Neural Style Transfer

Reference:
https://www.youtube.com/watch?v=imX4kSKDY7s&list=PLhhyoLH6IjfxeoooqP9rhU3HJIAVAJ3Vz&index=22

Based on two input images, one for content and one for style generate an image that represent the style from the style image applied on the content image.

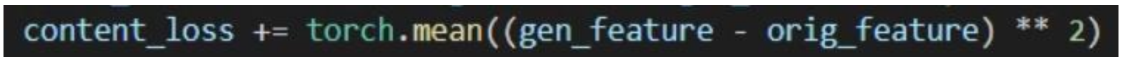
Loss function

C: content image

S: style image

G: generated image

$$\mathcal{L}_{\text{total}}(G) = \alpha\mathcal{L}_{\text{content}}(C, G) + \beta\mathcal{L}_{\text{style}}(S, G)$$



```
content_loss += torch.mean((gen_feature - orig_feature) ** 2)
```

Using mean will subtract it by 2 but it will only affect alpha value which we can adjust

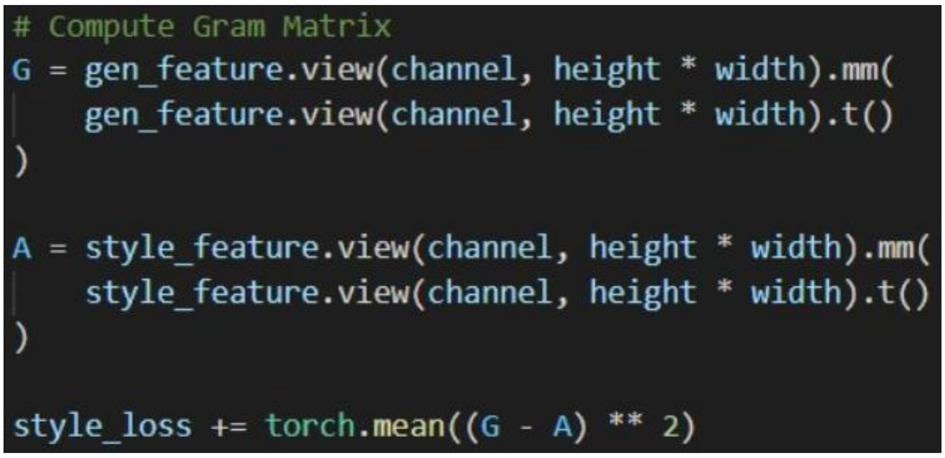
Content loss

$$J_{\text{content}}\ (C, G) = \frac{1}{2}\left\| \underset{\substack{\uparrow \\ \text{Original/Content}}}{a^{[\ell](C)}} - \underset{\substack{\uparrow \\ \text{Generated}}}{a^{[\ell](G)}} \right\|^2$$

a is the output of the l layer from C or G.

Style loss

Gram Matrix

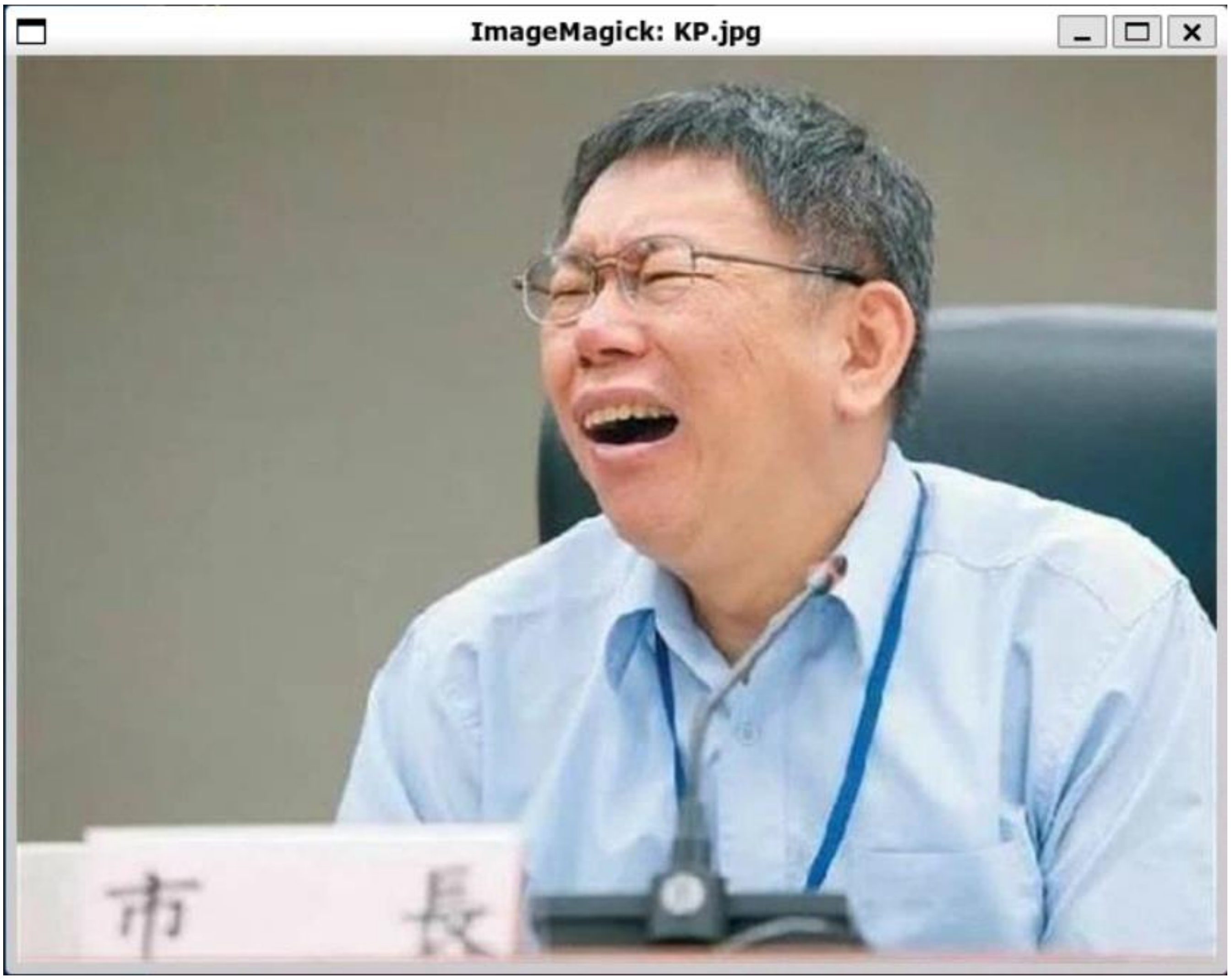


```
# Compute Gram Matrix
G = gen_feature.view(channel, height * width).mm(
    gen_feature.view(channel, height * width).t()
)

A = style_feature.view(channel, height * width).mm(
    style_feature.view(channel, height * width).t()
)

style_loss += torch.mean((G - A) ** 2)
```

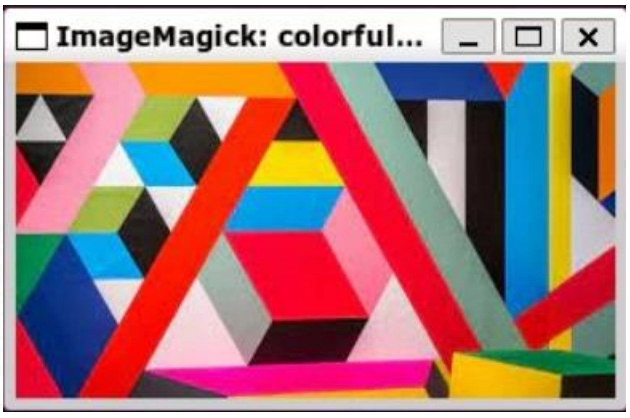
We multiply every pixel from every channel with other channels which end up with matrix with channel * channel size. By calculating gram matrix for both generated image layer feature and style image layer feature, the gram matrix is calculated the correlation between two images. If two images across the different channels have similar color, it considers that two images have similar style.

Result

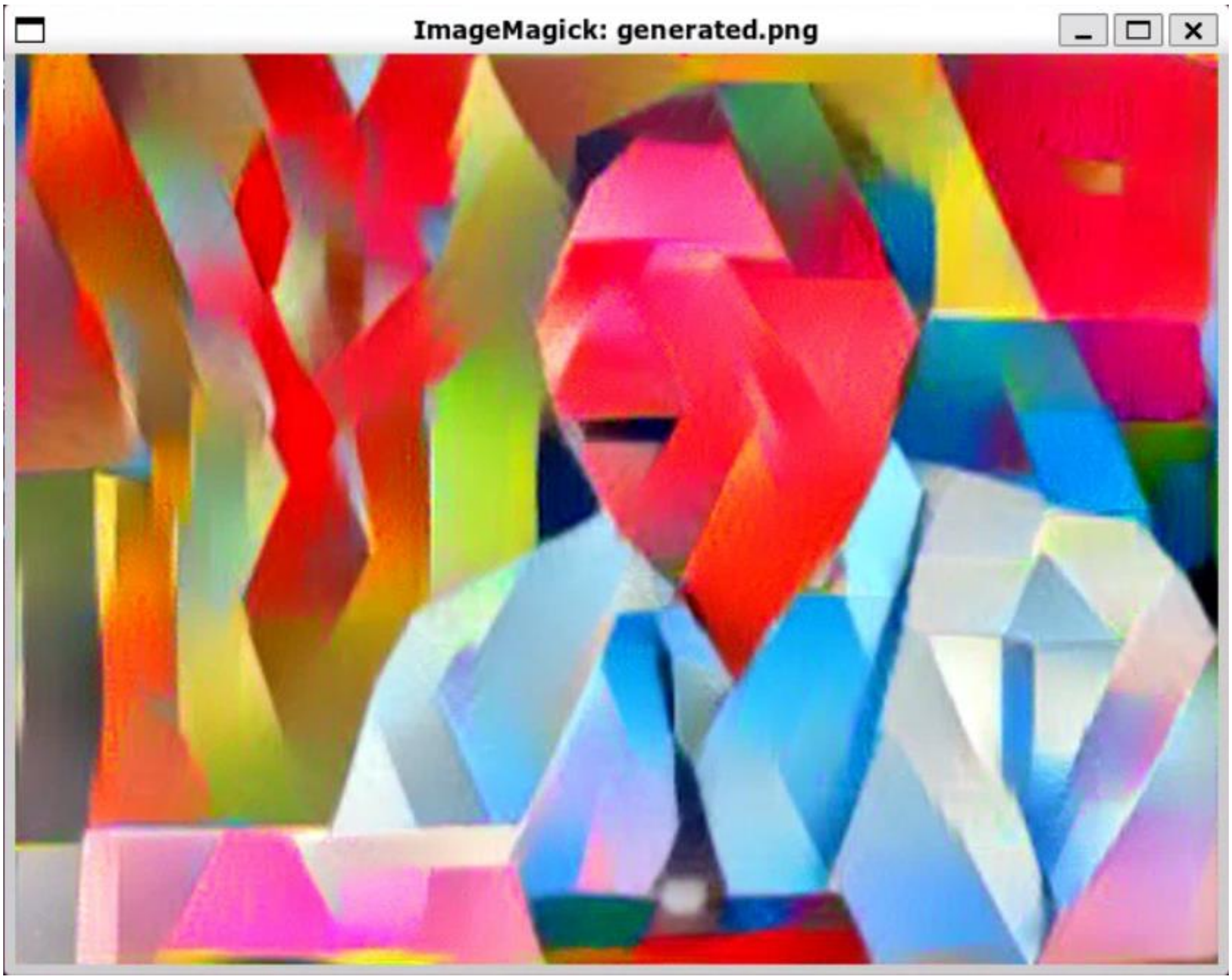
Original image:



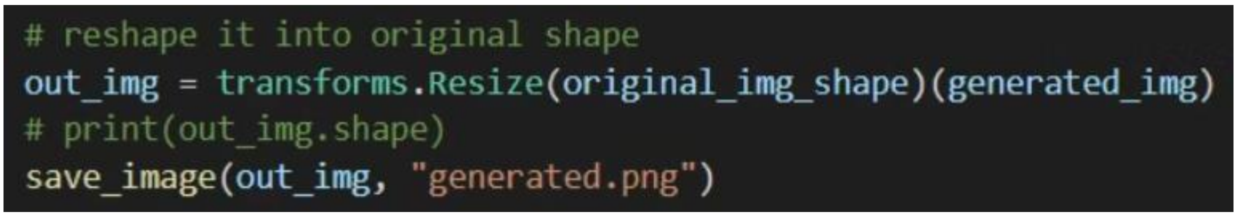
Style image:



Generated image:



In the program, the output tensor from the model is being resized back to the original image shape.



```
# reshape it into original shape
out_img = transforms.Resize(original_img_shape)(generated_img)
# print(out_img.shape)
save_image(out_img, "generated.png")
```