HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

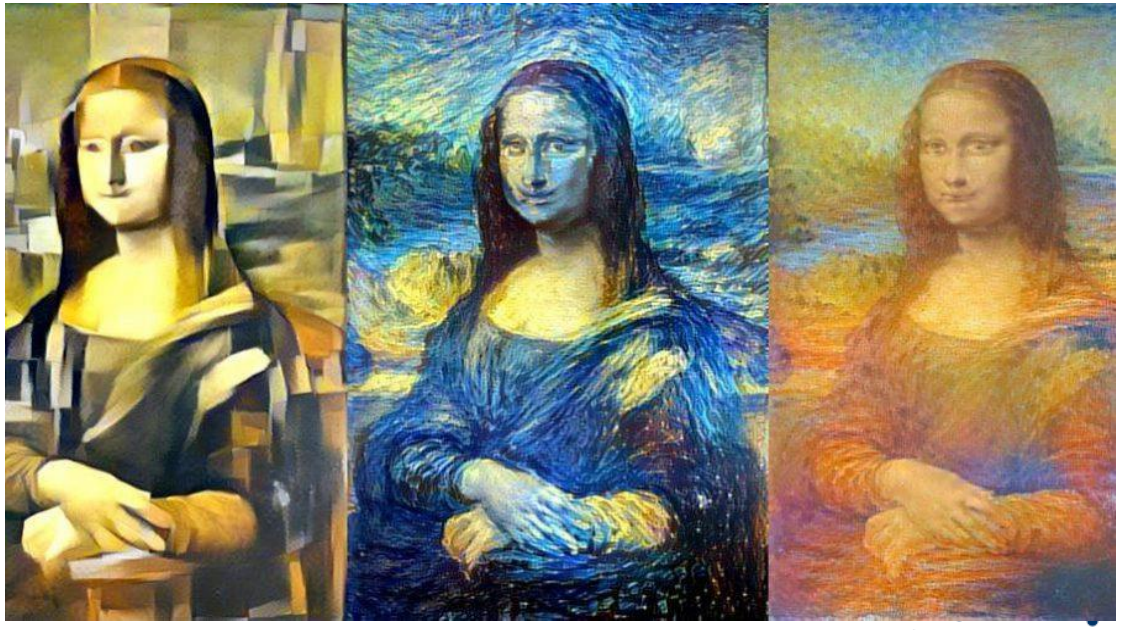
ONE LOVE. ONE FUTURE.

HUST

hust.edu.vn fb.com/dhbkhn

1. Introduction

Image style transfer is a technique in computer vision that allows us to recompose the content of an image in the style of another.



Convolutional Neural Network

Our objective:

Control the **smoothness** of the output image.
Able to transfer an input image to several styles **in a short time**.

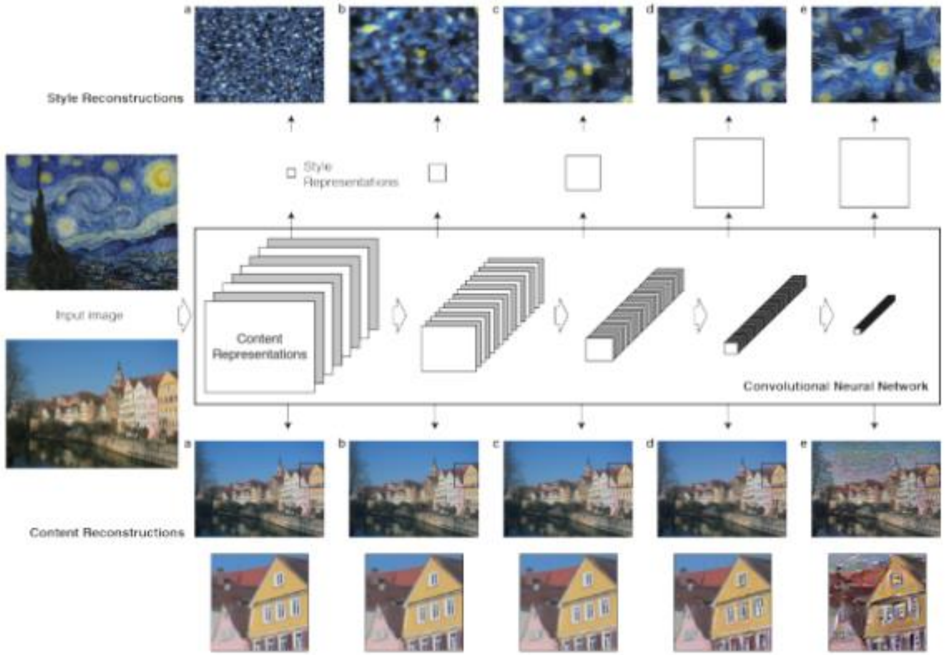
HUST

hust.edu.vn fb.com/dhbkhn

2. A Neural Algorithm of Artistic Style

Convolutional neural networks (CNNs) are a type of deep learning algorithm that are commonly used for image recognition tasks. CNNs are able to learn features from images in a hierarchical manner, starting with simple features at the early layers and progressing to more complex features at the later layers.

The idea of the method is to use a pretrained CNN to extract content representation of the content image and style representation of the style image.



We define the squared-error loss between the two feature representations

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F^l_{ij} - P^l_{ij}\right)^2 \ .$$

The derivative of this loss with respect to the activations in layer l equals

$$\frac{\partial \mathcal{L}_{content}}{\partial F^l_{ij}} = \begin{cases} \left(F^l - P^l\right)_{ij} & \text{if } F^l_{ij} > 0 \\ 0 & \text{if } F^l_{ij} < 0 \, . \end{cases}$$

On top of the CNN responses in each layer of the network, we built a style representation that computes the correlations between the different filter responses, where the expectation is taken over the spatial extend of the input image. These feature correlations are given by the Gram matrix $G^l \in R^{N_l \cdot M_l}$, where G^l_{ij} is the inner product between the vectorised feature map i and j in layer l:

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

The contribution of that layer to the total loss is then

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

And the total loss is

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

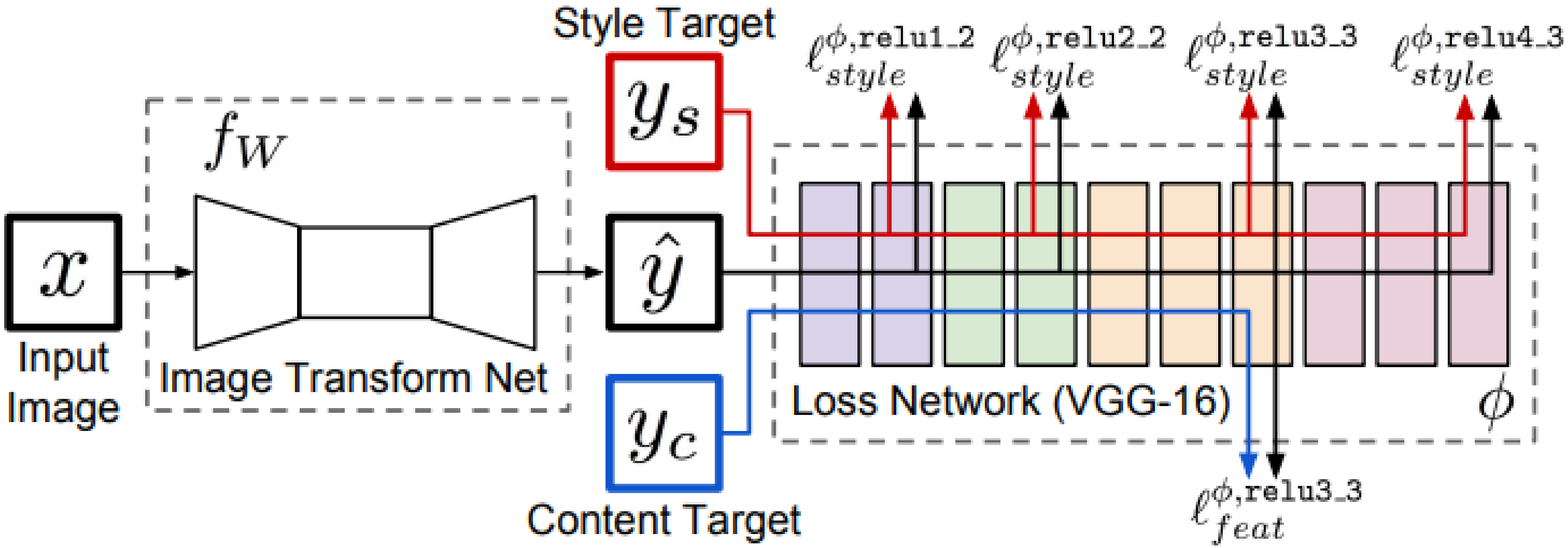
To generate images that mix the content of a photograph with the style of a painting we jointly minimise the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN. So let $\vec{p}$ be the photograph and $\vec{a}$ be the artwork. The loss function we minimise is

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

where α and β are the weighting factors for content and style reconstruction respectively.

3. Perceptual Losses for Real-Time Style Transfer

3.1. System Overview


Style Target
y_s
$\ell_{style}^{\phi,\texttt{relu1\_2}}$
$\ell_{style}^{\phi,\texttt{relu2\_2}}$
$\ell_{style}^{\phi,\texttt{relu3\_3}}$
$\ell_{style}^{\phi,\texttt{relu4\_3}}$
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Content Target
Loss Network (VGG-16)
ϕ
$\ell_{feat}^{\phi,\texttt{relu3\_3}}$

3.2. Image Transformation Networks

Overview:

Do not have any pooling layers, using strided convolutions instead.

Consists of five residual blocks:

- All non-residual convolutional layers: Spatial batch normalization and ReLU.
- Output layer: Scaled tanh.

Kernel:

- first and last layers: 9x9
- Other layers: 3x3

3.2. Image Transformation Networks

Inputs and Outputs: The input and output are both color images of shape 3 × 256 × 256. Since the image transformation networks are fully convolutional, at test-time they can be applied to images of any resolution.

Downsampling and Upsampling: The networks use two stride-2 convolutions to down sample the input followed by several residual blocks and then two convolutional layers with stride 1/2 to up sample.
-> Benefits:

- Computational: Can use a larger network for the same computational cost.
- Effective receptive field sizes: Giving larger effective receptive fields with the same number of layers.

Residual Connections: The body of the network consists of several residual blocks, each of which contains two 3 × 3 convolutional layers.
-> Make it easy for the network to learn the identify function .

3.3. Perceptual Loss Functions

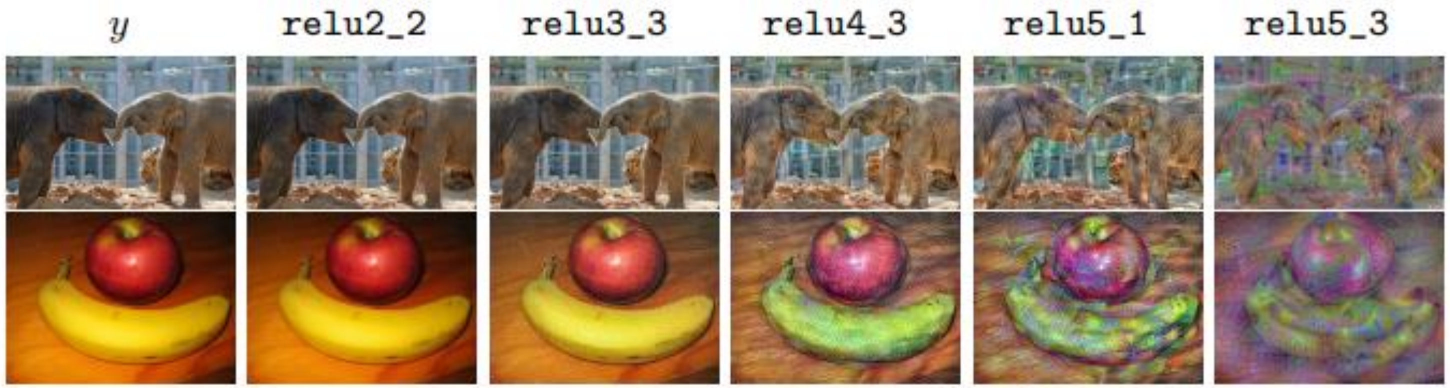
Feature Reconstruction Loss:

The pixels of the output image ŷ = $f_W(x)$ will have similar feature representations as computed by the loss network φ instead of exactly matching the pixels of the target image y. Let $\varphi_j(x)$ be the activations of the jth layer of the network φ when processing the image x; if j is a convolutional layer then $\varphi_j(x)$ will be a feature map of shape $C_j \times H_j \times W_j$. The feature reconstruction loss is the (squared, normalized) Euclidean distance between feature representations:

$$\ell_{feat}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\phi_j(\hat{y}) - \phi_j(y)\|_2^2$$

3.3. Perceptual Loss Functions

Feature Reconstruction Loss:



Finding an image ŷ that minimizes the feature reconstruction loss for early layers tends to produce images that are visually indistinguishable from y. As we reconstruct from higher layers, image content and overall spatial structure are preserved but color, texture, and exact shape are not.

3.3. Perceptual Loss Functions

Style Reconstruction Loss:

The feature reconstruction loss penalizes the output image ŷ when it deviates in content from the target y. We also wish to penalize differences in style: colors, textures, common patterns, etc.
Define the Gram matrix to be the $C_j \times C_j$ matrix whose elements are given by

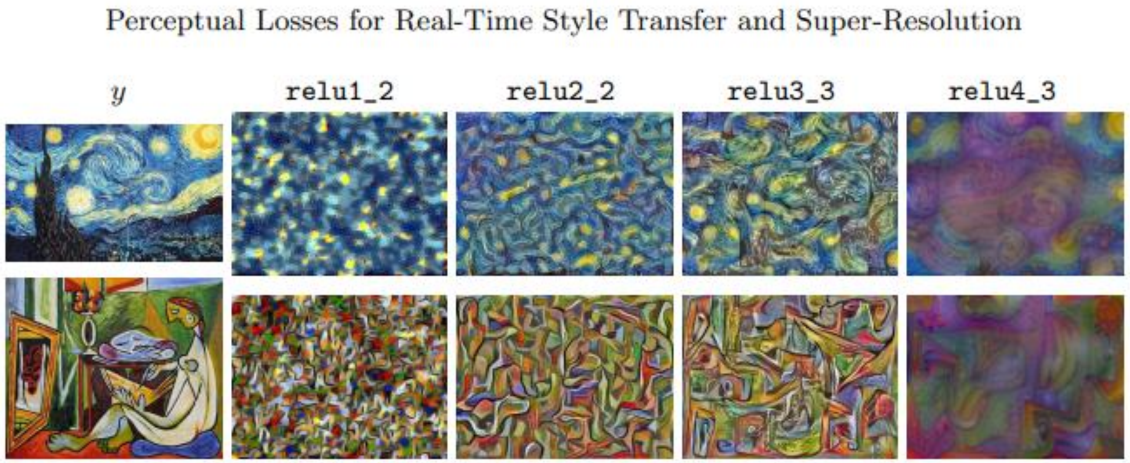
$$G_j^{\phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'} .$$

The style reconstruction loss is then the squared Frobenius norm of the difference between the Gram matrices of the output and target images:

$$\ell_{style}^{\phi,j}(\hat{y}, y) = \|G_j^{\phi}(\hat{y}) - G_j^{\phi}(y)\|_F^2 .$$

3.3. Perceptual Loss Functions

Style Reconstruction Loss:


Perceptual Losses for Real-Time Style Transfer and Super-Resolution
y
relu1_2
relu2_2
relu3_3
relu4_3

Generating an image ŷ that minimizes the style reconstruction loss preserves stylistic features from the target image, but does not preserve its spatial structure. Reconstructing from higher layers transfers larger-scale structure from the target image.

3.3. Perceptual Loss Functions

Final Loss:

Given style and content targets y_s and y_c and layers j and J at which to perform feature and style reconstruction, an image ŷ is generated by solving the problem:

$$\hat{y} = \arg\min_y \lambda_c \ell_{feat}^{\phi,j}(y, y_c) + \lambda_s \ell_{style}^{\phi,J}(y, y_s) + \lambda_{TV} \ell_{TV}(y)$$

Where: λ_c, λ_s, and λ_{TV} are scalars,

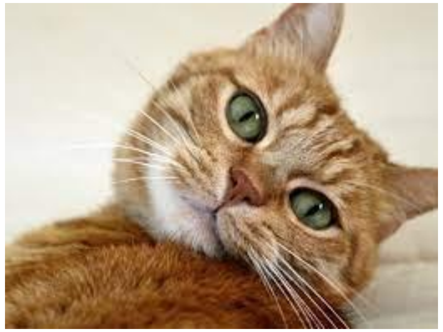
ℓ_{TV}(y) is total variation regularizer.

HUST

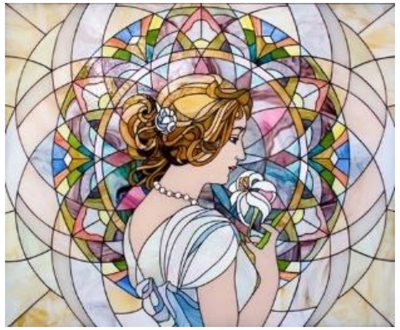
hust.edu.vn fb.com/dhbkhn

4. Experiment

These two images are used to describe the results of our experiment.



Content



Style

Resulting model of ‘A Neural Algorithm of Artistic Style’ implementation: Vanilla Style Transfer
Resulting model of ‘Perceptual Losses for Real-Time Style Transfer’ implementation: Fast Style Transfer

Vanilla Style Transfer

Training details

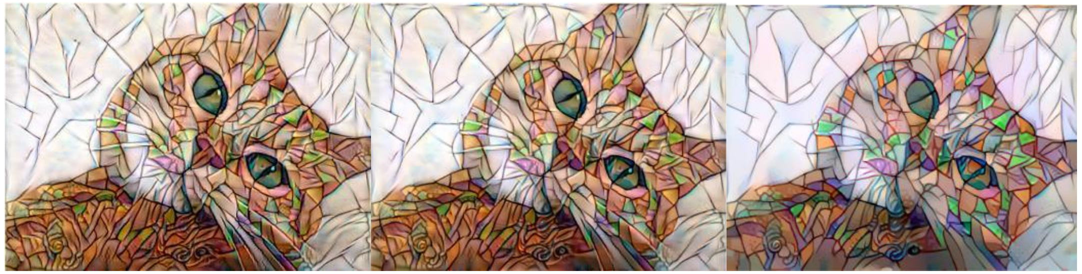
- $\alpha = 1, \beta = 10^3$
- Adam optimizer: learning rate = 0.01
- Iterations = 1000
- Use the content image instead of the white noise image as in the paper
- Network: VGG19
- Content representation layer: conv4_2
- Content layer weights = 1
- Style representation layers: conv1_1, conv2_1, conv3_1, conv4_1, conv5_1
- Style layer weights = 0.2, 0.2, 0.2, 0.2, 0.2

Vanilla Style Transfer

Total variation(tv) loss of a 2d image y:

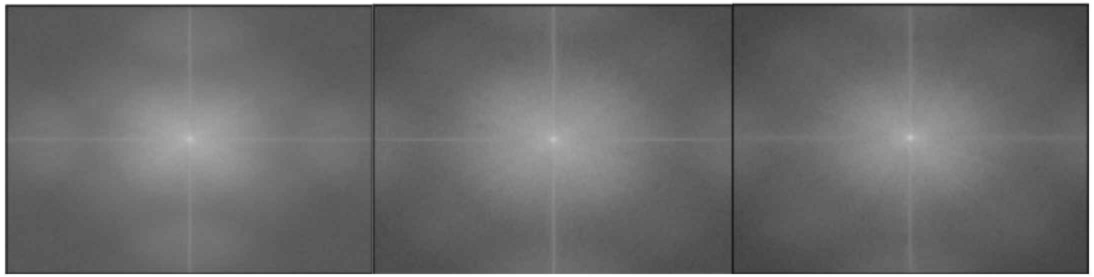
$$l_{tv} = \Sigma_{i,j}\left|y_{i+1,j} - y_{i,j}\right| + \left|y_{i,j+1} - y_{i,j}\right|$$

l_{tv} is weighted by a hyperparameter γ



$\gamma = 0$ $\gamma = 10$ $\gamma = 100$

Amplitude spectrum



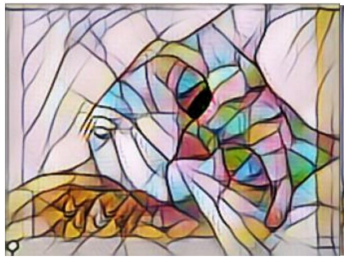
Fast Style Transfer

Training details

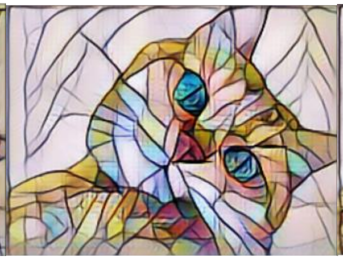
- Dataset: Microsoft COCO training set which contains 82,783 training images
- Preprocess: before being fetched to the network, images are resized into 256x256x3
- Adam optimizer: learning rate = 1e-3
- Batch size = 8
- Epochs = 2
- $\lambda_c = 2$
- $\lambda_s = 40$
- $\lambda_{TV} = 200$
- Loss network: VGG19
- Content representation layer: conv4_2
- Content layer weights = 1
- Style representation layers: conv1_1, conv2_1, conv3_1, conv4_1, conv5_1
- Style layer weights = 0.2, 0.2, 0.2, 0.2, 0.2

Fast Style Transfer

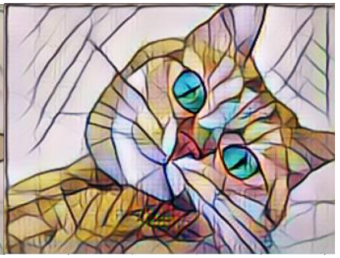
Content loss weight λ_c



$\lambda_c = 0.5$

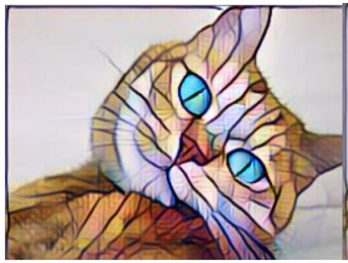


$\lambda_c = 1$

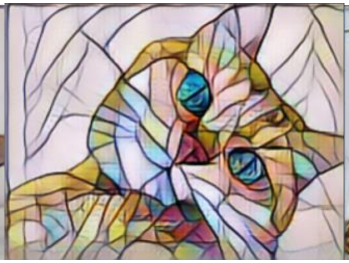


$\lambda_c = 2$

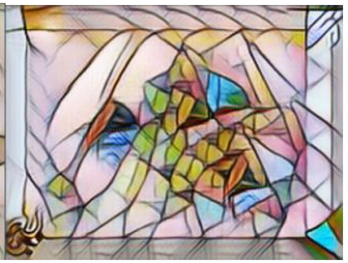
Style loss weight λ_s



$\lambda_s = 4$

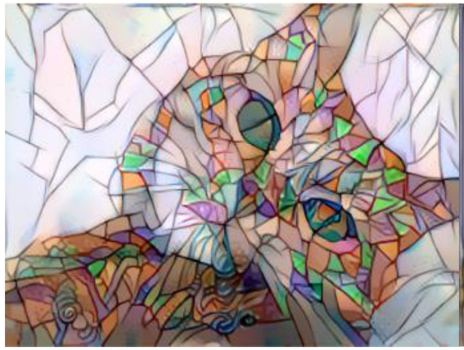


$\lambda_s = 40$

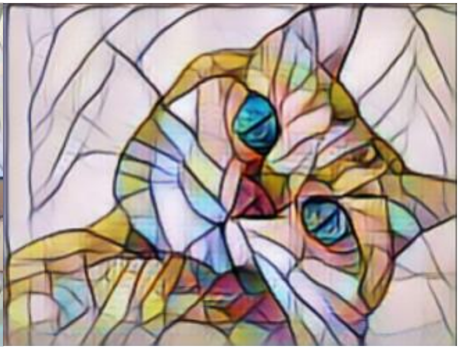


$\lambda_s = 400$

Comparison



Vanilla Style Tranfer



Fast Style Tranfer

Model	Time (seconds)
Vanilla Style Transfer	49.43
Fast Style Transfer	0.04

HUST

hust.edu.vn fb.com/dhbkhn

5. Demo

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !