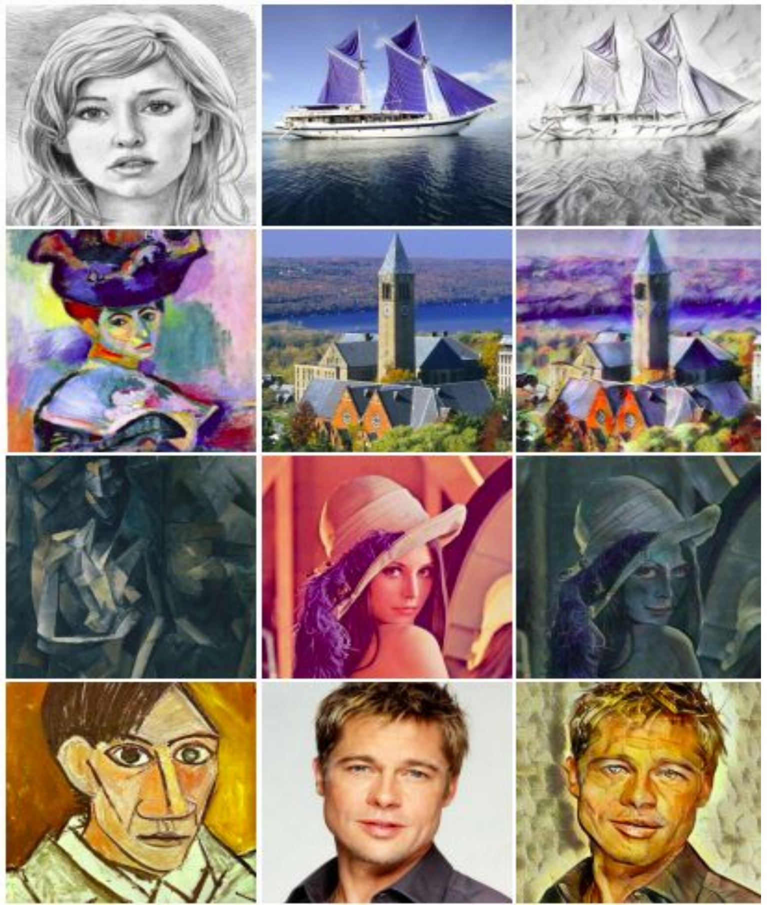
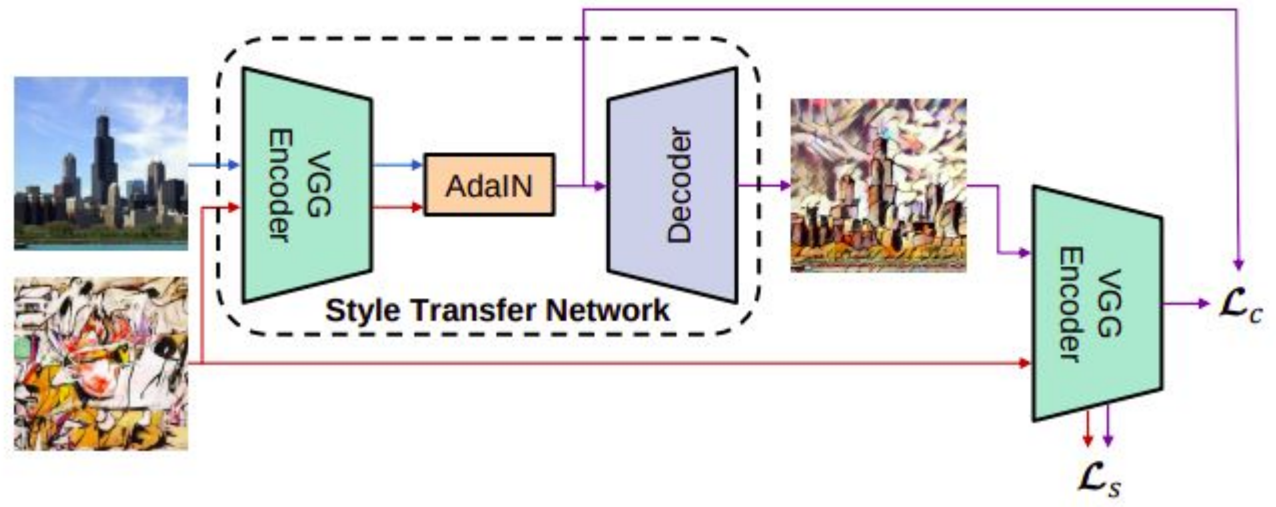
Papers:

- AdaAttN: Revisit Attention Mechanism in Arbitrary Neural Style Transfer in ICCV 2021
  Authors: Songhua Liu,Tianwei Lin,Dongliang He,Fu Li,Meiling Wang,Xin Li,Zhengxing Sun,Qian Li,Errui Ding

- Video Autoencoder: self-supervised disentanglement of static 3D structure and motion in ICCV 2021
  Authors: Zihang Lai,Sifei Liu,Alexei A. Efros,Xiaolong Wang

Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization in ICCV 2017





f is encoder,g is decoder

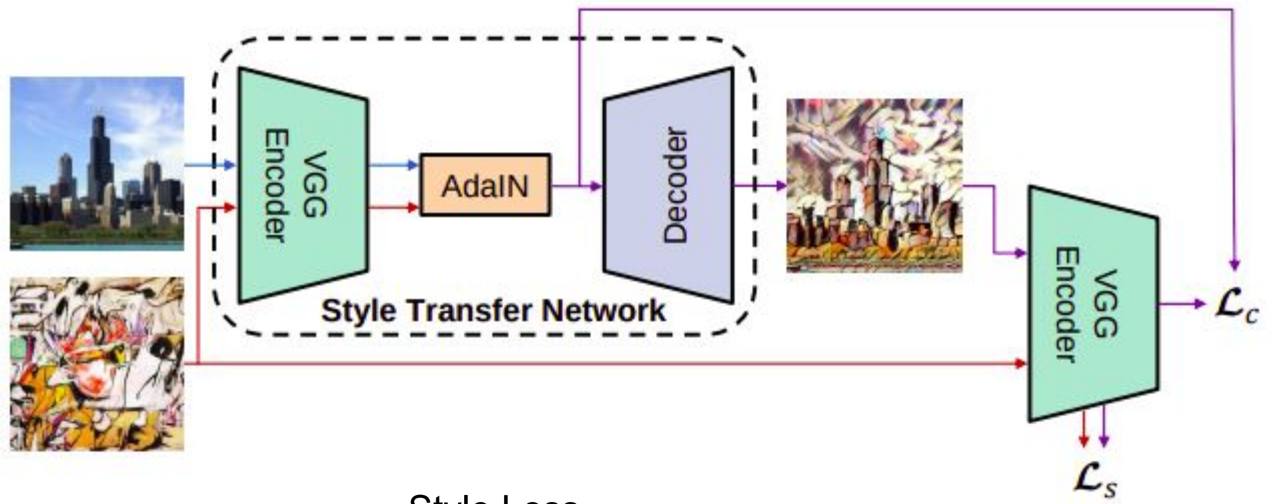
$$t = \text{AdaIN}(f(c), f(s))$$

Content Loss

$$\mathcal{L}_c = \|f(g(t)) - t\|_2$$

Overall Loss

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s$$

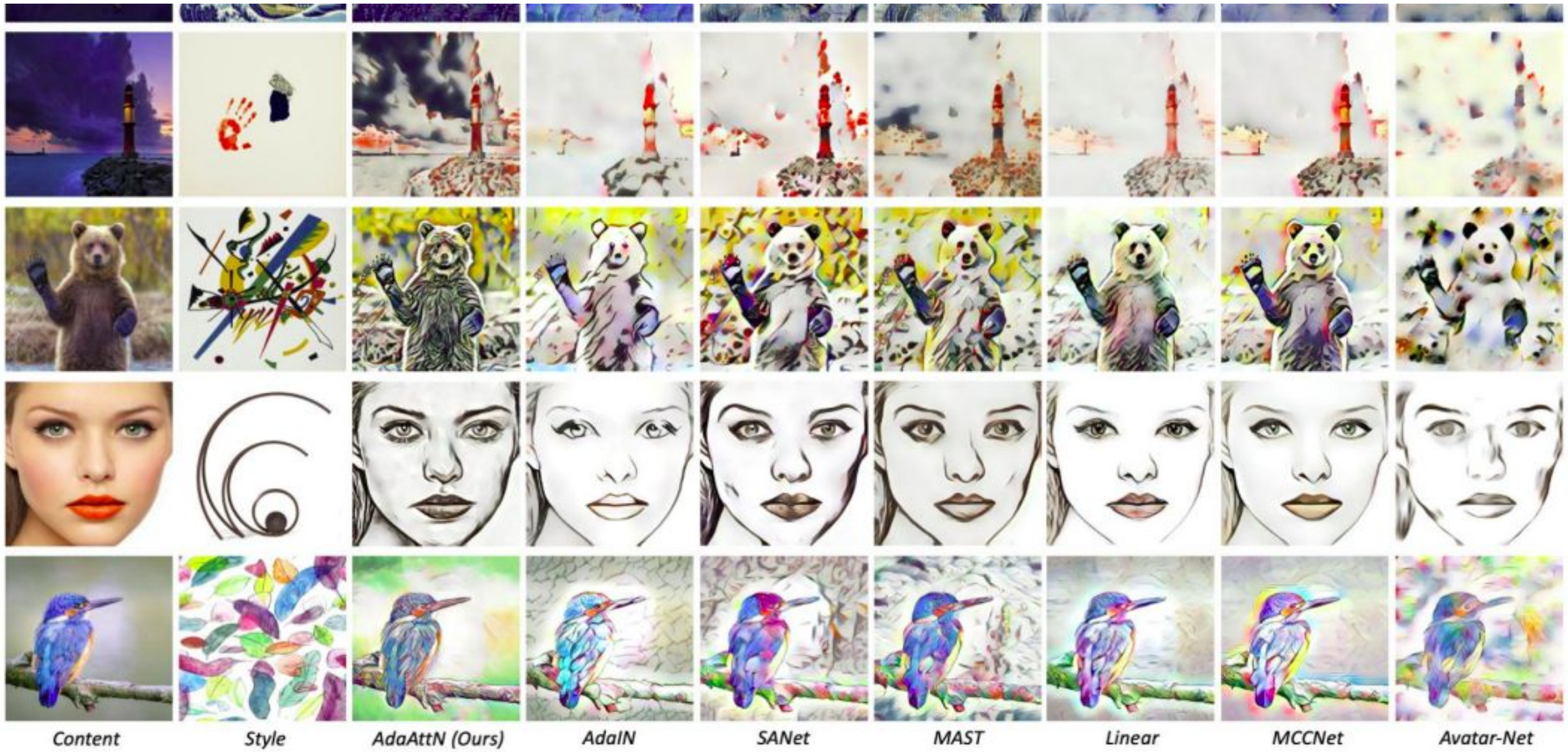


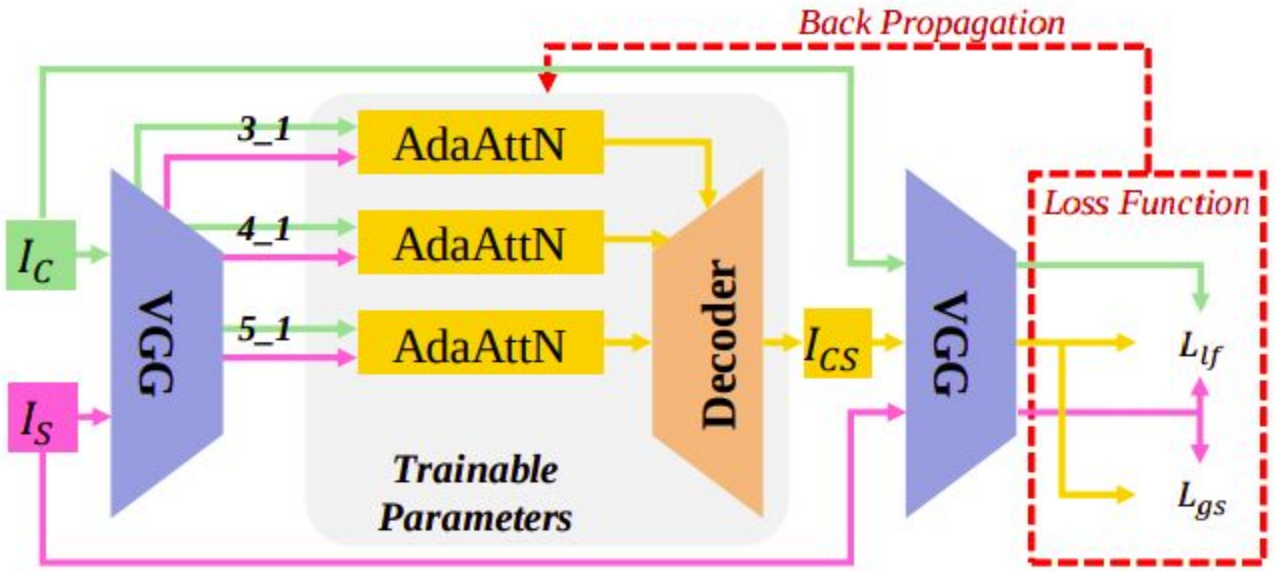
Style Loss

$$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu(\phi_i(g(t))) - \mu(\phi_i(s))\|_2 \quad + \sum_{i=1}^{L} \|\sigma(\phi_i(g(t))) - \sigma(\phi_i(s))\|_2$$

Φ_i is ith relu layer

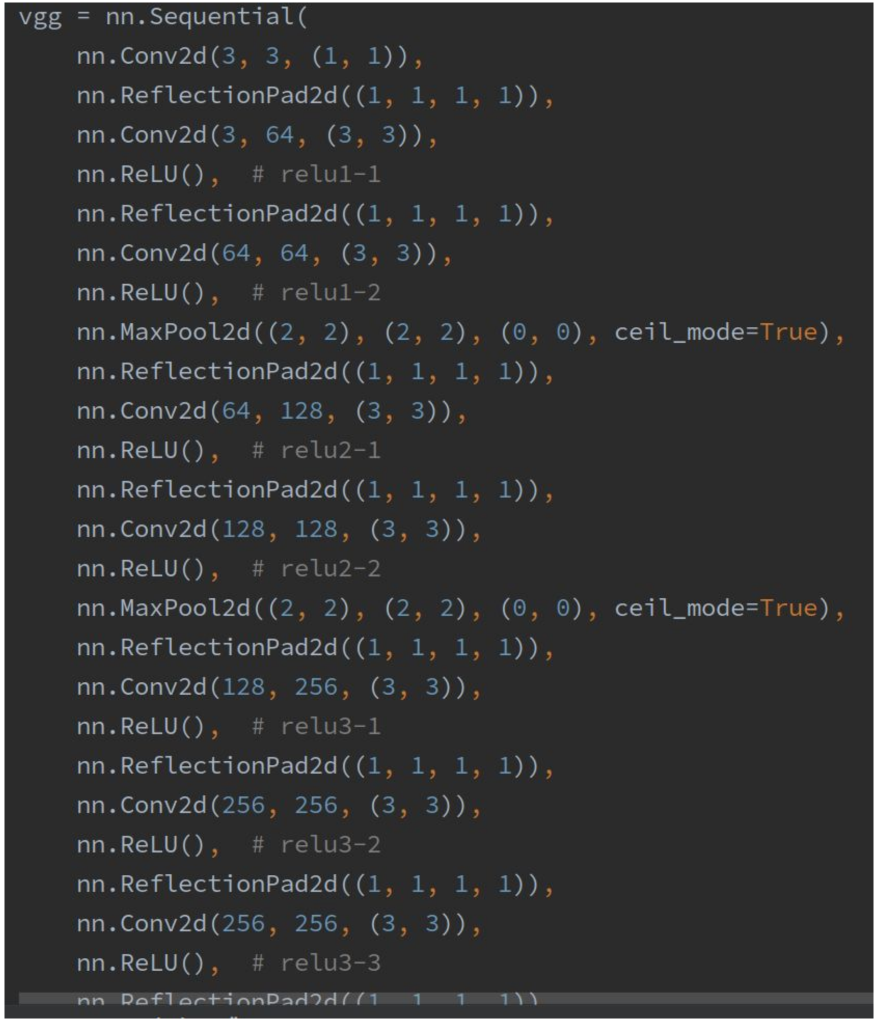
AdaAttN: Revisit Attention Mechanism in Arbitrary Neural Style Transfer in ICCV 2021



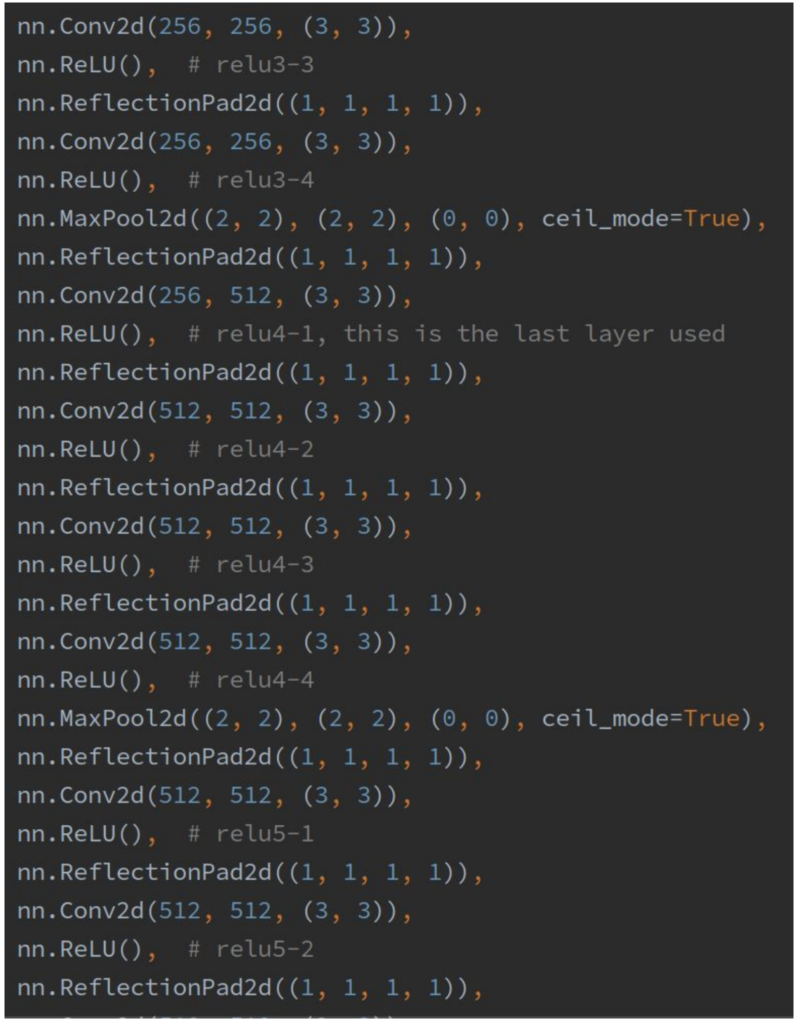


L_{lf}: Local feature loss

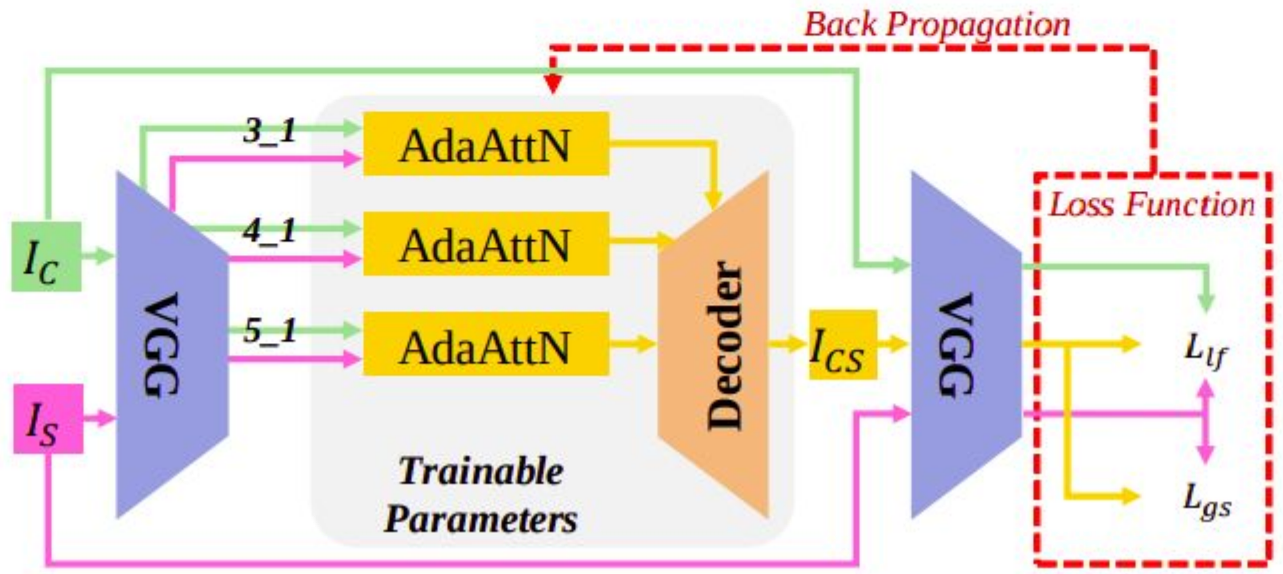
L_{gs}: GLobal style loss



```
vgg = nn.Sequential(
    nn.Conv2d(3, 3, (1, 1)),
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(3, 64, (3, 3)),
    nn.ReLU(),  # relu1-1
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(64, 64, (3, 3)),
    nn.ReLU(),  # relu1-2
    nn.MaxPool2d((2, 2), (2, 2), (0, 0), ceil_mode=True),
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(64, 128, (3, 3)),
    nn.ReLU(),  # relu2-1
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(128, 128, (3, 3)),
    nn.ReLU(),  # relu2-2
    nn.MaxPool2d((2, 2), (2, 2), (0, 0), ceil_mode=True),
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(128, 256, (3, 3)),
    nn.ReLU(),  # relu3-1
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(256, 256, (3, 3)),
    nn.ReLU(),  # relu3-2
    nn.ReflectionPad2d((1, 1, 1, 1)),
    nn.Conv2d(256, 256, (3, 3)),
    nn.ReLU(),  # relu3-3
    nn.ReflectionPad2d((1, 1, 1, 1))
```



```
nn.Conv2d(256, 256, (3, 3)),
nn.ReLU(),  # relu3-3
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(256, 256, (3, 3)),
nn.ReLU(),  # relu3-4
nn.MaxPool2d((2, 2), (2, 2), (0, 0), ceil_mode=True),
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(256, 512, (3, 3)),
nn.ReLU(),  # relu4-1, this is the last layer used
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(512, 512, (3, 3)),
nn.ReLU(),  # relu4-2
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(512, 512, (3, 3)),
nn.ReLU(),  # relu4-3
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(512, 512, (3, 3)),
nn.ReLU(),  # relu4-4
nn.MaxPool2d((2, 2), (2, 2), (0, 0), ceil_mode=True),
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(512, 512, (3, 3)),
nn.ReLU(),  # relu5-1
nn.ReflectionPad2d((1, 1, 1, 1)),
nn.Conv2d(512, 512, (3, 3)),
nn.ReLU(),  # relu5-2
nn.ReflectionPad2d((1, 1, 1, 1)),
```

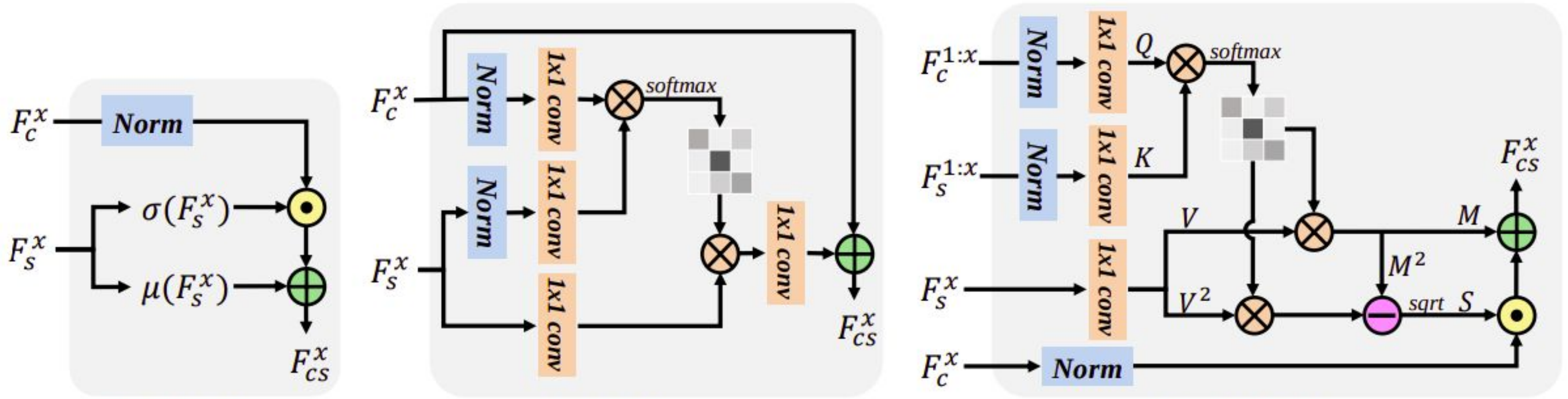


$$F_*^x \in R^{C \times H_* W_*}$$

* can be c or s

$$F_*^{1:x} = D_x(F_*^1) \oplus D_x(F_*^2) \oplus \cdots \oplus F_*^x$$

$$F_{cs}^x = AdaAttN(F_c^x, F_s^x, F_c^{1:x}, F_s^{1:x})$$



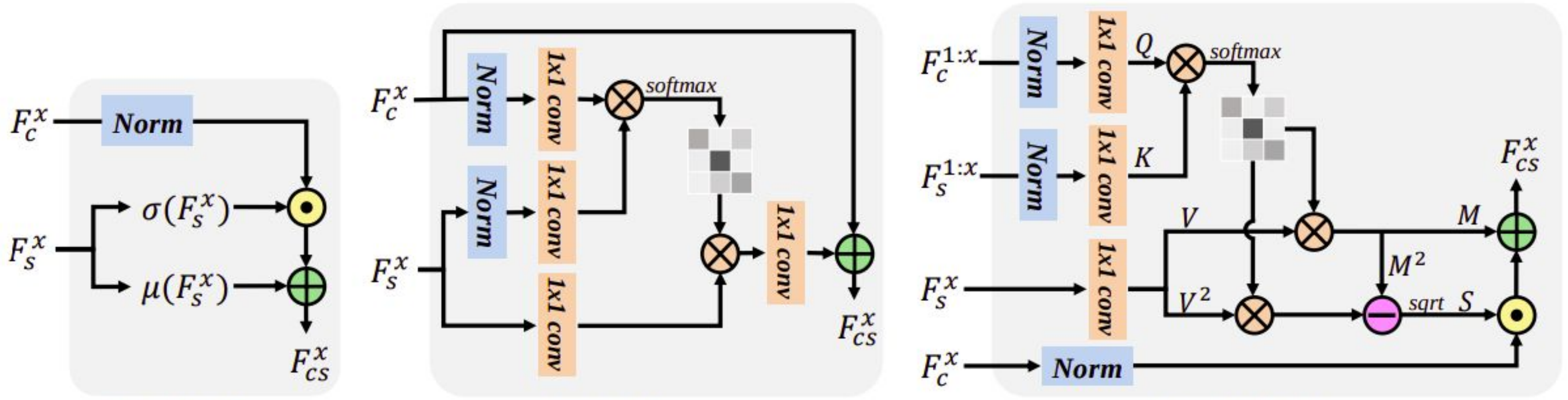
(a) AdaIN (b) SANet (c) AdaAttN

$$Q = f(Norm(F_c^{1:x})),$$ f, g, and h are 1×1 learnable convolution layers
$$K = g(Norm(F_s^{1:x})),$$
$$V = h(F_s^x),$$

Like.SANet calculates each target style feature point by weighted summation of all style feature points.Authors,considered Attention output as a distribution of all the weighted style feature points.

$$A = Softmax(Q^\top \otimes K)$$

where $\otimes$ denotes matrix multiplication



(a) AdaIN (b) SANet (c) AdaAttN

$M = V \otimes A^{\top}$ $M \in R^{C \times H_c W_c}$ $A \in R^{H_c W_c \times H_s W_s}$ and $V \in R^{C \times H_s W_s}$

$S = \sqrt{(V \cdot V) \otimes A^{\top} - M \cdot M}$ where $\cdot$ denotes element-wise product

$F_{cs}^x = S \cdot Norm(F_c^x) + M$

Loss Functions

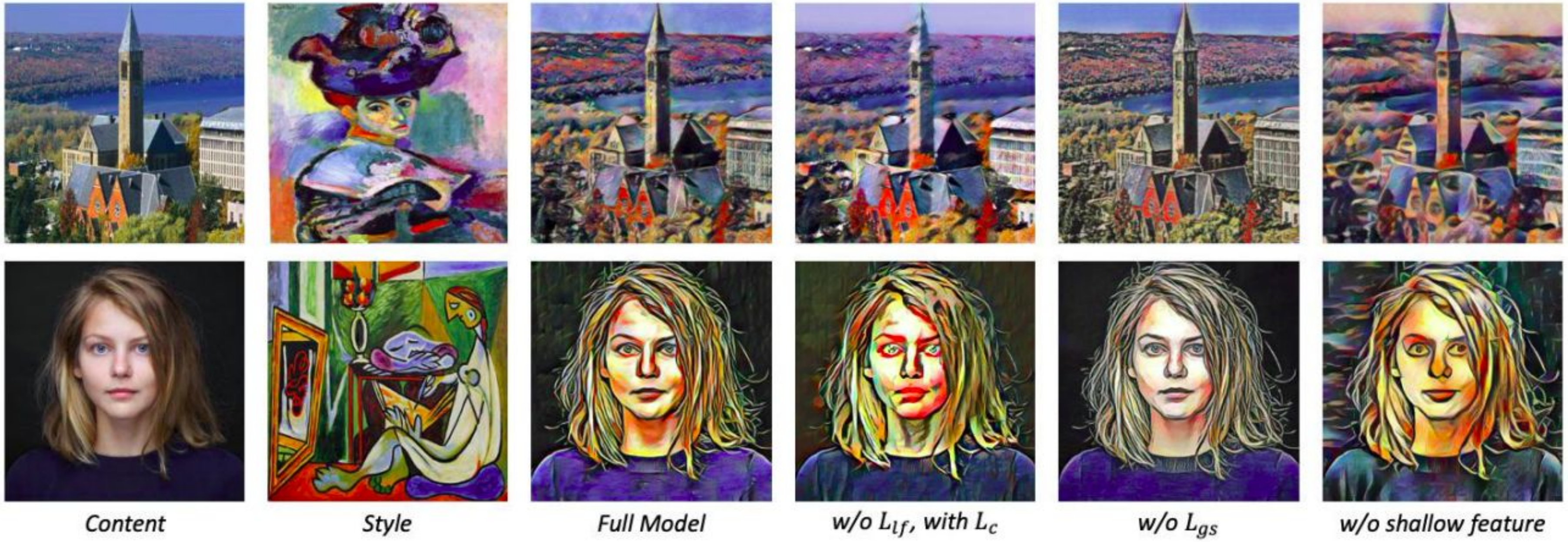
$$\mathcal{L} = \lambda_g \mathcal{L}_{gs} + \lambda_l \mathcal{L}_{lf}$$

$$\mathcal{L}_{gs} = \sum_{x=2}^{5} (||\mu(E^x(I_{cs})) - \mu(F_s^x)||_2 + ||\sigma(E^x(I_{cs})) - \sigma(F_s^x)||_2),$$

$$\mathcal{L}_{lf} = \sum_{x=3}^{5} \left|\left| E^x(I_{cs}) - AdaAttN^*(F_c^x, F_s^x, F_c^{1:x}, F_s^{1:x}) \right|\right|_2,$$

where E() denoted the feature of VGG encoder.

Ablation Studies



Qualitative Analysis

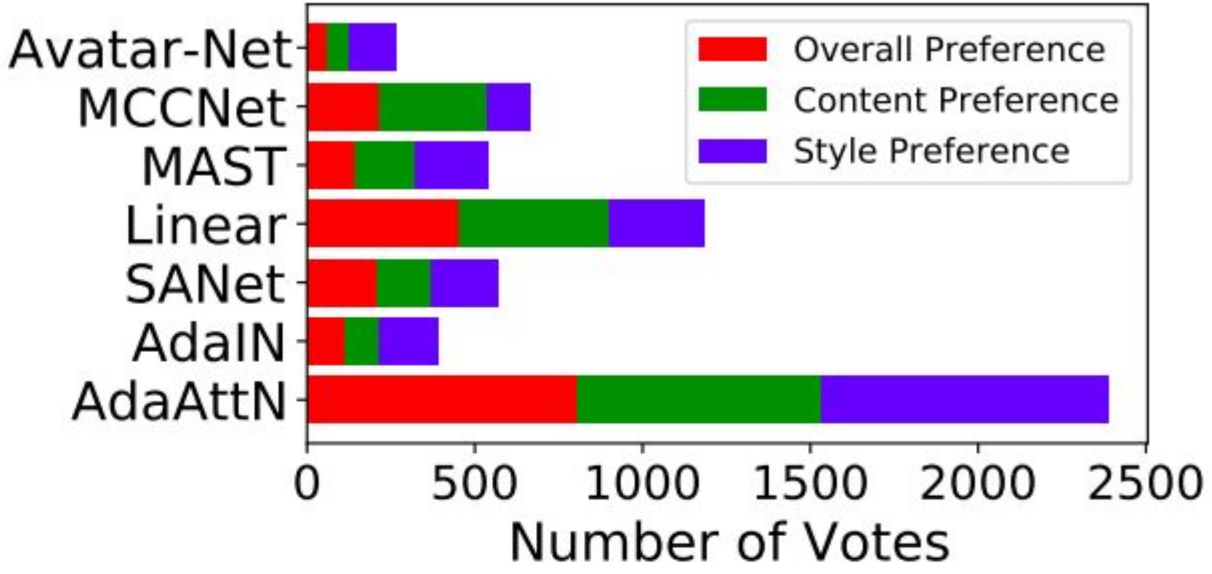


Figure 6. Results of user study.

Extension for Video Style Transfer

Authors observe properties about using softmax:

- Does exponential computation in attention
- major focus on local patterns
- negative influence in stabilization

Solution proposed is to use cosine similarity

$$A_{i,j} = \frac{S_{i,j}}{\sum_j S_{i,j}}, S_{i,j} = \frac{Q_i \cdot K_j}{||Q_i|| \times ||K_j||} + 1$$

Effect of using cosine similarity:

- More flat attention score distribution than softmax which leads to:
- More stable local feature statistics
- local style patterns will not be over emphasised
- More consistency

Extension for Video Style Transfer

Novel cross-image similarity loss based on attention mechanism

$$\mathcal{L}_{is} = \sum_{x=2}^{4} \frac{1}{N_{c_1}^{x} N_{c_2}^{x}} \sum_{i,j} \left| \frac{D_{c_1,c_2}^{i,j,x}}{\sum_i D_{c_1,c_2}^{i,j,x}} - \frac{D_{cs_1,cs_2}^{i,j,x}}{\sum_i D_{cs_1,cs_2}^{i,j,x}} \right|$$

$$D_{u,v}^{i,j,x} = 1 - \frac{F_u^{x,i} \cdot F_v^{x,j}}{\left\| F_u^{x,i} \right\| \times \left\| F_v^{x,j} \right\|},$$

- F_c^x: Content feature map
- N_c^x : size of spatial dimension of F_c^x.
- $F_*^{x,i}$: feature vector of F_*^x at ith position.
- $D^{i,j,x}_{u,v}$: cosine distance between numerator quantities.

Benefits of this feature loss intuitively:

- Two content images share similar local similarity patterns
- Awareness of inter-frame relationship

Datasets used:

- MS-COCO for content images
- WikiArt for style images

Quantitative Results

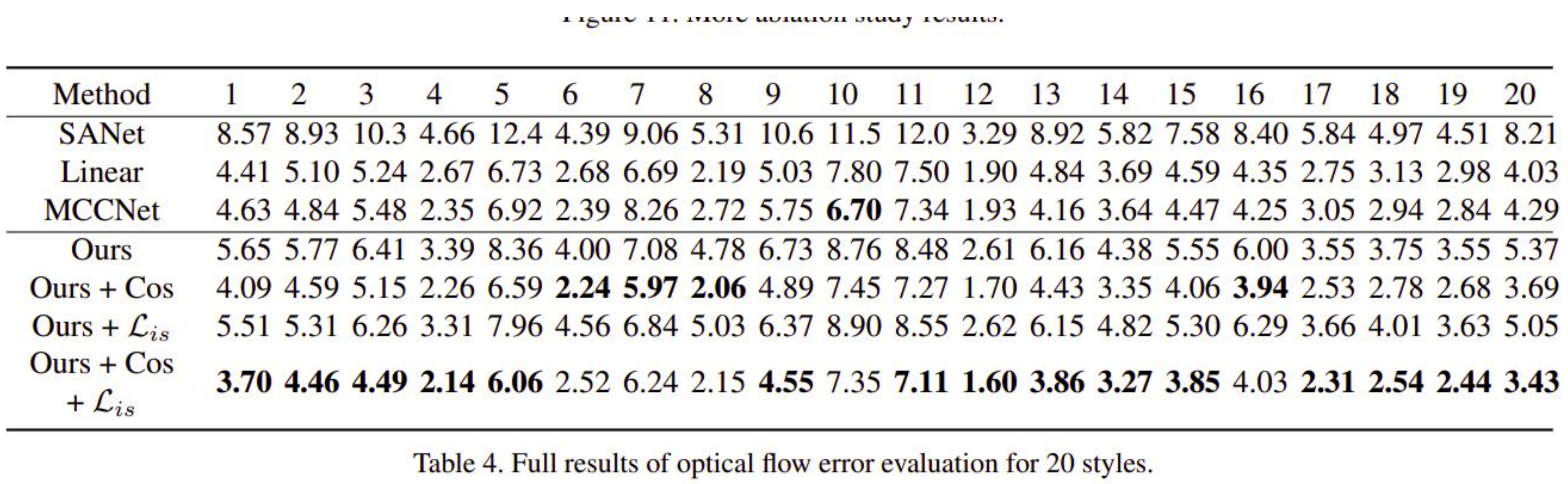


Figure 11. More ablation study results.

Method	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
SANet	8.57	8.93	10.3	4.66	12.4	4.39	9.06	5.31	10.6	11.5	12.0	3.29	8.92	5.82	7.58	8.40	5.84	4.97	4.51	8.21
Linear	4.41	5.10	5.24	2.67	6.73	2.68	6.69	2.19	5.03	7.80	7.50	1.90	4.84	3.69	4.59	4.35	2.75	3.13	2.98	4.03
MCCNet	4.63	4.84	5.48	2.35	6.92	2.39	8.26	2.72	5.75	**6.70**	7.34	1.93	4.16	3.64	4.47	4.25	3.05	2.94	2.84	4.29
Ours	5.65	5.77	6.41	3.39	8.36	4.00	7.08	4.78	6.73	8.76	8.48	2.61	6.16	4.38	5.55	6.00	3.55	3.75	3.55	5.37
Ours + Cos	4.09	4.59	5.15	2.26	6.59	**2.24**	**5.97**	**2.06**	4.89	7.45	7.27	1.70	4.43	3.35	4.06	**3.94**	2.53	2.78	2.68	3.69
Ours + $\mathcal{L}_{is}$	5.51	5.31	6.26	3.31	7.96	4.56	6.84	5.03	6.37	8.90	8.55	2.62	6.15	4.82	5.30	6.29	3.66	4.01	3.63	5.05
Ours + Cos + $\mathcal{L}_{is}$	**3.70**	**4.46**	**4.49**	**2.14**	**6.06**	2.52	6.24	2.15	**4.55**	7.35	**7.11**	**1.60**	**3.86**	**3.27**	**3.85**	4.03	**2.31**	**2.54**	**2.44**	**3.43**

Table 4. Full results of optical flow error evaluation for 20 styles.

Multi-Style Transfer

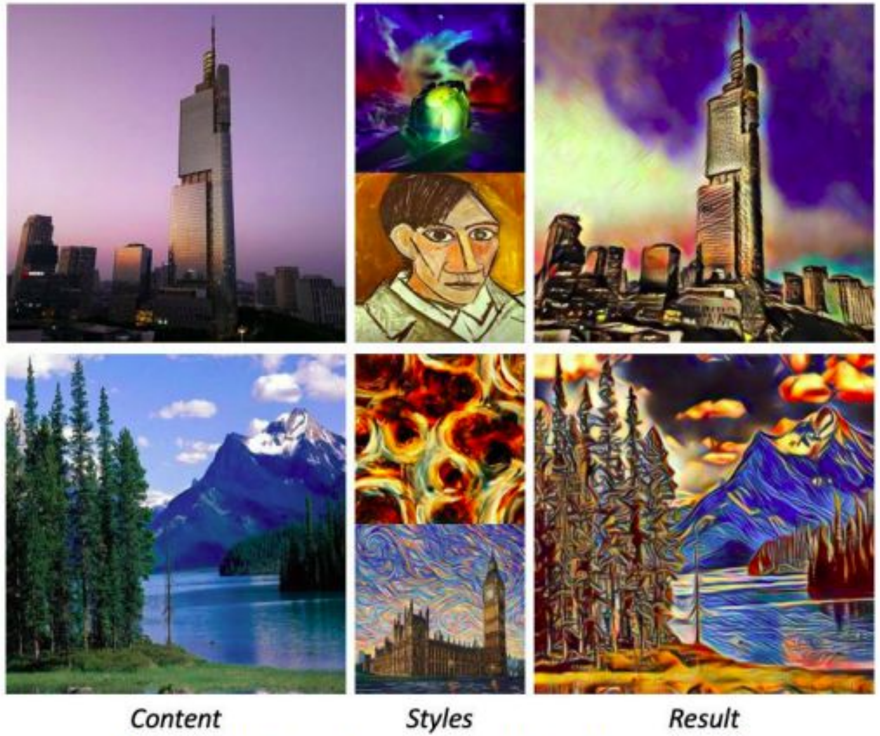
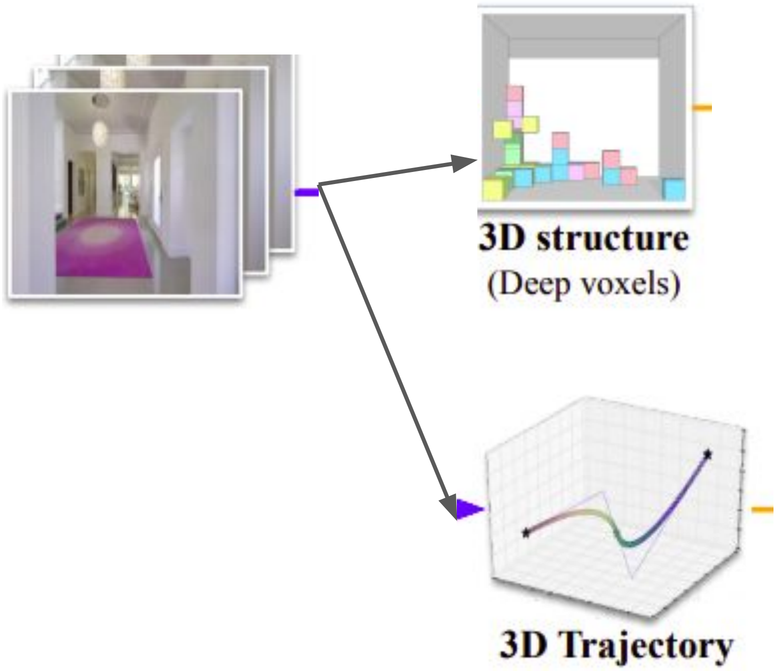


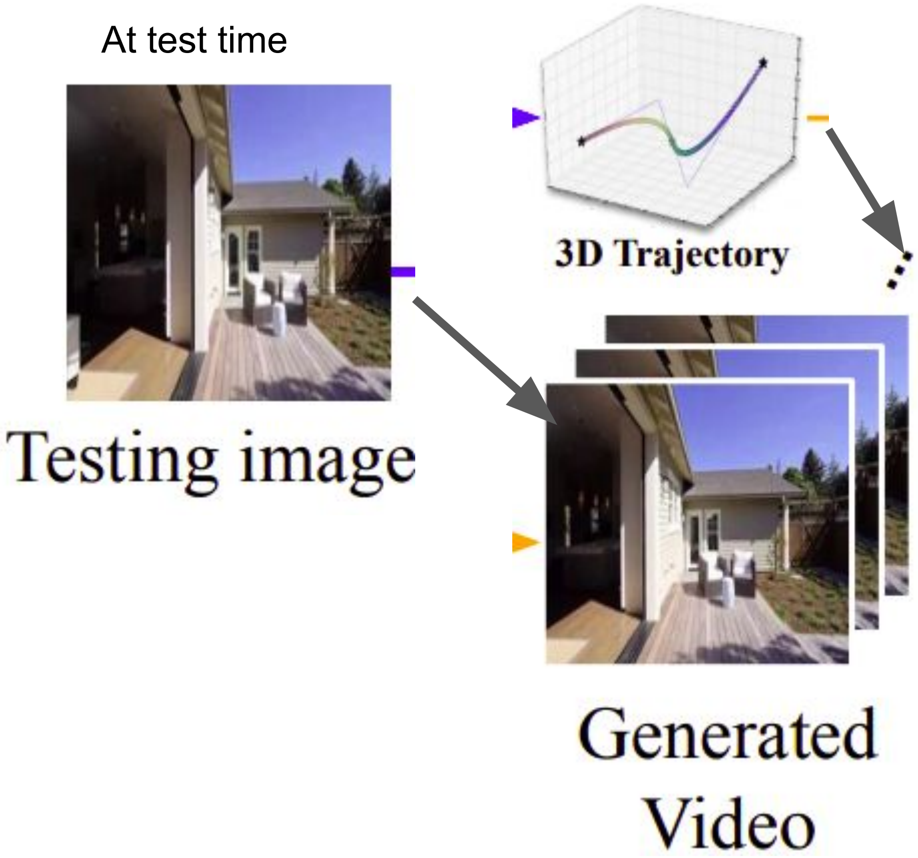
Figure 9. Results of multi-style transfer.

Concatenate style images into one style image.

Video Autoencoder: self-supervised disentanglement of static 3D structure and motion in ICCV 2021



At test time



Applications of method mentioned in the paper

- Novel view synthesis
- Camera pose estimation
- Video generation by motion following

Method overview and assumptions

- Learn a disentangled 3D scene representation,separating 3D scene structure and camera motion.

- Used videos as training data instead of dataset of stills.

- Assumption that local snippet of video is capturing a static scene,so changes in appearance must be due to camera motion.

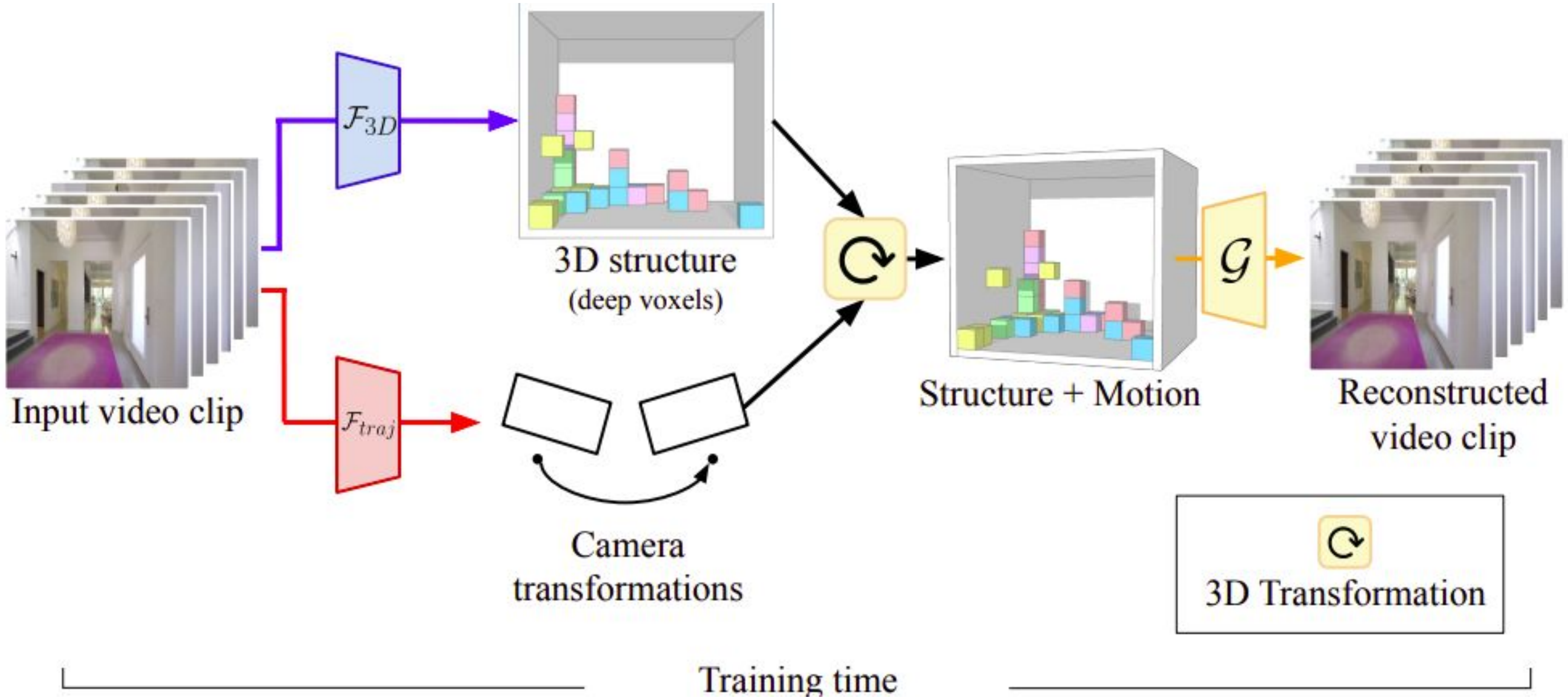
Method overview

- Input video is encoded in 3D scene structure(forced to remain fixed across frames) and camera trajectory(updated for every frame)

- 3D structure is represented by 3D deep voxels and camera pose with 6D vector containing rotation and translation.

Training(Encoding)

- First frame of 6-frame video clip pass through 3D Encoder which give voxel grid of deep features.
- Meanwhile,trajectory encoder use same video clip to produce a short trajectory of three points.
- The trajectory estimate camera pose of each frame w.r.t first frame.

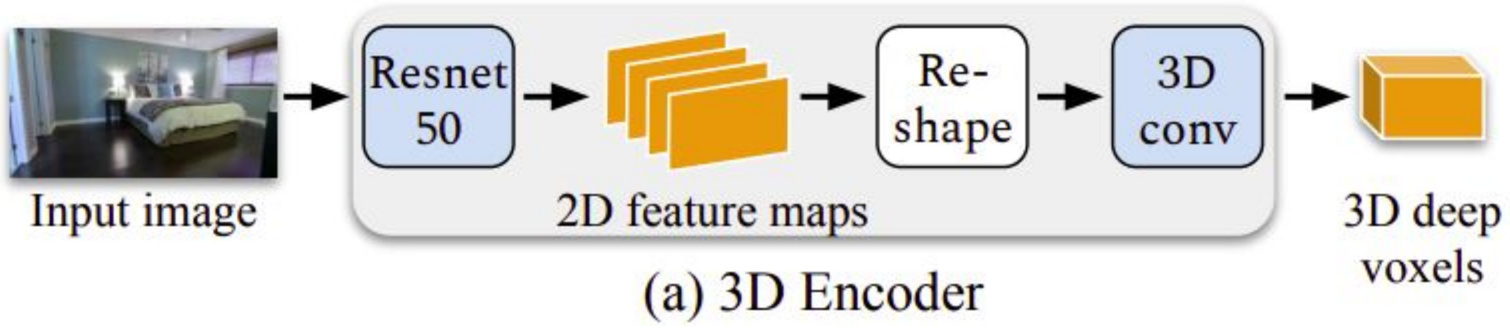

$\mathcal{F}_{3D}$
3D structure
(deep voxels)
Input video clip
$\mathcal{F}_{traj}$
Camera transformations
Structure + Motion
$\mathcal{G}$
Reconstructed video clip
3D Transformation
Training time

Training(Entanglement and Decoding)

- Entangle the 3D structure with one of the camera pose to generate a frame(in feature space) and repeat it to generate a whole reconstructed video clip(feature space).

- We assume since scene is static the reconstructed frame align with the corresponding frame in the feature space.

- Next, we pass this N rendered frame(feature space) to decoder to generate the reconstructed video clip.



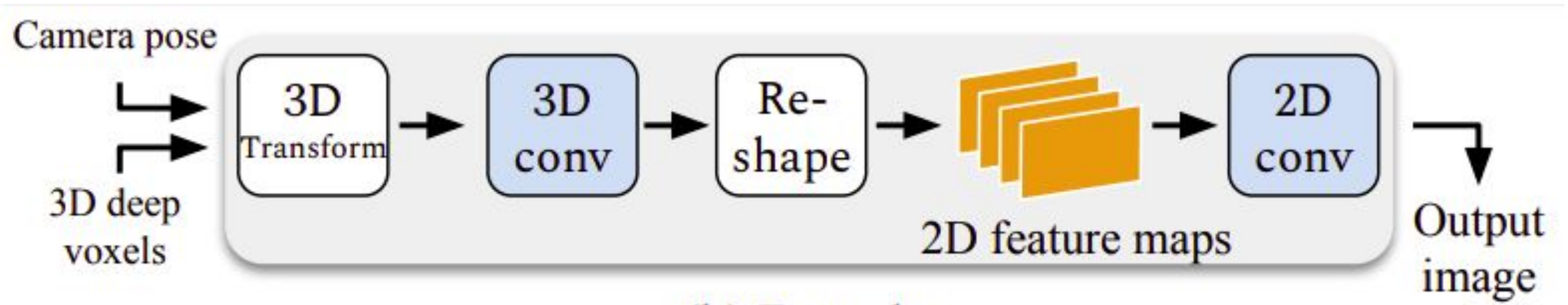
- In Reshape operation (H,W,C) -> (H,W,D,C/D) where D is depth dimension
- To,reconstruct image of high fidelity, we upsample and refine 3D Structure using strided 3D Conv.

Note: According to authors other representation like point cloud,polygon mesh will work as well but not as simple as that of voxels.

Trajectory Encoder

$$E = \mathcal{F}_{\text{traj}}(V) = \{\mathcal{H}(I_1, I_i)\}_{i=1}^{N}$$

- Encoder computes 6D vector camera pose for each frame and we have N such frames.
- The sequence of this camera poses is called trajectory.
- We need to find the relative poses for particular target frame and reference frame(first frame) by using ConvNet *H*.
- *H* has 7 2D-Conv layers whose input is reference,target image concatenated along channel dimension.



(b) Decoder

$$\hat{I} = \mathcal{G}(Z, e), \; e \in E$$

Steps followed in 3D Transform:

- They warp the grid such that the voxel at location p = (i, j, k)^T will be warped to pˆ, which is computed as

$$\hat{p} = Rp + t$$

- Warp is performed inversely and value at fractional grid location is trilinearly sampled.
- Due to coarseness of voxel representation misaligned voxel is refine using 2 3D Convolutions
- Reshape similar to Encoder phase.
- Output Image has the original resolution.

Losses

$$L_{\text{recon}}(\hat{I}_t, I_t) = \lambda_{L1}||\hat{I}_t - I_t||_1 + \lambda_{\text{perc}} L_{\text{perc}}(\hat{I}_t, I_t)$$

- L_{perc} is VGG-16 perceptual loss.
- To,enhance image quality WGAN-GP adversarial loss on each output frame.

$$\min_{G} \max_{\mathcal{F}_D \in \mathcal{D}} \mathop{\mathbb{E}}_{I \in \mathbb{P}_r} [\mathcal{F}_D(I)] - \mathop{\mathbb{E}}_{\hat{I} \in \mathbb{P}_g} [\mathcal{F}_D(\hat{I})]$$

Overall, our final loss is:

$$L = \sum_t L_{\text{recon}}(I_t, \hat{I}_t) + \lambda_{\text{GAN}} L_{\text{GAN}}(\hat{I}_t) + \lambda_{\text{cons}} L_{\text{cons}}(I_0, I_t)$$

In our experiments, we use $\lambda_{\text{cons}} = 1$, $\lambda_{\text{GAN}} = 0.01$.

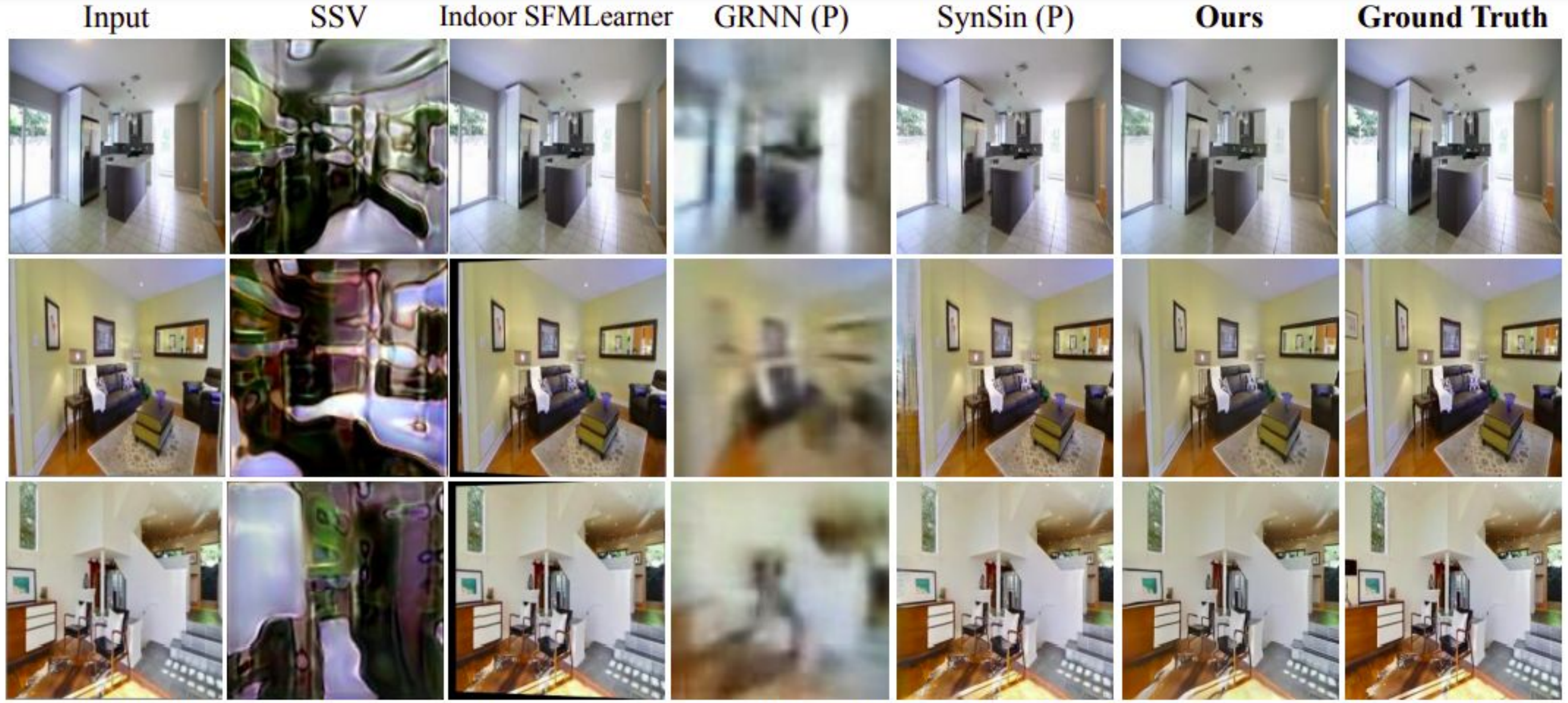


Figure 4: **Novel View Synthesis (our method vs. previous methods):** Other methods show systematic errors either in rendering or