Style Transfer for Videos

KU-BIG

Contents

1. Project Introduction
2. Paper Review & Results so far
3. Further plans

Project Introduction

Style Transfer for images

***P**: content image* ***A**: style image* ***X**: synthesized image*

"With pre-trained model"

Style Transfer for images

Artistic Style Transfer for Videos

Artistic Style Transfer for Videos

Temporal Constraint

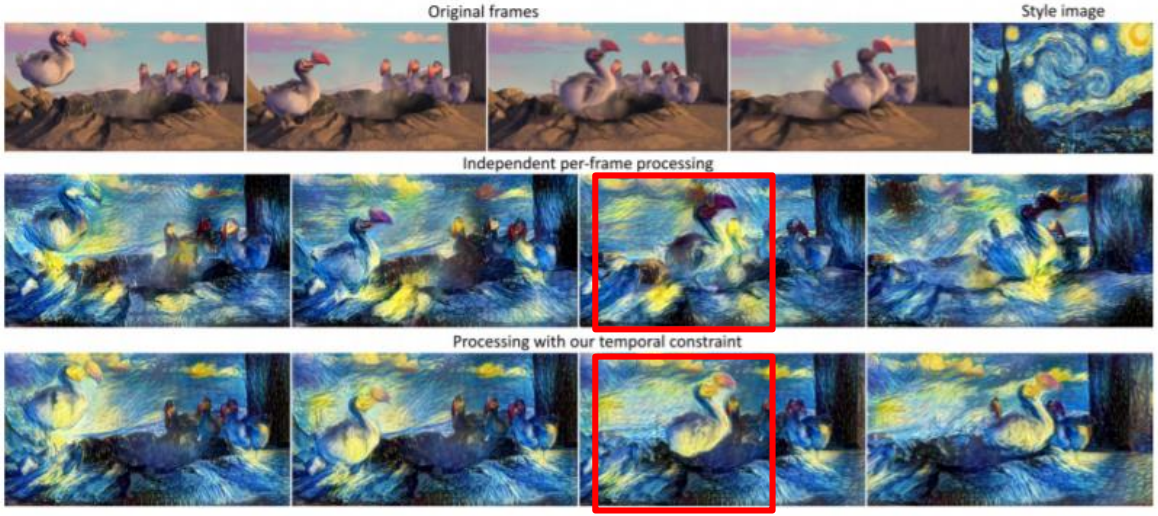
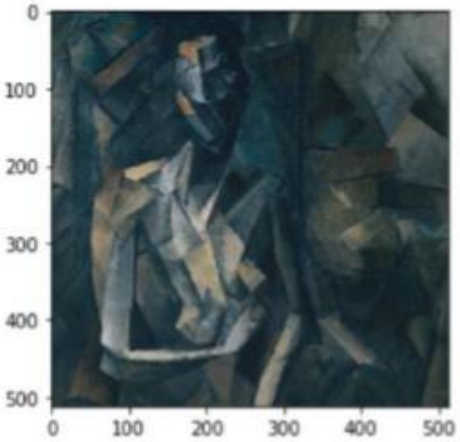


Fig. 1. Scene from *Ice Age* (2002) processed in the style of *The Starry Night*. Comparing independent per-frame processing to our time consistent approach, the latter is clearly preferable. Best observed in the supplemental video, see section 8.1.

Paper Review & Results so far

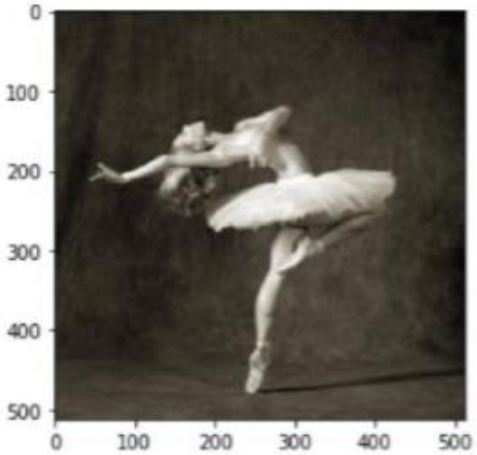
Results so far

Style Image



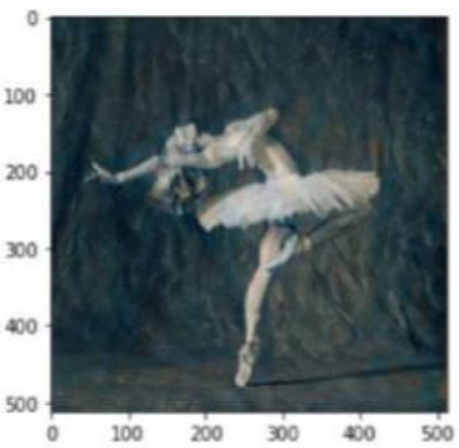
+

Content Image



=

Output Image



https://pytorch.org/tutorials/advanced/neural_style_tutorial.html#underlying-principle

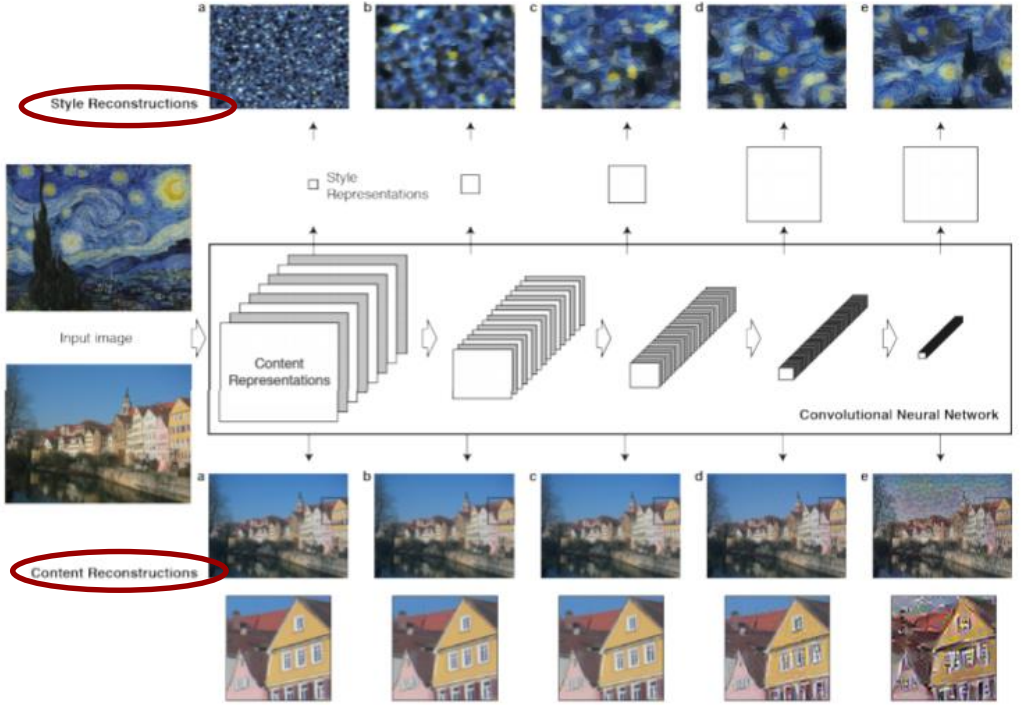
How to extract feature from various layers

Style Recon.

- Computes correlations between the different features in different layers of the CNN
- Subsets of CNN layers
- Use all layers : Zoom in

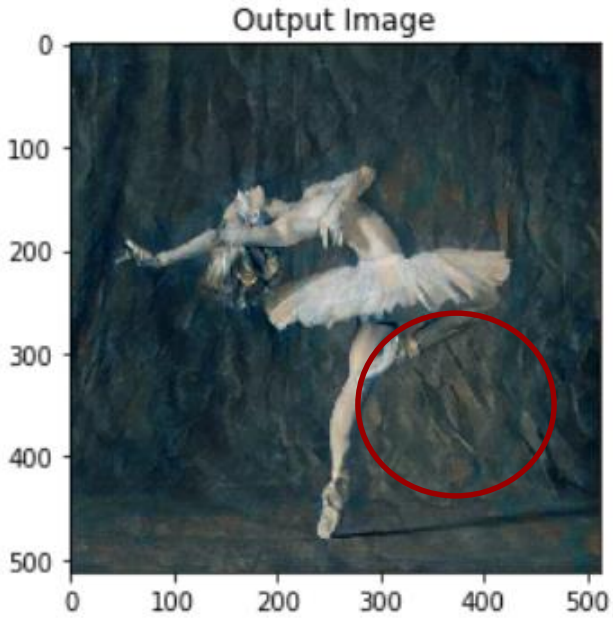
Content Recon.

- Lower layer : Contain Original Image
- Higher layer : Remain high level content

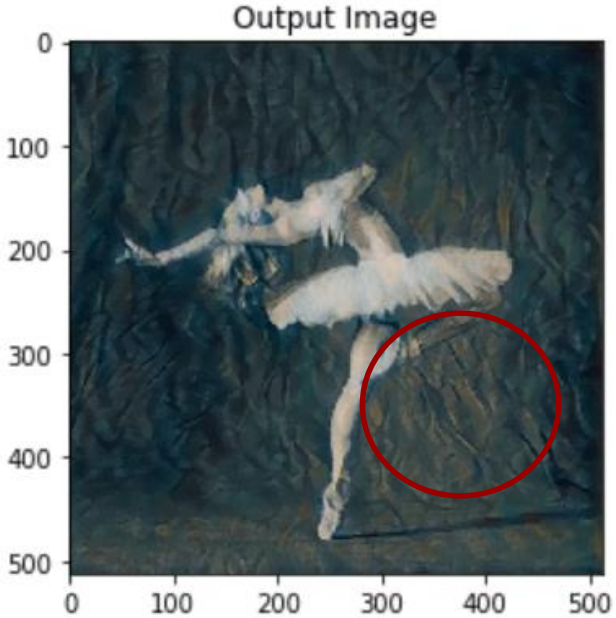


Results so far

Content Layer default : 'Conv_4'



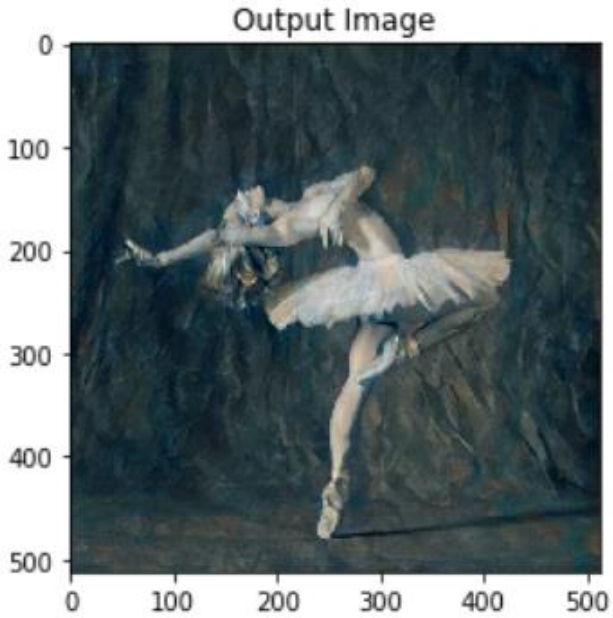
Content Layer default : 'Conv_1'



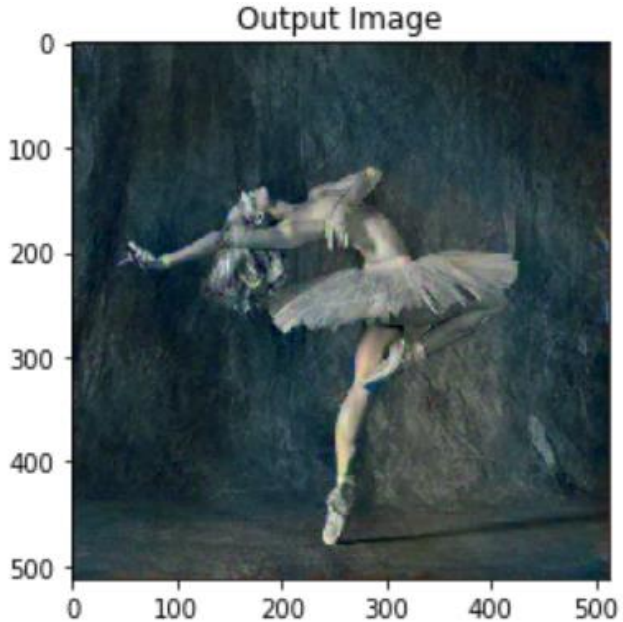
https://pytorch.org/tutorials/advanced/neural_style_tutorial.html#underlying-principle

Results so far

Style Layer default : 'Conv_1'-'Conv_5'



Style Layer default : 'Conv_1'-'Conv_3'



https://pytorch.org/tutorials/advanced/neural_style_tutorial.html#underlying-principle

How to deal with Loss and Hyperparameters

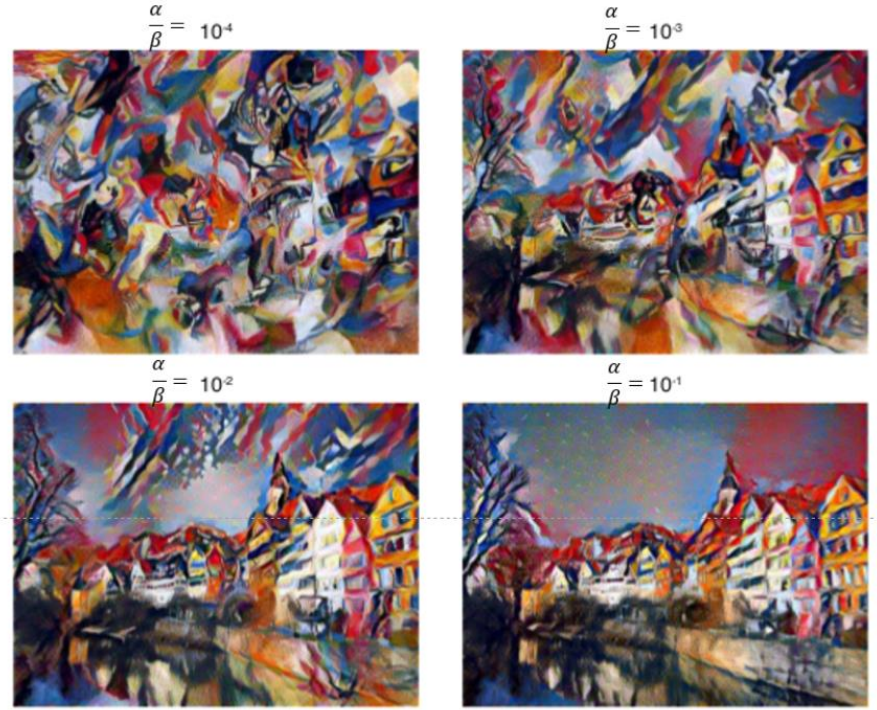
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

$\boldsymbol{\alpha} \downarrow \boldsymbol{\beta} \uparrow \Rightarrow$ Lower $\frac{\alpha}{\beta}$

More focused on Style, rather than Content

$\boldsymbol{\alpha} \uparrow \boldsymbol{\beta} \downarrow \Rightarrow$ Higher $\frac{\alpha}{\beta}$

More focused on Content, rather than Style



Results so far

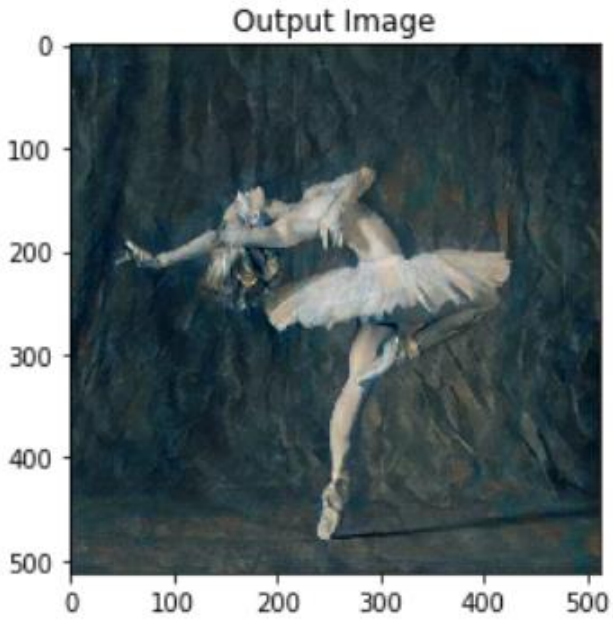
```
##function that performs the neural transfer.
#for each iteration of the networks, it is fed and updated input and computes new losses
#run the backward methods of each loss module to dynamicaly compute their gradients

def run_style_transfer(cnn, normalization_mean, normalization_std,
                       content_img, style_img, input_img, num_steps=300,
                       style_weight=1000000, content_weight=1):
    """Run the style transfer."""
    print('Building the style transfer model..')
    model, style_losses, content_losses = get_style_model_and_losses(cnn,
        normalization_mean, normalization_std, style_img, content_img)
    optimizer = get_input_optimizer(input_img)

    print('Optimizing..')
    run = [0]
    while run[0] <= num_steps:

        def closure():
            # correct the values of updated input image
            input_img.data.clamp_(0, 1)

            optimizer.zero_grad()
            model(input_img)
            style_score = 0
            content_score = 0

            for sl in style_losses:
                style_score += sl.loss
```

β α

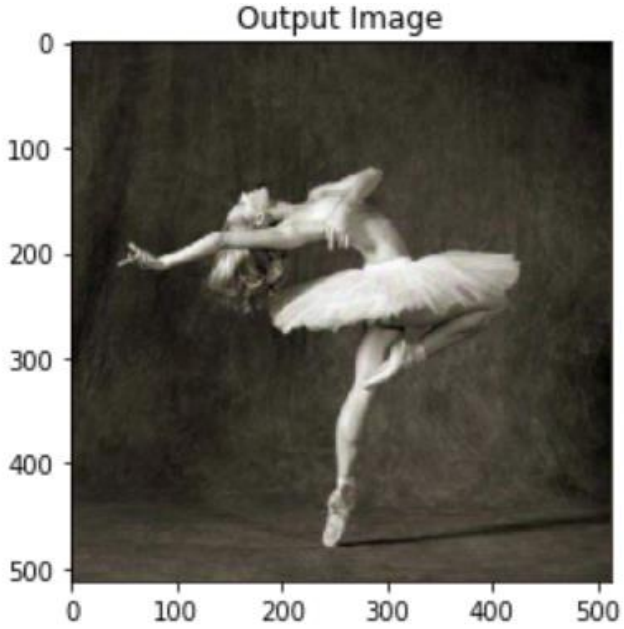
https://pytorch.org/tutorials/advanced/neural_style_tutorial.html#underlying-principle

Results so far

Style Weight(β) : 1000000



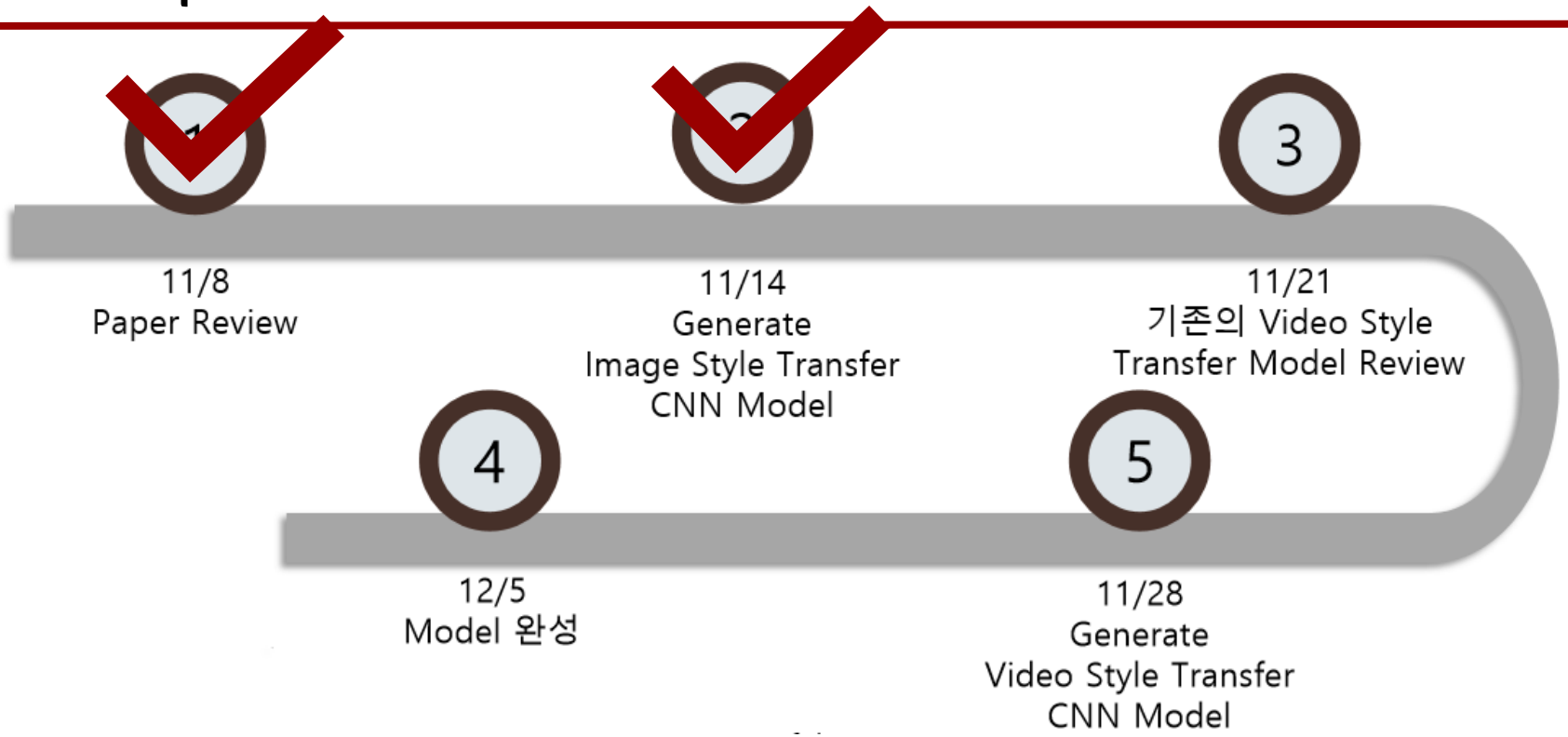
Style Weight(β) : 100



https://pytorch.org/tutorials/advanced/neural_style_tutorial.html#underlying-principle

Further Plans

Further plans



Thank You

KU-BIG