Style transfer Literature review

M.Generali, A.Reyero Lobo, N.Benali

Tuesday 14th March, 2023

Introduction

Why use Feature-wise transformation ?

Figure: My little brother during summer and Monet's painting "Paysage près de Montecarlo"

Introduction

Why use Feature-wise transformation ?

Figure: Feature-wise transformation (style transfer)

- Different conditioning approachs
- FiLM layer: the Feature-wise Linear Modulation

1. Loss Function
   - Content Representation
   - Style Representation
   - Optimization Problem
   - Optimization Problem
2. Models
   - Conditional Instance Normalization
   - N-Style network
   - Arbitrary Style Transfer
3. Experiments
   - Interpolation weights
   - β and γ parameters
   - Embedding space
   - Convex combination
4. Conclusion

Feature-wise transformation

Of the utility of contextualisation

Figure: Cross-reference of similarities

Feature-wise transformation

A first idea of conditioning

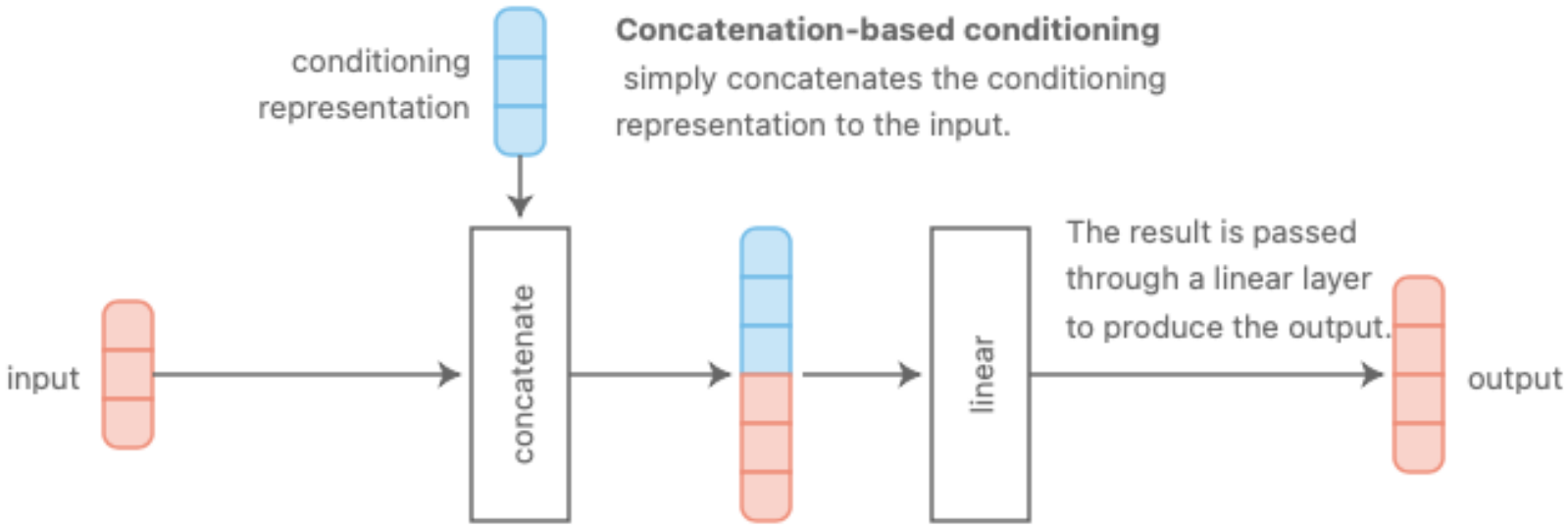


Figure: Concatenation-based conditioning

Feature-wise transformation

Another kind of concatenation

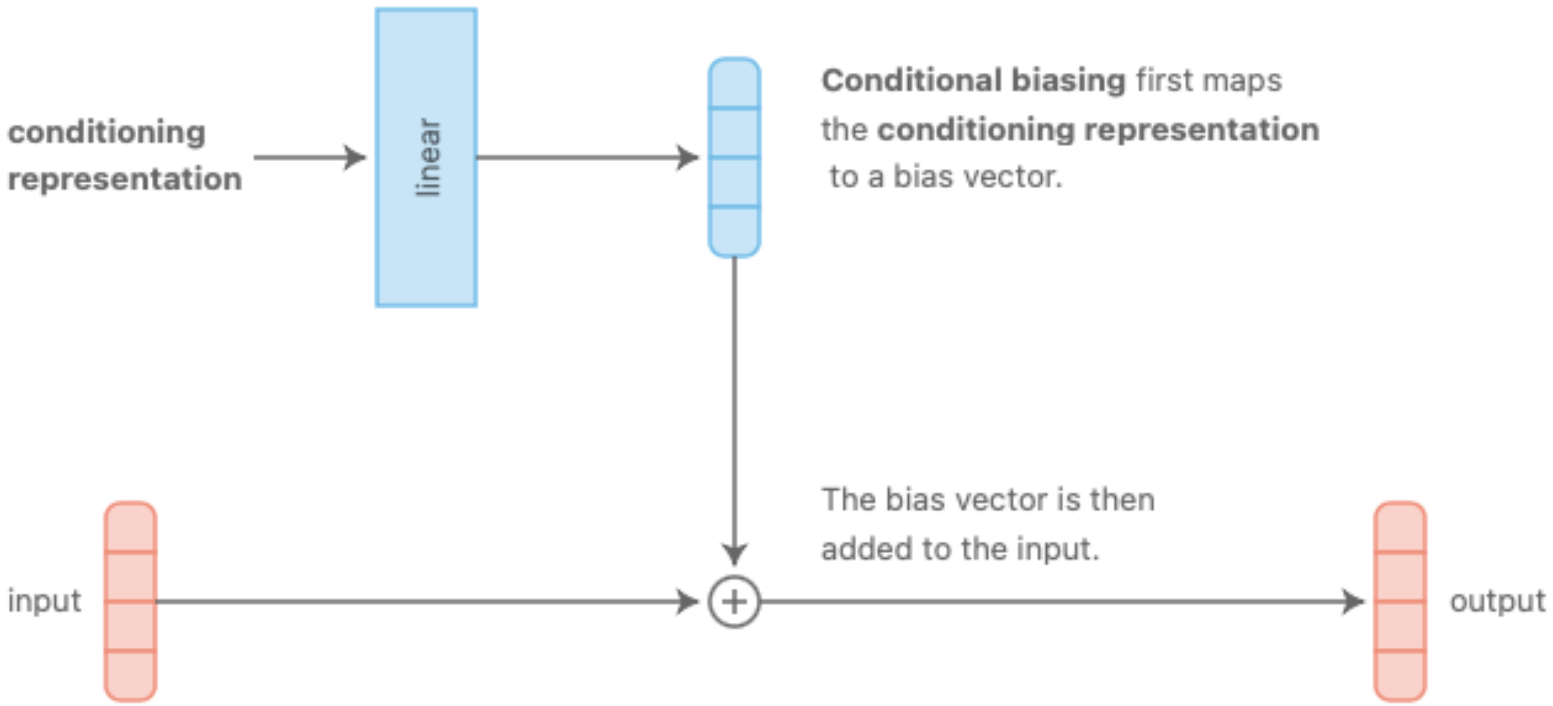


Figure: Conditional biasing: adding a bias to the hidden layers based on the conditioning representation

Feature-wise transformation

A second idea of conditioning

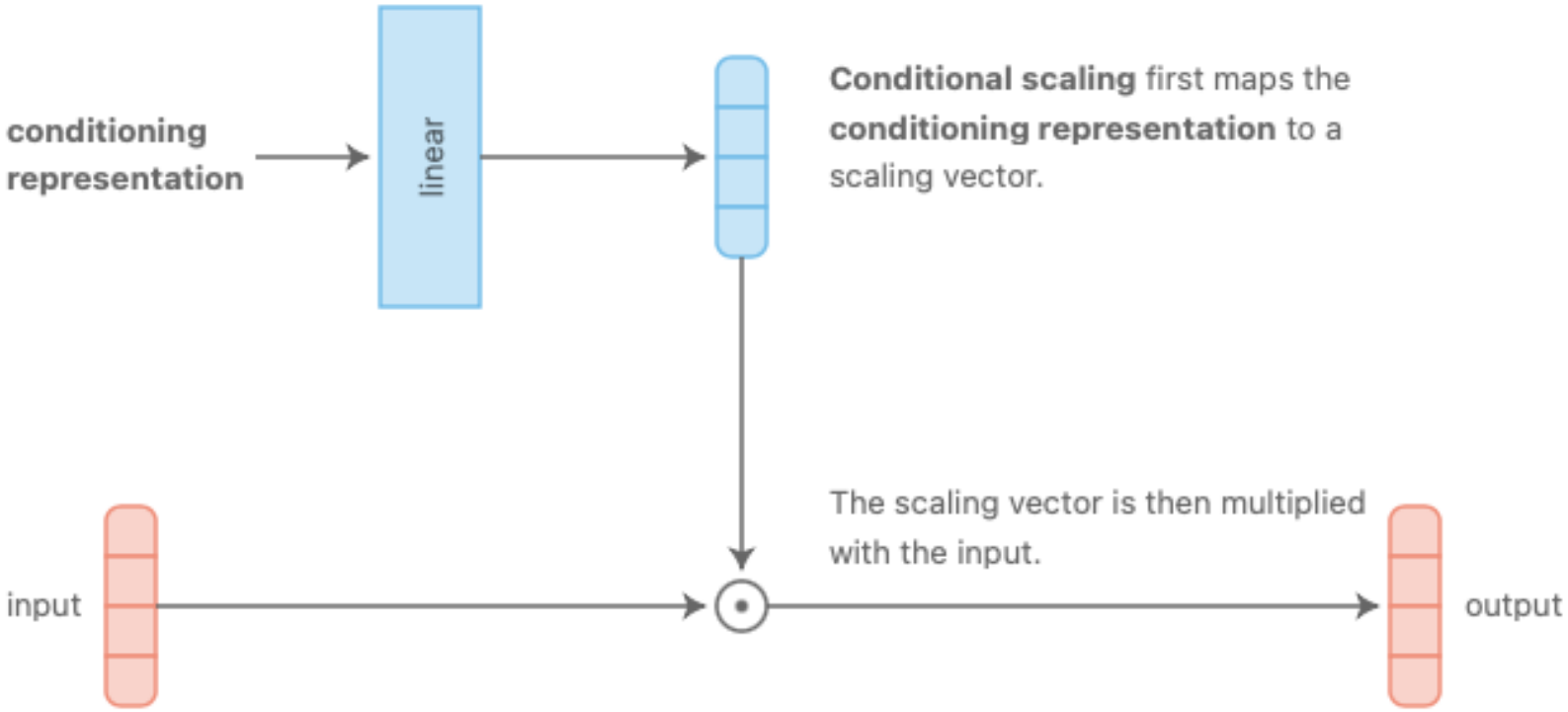


Figure: Conditional scaling : scaling hidden layers based on the conditioning representation.

FiLM layer

Perez and al nomenclature

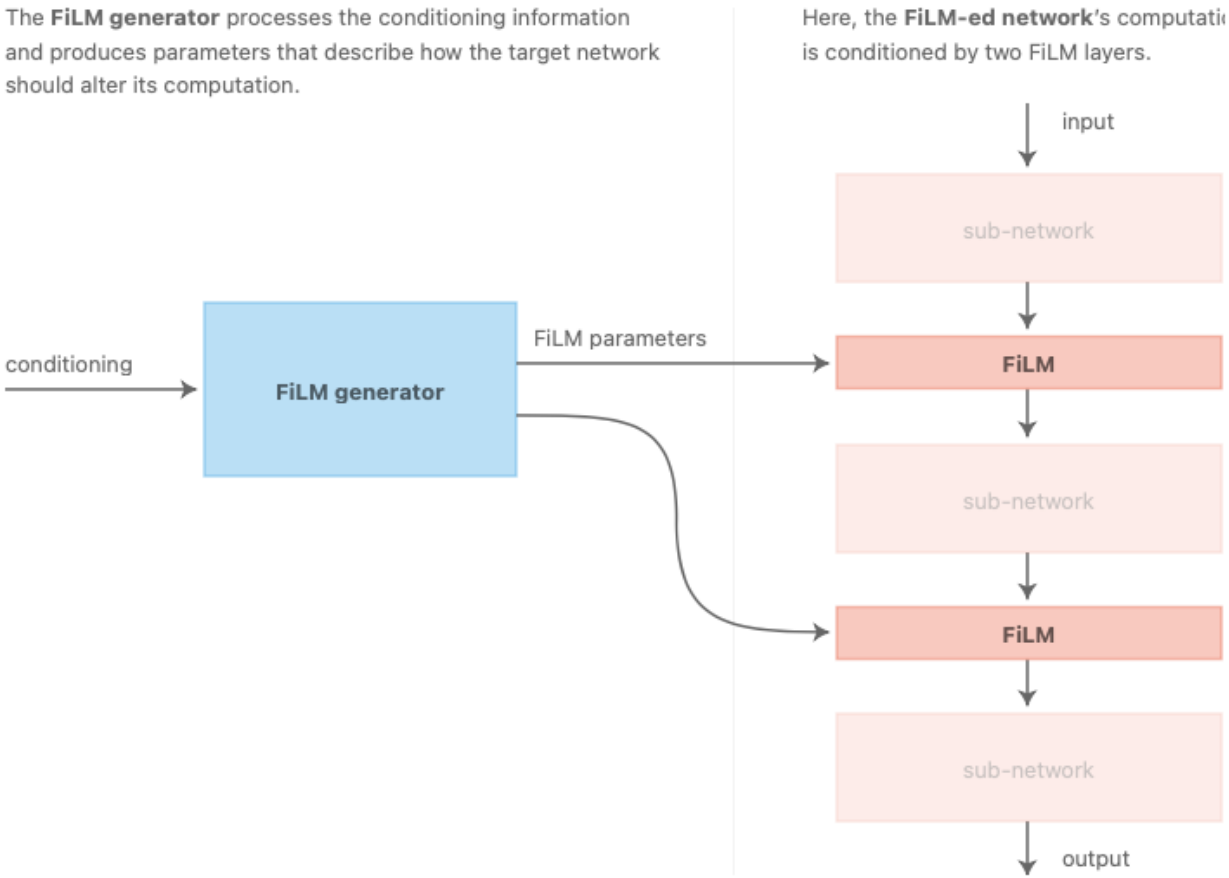


Figure: Conditional scaling : scaling hidden layers based on the conditioning representation.

FiLM layer

Perez and al nomenclature

$$\mathrm{FiLM}(\mathbf{x}) = \gamma(\mathbf{z}) \odot \mathbf{x} + \beta(\mathbf{z})$$

Figure: FiLM generator functioning

Deep Convolutional Neural Networks

VGG-16 Architecture

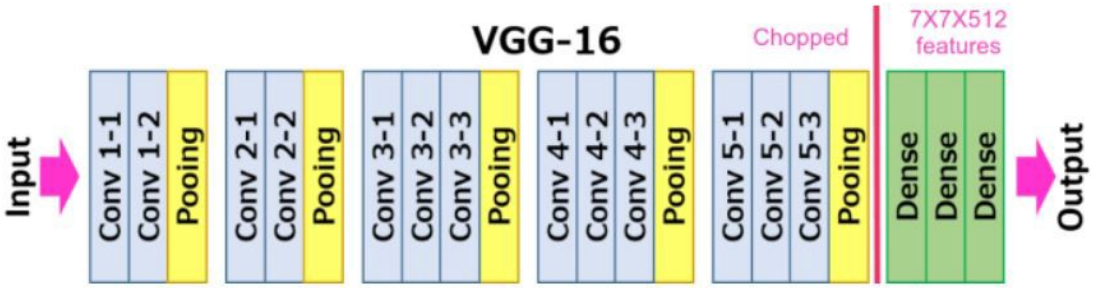


Figure: Architecture of the image recognition VGG-16 Deep Convolutional Neural Network.

Deep Convolutional Neural Networks

High-level features

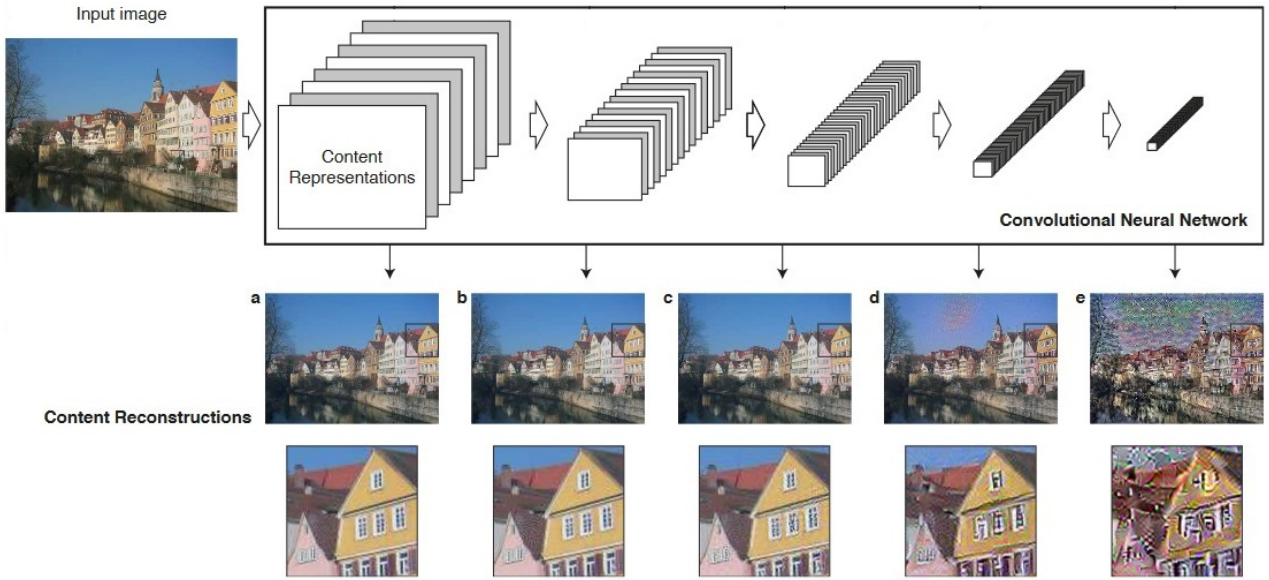


Figure: Image reconstruction through the VGG layers. We observe how we can reconstruct content images from the layers (particularly, 'conv1_2' (a), 'conv2_2' (b), 'conv3_2' (c), 'conv4_2' (d) and 'conv5_2' (e) of the original VGG-Network). We observe that in the lower layers we conserve perfectly the content, and that at the higher layers pixel information is lost, but we can conserve higher-level image content.

Deep Convolutional Neural Networks

Low-level features

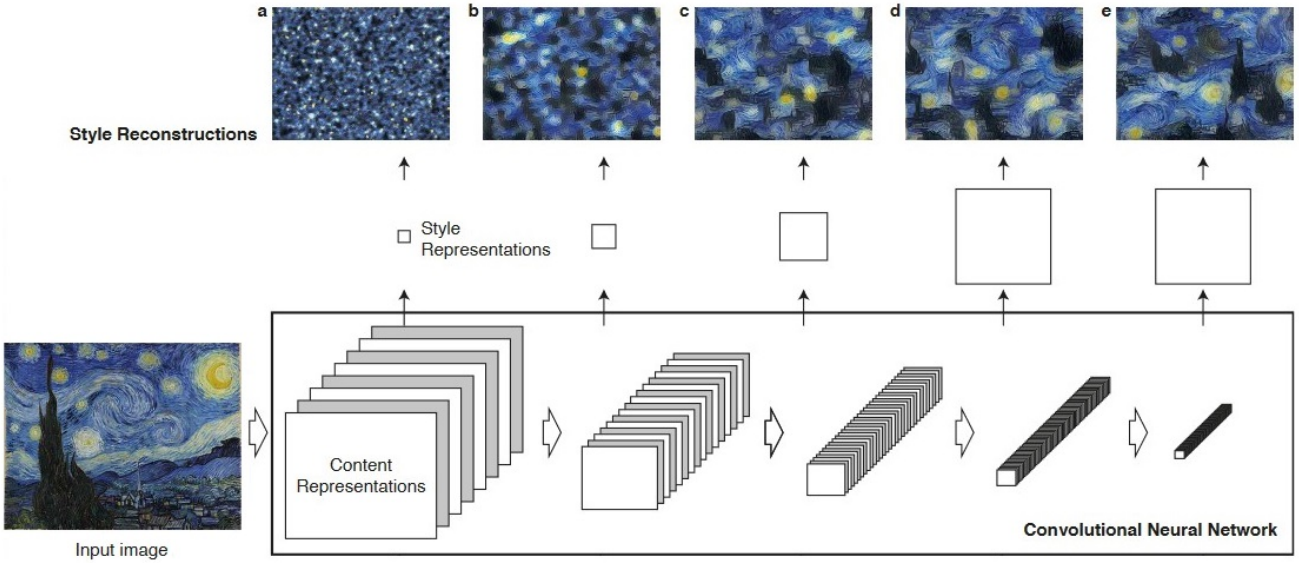


Figure: Image reconstruction through the VGG layers.

We are reconstructing the style image. To reconstruct this style we compute correlation between different feature maps and different layer sets. (Particularly, only 'conv1_1' to construct the (a), 'conv1_1' and 'conv2_1' for the (b), 'conv1_1', 'conv2_1' and 'conv3_1' for the (c), 'conv1_1', 'conv2_1', 'conv3_1' and 'conv4_1' for the (d), 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1' for the (e).

Content Loss $\mathscr{L}_{content}$

Let l be a layer with N_l feature maps, each one of size $M_l(height \times width)$. Let $F^l \in \mathbb{R}^{N_l \times M_l}$ be the output of the layer l, where F^l_{ij} is the activation of the i-th filter at position j in layer l.
Let p photograph, x the output, and P_l and F_l their respective feature representation in layer l. Then, we can compute the loss between p and x by

$$\mathscr{L}_{content}(p, x, l) = \frac{1}{2} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 \tag{1}$$

We will omit the l of the equation as we will just use one layer.

Content Loss $\mathscr{L}_{content}$

Figure: We have changed the content loss to be computed with the conv2_2 instead of the conv4_2, and we can observe that we are more pixel constrained.

Style Loss $\mathscr{L}_{style}$

Let $G^l \in \mathbb{R}^{N_l x N_l}$ (With N_l the number of filters in the layer) be the Gram matrix between multiple filters, where G^l_{ij} is the inner product between the vectorised feature maps i and j in layer l:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \tag{2}$$

Let a be the artwork and x the output. Then, let A_l and G_l the Gram Matrix computed for a and x at layer l. Then, we can define:

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{ij} - A^l_{ij})^2 \tag{3}$$

and the total style loss as:

$$\mathscr{L}_{style}(a, x) = \sum_{l=0}^{L} w_l E_l \tag{4}$$

where w_l are weighting factors of the contribution of each layer to the total style loss.

Total Loss $\mathscr{L}_{total}$

$$\mathscr{L}_{total} = \min_{x} \mathscr{L}_{content}(x, p) + \lambda_s \mathscr{L}_{style}(x, a) \tag{5}$$

with λ_s the weight that we give to the style loss.

Gatys Algorithm Backpropagation

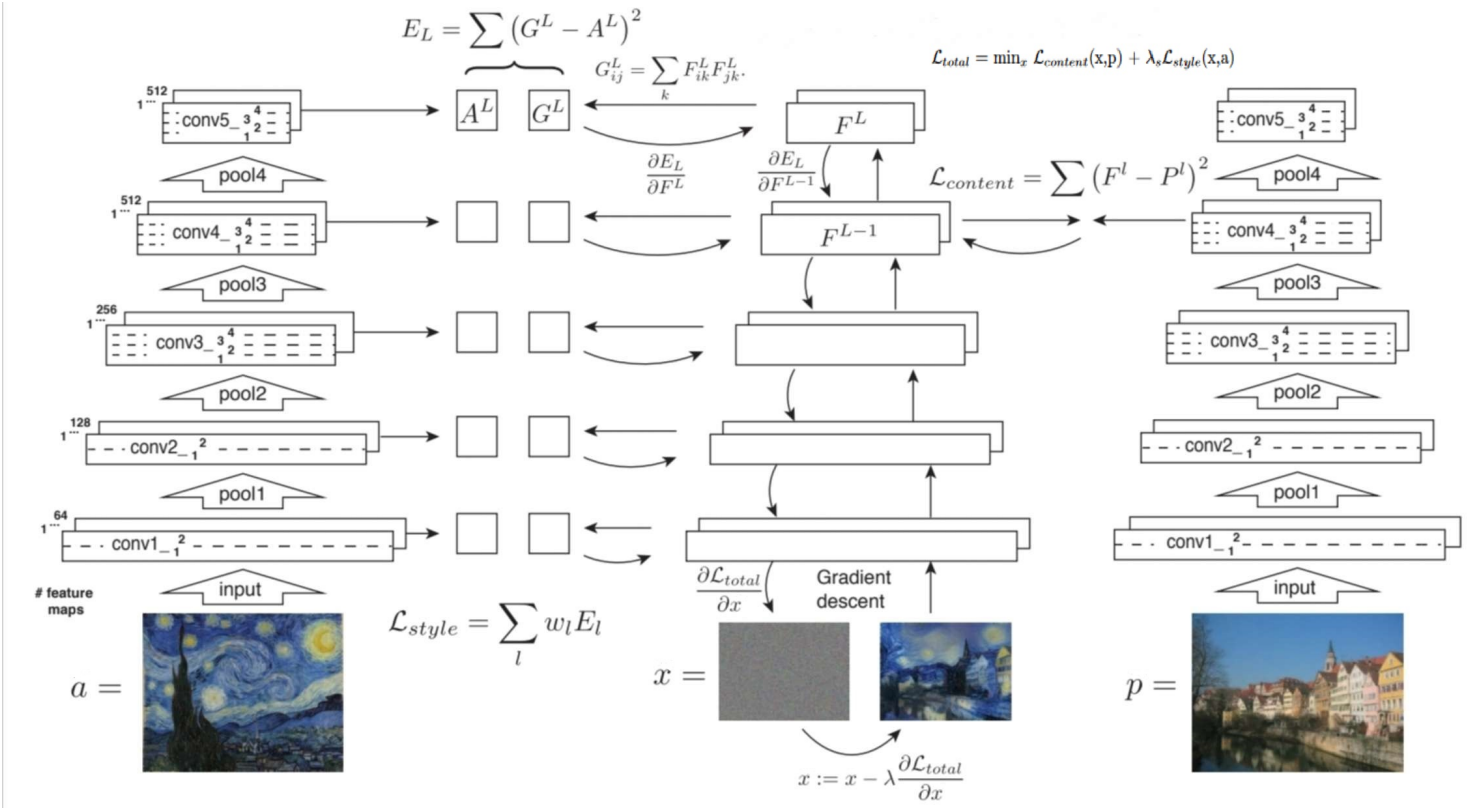
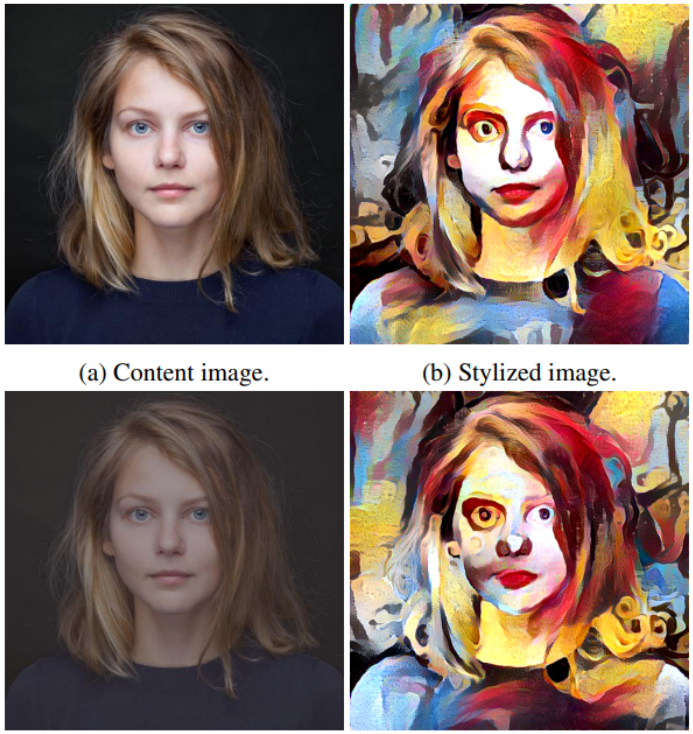


Figure: We pass a (left) and p (right) through the VGG network and we store the A_ls (left) and the content representation P_l (right). Then, a random white noise x is passed through the network to compute its content features F_l and style features G_l. We iteratively update the image x until it matches both the style features with a and the content features with p simultaneously.

Inconsistencies



(a) Content image. (b) Stylized image.

(c) Low contrast content image. (d) Stylized low contrast image.

Figure: Trained following the architecture from Gatys. We are able to see how the contrast of the content image alters the style of the pastiche when it should not.

Batch and Instance Normalization

Let x_{tijk} denote its $tijk$-th element, where k and j span spatial dimensions, i is the feature channel (color channel if the input is an RGB image), and t is the index of the image in the batch. Batch Normalization:

$$y_{tijk} = \frac{x_{tijk} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}, \quad \mu_i = \frac{1}{HWT} \sum_{t=1}^{T} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{tilm}, \quad \sigma_i^2 = \frac{1}{HWT} \sum_{t=1}^{T} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{tilm} - mu_i)^2.$$

Instance normalization:

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} x_{tilm}, \quad \sigma_{ti}^2 = \frac{1}{HW} \sum_{l=1}^{W} \sum_{m=1}^{H} (x_{tilm} - mu_{ti})^2.$$

Conditional Instance Normalization

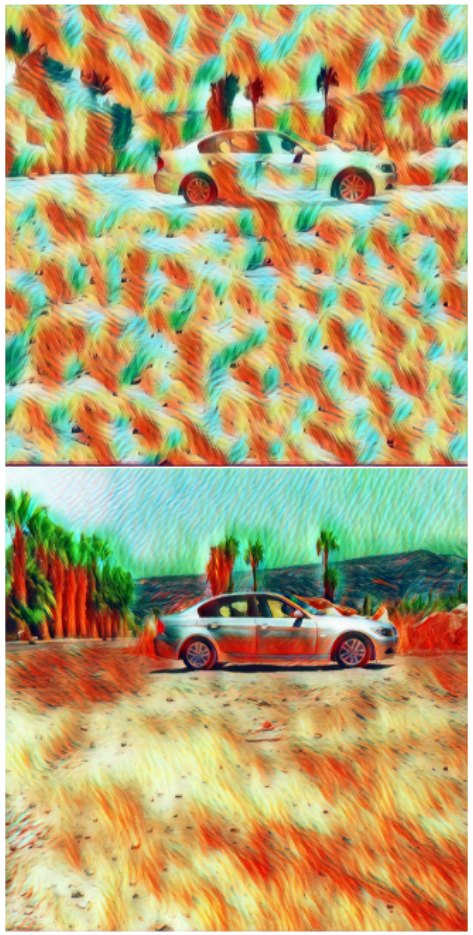


Figure: For the first image we use a conditional batch normalization, and for the second one a conditional instance normalization.

Conditional Instance Normalization

Let γ and β of dimensions $S \times C$ with S the number of styles and C the number of output maps. Then, for each style,

$$z = \gamma_s \left(\frac{x - \mu}{\sigma} \right) + \beta_s \tag{6}$$

N-styles network

N-Style Transfer model diagram

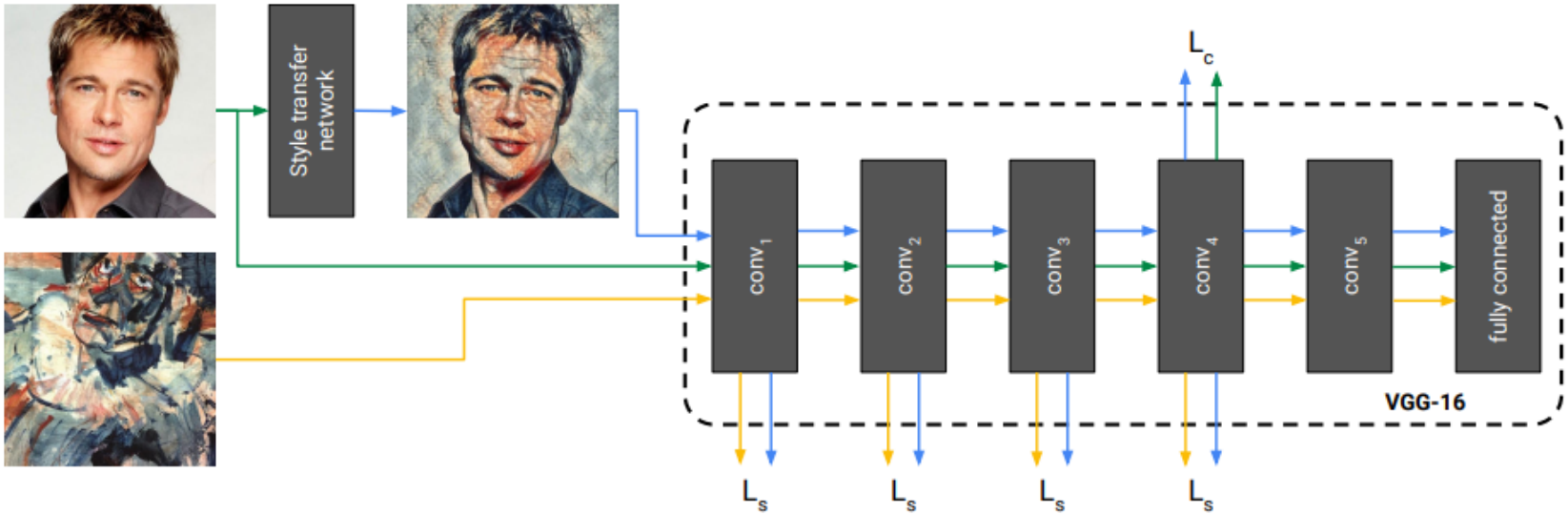


Figure: Style Transfer Network is able to do a simple style embedding. It computes the content loss and style loss in the same was as Gatys.

N-styles network

N-Style Transfer vs Single Purpose Style Transfer

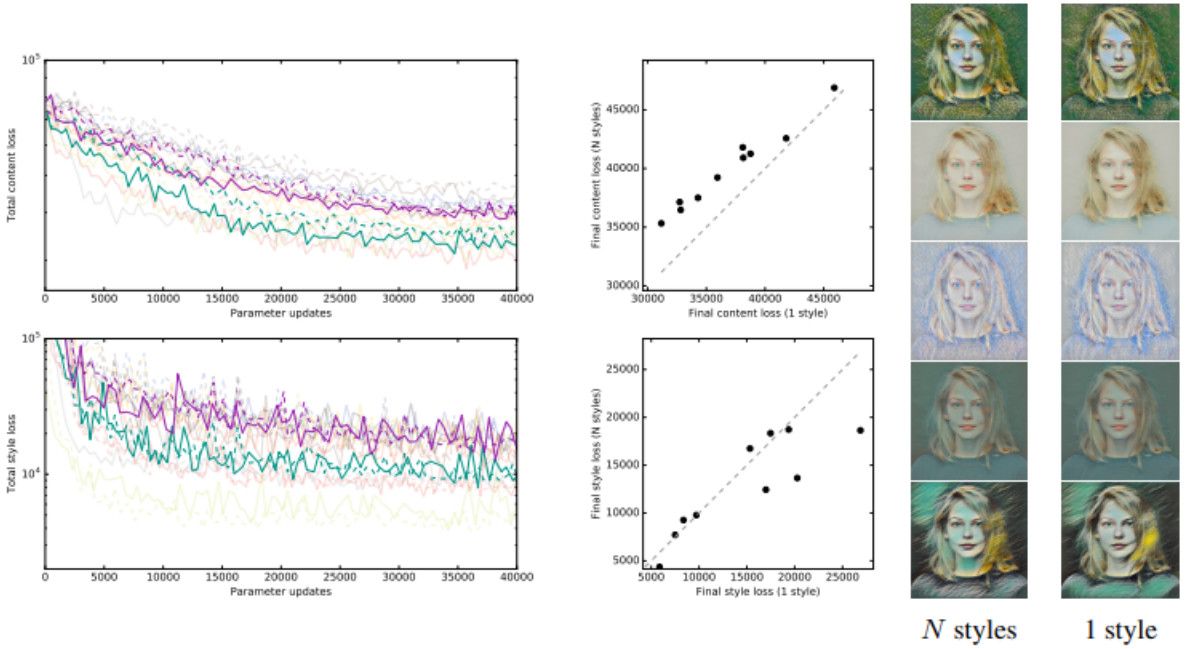


Figure: (Left)The dashed curve represents the N-styles model, and the full curves represent individual models. (Center) The N-styles model reaches a slightly higher final content loss than and a final style loss comparable to individual models.(Right)Pastiches produced by N-styles and 1-styles models.

Arbitrary Style Transfer

Corpus Size Importance

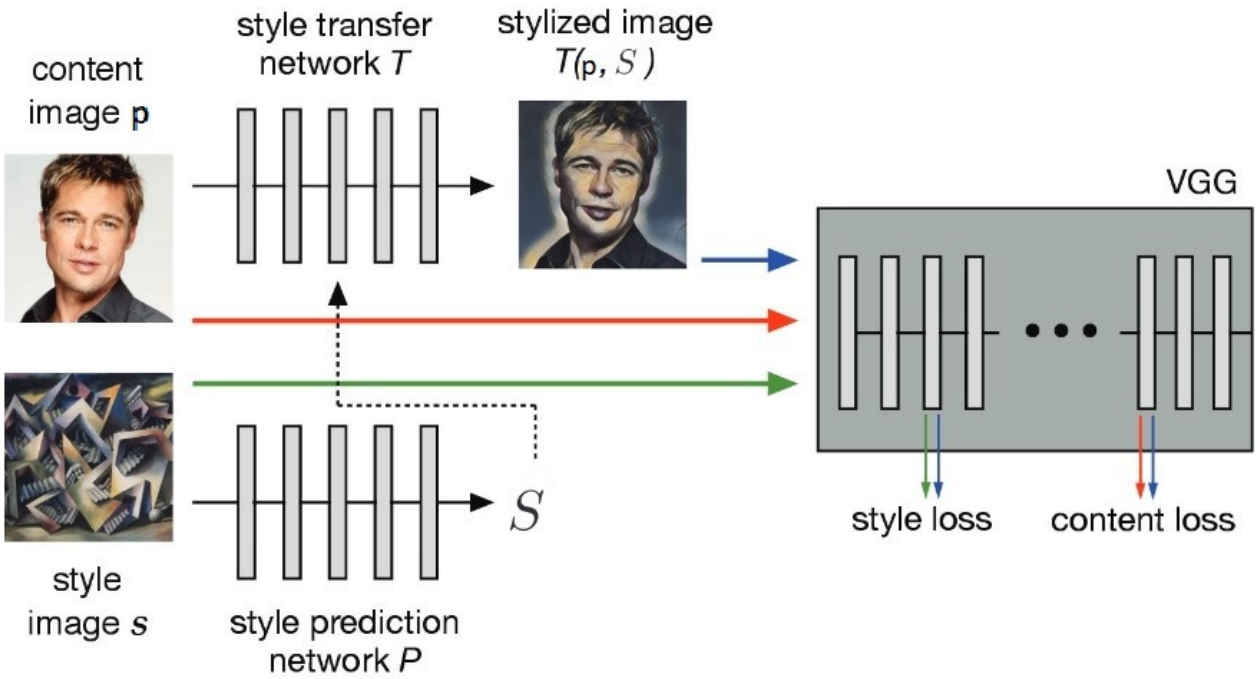
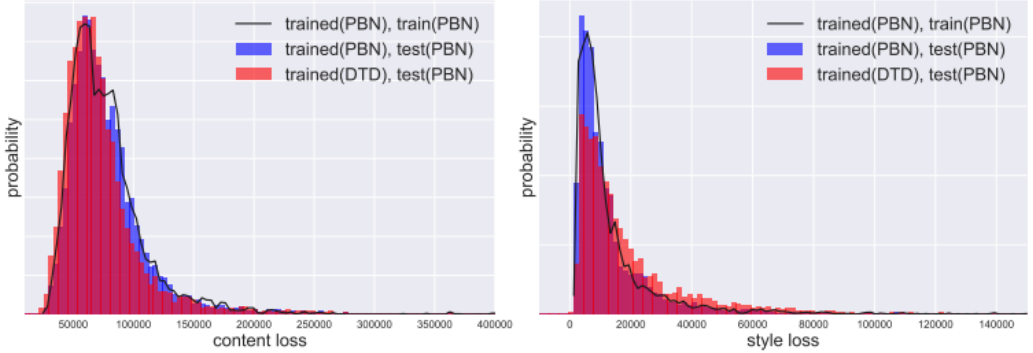


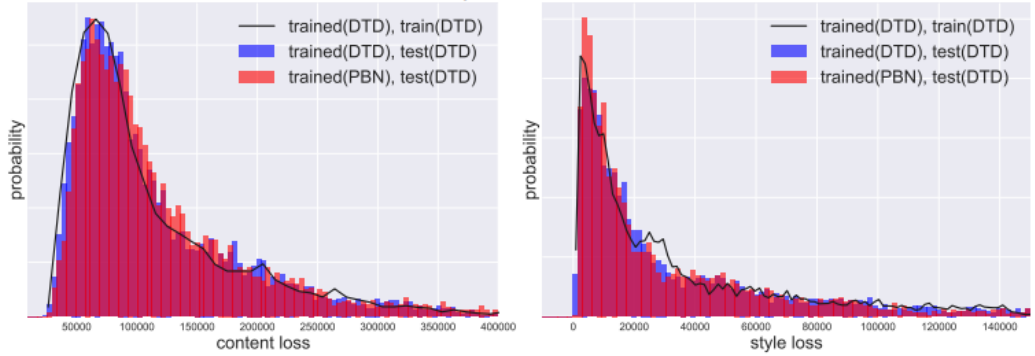
Figure: This model introduces a style prediction network to predict an embedding vector S=$\{\gamma_s, \beta_s\}$ from the input style image s. Then, the style transfer network transforms the photograph into a stylized representation. The content and style losses are derived from the distance in representational space of the VGG image classification network.

Arbitrary Style Transfer

Corpus Size Importance



(a) Content and style losses over PBN dataset



(b) Content and style losses over DTD dataset

Figure: These two models are trained with the same content images dataset (Imagenet), but with different style datasets. PBN has 79,433 paintings and DTD has 5,640.

Arbitrary Style Transfer

Corpus Size Importance

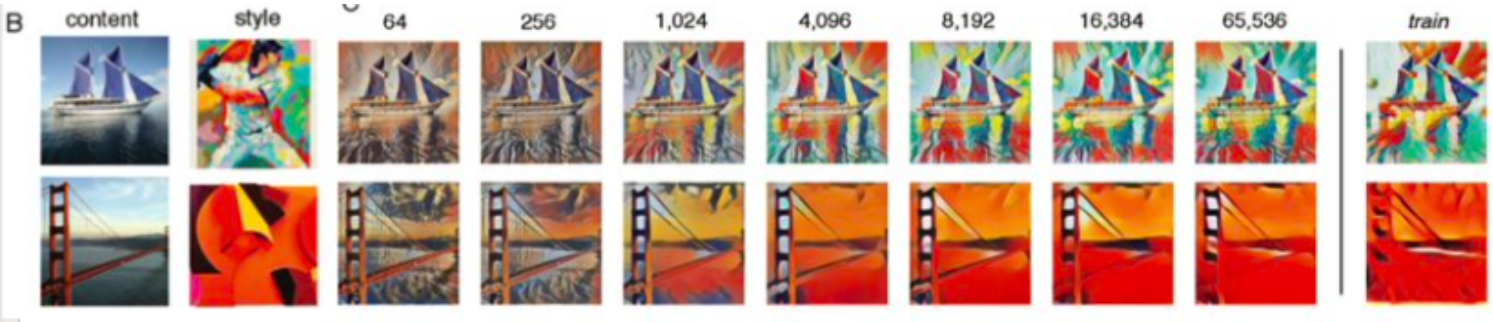


Figure: We can observe the importance if training with multiple images. The number over the images the number of instances used to train the models. When we use more images we are able to preserve more the style.

Experiments

Interpolation weights

$$\min_{x} \mathscr{L}_{content}(x,p) + \lambda_s \mathscr{L}_{style}(x,a) \tag{7}$$

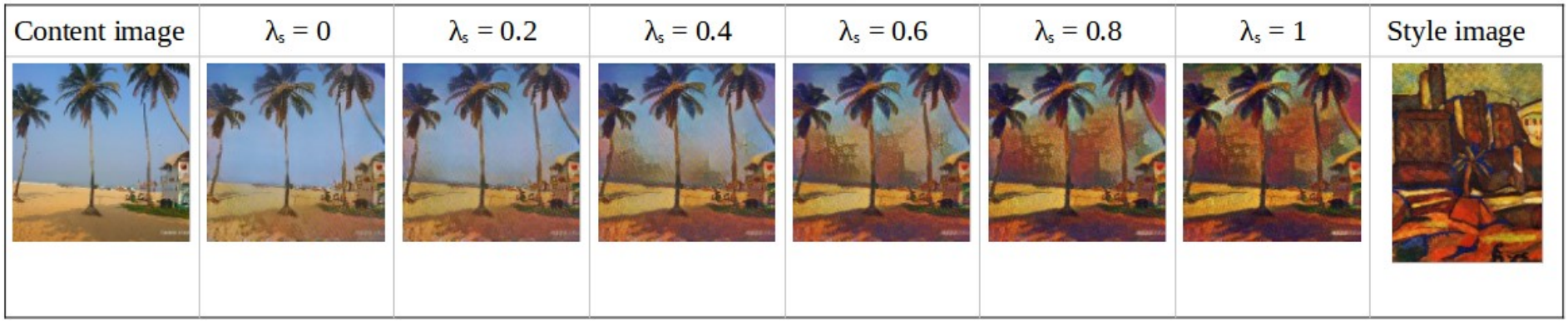
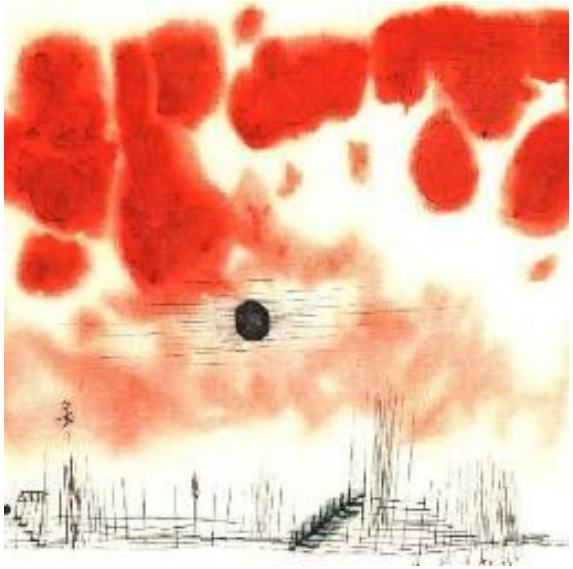


Figure: Representation of the transformation undergone by the content image for different interpolation weights such that $\lambda_s \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

Experiments

β and γ parameters



(a) Style image n°1: "Clouds over Bor, 1940"

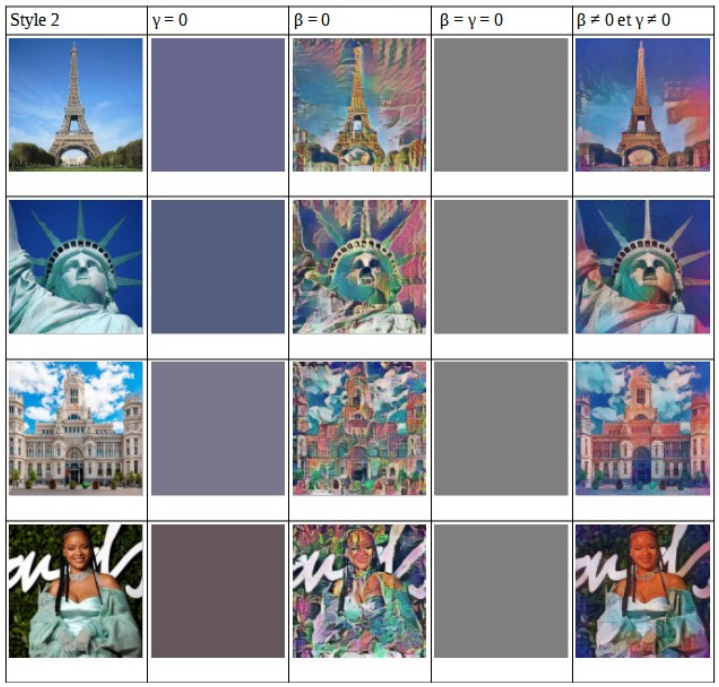
(b) Style image n°2: "Towers, 1916"

Figure: Style images used for testing β and γ parameters.

Experiment

β and γ parameters

(a) Representation of different values of β and γ on different content images and with style 1.



(b) Representation of different values of β and γ on different content images and with style 2.

Figure: The experiment was conducted on a set of five content images, with an interpolation weight of 0.5.

Experiments

Embedding space

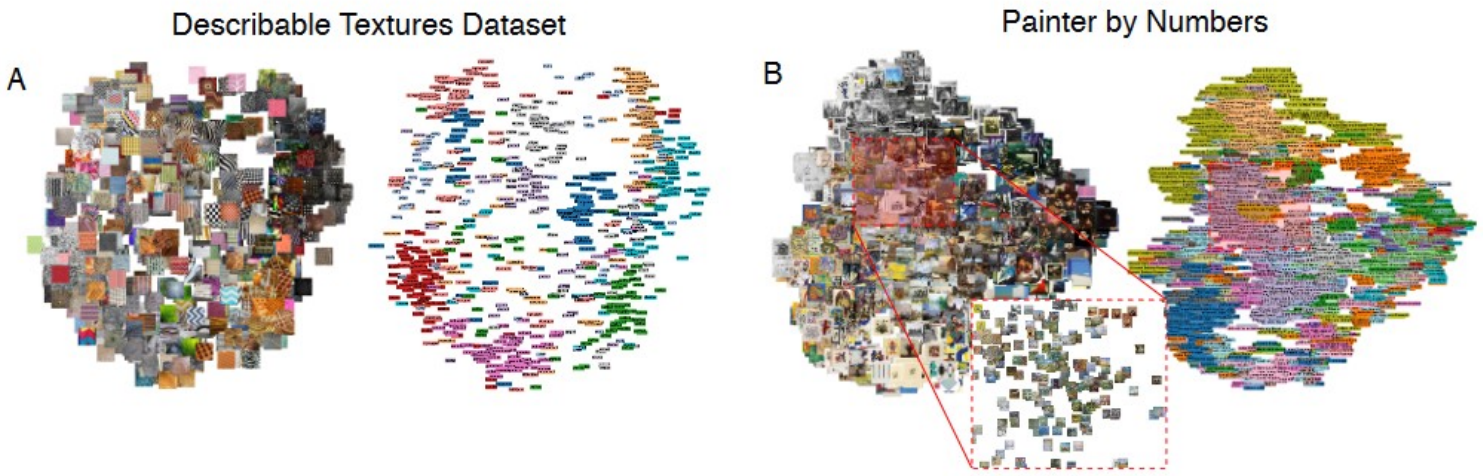


Figure: Low dimensional representation of the embedding space.

$$z = \gamma_s \left(\frac{x - \mu}{\sigma} \right) + \beta_s \tag{8}$$

Experiments

Embedding space

Figure: Representation of how the model generalizes with style images it was not trained on.

Experiments

Convex combination

Given a convex combination of $S = \{\gamma_s, \beta_s\}$, it is possible to generate an image with blended style.

$$\gamma = \alpha\gamma_1 + (1-\alpha)\gamma_2, \quad \beta = \alpha\beta_1 + (1-\alpha)\beta_2 \tag{9}$$

Experiments

Convex combination

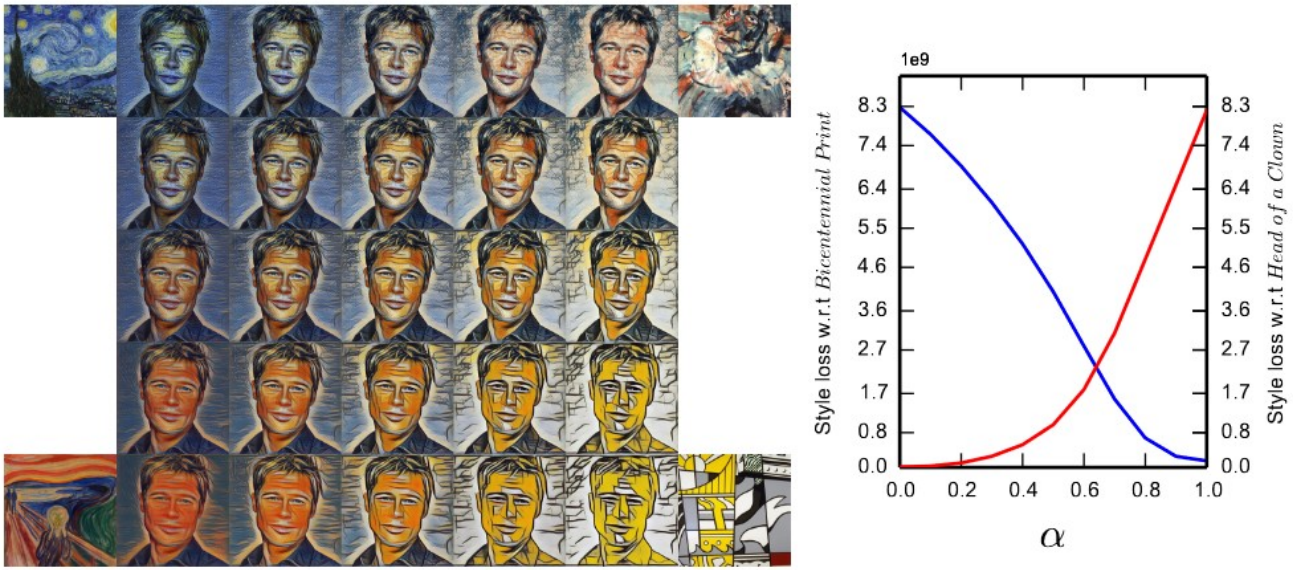


Figure: Left : Pastiches generated combining four different styles (in the corners). Each stylized image corresponds to a different convex combination of the four styles' γ and β values. Right: The style loss as a function of α

Conclusion

- Despite the model's excellent performance, there are still some limitations, such as low-level noise that can affect photorealism in photographs.
- We aim to separate image content from style, although this is not a clearly defined problem.
- The ability to abstract content from style may be a fundamental feature of our visual system's powerful inference capabilities.