# Homework #5

# Convolutional Neural Networks

Junfei Liu

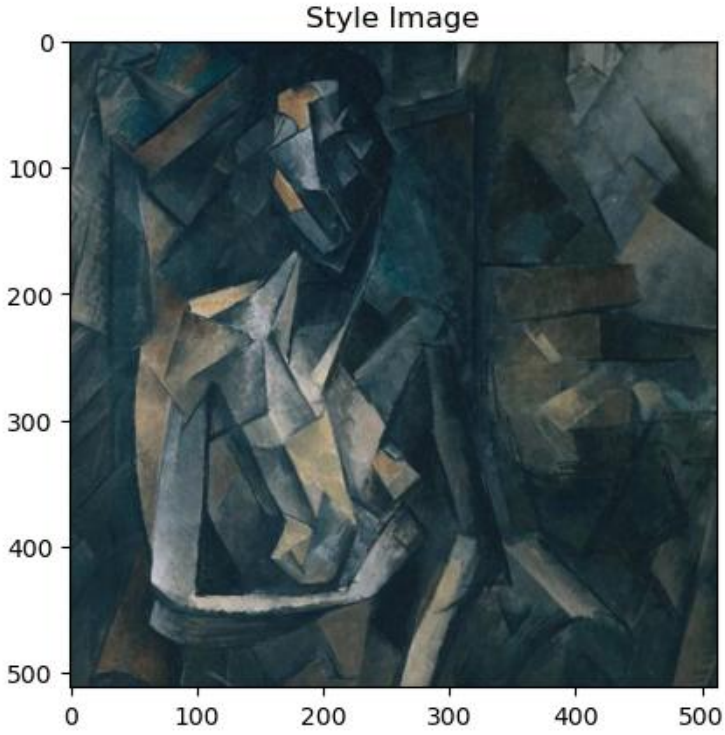
CSC249

March 29, 2023

# Static style transfer

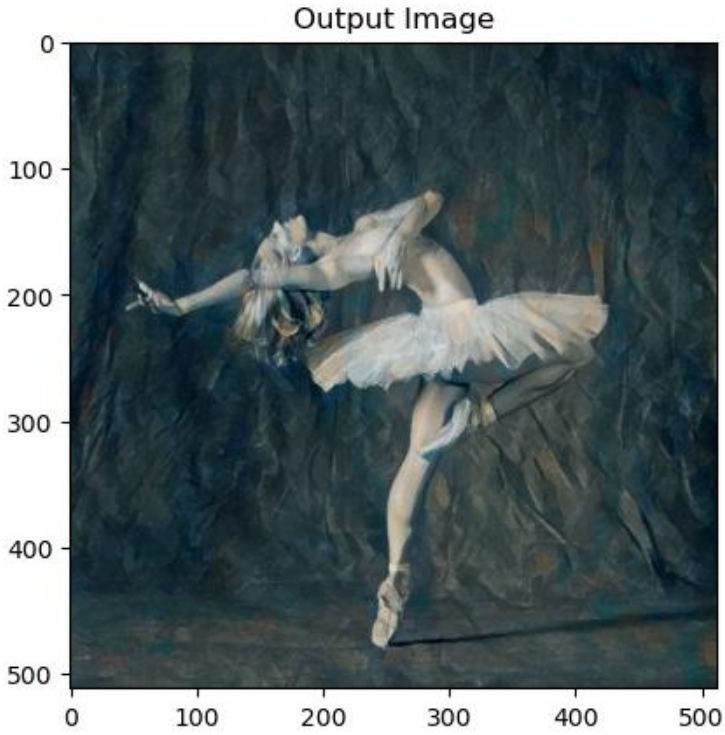
## Image I/O

Because the GATYS requires the content and style images to be of the same size while the provided content.jpg and style.jpg need manual clipping for transforms from torchvision.transforms to properly resize, I used picasso.jpg and dancing.jpg from neural style tutorial for convenience. The Picasso.jpg is used for style and dancing for content. These two images after getting loaded into memory and resized are shown below:




**Style Tranfered Image**

The input image is set to the content image, i.e. dancing.jpg. With all input image, content image, and style image set, the result image is generated as shown below:




This image is generated after 300 steps with style loss of 2.700416 and content loss of 9.909768.

## Experiments with different steps

I experimented with number of steps in [10, 50, 100, 200, 400, 800] as shown below:


It is apparent that the more steps it takes, the closer to the style image by both texture and color. It shows as a continuous descent in both content and style losses with increasing steps, which means it is transferring the style with better performance.

## Experiments with different style weights

I experimented with style weights in [10000000, 1000000, 100000, 10000, 1000, 100] with 300 steps as shown below:


It is also apparent that the higher the style weight the closer result image is to the style image. The “style_weight” stands for how important the style is regarded. When style weight is high, style loss is decreasing rapidly while the content loss might even increase for a small amount; when style weight is low, reducing the style loss will not be regarded as important.

## Count time of style transfer

The following data is obtained on GTX1660ti graphics card with cuda.

One trial of style transfer with 300 steps and default weights takes 35.9293 seconds.

```
Style transfer in 35.9232 seconds
```

One trail of style transfer with one step takes 1.4213 seconds. It takes significantly longer time than the average time with 300 steps could be that it involves optimizing and model setup before the style transfer runs.

```
Style transfer in 1.4213 seconds
```

# Arbitrary style transfer

## Style Tranfered Image

By setting up the AdaIN algorithm on local machine and running the test.py with directories to the content and style images, the result image is generated as follows:

It is noted that this image only has both horizontal and vertical dpi of 96, both with PIL image save by transforming tensor to PIL image and saving it. It could be the reason why it seems to be blurrier than the image generated by GATYS. However, the style is more obvious with AdaIN.

## Comparison between GATYS and AdaIN

From the perspective of running time, the style transfer of AdaIN is 3.5548s with the result image above, while GATYS finishes one step in 1.4213 seconds and 300 steps in 35.9232 seconds. Because the dpi of result images are different even given the same resolution (512 x 512), it would not be a reasonable evaluation criteria to compare running time directly. Moreover, GATYS requires a certain number of steps to reach an ideal result, while AdaIN only takes one trail.

With more steps taken, GATYS can reach a high standard of style transfer, yet the style is not visible with small number of steps, while AdaIN could produce a well-combined picture with style and content in one trail.

## Train AdaIN

By running train.py with max_iter and save_model_interval set to 1000 and only the target content and style images in corresponding folders, a .pth.tar file is generated.

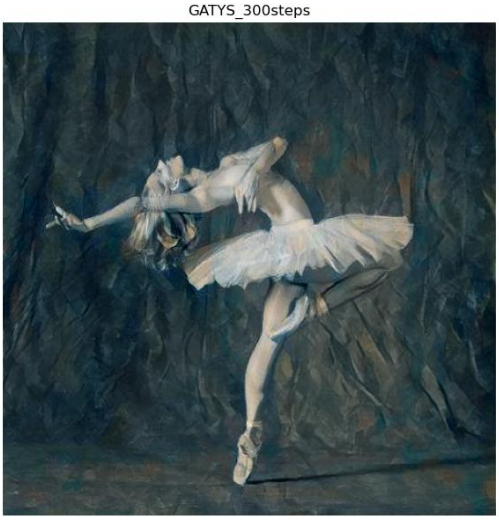
No huge difference in style is observed between the image generated with finetuned model and one with default parameters. However, the color of the content image seems to be better preserved as more parts remains yellow instead of turning into blue, which makes this image a slightly better one. The dpi difference still exists; if we do not take it into account, GATYS generates better picture.

I also trained the decoder from scratch with the dancing content and picasso style:

Training from scratch clearly does not generate pictures of high quality as the style has taken over the content in the result image above.

The combination of images below show a more complete view of comparisons:

GATYS_300steps



AdaIN_default

AdaIN_finetune

AdaIN_scratch

## Style transfer on a new pair of content and style

The new pair of content and style images come from the neural style transfer page on Tensorflow core tutorial site. The content image is the picture of a dog:

The style image is an artwork, Wassily Kandinsky's Composition 7:

The image generated with the AdaIN models with the original parameter is shown below:

The image generated with the finetuned parameter is shown below:

Although there is no huge difference, I tend to believe that the images generated with finetuned parameters is better as the dog face and fur are clearer. It is strange that the finetuned model with a different pair of style and content images could result in a better style transferred image. The similar art genre in the style image could be the reason.

When the image is finetuned with only a pair of content and style images, the model is fitting to this specific pair and will perform better with them. However, because it will result in overfitting, the model may get worse at generalizing to other pairs of contents and styles.