WISE: Whitebox Image Stylization by Example-based Learning

Winfried Lötzsch[1⋆], Max Reimann[1⋆], Martin Büssemeyer[1], Amir Semmo[2], Jürgen Döllner[1], and Matthias Trapp[1]

[1] Hasso Plattner Institute, University of Potsdam, Germany
[2] Digital Masterpieces GmbH, Germany

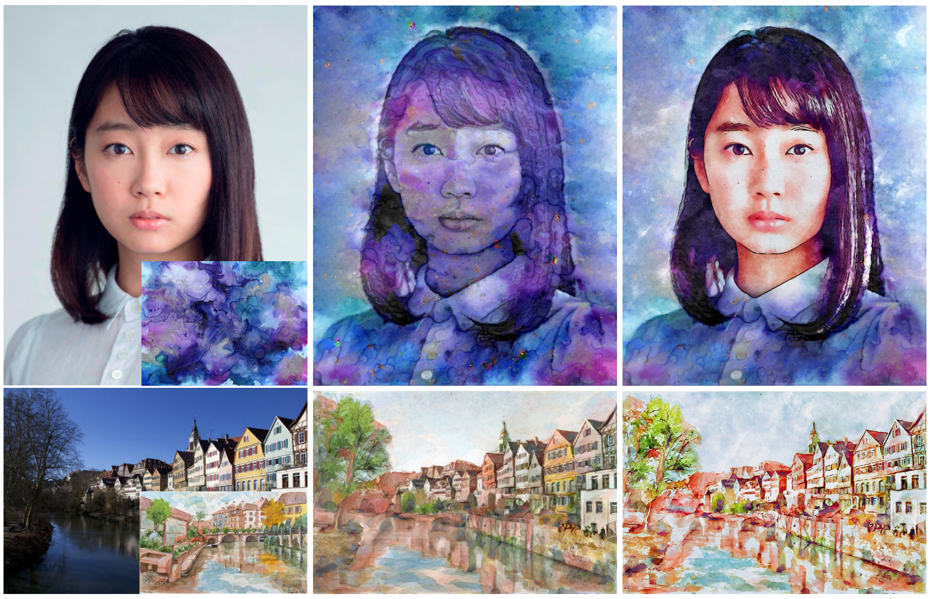
Abstract. Image-based artistic rendering can synthesize a variety of expressive styles using algorithmic image filtering. In contrast to deep learning-based methods, these heuristics-based filtering techniques can operate on high-resolution images, are interpretable, and can be parameterized according to various design aspects. However, adapting or extending these techniques to produce new styles is often a tedious and error-prone task that requires expert knowledge. We propose a new paradigm to alleviate this problem: implementing algorithmic image filtering techniques as differentiable operations that can learn parametrizations aligned to certain reference styles. To this end, we present WISE, an example-based image-processing system that can handle a multitude of stylization techniques, such as watercolor, oil or cartoon stylization, within a common framework. By training parameter prediction networks for global and local filter parameterizations, we can simultaneously adapt effects to reference styles and image content, e.g., to enhance facial features. Our method can be optimized in a style-transfer framework or learned in a generative-adversarial setting for image-to-image translation. We demonstrate that jointly training an XDoG filter and a CNN for postprocessing can achieve comparable results to a state-of-the-art GAN-based method. https://github.com/winfried-loetzsch/wise

1 Introduction

Image stylization has become a major part of visual communication, with millions of edited and stylized photos shared every day. At this, a large body of research in Non-photorealistic Rendering (NPR) has been dedicated to imitating hand-drawn artistic styles [37,53]. Traditionally, such *heuristics-based algorithms* [57] for image-based artistic rendering emulate a certain artistic style using a series of specifically developed *algorithmic* image processing operations. Thus, creating new styles is often a time-consuming process that requires the knowledge of domain experts.

Recently, deep learning-based techniques for stylization and image-to-image translation have gained popularity by enabling the learning of stylistic abstractions from example data. In particular, Neural Style Transfer (NST) [28,27] that

⋆ Both authors contributed equally to this work.



(a) Content/Style (b) *WISE* / without edits (c) *WISE* / with local edits

Fig. 1: **Example-based effect adjustment.** *WISE* optimizes effect parameters, e.g., of a watercolor stylization effect, to match stylized outputs to a reference style a. The results b can then be interactively adjusted by tuning the obtained parameters globally and locally for increased artistic control c[3].

transfers the artistic style of a reference image and Generative Adversarial Network (GAN)-based [15,24] methods for fitting style distributions have achieved impressive results and are increasingly used in commercial applications [1].

Classical heuristics-based filters and filter-based image stylization pipelines, such as the eXtended difference-of-Gaussians (XDoG) filter [65], cartoon effect [66], or watercolor effect [3,63], expose a range of parameters to the user that enable fine-grained global and local control over artistic aspects of the stylized output. By contrast, learning-based techniques are commonly limited in their modes of control, i.e., NST [10] only offers control over a general content-style tradeoff. Furthermore, their learned representations are generally not interpretable as a set of design aspects and configurations. Thus, these approaches often do not meet the requirements of interactive image editing tasks that go beyond one-shot global stylizations towards editing with high-level and low-level artistic control [23,19,14]. Additionally, deep network-based methods are often computationally expensive in both training and inference on high image resolutions [10,30,31]. This further limits their applicability in interactive or mobile applications [13] and their capability to simulate fine-grained (pigment-based) local effects and phenomena of artistic media such as watercolor and oil paint.

[3] View examples of editing in our supplemental video: https://youtu.be/wIndN7cr0PE

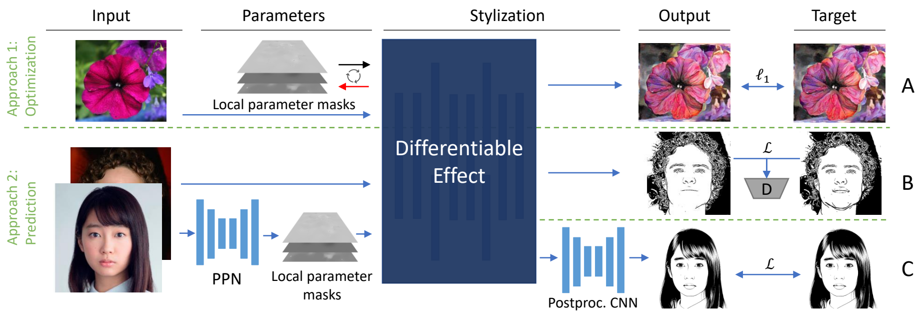


Fig. 2: **Overview of *WISE*.** Differentiable effects can be adapted to example data, demonstrated for three use-cases: *parametric style transfer* (**A**) optimizes parameter masks to match a hand-drawn or synthesized stylization target (e.g., from NST as in Fig. 1) which enables style transfer results to remain editable and resolution-independent. *Local parameter prediction* (**B**) trains PPNs to predict parameter masks to adapt the effect to the content (e.g., for facial structure enhancement as shown here). Combined with a postprocessing CNN, local parameter prediction can learn sophisticated *image-to-image translation* tasks such as learning hand-drawn sketch-styles (**C**).

To counterbalance these limitations, this work aims to combine the strengths of *heuristics-based* and *learning-based* image stylization by implementing algorithmic effects as *differentiable* operations that can be trained to learn filter-based parameters aligning to certain reference styles. The goal is to enable (1) the creation of complex, example-based stylizations using lightweight algorithmic approaches that remain interpretable and can operate on very high image resolutions, and enable (2) the editing with artistic control on a fine-granular level according to design aspects. To this end, we present *WISE*, a *whitebox* system for example-based image processing that can handle a multitude of stylization techniques in a common framework. Our system integrates existing algorithmic effects such as XDoG-based stylization [65], cartoon stylization [66], watercolor effects [3,63], and oilpaint effects [56], by creating a library of differentiable image filters that match their shader kernel-based counterparts. We show that the majority of filters (e.g., bilateral filtering) can be transformed into auto-differentiable formulations, while for the remaining filters, gradients can be approximated (e.g., for color quantization). Using our framework, effects can be adapted to reference styles using popular, deep network-based image-to-image translation losses. We train exemplary effects using both NST and GAN-based losses and show qualitatively and quantitatively that the results are comparable to state-of-the-art deep networks while retaining the advantages of filter-based stylization. To summarize, this paper contributes the following:

1. It provides an end-to-end framework for example-based image stylization using differentiable algorithmic filters. Fig. 2 shows an overview of the system.
2. It demonstrates the applicability of style transfer-losses to adapt stylization effects to a reference style (Fig. 1 and Fig. 2 A). The results remain editable and resolution-independent.

3. It shows that both global and local parametrizations of stylization effects can be optimized as well as predicted by a parameter prediction network (PPN). The latter can be trained on content-adaptive tasks (Fig. 2 B).
4. It demonstrates that filters can be trained in combination with CNNs for improved generalization on image-to-image translation tasks. Combining the XDoG effect with a simple post-processing Convolutional Neural Network (CNN) (Fig. 2 C) can achieve comparable results to state-of-the-art GAN-based image stylization for hand-drawn sketch styles, but at much lower system complexity.

2 Related Work

Heuristics-based Stylization. In NPR, image-based artistic rendering deals with emulating traditional artistic styles, using a pipeline of rendering stages [37,53,57]. Commonly, edge detection and content abstraction are important parts of such pipelines. The XDoG filter [65,66] is an extended version of the Difference-of-Gaussians (DoG) band-pass filter, and can be used to create smooth edge stylizations. Furthermore, edge-aware smoothing filters such as bilateral filtering [62] or Kuwahara filtering [36] can abstract image contents, and can be combined with image flow to adapt the results to local image structures [38,40]. These techniques can be found in heuristics-based effects such as cartoon stylization [66], oil-paint abstraction [58] and image watercolorization [3,63], each consisting of a series of rendering stages such as image blending, wobbling, pigment dispersion and wet-in-wet stylization. For a comprehensive taxonomy of techniques, the interested reader is referred to the survey by Kyprianidis *et al.* [37].

These effects are typically parameterized globally, and can be further adjusted within pre-defined parameter ranges, or locally on a per-pixel level using parameter masks [58]. In this work, we implement variants of the XDoG, cartoon filtering, and watercolor pipeline in our framework using auto-differentiable formulations of each rendering stage. At runtime, users choose between one of these different effect pipelines; and results are generally achieved by optimizing the exact chain of filters as introduced in [65,66,3,63,56].

Deep Learning-based Methods. With the advent of deep learning, CNNs for image generation and transformation have led to a range of impressive results.

In NST, first introduced by Gatys *et al.* [10], the stylistic characteristics of a reference image are transferred to a content image by matching deep feature statistics using an optimization process. Fast feed-forward networks have been trained to reproduce a single [28] or even arbitrarily many styles [22,16,47]. Furthermore, there have been efforts to increase controllability of NSTs, e.g., by control over color [11], sub-styles [51] or strokes [26,50], however, the representations are not interpretable. Recently, style transfer has been formulated as a neurally-guided stroke rendering optimization approach [35], that retains interpretability, however, is slow to optimize. We show that our method can obtain comparable results to a state-of-the-art NST [33], while retaining interactive editing control.

GANs, first introduced by Goodfellow *et al.* [15], learn powerful generative networks that model the input distribution. They have been widely used for conditional image generation tasks, with both paired [24] and unpaired [71] training data. In the stylization domain, it has found applications for collection style transfer [6,55], cartoon generation [7,61], and sketch styles [68]. However, domain-specific applications often require sophisticated losses and multiple networks to prevent artifacts [7,69]. We show that our differentiable implementation of the XDoG filter can be trained as a generator network in a GAN framework and can produce comparable sketches to state-of-the-art (CNN-based) GANs [68,69].

Learnable Filters. While the previous end-to-end CNNs deliver impressive results, they are limited in their output resolution. A few recent methods have proposed training fast algorithmic filters to operate efficiently at high resolutions. Getreuer *et al.* [12] introduce learnable approximations of algorithmic image filters, such as of the XDoG filter. At run-time, a linear filter is selected per image pixel according to the local structure tensor; filters can be combined in pipelines for image enhancement [9]. Gharbi *et al.* [13] train a CNN to predict affine transformations for bilateral image enhancement, e.g., to approximate edge-aware image filters or tone adjustments. The transform filters are predicted at a low resolution and then applied in full resolution to the image. "Exposure" framework [21] combines learning linear image filters with reinforcement learning, where an actor-critic model decides which filters to include to achieve a desired photo enhancement effect.

These methods have in common that they learn several simple, linear functions to approximate image processing operations [12,67,13,21,45]. Our framework consists of pipelines of differentiable filters as well. However, in contrast to previous work, we make a variety of heuristics-based stylization operators differentiable and learn to predict their parameterizations. Thereby, sophisticated stylization effects (e.g., those found in stylization applications) can be ported and directly used in our framework.

3 Differentiable Image Filters

Heuristics-based stylization effects consist of pipelines of image filtering operations. To compute gradients for effect input parameters, all image operations within the pipeline are required to be differentiable with respect to their parameters and the image input. Gradients throughout the pipeline can then be obtained by applying the chain rule. Fig. 3 outlines the gradient flow from a loss function to the effect parameters by the cartoon pipeline example, gradients for parameters can be obtained both globally as well as locally using per-pixel parameter masks.

In previous works, individual operations in such pipelines are typically implemented as shader kernels for fast GPU-based processing. To achieve an end-to-end gradient flow, we implement these operations in an auto-grad enabled

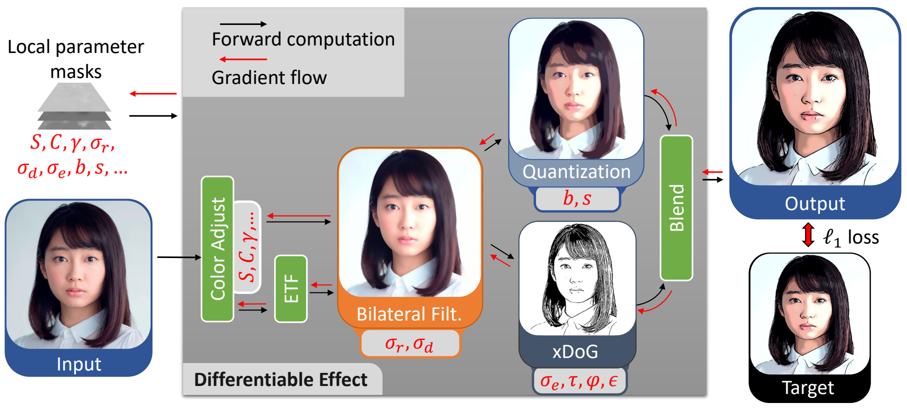


Fig. 3: **Exemplary differentiable effect pipeline.** Shown for the cartoon effect proposed by Winnemöller *et al.* [66]. Gradients are backpropagated from the loss to the filter parameter masks. In the effect, colors are first adjusted based on parameters such as saturation S, contrast C or gamma γ, and then, using an Edge Tangent Flow (ETF) [29], orientation-aligned bilateral filtering [38], and XDoG [65] are computed. Additionally, the image is quantized with respect to the number of bins b and softness s.

framework, the implemented filtering stages are listed in Tab. 1. Point-based and fixed-neighbourhood operations such as color space conversions, structure tensor computation [38], or DoG [65] can generally be converted into differentiable filters by transforming any kernelized function into a sequence of its constituent auto-differentiable transformations. The exception to this are functions which are inherently not differentiable, such as color quantization, and which require the approximation of a numeric gradient. An example for numeric gradient approximation of color quantization is shown in the supplemental material. Structure-adaptive neighborhood operations, such as the orientation-aligned bilateral filter and flow-based Gaussian smoothing filter, iteratively determine sampling locations based on the structure of the content (often oriented along a flow field). To preserve gradients and make use of the in-built fixed-neighborhood functions of auto-grad enabled frameworks, per-pixel iteration is transformed into a grid-sampling operation where neighborhood values are accumulated into a new dimension with size D which represents the expected maximum kernel neighbourhood. A structure-adaptive filter transformation by example of the orientation-aligned bilateral filter is shown in the supplementary material.

Implementation Aspects. We implement differentiable filters in PyTorch and create reference implementations of the same effects using OpenGL shaders. The learnability of each effect parameter is validated in a functional benchmark (shown in supplemental material) by optimizing the differentiable effect to match reference effects with randomized parameters. At inference time, OpenGL shaders can be interchangeably used with the differentiable effect, and can be efficiently executed on high-resolution images using parameters predicted on low-

Table 1: Differentiable filters by type and effect. Filters can be classified by their sampling approach, which is either point-based (PB) or in a fixed neighborhood (FN) or structure-adaptive neighborhood (SN). Some filters are non-differentiable and require numerical gradient (NG) approximations for training.

Filtering operation	Differentiable Filter Type	Cartoon	Watercolor	Oilpaint
Anisotropic Kuwahara [40]	SN		✓	
Bilateral [62]	FN		✓	✓
Bump Mapping / Phong Shading [48]	FN			✓
Color Adjustment	PB	✓	✓	✓
Color Quantization [66]	PB,NG	✓		
Flow-based Gaussian Smoothing [38]	SN	✓	✓	✓
Gaps [44]	FN		✓	
Joint Bilateral Upsampling [34]	SN			✓
Flow-based Laplacian of Gaussian [39]	SN			✓
Image Composition [49]	PB		✓	
Orientation-aligned Bilateral [38]	SN	✓		✓
Warping / Wobbling [3]	FN		✓	
Wet-in-Wet [63]	SN		✓	
XDoG [65]	SN	✓	✓	✓

resolution images. At training time, memory usage of differentiable filters can be reduced by controlling their kernel-size (shown in supplemental material).

4 Parameter Prediction

With the introduced differentiable filter pipelines, parameters can be optimized using image-based losses. To generalize to unseen data, we explore Parameter Prediction Networks (PPNs) that are trained to predict global parameters or spatially varying (local) parameters.

4.1 Parameter Prediction Networks

Global Parameter Prediction. We construct a PPN that predicts the effect parameters of a stylized example image, given both the stylized and source image. Thereby, the network is trained to effectively reverse-engineer the stylization effect. During training, gradients are back-propagated through the effect, the parameters, and finally to the PPN. Formally, let I denote the input image, T the target image, $O(\cdot)$ the differentiable effect, $P_G(\cdot)$ the PPN network. The loss for the global PPN is computed using an ℓ_1 image space-based loss as:

$$\mathcal{L}_{\text{global}} = \|O(I, P_G(I))) - T\|_1 \tag{1}$$

Our global PPN architecture consists of a VGG backbone [59] that extracts features of the input and stylized image and computes layer-wise Gram matrices

Table 2: **Global PPN loss functions.** The PPNs are trained with different loss functions on the NPR benchmark [54]. Networks trained in parameter space use reference parameters as the loss signal directly. We measure the Structural Similarity Index Measure (SSIM) and Peak Signal-to-Noise Ratio (PSNR)

Loss domain	SSIM	PSNR	Parameter loss
Parameter space ℓ_1/ℓ_2	0.738/0.737	12.530/12.927	**0.158**/0.162
Image space ℓ_1/ℓ_2	**0.780**/0.764	**13.875**/13.286	0.183/0.190

to encode important style information [10]. The accumulated features are passed to a multi-head module to predict the final global parameters. We found this network architecture to perform superior against other common architecture variants, please refer to the supplementary material for an ablation study and details on the architecture.

Local Parameter Prediction. While the global PPN can predict settings of similar algorithmic effects, real-world, hand-drawn images often vary significantly based on the local content, which cannot be modeled by a global parameterization. Therefore, we construct a PPN to predict local *parameter masks.* We use a U-Net architecture [52] for mask prediction at input resolution, where each output channel represents a parameter. Gradients are back-propagated to the PPN through the differentiable effects for all parameter masks.

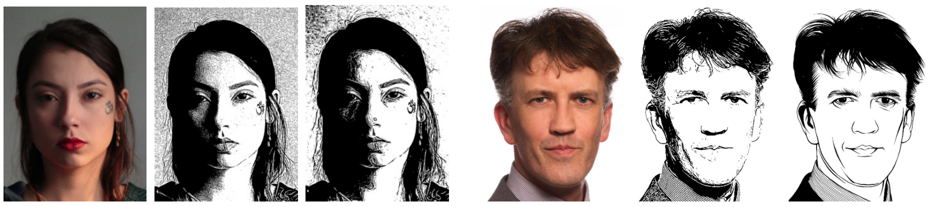
For training, a paired data GAN approach is used, where a patch-based Pix2Pix discriminator [24] matches the distribution of patches in the reference image and an additional weighted ℓ_1 image space loss enforces a more strict pixel-wise similarity. Formally, let $D(\cdot)$ denote the discriminator, $\mathcal{L}_{TV}(\cdot)$ the total variation regularizer [28] to enforce smooth parameter masks, and $P_L(\cdot)$ the local PPN which acts as the generator network. The final loss $\mathcal{L}$ for the PPN generator is computed as:

$$\begin{aligned}\mathcal{L} = \mathbb{E}\big[&\alpha \log(1 - D(I, O(I, P_L(I)))) \\ &+ \beta \|O(I, P_L(I)) - T\|_1 + \gamma \mathcal{L}_{TV}(P_L(I))\big]\end{aligned} \tag{2}$$

4.2 PPN Experiments

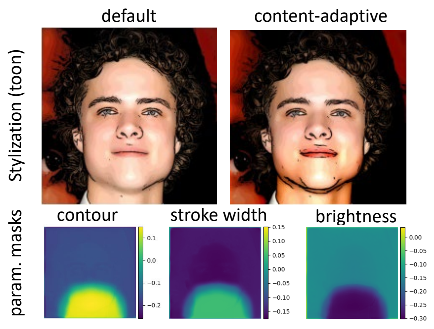
We conduct several experiments to validate our approach for global and local parameter prediction.

Global Parameter Prediction. We compare the loss function and loss space of global PPNs in (Tab. 2). We find that while directly predicting in parameter space (without obtaining gradients from the effect) yields closer parameter values, the highest visual accuracy is achieved using an ℓ_1 image space-based loss. This validates the usefulness of the differentiable effect being part of the training pipeline. Global PPNs can accurately match reference stylizations created by the same effect, as shown in Fig. 4b. Furthermore, they can approximate similar hand-drawn styles, albeit with significant local deviations, as shown in Fig. 4e.

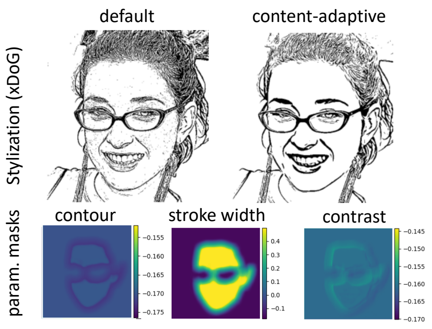


(a) Source (b) Predicted (c) Reference (d) Source (e) Predicted (f) Reference

Fig. 4: **Predictions using the global PPN for XDoG.** The stylized reference c is synthetic (generated using the reference XDoG implementation), while the reference f is hand-drawn, taken from APDrawing [68].



(a) Chin contour enhancement



(b) Selective detail reduction

Fig. 5: **Local PPN results.** Networks are trained on CelebAMask-HQ [41] to generate selective enhancements by predicting parameter masks. The default result is created by a global parameter configuration, while the shown predicted parameter masks create the content-adaptive result.

Content-adaptive Effects. Using the previously described approach for local PPN training, we demonstrate its applicability to several common problems that are often present in purely algorithmic image-stylization techniques. We consider three tasks to improve stylization quality: (1) highlighting facial features, for example by increasing contours at low-contrast edges such as the chin (Fig. 5a), (2) selectively reducing details such as small wrinkles in the face (Fig. 5b), and (3) background removal (refer to supplemental material). We use the CelebAMask-HQ dataset [41] for training, which consists of 30,000 high-resolution face images and segmentation masks for all parts of the face. For the above tasks, we each create a synthetic training dataset by stylizing images using a reference effect and adjusting its parameters for certain parts of the face (obtained from dataset annotations) according to the task, e.g., increasing the amount of contours in the chin area. In Fig. 5 trained PPN networks are evaluated by plotting the predicted local parameter masks together with the generated stylizations. It shows that the networks learn to predict parameter masks for the relevant regions accurately solely by observing pixels without additional supervision.

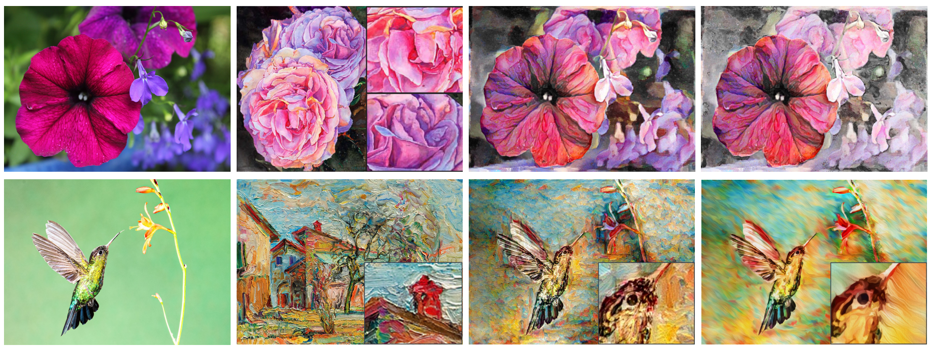


Fig. 6: **Parametric style transfer.** Effect parameters are optimized (watercolor effect in top row, oilpaint effect in bottom row) to match the stylistic reference image b. Users can then interactively edit the result c by adjusting resulting parameter masks. In the top row, the saturation parameter mask is adjusted to highlight foreground objects d. In the bottom row, the oilpaint-specific bump scale and flow-smoothing are adjusted in d to make the background appear to be painted wet-in-wet with long brushstrokes.

5 Applications

Using our framework for global and local parameter prediction for differentiable algorithmic pipelines, example-based stylization with closely related references is made possible. To adapt to real-world, more diverse example data, our framework can be integrated with existing stylization paradigms. In the following, we demonstrate our approach for the task of (statistics-based) style transfer reconstruction and GAN-based image-to-image translation based on the APDrawing dataset [68].

5.1 Style Transfer

We investigate the combination of iterative style transfer and algorithmic effects. We use Style Transfer by Relaxed Optimal Transport and Self-Similarity (STROTSS) by Kolkin *et al.* [33] to create stylized references for our effect. We subsequently try to recreate the style transfer result with our algorithmic effects by optimizing parameter masks (Fig. 6). For this, a ℓ_1 loss in image space is again used to match effect output and reference image.

Optimization. We run the style transfer algorithm [33] for 200 steps to create a stylized reference with a resolution of 1024×1024 pixels. The local parameter masks are then optimized using 1000 iterations of Adam [32] and a learning rate of 0.1. The learning rate is decreased by a factor of 0.98 every 5 iterations starting from iteration 50. To avoid the generation of artifacts in parameter masks, we smooth all masks at increasing iterations (10, 25, 50, 100, 250, 500) using a

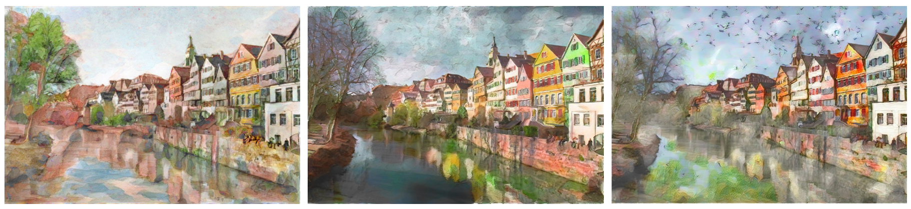

(a) Watercolor ℓ_1 optim
(b) Oilpaint $\mathcal{L}_C + \mathcal{L}_S$
(c) Watercolor $\mathcal{L}_C + \mathcal{L}_S$

Fig. 7: Optimization of parametric style transfer with different losses. In a), the effect is optimized by matching the result of STROTSS style transfer [33] using an ℓ_1 loss. In b) and c), the effect is directly optimized using neural style transfer [10] losses $\mathcal{L}_C + \mathcal{L}_S$. The content/style image are the same as in Fig. 1.

Table 3: Comparison of our method to STROTSS [33] for style loss $\mathcal{L}_S$, content loss $\mathcal{L}_C$, and the ℓ_1 difference between respective results. We test against in-domain style images and against a set of common (arbitrary domain) NST styles. In each case, results are averaged over 10 styles and NPRB [54] as content[4]

		XDoG	Cartoon	Watercolor		Oilpaint	
	style domain:	bw-drawing	cartoon	watercolor	common	oilpaint	common
$\mathcal{L}_S$	STROTSS	0.340	0.289	0.351	0.39	0.209	0.39
	Our results	0.246	0.406	0.359	0.42	0.384	0.52
$\mathcal{L}_C$	STROTSS	0.099	0.094	0.081	0.036	0.037	0.036
	Our results	0.172	0.148	0.092	0.034	0.023	0.033
	ℓ_1 Difference	0.188	0.136	0.007	0.021	0.036	0.039

Gaussian filter. Alternatively to image-space matching, parameters can also be directly optimized using NST [10] losses $\mathcal{L}_S + \mathcal{L}_C$, however this often fails to transfer more complex stylistic elements Fig. 7. As optimizing effect parameters to affect the output is harder than direct pixel optimization (as done in NST), stricter supervision (i.e., pixel-wise losses) is necessary to achieve similar outputs. Directly using $\mathcal{L}_S + \mathcal{L}_C$ could however be appropriate for photographic style transfer.

Results. As Tab. 3 shows, parametric style transfer works better for effects that have a high expressivity and are closer to hand-drawn styles, such as the watercolor or oilpaint, compared to more restricted effects such as XDoG or cartoon. After the generation of local parameter masks, the parameters can be refined by the user as shown in Fig. 6d. By optimizing parameters, we obtain an interpretable "whitebox" representation of a style that, in contrast to current pixel-optimizing NSTs [10,28,33], retains controllability according to artistic design aspects. Furthermore, our method is resolution independent, i.e., parameter masks can be optimized at lower resolutions and then scaled up to high resolutions for editing. In Fig. 6d (bottom row) the effect is applied at 4096 $\times$ 4096

[4] Exemplary style images and results in suppl. material.

pixels, while current style transfers are mostly memory-limited to much lower resolutions. We further compare matching performance of in-domain styles with a set of common NST styles and observe that the ℓ_1 difference (Tab. 3) is only marginally higher for the latter. Thus, highly parameterized effects such as watercolor or oilpaint can emulate any out-of-domain style by per-pixel optimization of parameter combinations reasonably well. However, as this creates highly fragmented parameter masks, a tradeoff between generalizability and interpretability of masks can be made using a weighted total variation-loss.

5.2 GAN-based image-to-image translation with PPNs

For learning a style distribution, i.e., the characteristics of an artistic style over a larger collection of artworks, GAN-based approaches have achieved impressive results [7,64,68]. We investigate training PPNs in a GAN-setting for image-to-image translation. While the global and local PPNs discussed in Sec. 4 can match in-domain styles very well (Sec. 4.2), they cannot produce local image structures that are not synthesizable by their constituent image filters. Artistic reference styles, however, often contain stylistic elements that have not been modeled in the heuristics-based filter - this holds especially true for more simple effects such as xDoG. For reference styles that are stylistically close to such effects (e.g., xDoG), such as line-drawings, we hypothesize that combining our filter pipeline with a lightweight CNN-based post-processing operation and learning them end-to-end can close the domain gap while retaining the positive properties of the filter parametrization and beeing computationally efficient.

Dataset. For our experiment, we select the APDrawing [68] dataset which consists of closely matching photos and their hand-drawn stylistic counterparts. Its hand-drawn images are reasonably similar to the XDoG results, while still containing many stylistic abstractions that cannot be emulated solely by the XDoG. We hypothesize, that re-creating such an effect entails both edge detection and content abstraction, which could be performed by our differential XDoG pipeline combined with a separate convolution network for content abstraction.

APDrawing contains a set of 140 portrait photographs along with paired drawings of these portraits. The GAN-based local PPN approach introduced in Sec. 4 is used to train on this paired dataset, where solely the generator is extended using a CNN $N(\cdot)$ to post-process the XDoG output, i.e., following Eq. (2) our generator now combines PPN, effect and CNN: $N(O(I, P_L(I)))$.

Architecture. We investigate the efficacy of each component in our proposed approach for APDrawing in Tab. 4. Following Yi *et al.* [69], we measure the Fréchet Inception Distance (FID) score [20] and Learned Perceptual Image Patch Similarity (LPIPS) [70] to the test set. We train for 200 epochs and otherwise use the same hyperparameters as Pix2Pix [24]. We observed that using the ResNet-based architecture for image translation introduced by Johnson *et al.* [28] works best for the post-processing CNN. Furthermore integrating the XDoG in the pipeline improves the results vs. a convolutional-only pipeline. Note that this

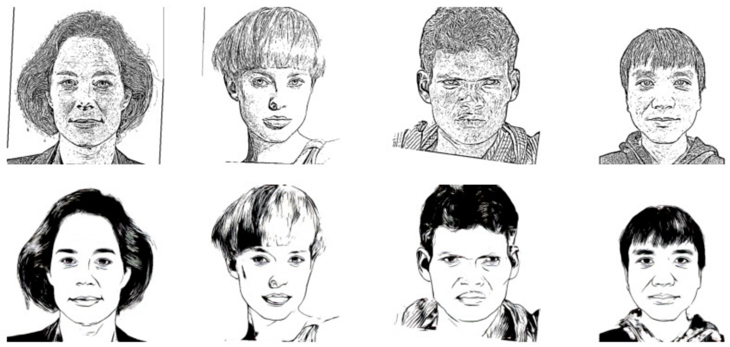


Fig. 8: **Learned task seperation**. Results on APDrawing [68] after XDoG (top) and then after convolution (bottom) are shown.

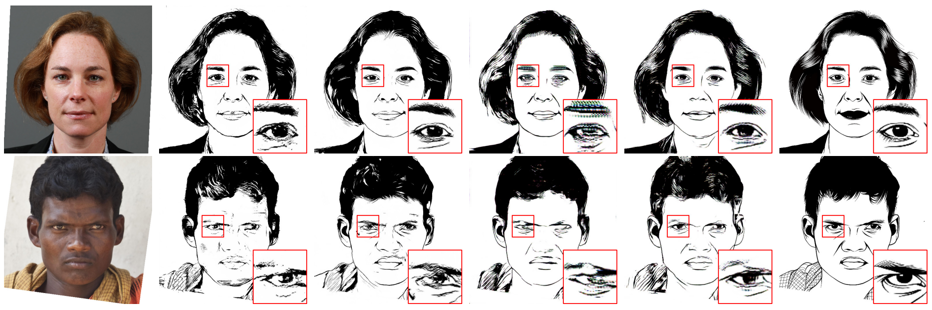

(a) Input (b) APDraw. -GAN [68] (c) APDrawing++ [69] (d) Ours - CNN-only (e) Ours - XDoG+CNN (f) Ground truth

Fig. 9: **Results on APDrawing.** While APDrawing GAN b and APDrawing++ GAN c can produce inconsistent lines, our proposed method e generally produces flow-consistent lines. The differentiable filter in our approach is important for consistent quality, as solely using an image translation CNN [28] often produces local dithering artefacts (upper row) and patchy features (lower row) d.

combination of algorithmic effects and CNNs in a training pipeline is only made possible by our introduced approach for end-to-end differentiable filter pipelines and PPNs. Omitting the PPN and using fixed parameters for XDoG significantly degrades the results, which validates the integrated training of filter and CNN. Fig. 8 visualizes the results after the XDoG stage and after post-processing using a CNN; the learned separation of edge detection and abstraction is apparent. Further, we observe that training with XDoG as a postprocessing instead of as a preprocessing step does not converge. All architecture choices are extensively evaluated in an ablation study, please refer to the supplemental material.

Results. While the CNN alone (without XDoG) already achieves good FID and LPIPS scores, we show in Fig. 9 that it creates major artifacts especially around eyebrows and eyes, which are not detected by those metrics.

Table 4: Our results on APDrawing [68]

PPN	XDoG	CNN	FID	LPIPS
✗	✗	U-Net	71.26	0.322
✗	✗	ResNet	62.44	**0.275**
✗[1]	✓	U-Net	75.40	0.329
✗[1]	✓	ResNet	71.56	0.305
✓	✓	U-Net	89.93	0.366
✓	✓	ResNet	**60.55**	0.285
APDrawing GAN			62.14[2]	0.291[2]
APDrawing++			**54.40**	**0.258**[2]
Train vs. Test			49.72	-

[1] a fixed parameter preset is used
[2] results obtained from [68][69]

Compared to the APDrawing GAN approach by Yi *et al.* [68], our model improves the FID score (Tab. 4). The state-of-the-art APDrawing++ [69] improves on these metrics and quantitatively performs better than our model, however qualitatively it can suffer from artifacts in small structures such as the eyes (Fig. 9c) whereas our approach leads to more consistent lines. We note that their approach consists of a sophisticated combination of several losses and task-specific discriminators that require facial landmarks to train multiple local generator networks for facial features such as eyes, nose, and mouth separately. This limits their generalizability to other datasets, while our approach, on the other hand, represents a general setup for image-to-image translation consisting of a globally trained CNN and a simple effect, making it applicable to any paired training data without further annotation requirements.

6 Discussion

Applicability. In the previous sections, we have demonstrated the applicability of differentiable filters to several example-based stylization tasks using four established heuristics-based filter pipelines. Their constituent image filters (Tab. 1) form a common basis of many image-based artistic rendering approaches [37]. We expect that other filtering-based effects, such as pencil-hatching [43], or stippling [60], can be transferred to our framework with relative ease due to their pipeline-based, GPU-optimized formulations. Stroke-based rendering approaches, on the other hand, are typically optimized globally [46] or locally [18], and are thus challenging to transform into differentiable formulations. However, a recent approach by Liu *et al.* [42] has shown that strokes can be predicted in a single feedforward pass of a CNN, which could be regarded as a complementary approach.

Limitations. Our PPN-based approaches make use of a paired data training regime. While paired data can be synthetically generated for content-adaptive effects aiming at solving filter-specific problems, datasets with paired real-world paintings are subject to limited availability. As our training approach follows Pix2Pix GAN [24], future work extending the method to train with unpaired training losses, such as cycle-consistency losses [71], could alleviate this limitation. An inherent limitation of predicting parameters in comparison to directly predicting pixels (as with convolutional GANs), remains the constraint of only being able to produce styles that lie in the manifold of achievable effects of the underlying image filters. While this can be mitigated using a post-processing CNN, this represents a trade-off with respect to interpretability and range of low-level control (we examine this aspect in the supplemental material). On the other hand, our parametric style transfer is able to match arbitrary styles when optimizing highly parameterized effects such as watercolor. Training a PPN with such an effect on a large dataset, e.g., using unpaired training, could similarly already have sufficient representation capability without postprocessing CNNs.

7 Conclusions

In this work, we propose the combination of algorithmic stylization effects and example-based learning by implementing heuristics-based stylization effects as differentiable operations and learning their parametrizations. The results show that both optimization of parameters, e.g., to achieve style transfers, and their global and local prediction, e.g., for content-adaptive effects, are viable approaches for example-based algorithmic stylizations. Our experiments demonstrate that our approach is especially suitable for applications that require fast adaptation to new styles while retaining full artistic control and low computation times for high image resolutions. Furthermore, stylizations beyond the filters' abstraction capabilities are achieved by adding convolutional post-processing. This approach can generate results on-par with state-of-the-art CNN-based methods. For future research, learning the composition of filters as building blocks of a generic algorithmic effect pipeline would allow for seamless integration of user control and example-based stylization without the limitation to a specific stylization technique.

Acknowledgments. We thank Sumit Shekhar and Sebastian Pasewaldt for comments and discussions. This work was partially funded by the BMBF through grants 01IS18092 ("mdViPro") and 01IS19006 ("KI-LAB-ITSE").

References

1. Adobe. Photoshop neural filters overview. `https://helpx.adobe.com/photoshop/using/neural-filters.html`, 2021. 2
2. Yoshua Bengio, Nicholas Leonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 23
3. Adrien Bousseau, Matt Kaplan, Joëlle Thollot, and François X Sillion. Interactive watercolor rendering with temporal coherence and abstraction. In *Proc. International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 141–149, 2006. 2, 3, 4, 7
4. Thomas Brox, Rein Van Den Boomgaard, François Lauze, Joost Van De Weijer, Joachim Weickert, Pavel Mrázek, and Pierre Kornprobst. Adaptive structure tensors and their applications. In *Visualization and Processing of Tensor Fields*, pages 17–47. 2006. 22

5. Benoit Brummer and Christophe De Vleeschouwer. Natural image noise dataset. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1777–1784, 2019. 25
6. Xinyuan Chen, Chang Xu, Xiaokang Yang, Li Song, and Dacheng Tao. Gated-GAN: Adversarial gated networks for multi-collection style transfer. *IEEE Transactions on Image Processing*, 28(2):546–560, 2018. 5
7. Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. CartoonGAN: Generative Adversarial Networks for Photo Cartoonization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9465–9474, 2018. 5, 12
8. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009. 26
9. Ignacio Garcia-Dorado, Pascal Getreuer, Bartlomiej Wronski, and Peyman Milanfar. Image stylisation: from predefined to personalised. *IET Computer Vision*, 14(6):291–303, 2020. 5
10. Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image Style Transfer Using Convolutional Neural Networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 2, 4, 8, 11
11. Leon A. Gatys, Alexander S. Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Controlling Perceptual Factors in Neural Style Transfer. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3730–3738, 2017. 4
12. Pascal Getreuer, Ignacio Garcia-Dorado, John Isidoro, Sungjoon Choi, Frank Ong, and Peyman Milanfar. BLADE: Filter learning for general purpose computational photography. In *Proc. IEEE International Conference on Computational Photography (ICCP)*, pages 1–11, 2018. 5
13. Michaël Gharbi, Jiawen Chen, Jonathan T Barron, Samuel W Hasinoff, and Frédo Durand. Deep Bilateral Learning for Real-Time Image Enhancement. *ACM Transactions on Graphics*, 36(4):1–12, 2017. 2, 5
14. Amy A. Gooch, Jeremy Long, Li Ji, Anthony Estey, and Bruce S. Gooch. Viewing Progress in Non-photorealistic Rendering through Heinlein's Lens. In *Proc. International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 165–171, 2010. 2
15. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems (NIPS)*, 2014. 2, 5
16. Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. Arbitrary Style Transfer with Deep Feature Reshuffle. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8222–8231, 2018. 4
17. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 26
18. Aaron Hertzmann. Painterly Rendering with Curved Brush Strokes of Multiple Sizes. In *Proc. of the 25th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 453–460, 1998. 14
19. Aaron Hertzmann. Computers do not make art, people do. *Communications of the ACM*, 63(5):45–48, 2020. 2
20. Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems (NIPS)*, 2017. 12

21. Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A White-Box Photo Post-Processing Framework. *ACM Transactions on Graphics*, 37(2):1–17, 2018. 5
22. Xun Huang and Serge Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 1501–1510, 2017. 4, 26
23. Tobias Isenberg. Interactive NPAR: What Type of Tools Should We Create? In *Proc. Non-Photorealistic Animation and Rendering (NPAR)*, 2016. 2
24. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image Translation with Conditional Adversarial Networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1125–1134, 2017. 2, 5, 8, 12, 14, 33
25. Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2017–2025, 2015. 22
26. Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, Dacheng Tao, and Mingli Song. Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields. In *Proc. European Conference on Computer Vision (ECCV)*, 2018. 4
27. Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural Style Transfer: A Review. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3365–3385, 2019. 1
28. Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In *Proc. European Conference on Computer Vision (ECCV)*, pages 694–711, 2016. 1, 4, 8, 11, 12, 13, 33
29. Henry Kang, Seungyong Lee, and Charles K. Chui. Flow-Based Image Abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):62–76, 2009. 6, 22
30. Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *Proc. International Conference on Learning Representations (ICLR)*, 2018. 2
31. Tero Karras, Samuli Laine, and Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4401–4410, 2019. 2, 28, 29
32. Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2015. 10, 28
33. Nicholas Kolkin, Jason Salavon, and Gregory Shakhnarovich. Style Transfer by Relaxed Optimal Transport and Self-Similarity. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10051–10060, 2019. 4, 10, 11, 29, 30, 31, 32
34. Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint Bilateral Upsampling. *ACM Transactions on Graphics (ToG)*, 26(3):96–es, 2007. 7
35. Dmytro Kotovenko, Matthias Wright, Arthur Heimbrecht, and Björn Ommer. Rethinking Style Transfer: From Pixels to Parameterized Brushstrokes. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12196–12205, 2021. 4
36. Michiyoshi Kuwahara, Kozaburo Hachimura, S Eiho, and Masato Kinoshita. Processing of RI-Angiocardiographic Images. In *Digital Processing of Biomedical Images*, pages 187–202. 1976. 4
37. Jan Eric Kyprianidis, John Collomosse, Tinghuai Wang, and Tobias Isenberg. State of the "Art": A Taxonomy of Artistic Stylization Techniques for Images and Video.

IEEE Transactions on Visualization and Computer Graphics, 19(5):866–885, 2012. 1, 4, 14
38. Jan Eric Kyprianidis and Jürgen Döllner. Image Abstraction by Structure Adaptive Filtering. In *Proc. EG UK Theory and Practice of Computer Graphics (TPCG)*, pages 51–58, 2008. 4, 6, 7, 22
39. Jan Eric Kyprianidis and Henry Kang. Image and Video Abstraction by Coherence-Enhancing Filtering. In *Computer Graphics Forum*, volume 30, pages 593–602. Wiley Online Library, 2011. 7
40. Jan Eric Kyprianidis, Henry Kang, and Jürgen Döllner. Image and Video Abstraction by Anisotropic Kuwahara Filtering. *Computer Graphics Forum*, 28(7):1955–1963, 2009. 4, 7
41. Cheng-Han Lee, Ziwei Liu, Lingyun Wu, and Ping Luo. MaskGAN: Towards Diverse and Interactive Facial Image Manipulation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5549–5558, 2020. 9
42. Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Ruifeng Deng, Xin Li, Errui Ding, and Hao Wang. Paint Transformer: Feed Forward Neural Painting with Stroke Prediction. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 6598–6607, 2021. 14
43. Cewu Lu, Li Xu, and Jiaya Jia. Combining Sketch and Tone for Pencil Drawing Production. In *Proc. International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, 2012. 14
44. Santiago E Montesdeoca, Hock Soon Seah, Pierre Bénard, Romain Vergne, Joëlle Thollot, Hans-Martin Rall, and Davide Benvenuti. Edge- and substrate-based effects for watercolor stylization. In *Proc. International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 1–10, 2017. 7
45. Sean Moran, Pierre Marza, Steven McDonagh, Sarah Parisot, and Gregory Slabaugh. DeepLPF: Deep Local Parametric Filters for Image Enhancement. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12826–12835, 2020. 5
46. Reiichiro Nakano. Neural Painters: A learned differentiable constraint for generating brushstroke paintings. *arXiv preprint arXiv:1904.08410*, 2019. 14
47. Dae Young Park and Kwang Hee Lee. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5880–5888, 2019. 4
48. Bui Tuong Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, 1975. 7
49. Thomas Porter and Tom Duff. Compositing Digital Images. *SIGGRAPH Comput. Graph.*, 18(3):253–259, 1984. 7
50. Max Reimann, Benito Buchheim, Amir Semmo, Jürgen Döllner, and Matthias Trapp. Controlling strokes in fast neural style transfer using content transforms. *The Visual Computer*, pages 1–15, 2022. 4
51. Max Reimann, Mandy Klingbeil, Sebastian Pasewaldt, Amir Semmo, Matthias Trapp, and Jürgen Döllner. Locally controllable neural style transfer on mobile devices. *The Visual Computer*, 35(11):1531–1547, 2019. 4
52. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proc. International Conference on Medical Image Computing and Computer-assisted Intervention*, pages 234–241, 2015. 8, 33
53. Paul Rosin and John Collomosse. *Image and video-based artistic stylisation*, volume 42. 2012. 1, 4
54. Paul L. Rosin, Yu-Kun Lai, David Mould, Ran Yi, Itamar Berger, Lars Doyle, Seungyong Lee, Chuan Li, Yong-Jin Liu, Amir Semmo, Ariel Shamir, Minjung

Son, and Holger Winnemöller. NPRportrait 1.0: A three-level benchmark for non-photorealistic rendering of portraits . *Computational Visual Media*, 8(3):445–465, 2022. 8, 11, 24, 28, 29, 38, 40
55. Artsiom Sanakoyeu, Dmytro Kotovenko, Sabine Lang, and Björn Ommer. A Style-Aware Content Loss for Real-time HD Style Transfer. In *Proc. European Conference on Computer Vision (ECCV)*, 2018. 5
56. Amir Semmo, Tobias Dürschmid, Matthias Trapp, Mandy Klingbeil, Jürgen Döllner, and Sebastian Pasewaldt. Interactive Image Filtering with Multiple Levels-of-Control on Mobile Devices. Proc. SIGGRAPH ASIA Mobile Graphics and Interactive Applications (MGIA), pages 2:1–2:8, 2016. 3, 4
57. Amir Semmo, Tobias Isenberg, and Jürgen Döllner. Neural Style Transfer: A Paradigm Shift for Image-based Artistic Rendering? In *Proc. International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 1–13, 2017. 1, 4
58. Amir Semmo, Daniel Limberger, Jan Eric Kyprianidis, and Jürgen Döllner. Image Stylization by Interactive Oil Paint Filtering. *Computers & Graphics*, 55:157–171, 2016. 4, 24
59. Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proc. International Conference on Learning Representations (ICLR)*, 2015. 7
60. Minjung Son, Yunjin Lee, Henry Kang, and Seungyong Lee. Structure grid for directional stippling. *Graphical Models*, 73(3):74–87, 2011. 14
61. Hao Su, Jianwei Niu, Xuefeng Liu, Qingfeng Li, Jiahe Cui, and Ji Wan. Manga-GAN: Unpaired Photo-to-Manga Translation Based on The Methodology of Manga Drawing. In *Proc. AAAI Conference on Artificial Intelligence*, pages 2611–2619, 2021. 5
62. Carlo Tomasi and Roberto Manduchi. Bilateral Filtering for Gray and Color Images. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, pages 839–846, 1998. 4, 7, 21
63. Miaoyi Wang, Bin Wang, Yun Fei, Kanglai Qian, Wenping Wang, Jiating Chen, and Jun-Hai Yong. Towards Photo Watercolorization with Artistic Verisimilitude. *IEEE Transactions on Visualization and Computer Graphics*, 20(10):1451–1460, 2014. 2, 3, 4, 7
64. Xinrui Wang and Jinze Yu. Learning to Cartoonize Using White-box Cartoon Representations. In *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8090–8099, 2020. 12
65. Holger Winnemöller. XDoG: Advanced Image Stylization with eXtended Difference-of-Gaussians. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, pages 147–156, 2011. 2, 3, 4, 6, 7, 28
66. Holger Winnemöller, Sven C Olsen, and Bruce Gooch. Real-Time Video Abstraction. *ACM Transactions On Graphics (TOG)*, 25(3):1221–1226, 2006. 2, 3, 4, 6, 7, 23
67. Zhicheng Yan, Hao Zhang, Baoyuan Wang, Sylvain Paris, and Yizhou Yu. Automatic Photo Adjustment Using Deep Neural Networks. *ACM Transactions on Graphics (TOG)*, 35(2):1–15, 2016. 5
68. Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. APDrawingGAN: Generating Artistic Portrait Drawings from Face Photos with Hierarchical GANs. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10743–10752, 2019. 5, 9, 10, 12, 13, 14, 21, 33, 34, 35, 37
69. Ran Yi, Mengfei Xia, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Line Drawings for Face Portraits from Photos using Global and Local Structure based GANs. In

Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2020. 5, 12, 13, 14, 34, 35

70. Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 586–595, 2018. 12, 28
71. Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Proc. IEEE international Conference on Computer Vision (ICCV)*, pages 2223–2232, 2017. 5, 14

Supplemental Material

Due to space restrictions, some details had to be omitted from the main paper; we present those details here. In Sec. 1 we describe filter modifications to obtain gradients using auto-grad enabled frameworks on the example of the seperated orientation-aligned bilateral filter and color quantization, and elaborate on filter pipelines concerning their structure, learnability and memory consumption. In Sec. 2, we elaborate on training global PPNs by presenting three PPN architectures and evaluating their performance in an ablation study. In Sec. 3 we give visual examples for the style transfer capability of different effects. In Sec. 4, we provide additional details and results for the architecture ablation study on image-to-image translation tasks using the APDrawing dataset[68] as well as editability of such effects. Finally, in Sec. 5, additional results for parametric style transfer, image-to-image translation, and effect variants are shown. Further, our supplemental video[1] demonstrates parametric style transfer optimization and interactive editing for the images shown in the main paper.

1 Differentiable Filters

In the main paper we note that while most filters are straight-forward to implement in auto-grad enabled frameworks, for some several filters re-formulations or approximations are needed. Specifically, for structure-adaptive neighborhood operations, a re-formulation to retain sub-gradients in adaptive kernel neighbourhoods must be employed, and for non-differentiable operations a numeric gradient approximation is needed. In the following, we show an example for each.

1.1 Bilateral Filter

Commonly used techniques in image filtering pipelines are bilateral filters. The bilateral filter [62] using Gaussians G of size σ_d (distance kernel) and σ_r (range kernel) for an image $I \in \mathbb{R}^{C\times W\times H}$ is defined for a pixel coordinate x by

$$\hat{I}(x) = \frac{1}{W} \sum_{y\in\Omega(x)} I(y) G_{\sigma_d}(\|y-x\|)\, G_{\sigma_r}(\|I(y)-I(x)\|) \qquad (1)$$

where $\Omega(x)$ denotes the window centered in x and $y \in \Omega$ represent pixel coordinates of the kernel neighbourhood. W represents the weight normalization term, which we omit in the following for the sake of brevity. Computing the bilateral filter for large images and at large kernel neighbourhoods is, however, computationally expensive. Several approaches have been proposed to approximate the filter by separation into multiple passes for improved efficiency.

[1] https://youtu.be/wIndN7cr0PE

For image abstraction and stylization, the *orientation-aligned separated bilateral filter*[38,29] has shown a good trade-off between quality and performance. The filter is guided in two passes along the gradient and tangent direction of an edge tangent field. The tangent field of the input image is obtained by an eigenanalysis of the smoothed structure tensor [4]; for more details the reader is kindly referred to the work of Kyprianidis and Döllner [38]. The tangent field computation consists of point-based and fixed-neighbourhood kernels only, and thus is straightforward to implement using common auto-differentiable functions. However, the following separated bilateral filter is iterative and structure adaptive (i.e., the size of the kernel neighbourhood depends on the content), and thus cannot be ported to the fixed-neighbourhood functions (e.g., convolutions) of auto-grad enabled frameworks in a straightforward way.

Specifically, in the work of Kyprianidis and Döllner [38], the filter response at a sampling point x of a pass of the separated orientation-aligned bilateral filter in direction t (gradient or tangent direction) is defined as:

$$\hat{I}(x) = I(x) + \sum_{y \in \Omega_t(x)} I(y) G_{\sigma_d}(\|y - x\|)\, G_{\sigma_r}(\|I(y) - I(x)\|) \tag{2}$$

where $\Omega_t(x) = \{x + kt, \mid k \in [-N, N] \wedge k \in \mathbb{Z}\}$ represents sampling positions along the direction t defined in x, and N denotes the cut-off kernel radius. N is typically based on σ_d, e.g., set to $\lfloor 2\sigma_d / \|t\| \rfloor$, and the kernel size locally depends on the magnitude of the direction vector t.

It would be possible to implement a custom kernel with associated backward pass for the above to manually compute (sub-) gradients for both input and parameters, and similarly repeat the procedure for every other structure-adaptive filter in Tab. 1 of the main paper. However, as our goal is to minimize the effort of porting existing (shader-based) filters to our framework while maximizing reusability and portability, we propose to implement the filters using common auto-differentiable components. To this end, we reformulate Eq. (2) into a grid sampling-based operation. Specifically, we use spatial coordinate grids $C \in \mathbb{R}^{2 \times W \times H}$ and $T \in \mathbb{R}^{W \times H \times 2D}$ that represent the mapping of output pixel locations to input pixels for grid sampling. Grid C is the identity mapping, i.e., contains coordinates $C_{wh} = (w, h)$. Grid T contains the sampling offsets in dimension D: $T_{wh} = \{kt_{wh}, \mid k \in [-D, D] \wedge k \in \mathbb{Z}\}$. Then the neighbourhood sampling values $V \in \mathbb{R}^{C \times W \times H \times 2D}$ for the entire image I are computed as:

$$V = \mathcal{I}(C + T)\delta\left[G_{\sigma_d}(\|T\|)\right] G_{\sigma_r}(\|\mathcal{I}(C + T) - I\|) \tag{3}$$

where in slight abuse of notation, $\mathcal{I}(C+T)$ denotes bilinear grid sampling [25] of I over the coordinate grid of unfolded kernels. It is thus equivalent to retrieving $I(\Omega_t(x))$ for every pixel x with fixed number of samples. To clip values outside of the adaptive kernel neighborhood, δ is computed as:

$$\delta_{whd}(v) = \begin{cases} v & \text{if } \|T_{whd}\| \le N_{wh} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Note that multiplications between tensors in Eq. (3) are performed elementwise. Filter responses in dimension D are then folded (summed) to obtain the response over the full kernel neighborhood (i.e., the filtered image $\hat{I}$):

$$\hat{I} = I + \sum_{d=1}^{D} V_d \tag{5}$$

The size of dimension D can be varied for performance-quality trade-offs (as shown in Fig.4 in the main paper).

1.2 Color Quantization

Color quantization is often applied to achieve a flat, cartoon-like impression (Fig. 3, main paper). It is usually defined using the floor function: $y = {}^{\lfloor xb \rfloor + 0.5}/_{b}$ [66], where x denotes the color value before and y the color value after quantization, and b (number of quantization bins) is the parameter that should receive gradients. Due to the non-differentiable nature of the floor function, we introduce a differentiable approximation. Common differentiable proxies like the straight-through estimator[2] work well for approximating the gradient of quantization functions with fixed numbers of quantization bins. However, for color quantization we want to differentiate with respect to the strength of the quantization or the number of bins, which is not possible using common differentiable proxies. As reasoning about the number of quantization bins requires information about the complete image, a global optimization is formulated instead:

$$f(r) = \sum_i \operatorname{sign}\left(\boldsymbol{y}_i - \frac{\lfloor r\boldsymbol{x}_i \rfloor + 0.5}{r}\right) \operatorname{sign}(\boldsymbol{g}_i) \tag{6}$$

where $\boldsymbol{x}$ denotes a vector containing the input pixel values, $\boldsymbol{y}$ the outputs of $y(\boldsymbol{x})$, and $\boldsymbol{g}$ the gradient vector for $\boldsymbol{y}$.

Intuitively, f sums values for each pixel, which are positive if the pixel's gradient points in the direction of r and negative otherwise. The maximum number of per-pixel gradients should point in the direction of r.

We choose $\hat{b} = \arg\max_r f(r)$ to derive the optimal value for the parameter b, which minimizes the learning target. This value is obtained in a single optimization step. To obtain a continuous gradient, the difference $b - \hat{b}$ is scaled by the gradient magnitude $|g_i|$ of the loss function computed with respect to all pixels $y_i \in \boldsymbol{y}$, such that gradients for b decrease, once the optimal value is approached:

$$\frac{\partial y}{\partial b} := \sum_i |g_i|(b - \hat{b}) \tag{7}$$

1.3 Differentiable filter pipeline - Oilpaint

As noted in the main paper, we implement differentiable filter pipelines analogous to their originally published versions and show an example for the toon

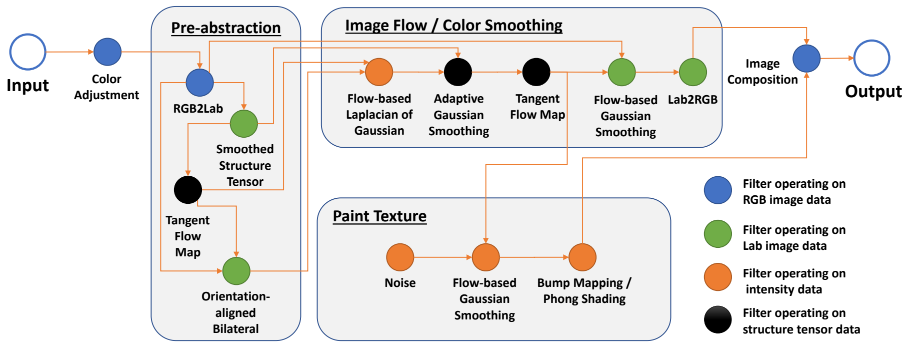

Input
Color Adjustment
Pre-abstraction
RGB2Lab
Smoothed Structure Tensor
Tangent Flow Map
Orientation-aligned Bilateral
Image Flow / Color Smoothing
Flow-based Laplacian of Gaussian
Adaptive Gaussian Smoothing
Tangent Flow Map
Flow-based Gaussian Smoothing
Lab2RGB
Image Composition
Output
Paint Texture
Noise
Flow-based Gaussian Smoothing
Bump Mapping / Phong Shading
Filter operating on RGB image data
Filter operating on Lab image data
Filter operating on intensity data
Filter operating on structure tensor data

Fig. 1: Oilpaint filter pipeline adapted from Semmo *et al.* [58].

pipeline in Fig. 3 of the main paper. In Fig. 1 we further show the filter pipeline of the oilpaint effect [58]. In contrast to Semmo *et al.* we do not use color palette extraction and color quantization and instead use a color adjustment layer.

Table 1: Parameter optimization to compare learned and target stylizations. Scores are computed on NPR benchmark [54] for both high-quality (level I) and low-quality (level III) portraits.

	NPR level I		NPR level III	
Loss	SSIM	PSNR	SSIM	PSNR
Cartoon ℓ_2	0.937	28.144	0.958	34.124
Cartoon ℓ_1	0.931	26.939	0.947	30.429
Watercolor ℓ_2	0.922	30.634	0.897	32.342
Watercolor ℓ_1	0.883	31.032	0.895	29.048

1.4 Functional Benchmark

To verify that all parameters of an effect pipeline can be learned, we set up a functional benchmark to evaluate how well effects can be adapted to an example stylization in the same domain. We developed an OpenGL-based "ground truth" implementation to generate reference stylizations, where parameter values are randomized during the benchmark. Using an image-based loss, gradients are then backpropagated to optimize the parameters of the differentiable effect using gradient descent. We measure the SSIM and PSNR. Tab. 1 shows that both the cartoon and watercolor effect achieve very high similarity to their reference using both ℓ_1 and ℓ_2 losses. The photographic quality level of the input images does

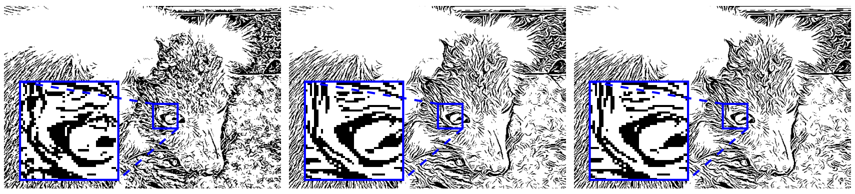

(a) 0.98 GiB, 0.12 s (b) 1.51 GiB, 0.17 s (c) OpenGL reference

Fig. 2: GPU VRAM and run-time in seconds for varying kernel sizes $D = 1$ a and $D = 5$ b for XDoG. (Image resized to 1MP from [5]).

not seem to play a role in the style-matching capability. As the XDoG filter is contained in both cartoon and watercolor pipelines, it is not evaluated separately.

1.5 Implementation Aspects - Memory

During training, limiting the memory consumption of individual filters is important, as they operate in the full input resolution. Generally, filters that accumulate sampling of multiple different locations in a single tensor have the highest memory usage. For our filters, wet-in-wet stylization, Kuwahara, and bilateral filtering are the most expensive with 3 GB to 5 GB peak memory usage for 1 MP input images. The trade-off between the quality of the generated results and computing resources can be controlled by the kernel size parameter D as shown in Fig. 2. To further reduce memory consumption during training, we use gradient checkpointing to recompute intermediate gradients on the fly. Thereby, activations are stored only at the end of each filter, which is usually a single RGB image. Intermediate activations are re-computed on the fly during back-propagation. The peak memory usage thus then depends on the single most memory-intensive filter. For example, the non-checkpointed XDoG uses 4GB of GPU memory at peak for 1MP resolution input, while the checkpointed version uses 2GB. The loss in speed is small and can be mitigated through the larger batch size that can be processed with the available memory.

2 Global PPN

This section provides details on the global PPN, as well as an ablation study for architecture variants and losses.

2.1 Architecture

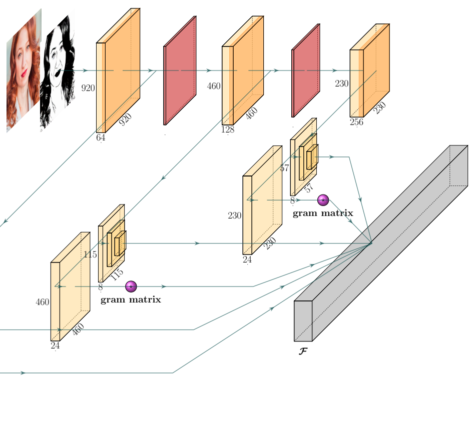
The network architecture receives two images: an input image and a stylized image, which has been generated with unknown parameters. We develop and benchmark three network architectures for parameter prediction, as follows:

1. **SimpleNet**: A simple convolution network with three Conv-ReLU layers (channel count: 16,32,64) followed by MaxPooling layers. All features are

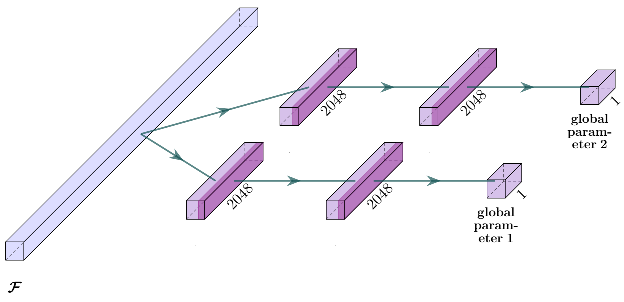
concatenated at the end and transformed with an additional linear layer to yield the global parameters. Input and stylized image are concatenated along the channel dimension and subsequently passed through the network.

2. **ResNet + Multi-head**: A ResNet50 architecture [17] without BatchNormalization layers is used to extract features. We initialize the ResNet with pre-trained weights on the ImageNet dataset [8]. Again, input and stylized image are concatenated along the channel dimension before passing them through the network. Features $\mathcal{F}$ are extracted after the last convolution layer and passed to a custom multi-head module depicted in Fig. 3b. The streams of linear layers process the features for each global parameter independently.
3. **Multi-feature + Multi-head**: A VGG backbone without BatchNormalization and pre-trained weights on ImageNet is used. Input and stylized image are passed separately through the feature extractor in two passes and all compact features are accumulated afterward. We extract features and compute Gram matrices for every convolution step as depicted in Fig. 3a. As the Gram matrices contain essential information about style, we hypothesize that these features help understand the input stylizations. The accumulated features $\mathcal{F}$ from both images are passed to the multi-head module of Fig. 3b to predict the final global parameters.

The design of the PPN architectures is motivated by two assumptions: (1) using a multi-head module for independent processing of each parameter should improve the accuracy as parameters for algorithmic effects model independent aspects of the stylization process and thus benefit from learning parameter-specific representations; (2) extraction of features at multiple scales (multi-feature) should be beneficial for the PPN's performance, as parameter changes can affect larger parts of the image as well as small details. As Huang *et al.* [22] find that normalization layers also perform normalization of style, we suspect that leaving out BatchNormalization in all architectures would yield better results



(a) Multi-feature extractor



(b) Multi-head module

Fig. 3: **Multi-feature extractor a:** Features are extracted from all convolution stages of a pre-trained VGG-11 network. We only show the first 3 convolution stages of the VGG-11 here for brevity. The features are passed through small stacks of strided convolutions with stride 4 and kernel size 4×4 pixels to generate compact representations. In parallel, the same features are passed through a single 1×1 pixel convolution and a subsequent operation to calculate the Gram matrices. All compact features $\mathcal{F}$ are concatenated. Convolution operations are visualized in yellow, ReLU layers in orange, and pooling layers in red. **Multi-head module b:** After the extraction of all features, we continue with multiple streams of linear layers. A separate stream of 3 linear layers is created for each global parameter: The first two layers process the features and have ReLU activations. The last layer outputs the final prediction as a single number. Fully connected layers are visualized in light violet and ReLU layers in dark violet.

2.2 Ablation Study

We conduct an ablation study for the previously discussed PPN architectures and loss functions. We use the XDoG filter effect [65] and compute the SSIM and PSNR metric in image space between the predicted and reference stylizations. We also indicate the mean absolute difference (ℓ_1) between the predicted and ground truth global parameters.

Loss Functions. For our ablation study, we compare parameter space- and image space losses. Parameter space losses are computed from the output of the PPN directly (i.e., in parameter space), thus the differentiable effect is not used during training. For image space losses, the predicted parameters are used to parameterize the differentiable effect, its output is then compared to the target output image using a pixel-wise similarity metric and gradients are back-propagated through the effect back to the PPN.

Network Training. For training, we use a random extract of 7000 images from the FFHQ-dataset of portrait images [31] at a resolution of 1024×1024 pixels. For data augmentation, images are randomly cropped to resolution 920×920 pixels. We use Adam [32] with a learning rate of 10^{-5} and train for 25 epochs, using a virtual batch size of 64. For testing, we use all 60 portrait images of the NPR benchmark portrait [54] and randomly generated global parameters. The final parameter predictions are passed through a tanh activation function for all networks and multiplied by 0.5 to yield parameters in the range $[-0.5, 0.5]$.

Results. Results are summarized in Tab. 2. Interestingly, the simple network architecture performs better than or similar to the ResNet in most experiments. We argue that for the ResNet, too much spatial information gets lost, as features are derived only after all downsampling and convolution operations. The multi-feature architecture effectively recovers the ability to derive representative features at all scales, as shown by Zhang *et al.* [70], and outperforms the other two options. We observe that computing the loss in image space instead of parameter space generally yields better results in terms of SSIM and PSNR, even though the ground-truth parameters are not directly available to the network in this case. As expected, computing the loss in parameter space still leads to a closer approximation of the ground truth parameters as measured by the ℓ_1 distance between ground truth and predicted parameters. If gradients are used as a learning signal, the model learns to use parameters more effectively to achieve better results, sometimes deviating more from the ground truth parameters. Some parameter changes within the XDoG effect (e.g., changing either contour or blackness) can similarly impact the final outcome. We reason that the gradients, which are derived from our differentiable effects, enable the model to build a better understanding of how the various parameters influence the final outcome.

Table 2: Global parameter prediction: ablation study for XDoG on FFHQ[31]. For network variants SimpleNet (SN), ResNet + Multi-head (R+M), Multi-feature + Multi-head (M+M), using image space-based losses $I[\ell_{1/2}]$ and parameter space-based losses $P[\ell_{1/2}]$.

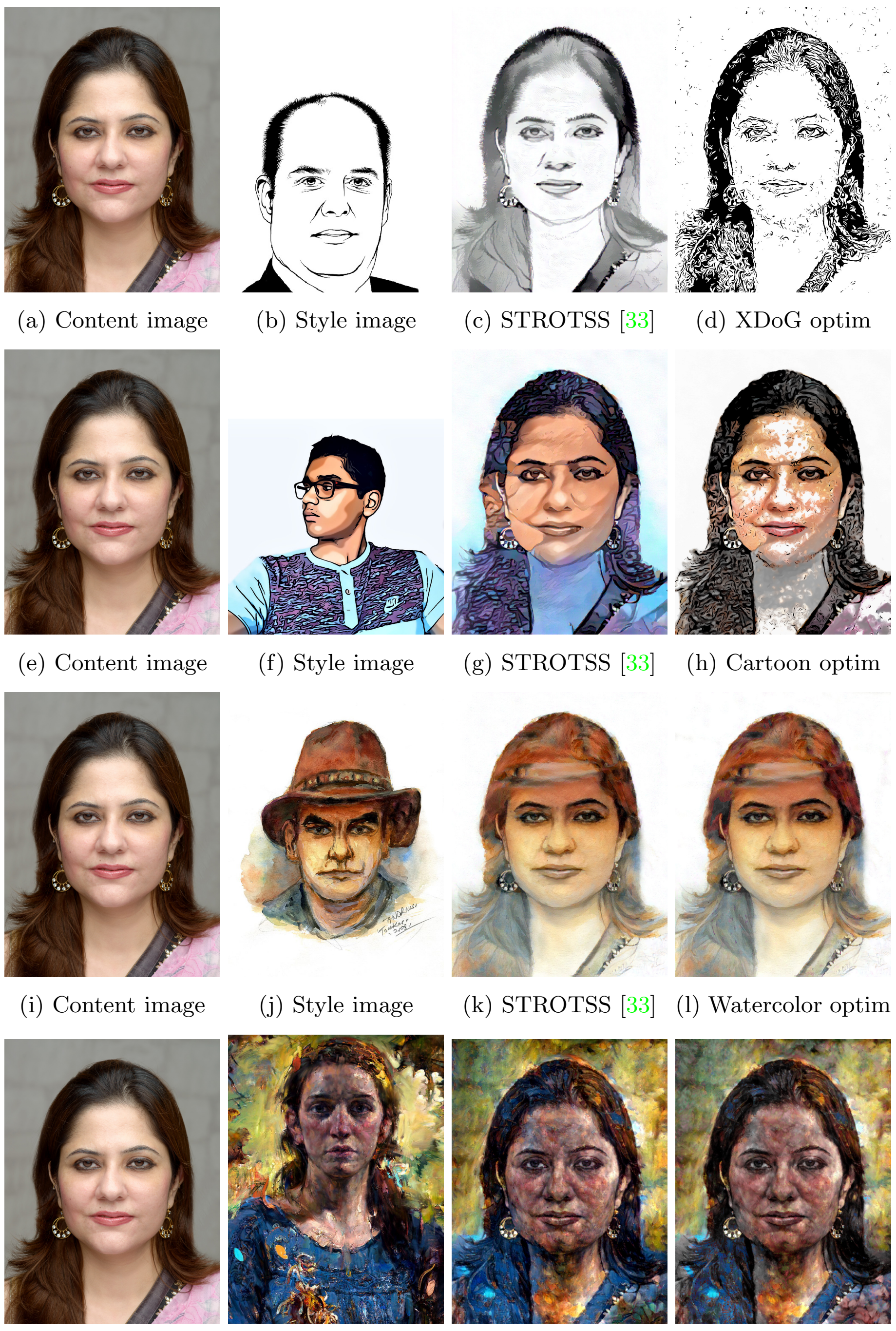
Network	Loss	SSIM	PSNR	$P[\ell_1]$
		– across all NPR levels –		
M+M	$I[\ell_2]$	0.764	13.286	0.190
SN	$I[\ell_2]$	0.733	12.523	0.218
R+M	$I[\ell_2]$	0.726	12.234	0.235
M+M	$P[\ell_2]$	0.738	12.530	**0.158**
SN	$P[\ell_2]$	0.686	11.277	0.197
R+M	$P[\ell_2]$	0.677	10.874	0.205
M+M	$I[\ell_1]$	**0.780**	**13.875**	0.183
SN	$I[\ell_1]$	0.728	12.407	0.227
R+M	$I[\ell_1]$	0.721	12.038	0.236
M+M	$P[\ell_1]$	0.737	12.927	0.162
SN	$P[\ell_1]$	0.697	11.805	0.200
R+M	$P[\ell_1]$	0.692	11.408	0.200

3 Style Transfer Capability

In Tab. 3 of the main paper we compared the potential of our differentiable effects to be optimized in a (general) style transfer framework. In Fig. 4 we show example images from the in-domain optimization for the implemented effects. For this, we used content images from the NPR benchmark dataset[54] and several style images from the same artistic domain for each effect (one example per effect is shown Fig. 4 b,f,j,n).

The reference for optimization is created by STROTSS [33] style transfer. Note that it does not necessarily produce references that would be considered strictly in-domain with the style image (e.g., cartoon or line drawing domains) as the style transfer method has no concept of the actual drawing techniques used. The created reference (e.g., Fig. 4k) is then matched as closely as possible during local parameter optimization. As is visible, only the Watercolor and Oilpaint effects are able to produce closely matching results Fig. 4l. The Cartoon and XDoG effects can, on the other hand, only be optimized with references that are closer to their range of achievable effects and are not suited for the general style transfer case.

In Fig. 5 and Fig. 6 we further show the general style transfer capability of oilpaint and watercolor on the set of common NST styles measured in Tab. 2. It is visible that oilpaint achieves a decent matching on most styles, but fails to create structure in some regions, while watercolor achieves an almost perfect matching of all references.



(a) Content image (b) Style image (c) STROTSS [33] (d) XDoG optim

(e) Content image (f) Style image (g) STROTSS [33] (h) Cartoon optim

(i) Content image (j) Style image (k) STROTSS [33] (l) Watercolor optim

(m) Content image (n) Style image (o) STROTSS [33] (p) Oilpaint optim

Fig. 4: Local parameter optimization using different algorithmic effects

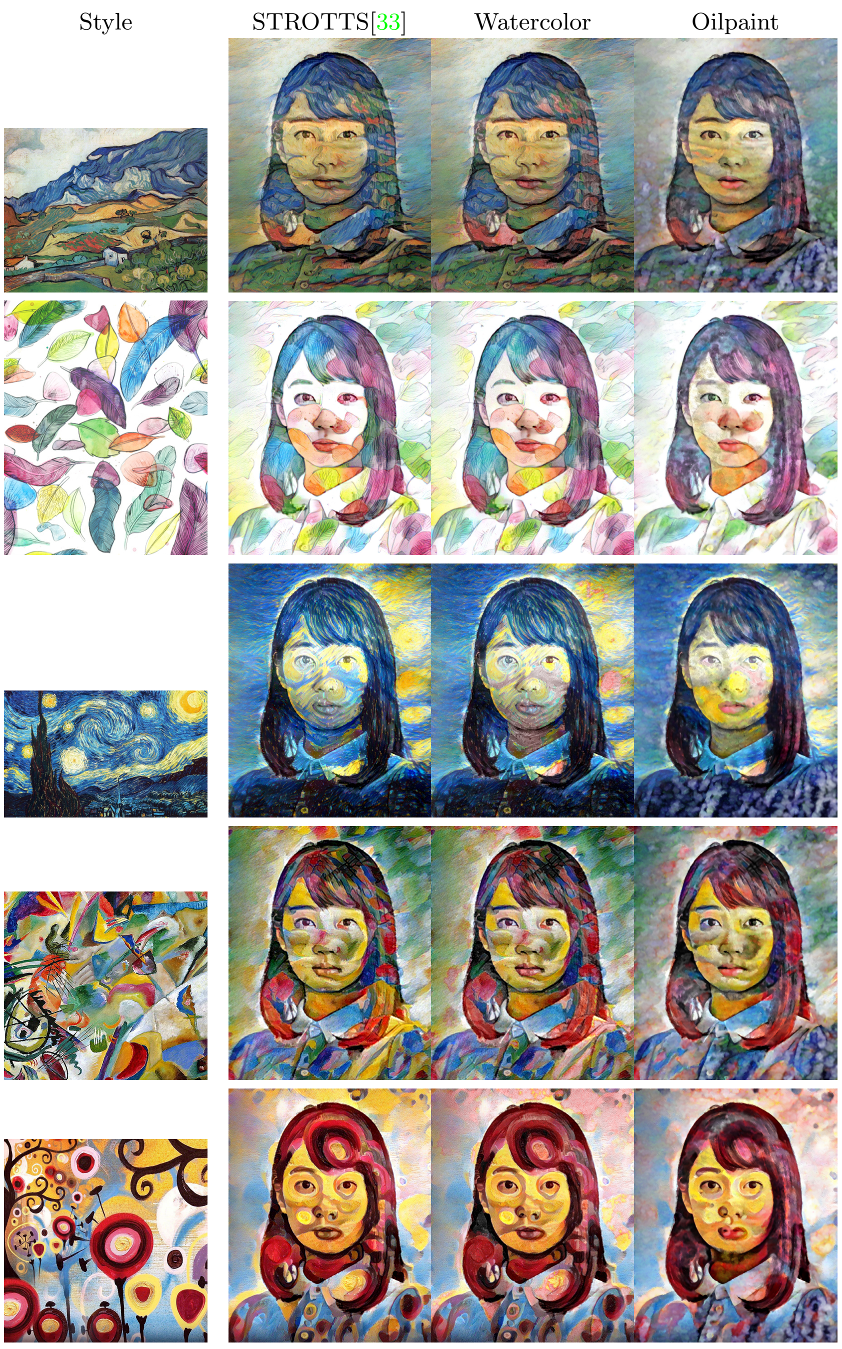


Fig. 5: Style transfer capability on common NST styles.

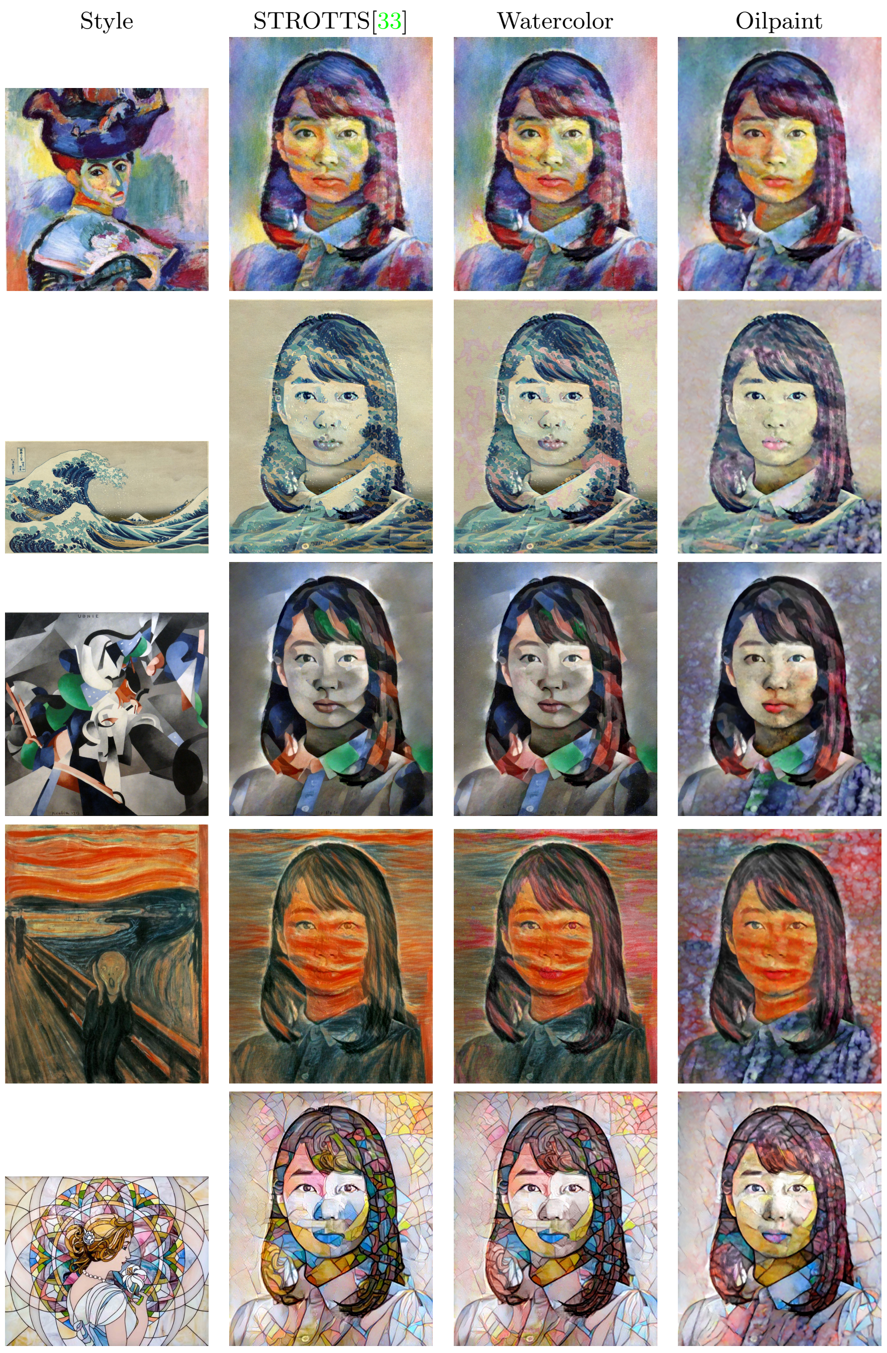


Fig. 6: Style transfer capability on common NST styles.

4 GAN-based image-to-image translation with PPNs

Here, we provide additional details to GAN-based image-to-image translation with PPNs. In an ablation study (Sec. 4.1) we elaborate on the architecture evaluation section on combining algorithmic effects and CNNs in the main paper (Sec. 5.2). We then provide additional comparisons to state of the art method, show predicted parameter masks and evaluate the methods editability in Sec. 4.2.

4.1 Ablation study for APDrawing Stylization

Table 3: Ablation study for our model on the APDrawing [68] dataset. The FID score between the train and the test set can be used as a baseline for all results.

XDoG	PPN weights	Postprocessing Architecture	Dropout	FID score	Key in Fig. 7
✗	-	U-Net	✓	75.27	-
✗	-	U-Net	✗	71.26	-
✗	-	ResNet	✓	70.00	-
✗	-	ResNet	✗	62.44	(a)
✓	Random	U-Net	✓	86.73	-
✓	Random	U-Net	✗	89.93	-
✓	Random	ResNet	✓	71.92	-
✓	Random	ResNet	✗	**60.55**	(c)
✓	ImageNet	U-Net	✓	100.41	-
✓	ImageNet	U-Net	✗	79.56	-
✓	ImageNet	ResNet	✓	76.25	-
✓	ImageNet	ResNet	✗	64.81	-
✓	-	U-Net	✓	71.64	-
✓	-	U-Net	✗	75.40	-
✓	-	ResNet	✓	71.56	-
✓	-	ResNet	✗	73.77	-
APDrawing GAN [68]				62.14	(b)
Train vs. Test				49.72	-

The results are summarized in Tab. 3. For the ResNet-50 backbone of the PPN, we compare initialization with random weights and weights pre-trained on ImageNet. For the convolutional postprocessing network in our generator, we compare the ResNet-based network architecture of Johnson *et al.* [28] and a classical U-Net[52] without residual blocks following Isola *et al.* [24][2]. We also investigate the effect of dropout. As it decreases the FID compared to architectures trained without dropout, the model clearly benefits from having the full learning capacity at its disposal during training. Also, pre-trained weights on ImageNet do not increase the ability of the model to adapt to data successfully.

[2] We adapt the implementation of https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

We also test whether parameter prediction and thus differentiable effects are necessary by comparing our method to the same approach using a fixed parameter preset, which is optimized for portrait data specifically. Our PPN network can dynamically adapt parameters locally to input images. As the results using our PPN and those with a fixed preset differ by a large margin, we conclude that parameter prediction and thus differentiable effects play an important role in the success of our method. We do not include results using only the differentiable XDoG effect in conjunction with a PPN. All experiments without a separate convolution network failed with mode collapse of the generator, i.e., the PPN started to predict the same parameters regardless of the input. We argue that this behavior is related to the missing ability of the XDoG effect to model the artist's style accurately.

4.2 Additional results and editability

Results. Fig. 7 shows additional comparisons of our model trained on APDrawing [68] against state of the art methods. We plot the parameter masks of our results in the latter figure in Fig. 8.

Editability. In Fig. 9 we show that while the postprocessing CNN reduces the stylistic variance that can be achieved with XDoG, results main editable vs. results of APDrawingGAN [68] and also remain stylistically close to the reference. Combining pretrained GANs such as APDrawingGAN++ [69] with a XDoG effect for postprocessing, on the other hand, does not yield visually appealing results and results deviate from the style of the reference, as shown in the last row, thus validating our integrated training approach.

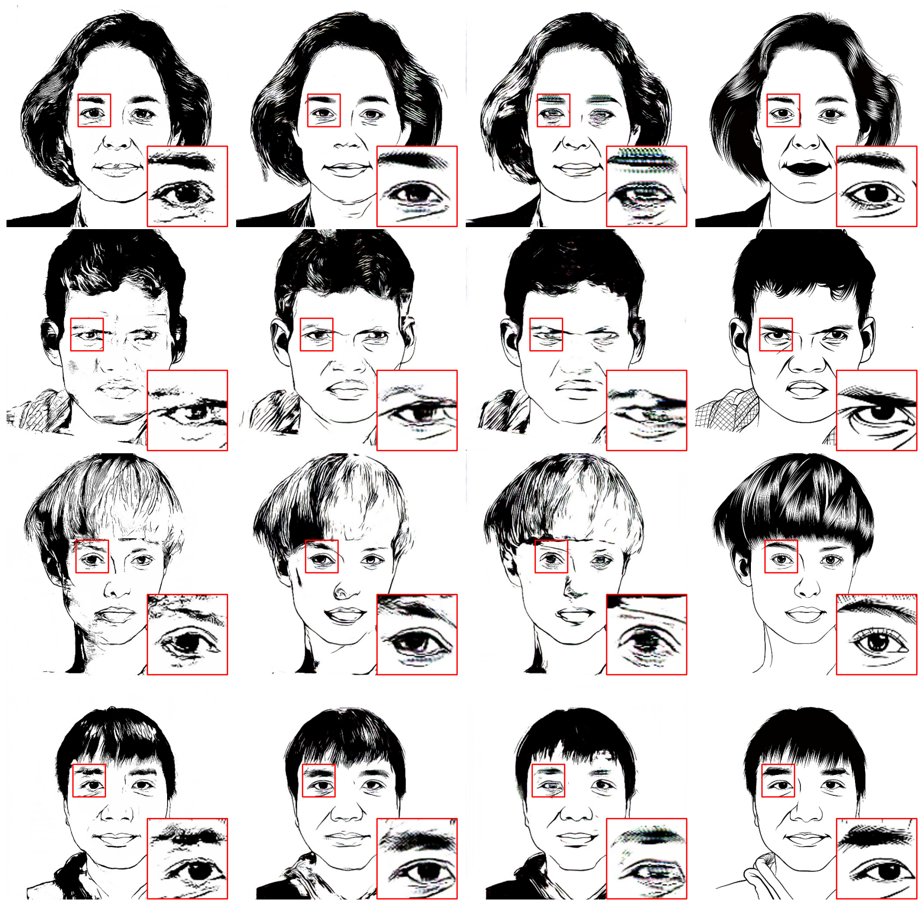


Fig. 7: Results on the APDrawing [68] dataset, our method uses XDoG and a postprocessing CNN. Our method, in contrast to APDrawing GANs[68,69], often produces more consistent lines e.g., for the eyes, and does not need any known facial landmarks at both training and inference time.

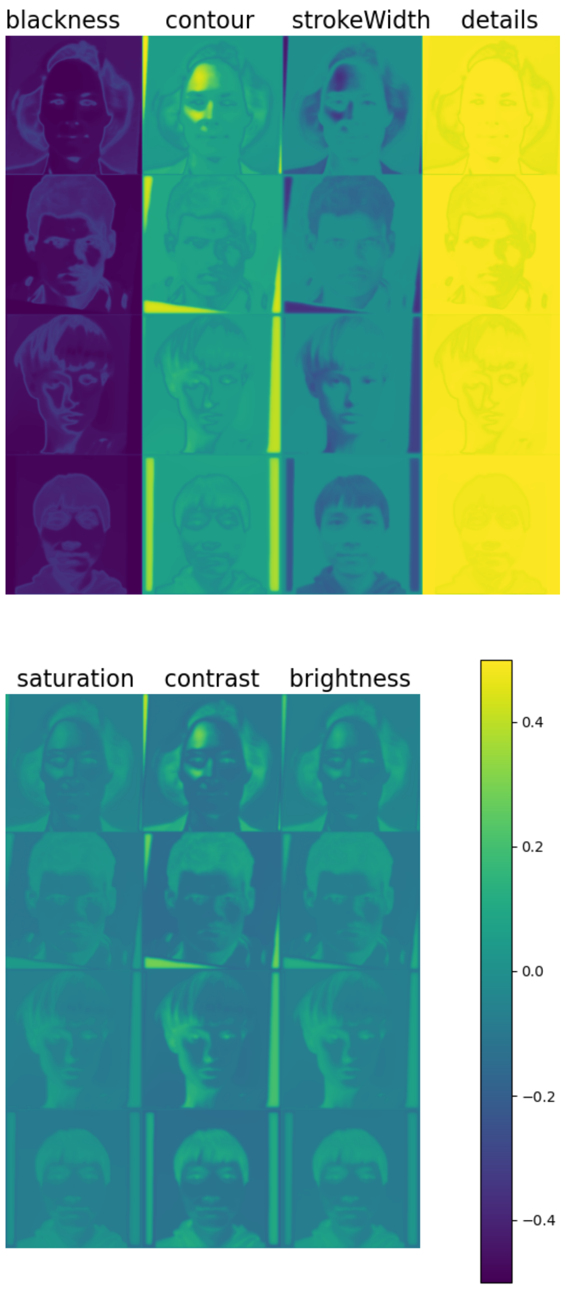


Fig. 8: In addition to the generated outputs in figure 7, we show the parameter masks that have been predicted for the first stage of our method (XDoG). See Fig. 8 of the main paper for a plot of the intermediate results after processing with XDoG.

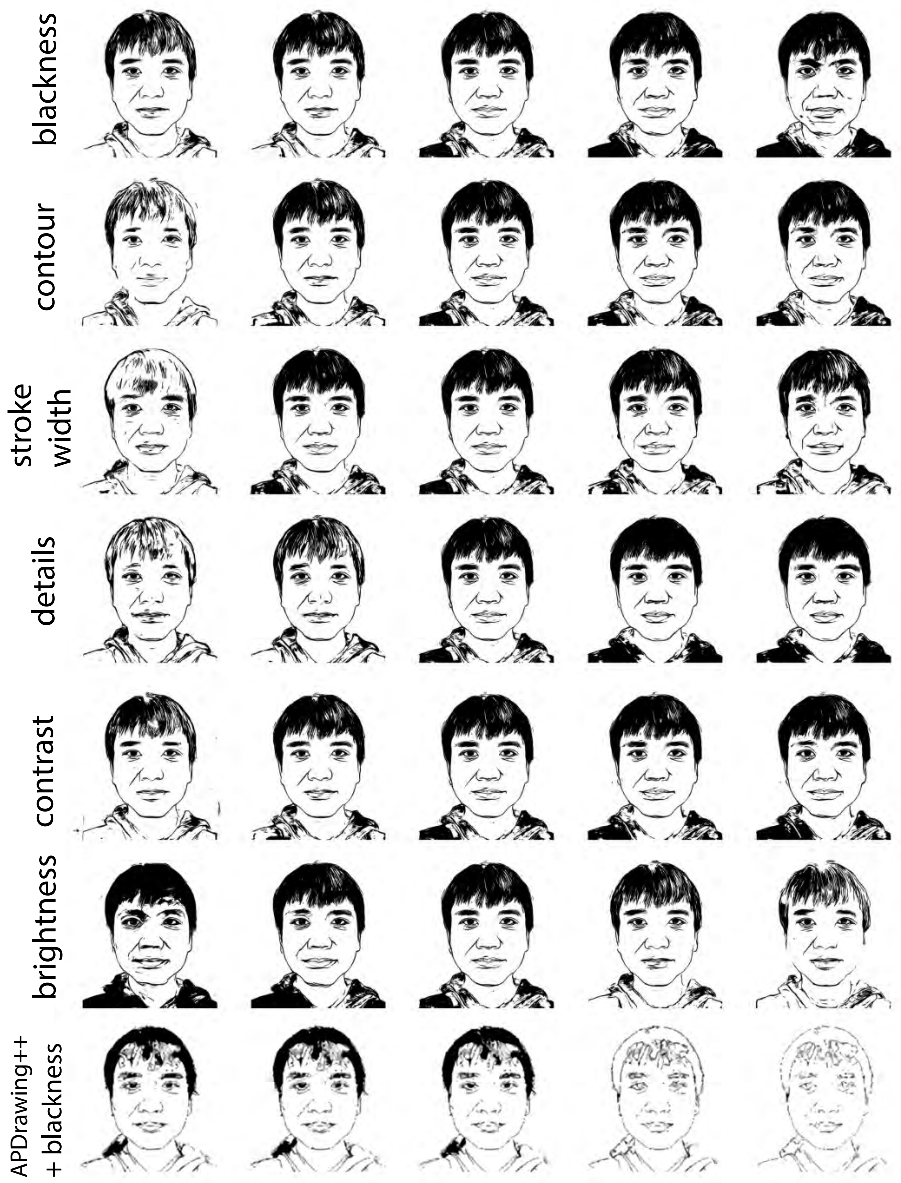


Fig. 9: Adjusting parameters of the XDoG effect after prediction using our XDoG+CNN method trained on APDrawing [68]. Each row corresponds to global changes of one parameter from lowest to highest value in the visual range, where unedited results (i.e., the predicted parameters) are shown in the middle column.

5 Additional Results

This section provides additional results of our method, i.e., for content-adaptive effects Sec. 5.1, and parametric style transfer Sec. 5.2. Further results on AP-Drawing depicted in Fig. 13 demostrate that our model generalizes to portraits from other datasets, such as the NPR benchmark [54]. Finally, we show baseline results of our implemented algorithmic differentiable effects with different parameter variants in Sec. 5.4.

5.1 Content-adaptive Effects

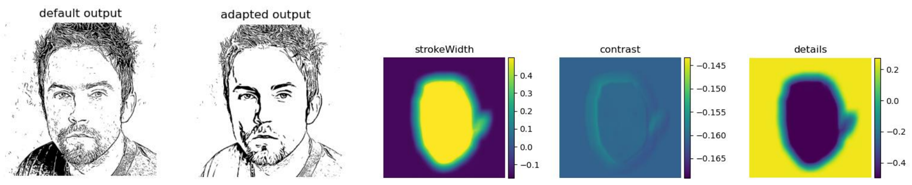


Fig. 10: Facial feature enhancement and predicted local parameter maps.

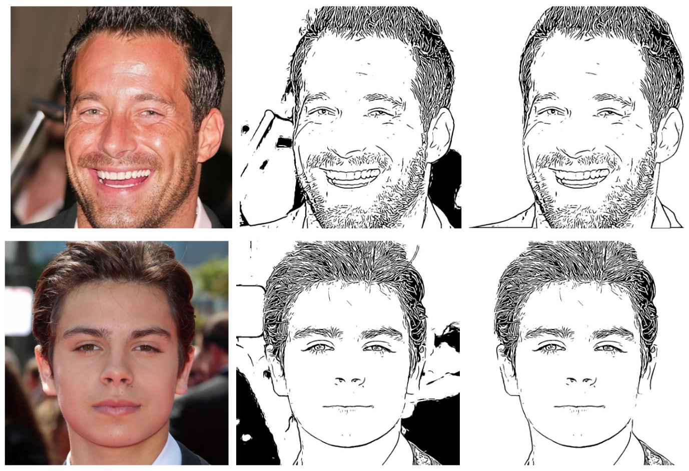


Fig. 11: Background removal. The PPN learns to reduce details and stroke-width for the XDoG filter in such a way that background details seen in the default output (middle) are removed the output with locally adapted parameters (right).

5.2 Style Transfer

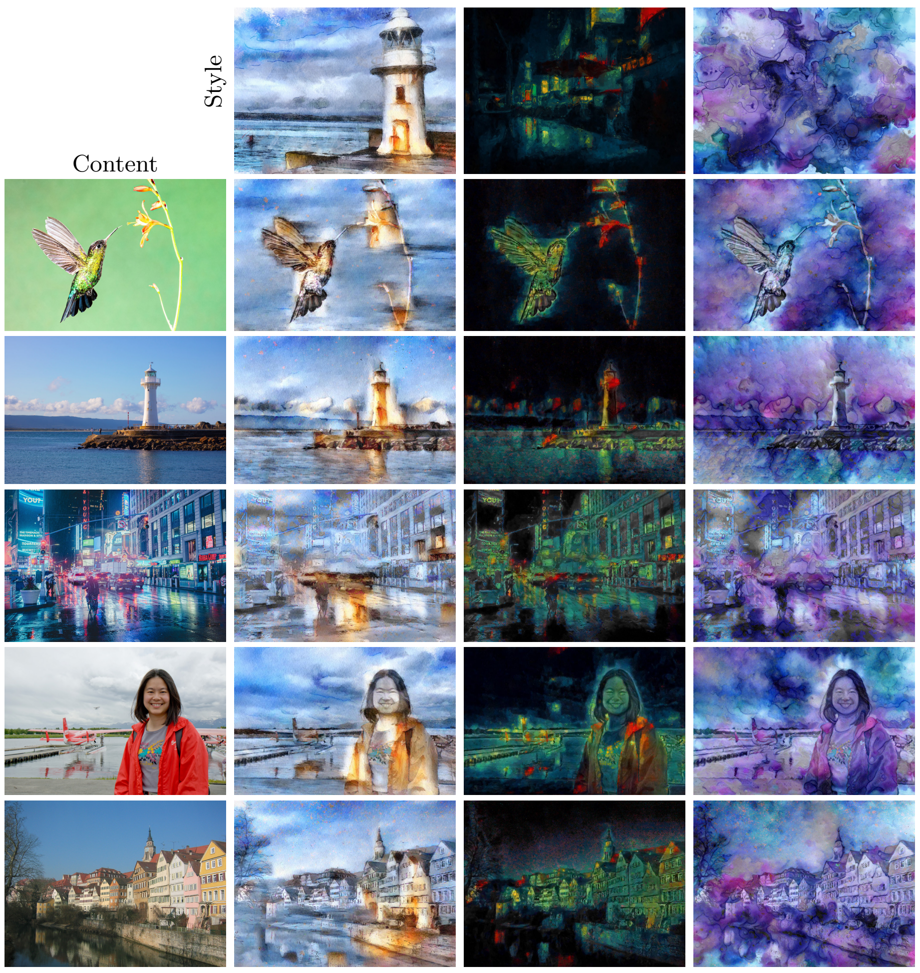


Fig. 12: Additional results for style transfer optimization. Our parametric style transfer can match a wide variety of styles and remains editable after optimization (not performed here). While the presented parameter smoothing schedule removes most artefacts stemming from parameters falling into local optima, some can remain in the final image (best seen zoomed in). These can be easily removed in a final, manual parameter editing step, as shown in the supplemental video.

5.3 Image Translation on APDrawing

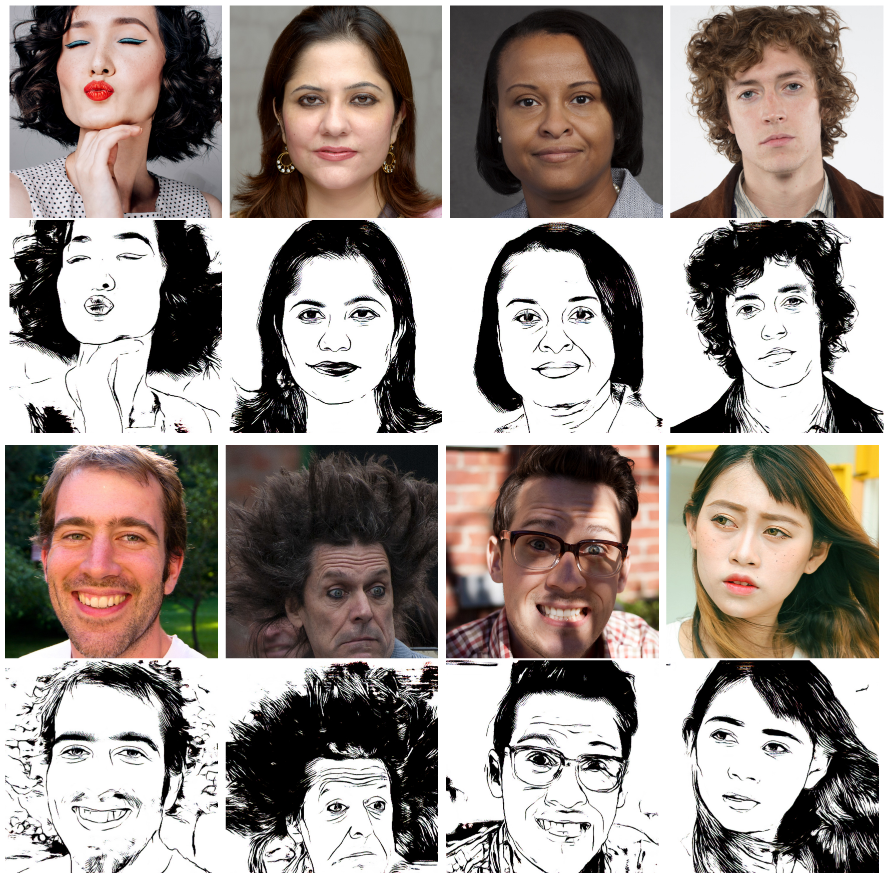
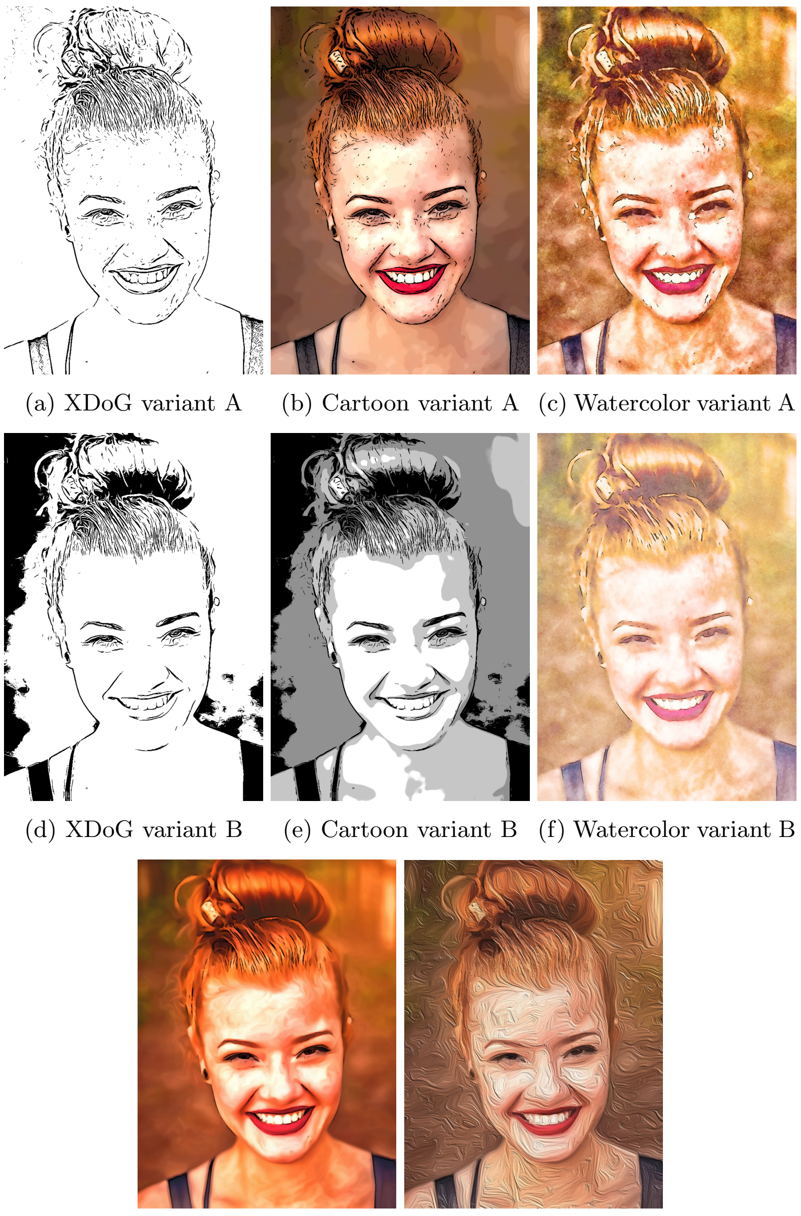


Fig. 13: Our model generalizes to example images[3] drawn from the NPR benchmark [54]. Our method retains only salient lines in the face, abstracting irrelevant details. Furthermore, lines generally flow consistently without unnatural discontinuities. Our model has been trained on portraits with uniform background only, thus prediction on images with non-uniform backgrounds (e.g., images in bottom row) may lead to background artefacts in the stylization.

[3] We show only images that are not part of APDrawing trainset

5.4 Effect Variants



(a) XDoG variant A (b) Cartoon variant A (c) Watercolor variant A

(d) XDoG variant B (e) Cartoon variant B (f) Watercolor variant B

(g) Oilpaint variant A (h) Oilpaint variant B

Fig. 14: A selection of global parametrization variants of our differentiable effects are shown. These represent default states of the effect without any parameter learning applied.