

# Variational Style Transfer

Jan Schopohl, Dominik Fuchsgruber

Supervisor: Maxim Maximov

Department of Informatics, Technical University of Munich

December 3rd, 2019

# Artistic Style Transfer

Given a **content image  $C$**   
and an artistic **style image  $S$** :

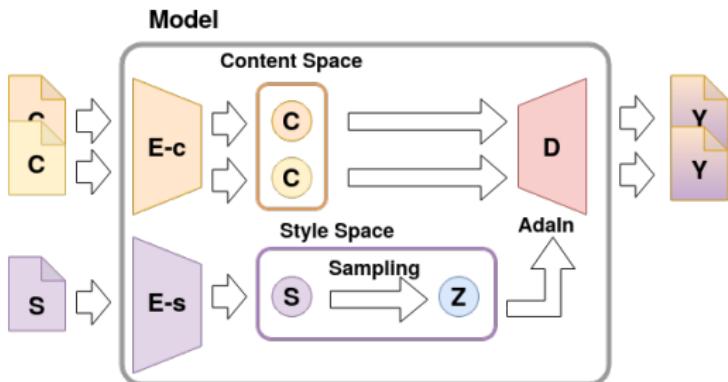
→ **Stylization:** Image with  
content similar to  $C$  and a  
style similar to  $S$

- How to combine  
**different styles?**
- Generating **new styles?**



- [1]: Optimize random image to fit  $C$  content-wise and  $S$  style-wise → **very slow!**
- [2]: **Autoencoders** to encode  $C$  and  $S$ , use Adaptive Instance Normalization (**AdaIn**) to transfer the style
- [5]: Separate Encoders for **content** and **style**  
→ **Disentanglement**
- [4]: **Variational Autoencoders** encode into a **smooth latent space**  
→ Good for **interpolation between encodings**
- [3]: **Perceptual losses** for style transfer

# Our Approach - Pipeline

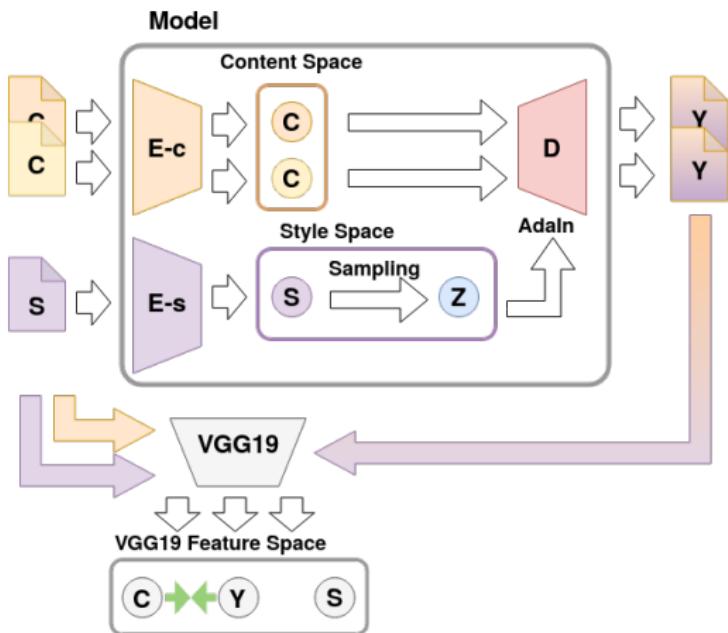


- **Content Encoder E-c and Style Encoder E-s**
- **Adaln** for style transfer

**Key contribution:**

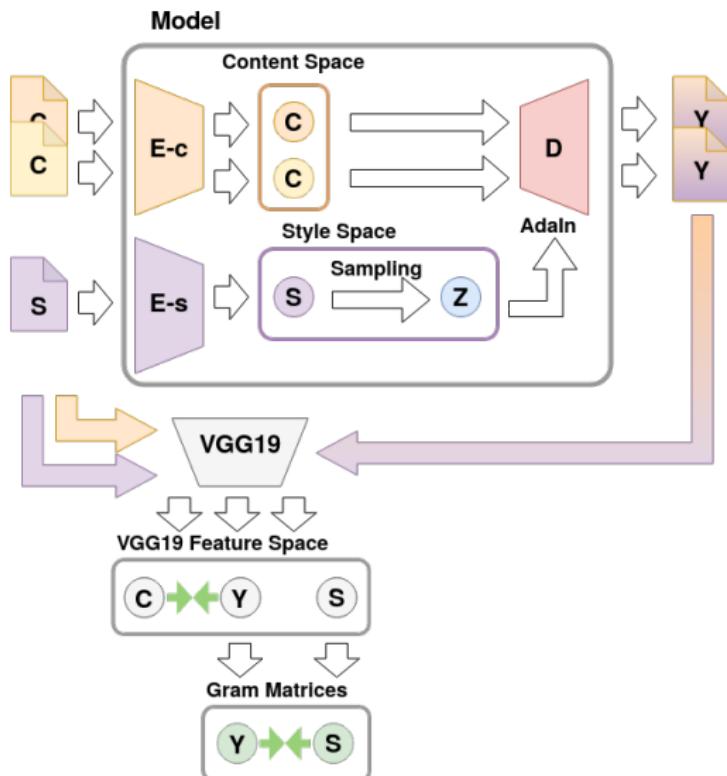
- **Smooth latent style space** due to **random sampling** (Variational Autoencoders)  
→ **Interpolation** and **style generation**

# Our Approach - Content Loss



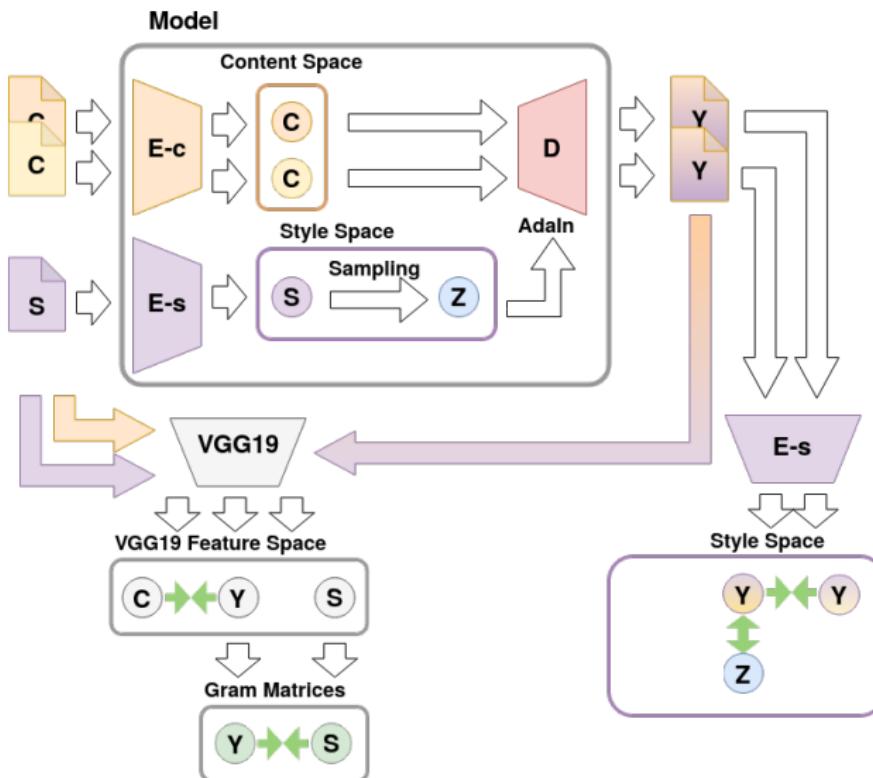
- **Content Loss:** Perceptual loss using pretrained **VGG19**

# Our Approach - Style Loss



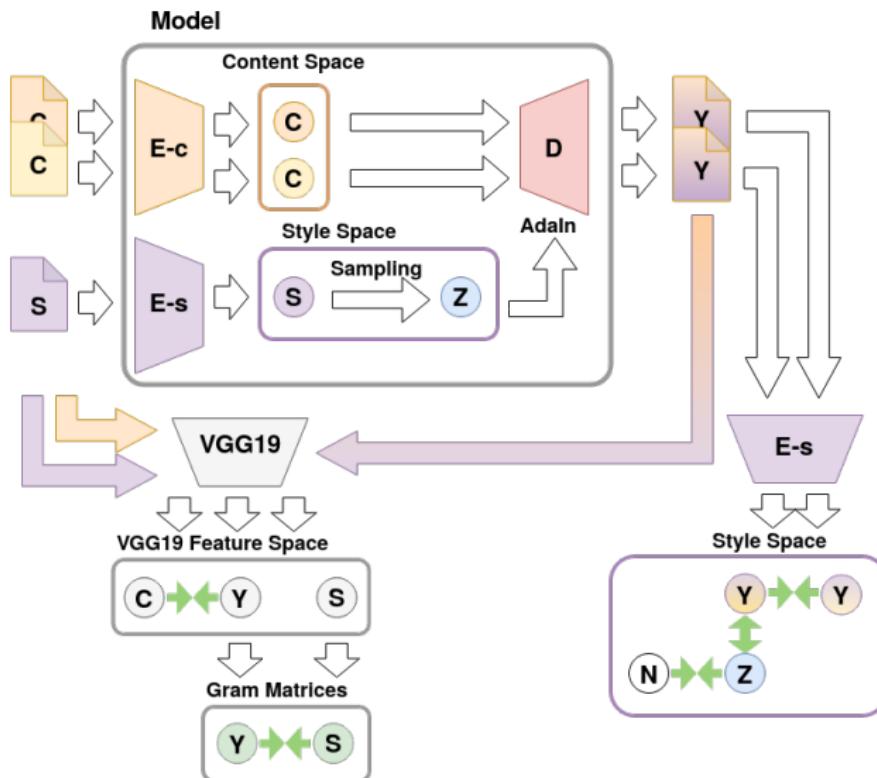
- **Style Loss:** Gram matrices of feature activations

# Our Approach - Disentanglement Loss



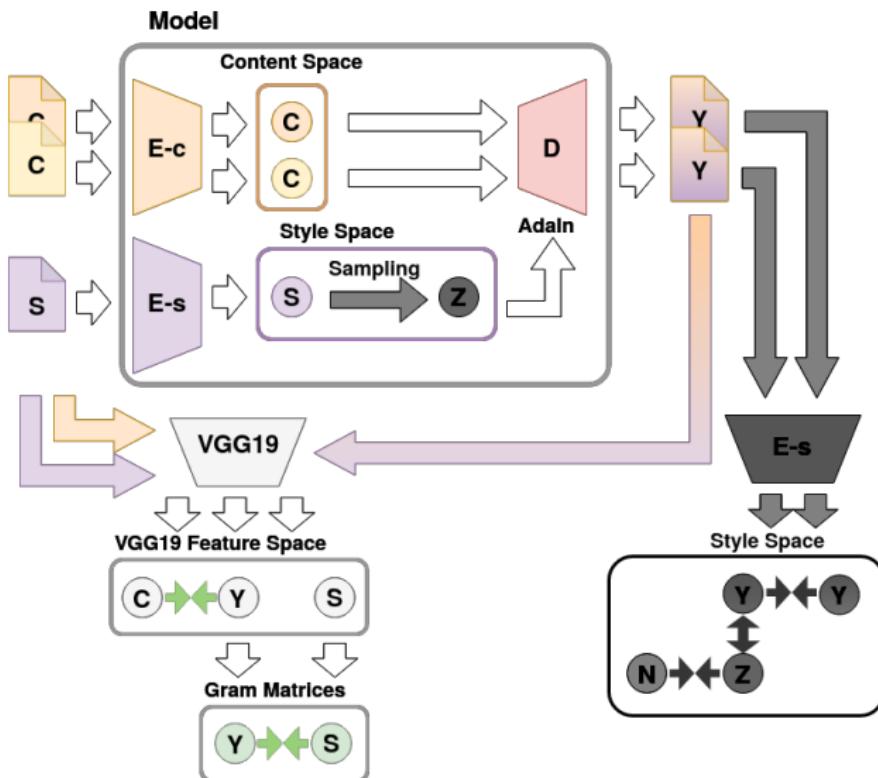
- **Disentanglement Loss:** Disentangles content and style

# Our Approach - Regularization Loss



- **Regularization:** Regularizes style distribution

# Our Approach - Implementation Progress



Gray components are not yet implemented.

Available implementations:

- **Loss network**: `torchvision`'s<sup>1</sup> pretrained VGG19
- **Content** and **style** losses and **AdaIn**: Unofficial implementation<sup>2</sup> of [2]
- **Perceptual** and **style** losses pytorch implementation<sup>3</sup> of [1]

Unfortunately, for [5], **no implementation is available** so far.

---

<sup>1</sup><https://pytorch.org/docs/stable/torchvision/models.html>

<sup>2</sup><https://github.com/naoto0804/pytorch-AdaIN>

<sup>3</sup><https://github.com/leongatys/PytorchNeuralStyleTransfer>

**Content images:** Places365<sup>4</sup> 1.839.960 different sceneries



**Style images:** WikiArt<sup>5</sup> contains 81.444 artistic images



Lack of large GPU resources: → **downsample** images to a **256x256** resolution and only **shallow** networks.

<sup>4</sup><http://places2.csail.mit.edu/download.html>

<sup>5</sup><https://github.com/cs-chan/ArtGAN>

To our best knowledge there is **no data-driven metric** to assess the quality of stylizations.

Common subjective assessment methods:

- **Preference Rate**: Probands **select most appealing** result among different stylizations
- **Deception Rate**: Probands try to **identify a real artistic image** mixed between stylizations

We focus on **interpolations between multiple styles!**

We would be glad if course participants and / or chair members could participate!

# First Results

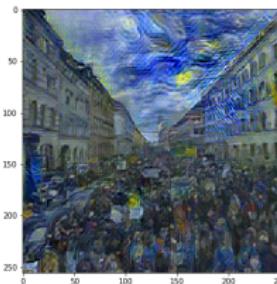
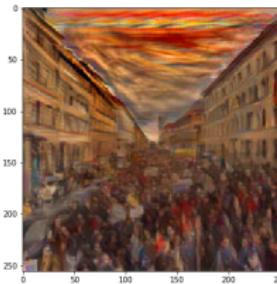
**Content Image**



**Style Image**



**Stylization**



What we want to achieve until the next presentation:

- Implement the **disentanglement loss**
- Implement **sampling** from a latent style distribution
- Train model on **entire dataset**
- Fine-tune **loss weights** → model for evaluation

After that:

- Compare to **baseline results**
- Set up a **survey** and **participants** for our evaluation

Possible baselines for evaluation on **preference rate** include:

- The original style transfer paper [1]
- The **AdaIn** paper [2]: Same decoder for **content** and **style**
- Learning a **linear transformation** on an image embedding for style transfer [6]

Possibly problematic may be **model sizes** and **computation times** due to our limited resources.

# References I

-  Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. “A Neural Algorithm of Artistic Style”. In: *CoRR* abs/1508.06576 (2015). arXiv: 1508.06576. URL: <http://arxiv.org/abs/1508.06576>.
-  Xun Huang and Serge J. Belongie. “Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization”. In: *CoRR* abs/1703.06868 (2017). arXiv: 1703.06868. URL: <http://arxiv.org/abs/1703.06868>.
-  Justin Johnson, Alexandre Alahi, and Fei-Fei Li. “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”. In: *CoRR* abs/1603.08155 (2016). arXiv: 1603.08155. URL: <http://arxiv.org/abs/1603.08155>.
-  Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2013. arXiv: 1312.6114 [stat.ML].

-  Dmytro Kotovenko et al. “Content and Style Disentanglement for Artistic Style Transfer”. In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.
-  Xuetong Li et al. “Learning Linear Transformations for Fast Arbitrary Style Transfer”. In: *CoRR abs/1808.04537* (2018). arXiv: 1808.04537. URL:  
<http://arxiv.org/abs/1808.04537>.

Instead of a pixel-wise loss between input and output:

→ **error between feature activations** of layer(s)  $i$  provided by a pre-trained model  $\Phi$

$$\mathcal{L}_\Phi(y, \hat{y}) = \frac{1}{C_i \times H_i \times W_i} \sum_i \|f_\Phi^i(y) - f_\Phi^i(\hat{y})\|_2^2$$

Where  $f^i$  are the **activation maps** of layer  $i$  with a dimensionality of  $C_i \times H_i \times W_i$ .

Penalizes **semantic discrepancies** between  $y$  and  $\hat{y}$  (e.g. objects, composition, etc.)

**Gram matrices**  $G_i$  of feature activation maps  $f_i$  capture **correlations between channels**:

$$G_i(y) = \tilde{f}_\Phi^i(y) \tilde{f}_\Phi^i(y)^T$$

Where  $\tilde{f}_\Phi^i$  are **flattened feature activation maps** of shape  $C_i \times (H_i W_i)$ .

$G_i(y)(c_1, c_2)$  captures how **strongly correlated** the features  $c_1$  and  $c_2$  are in the activation map  $f_i$ .

The **style loss** is given as the **error between Gram matrices of feature maps**:

$$\mathcal{L}_\Phi(y, \hat{y}) = \sum_i \frac{1}{C_i^2} \|G_i(y) - G_i(\hat{y})\|_2^2$$

# Disentanglement Loss

Try to **disentangle content and style**:

→ Minimize influence of content on style representation [5]

Let  $y_1$  and  $y_2$  denote stylizations of content images with style encoding  $s$

$$\mathcal{L}(y_1, y_2, s) = \max(0, \|y_1 - y_2\|_2^2 - \|y_1 - s\|_2^2) \quad (1)$$

Enforces two stylizations of **same style** to be **closer in style space** than a **stylization and the style source image**.

# Adaptive Instance Normalization

**Instance Normalization** is related to **Batch Normalization** but calculates **instance-wise** mean  $\mu$  and variance  $\sigma^2$ , instead of using batch-wise statistics.

$\mu$  and  $\sigma$  **encapsulate style information** [2]: → **Adaptive Instance Normalization:**

$$\text{AdaIn}(x, y) = \hat{\sigma} \frac{x - \mu(x)}{\sigma(x)} + \hat{\mu}$$

$\hat{\mu}$  and  $\hat{\sigma}$  are derived from **style encoding**

# Variational Autoencoders I

**Autoencoders** encode an input  $X$  to a latent space representation  $z$  and try to reconstruct  $X$  using only  $z$ .

- **neglects variations in input distribution** (e.g. noise)
- Latent space **may not be smooth**

**Variational Autoencoders** sample  $s$  from a **distribution parametrized by  $z$**  (usually Gaussian) instead

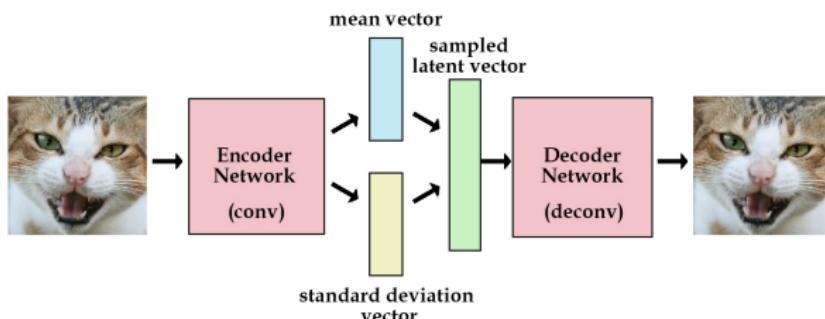


Figure: taken from

<http://kvfrans.com/variational-autoencoders-explained/>

To enforce a **smooth latent space** the learned distribution is **regularized** using the KL-divergence:

$$\mathcal{L} = \text{KL}(q_{\text{enc}}(s|X) \| p(z))$$

Where  $q_{\text{enc}}(s|X)$  describes the distribution of latent representations  $s$  given an input  $X$  and  $p(z)$  is set to be a standard normal distribution.

→ forces **latent distribution** to be **close to a standard normal distribution**.