implementation details on the self-supervised baseline methods we use in our experiments. To allow for a fair comparison, we use the same MobileNetV2 [33] architecture for all methods. Each model is initialized with ImageNet weights. Additionally, all algorithms are applied to our collected highly aesthetic dataset to make sure that all methods have access to the same images while pretraining. Fine-tuning on AVA [29] stays the same for all pretrained models.

C.1. RotNet

For RotNet [22], we use the implementation from https://github.com/gidariss/FeatureLearningRotNet, which is the official code from the paper. We follow the same procedure as in the original paper and only change the given network architecture and dataset.

C.2. SimCLR

For this baseline [2], we take the implementation from https://github.com/Spijkervet/SimCLR. While this is not the official implementation, the authors were able to reproduce the results from the paper. The code provides the choice between the Adam optimizer and the LARS optimizer [2]. For our experiments we selected the latter with a Cosine annealing learning rate schedule as in the original paper. We are therefore positive that this is comparable to the original implementation.

D. Accuracy of Classification Multi-Task

We found that the accuracy of the classification layer predicting the applied distortion vastly differs between the different aesthetic aspects, as discussed in Section 5.

ACC	technical	style	composition
ranking+classification	0.445	0.127	0.257
random	0.167	0.111	0.167

Table 5. Prediction accuracy for the correct distortion by the classification layer. Included is a random baseline guessing a random distortion.