

Курс МАДМО2

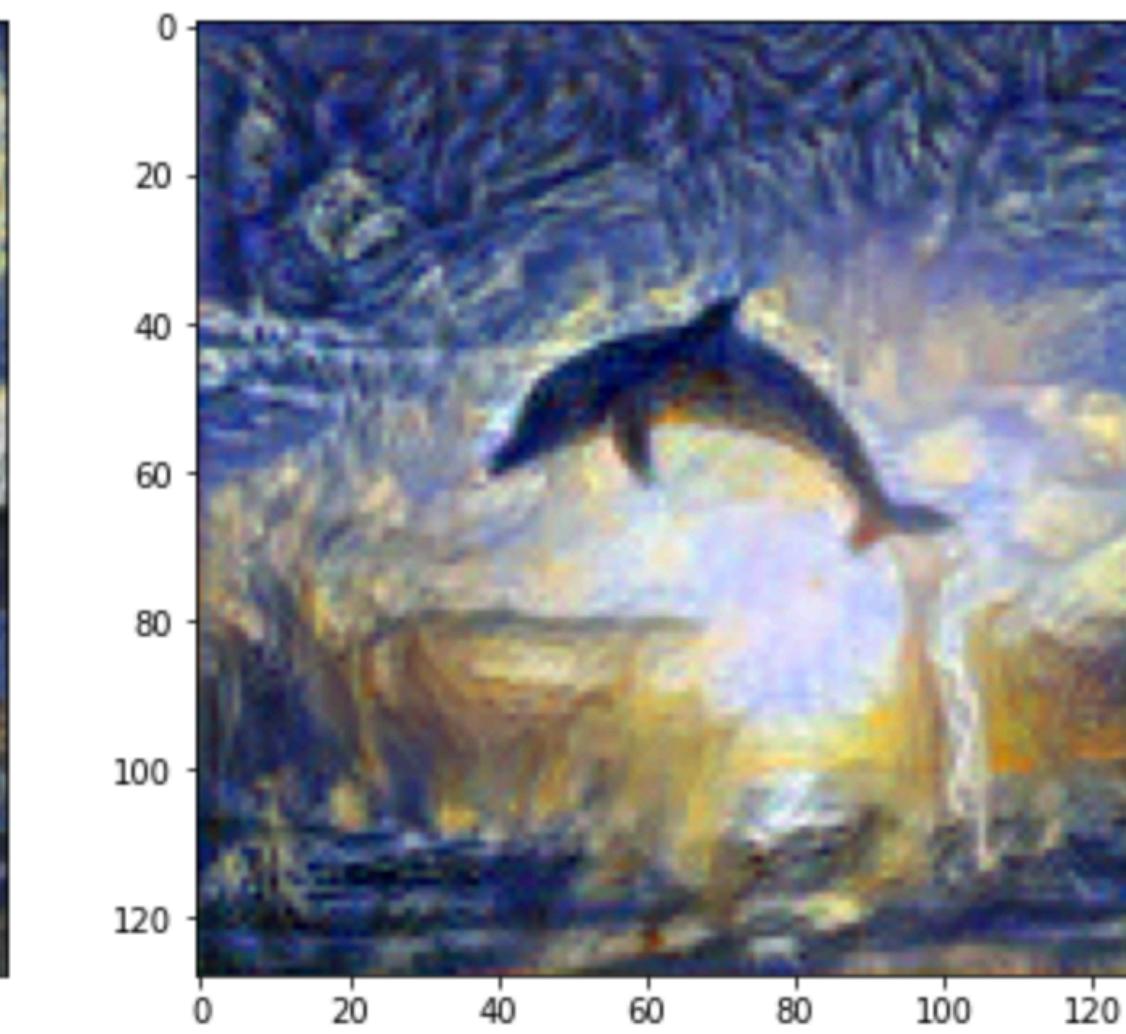
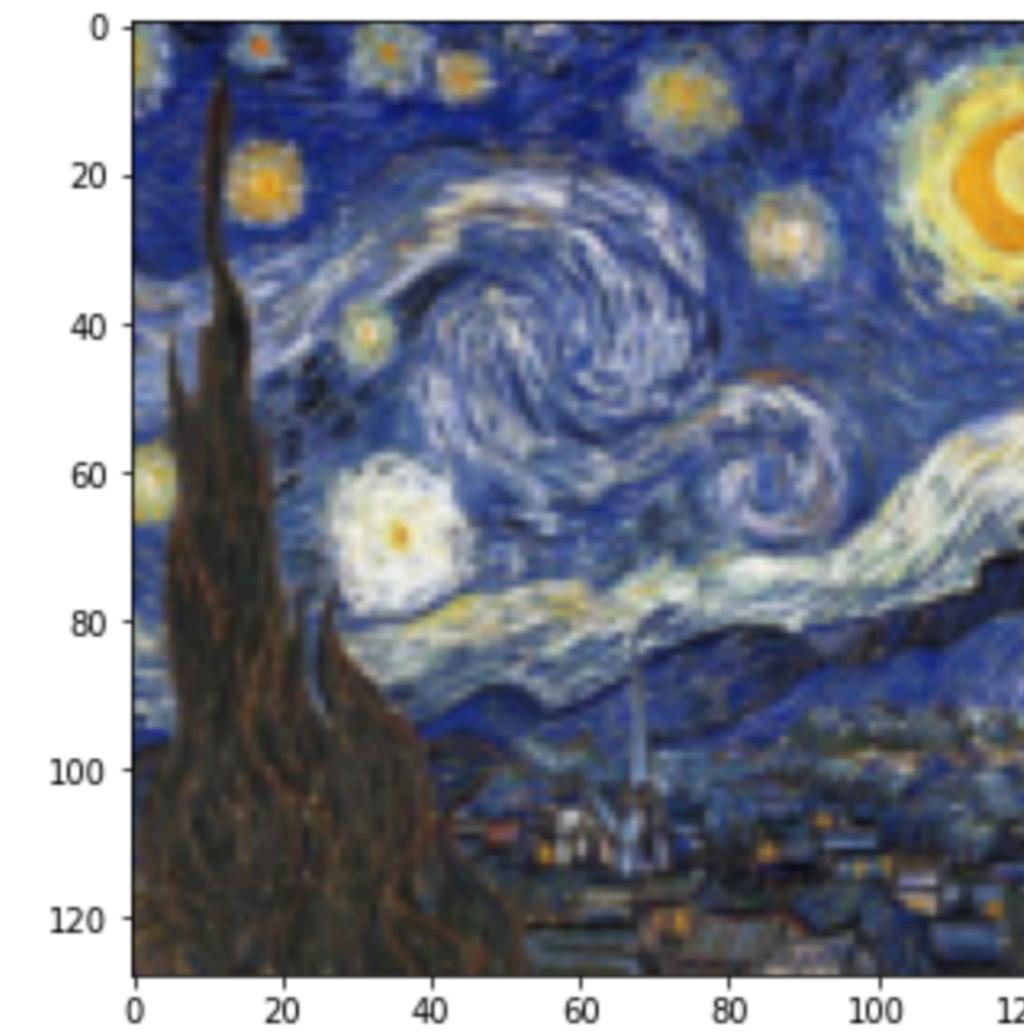
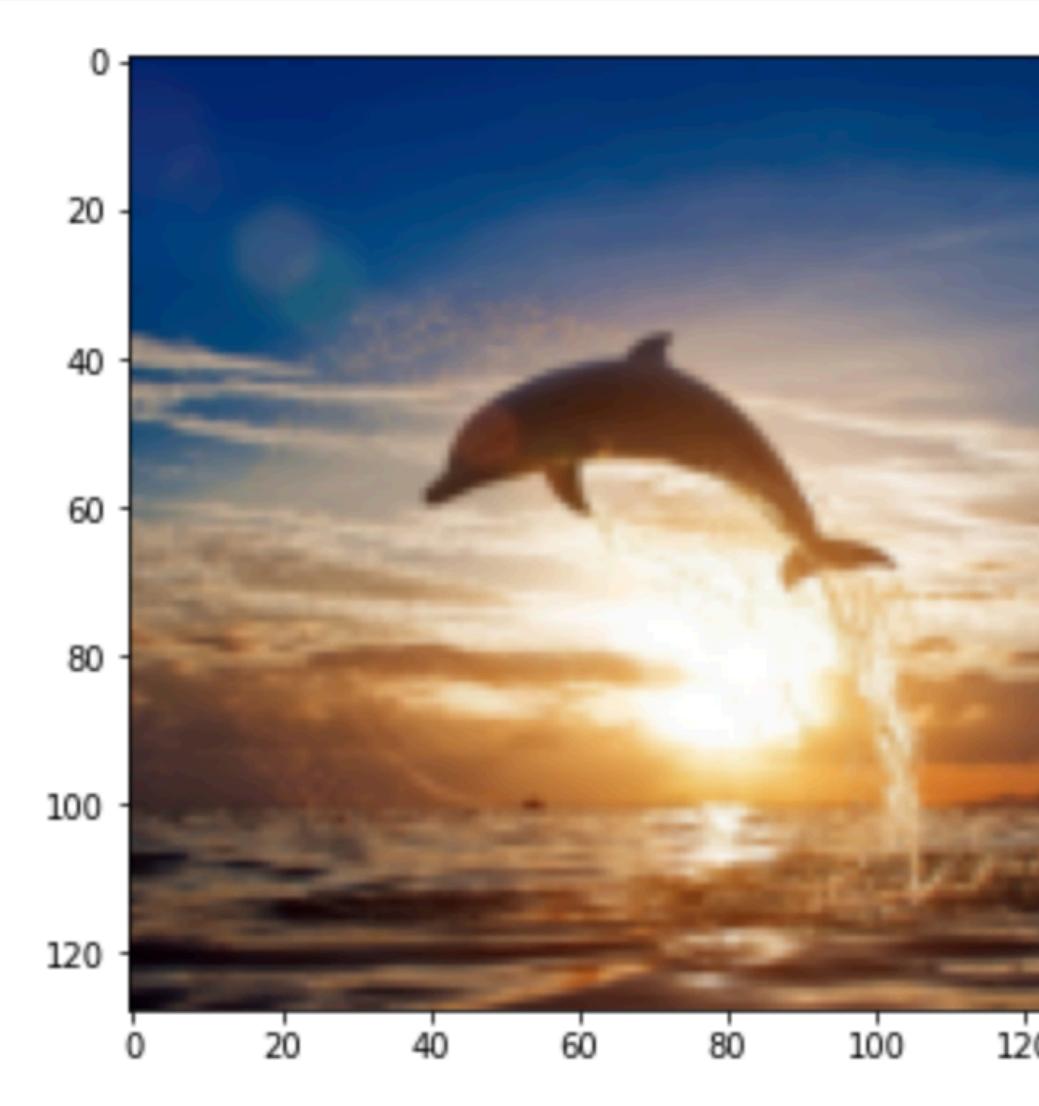
Artistic style transfer

Или что под капотом Prisma

Данилов Степан

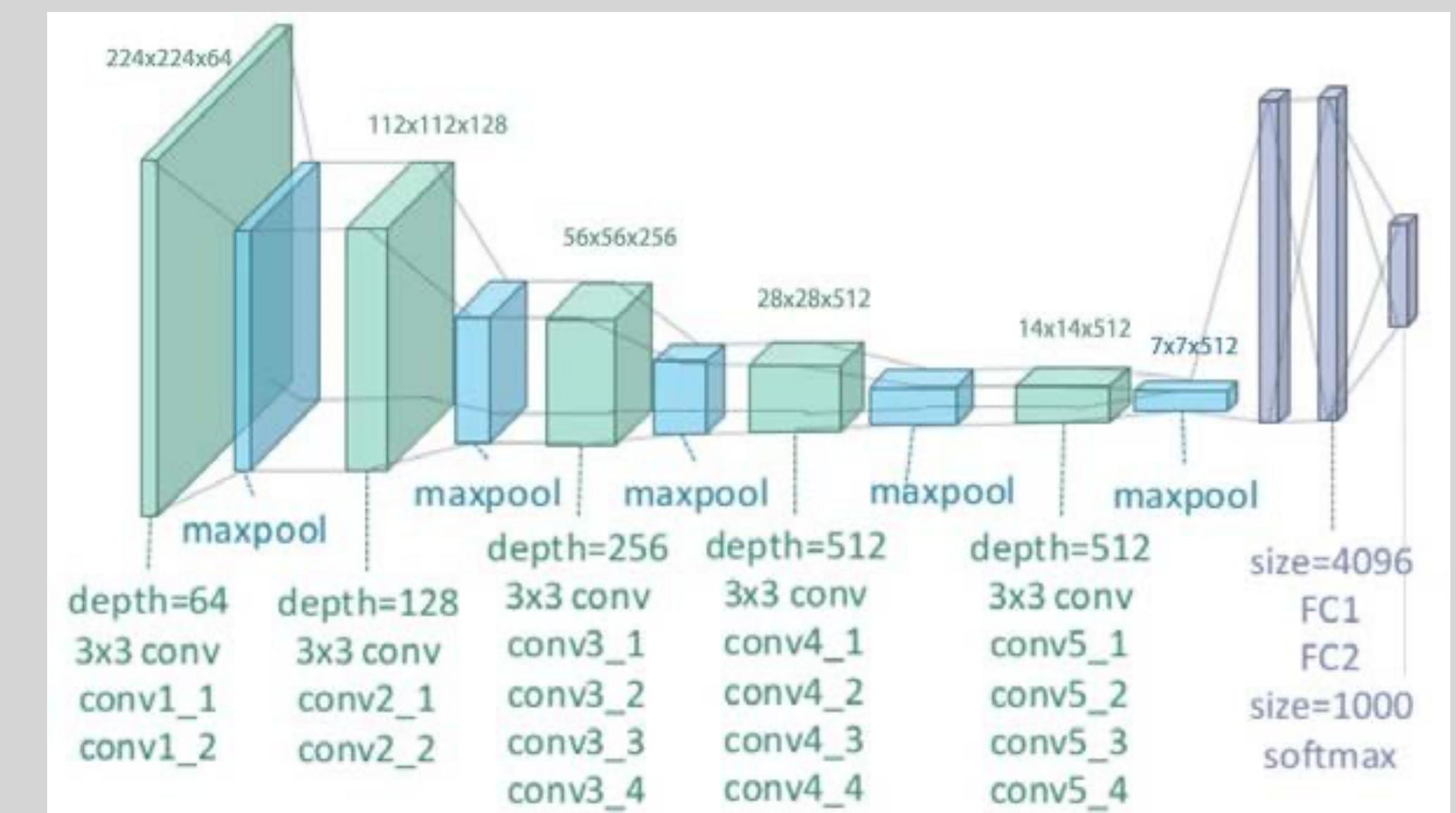
Задача

- На входе, два изображения: content, style images
- Перенести стиль изображения на content image, не теряя content
- Подобрать интересные сочетания



Пример реализации

- Взять достаточно глубокую предобученную сеть (Vgg19)
- Выдвинуть гипотезу, что первые слои в сети отвечают за стиль, более глубокие за контент в изображении
- Выбрать слои для подсчета style transfer loss и content loss
- Добавить подсчет content / style losses между выбранными слоям, обрезать неиспользуемые слои предобученной сети
- Произвести оптимизацию loss в пространстве параметров выходной картинки



Loss function

$$\text{style loss} \quad L_{\text{style}}(x; t) = \|G^l(x) - G^l(t)\|_2^2$$

$$\text{Content loss} : L_{\text{content}}(x; c) = \|F^l(x) - F^l(c)\|_2^2$$

$$\text{Total loss} : \quad L_{\text{total}}(x; c, t) = \alpha L_{\text{content}}(x; c) + \beta L_{\text{style}}(x; t)$$

```
def gram_matrix(inp):
    b, ch, h, w = inp.size()
    features = inp.view(b, ch, w * h)
    features_t = features.transpose(1, 2)
    gram = torch.bmm(features, features_t) / (ch * h * w)
    return gram

class ContentLoss(nn.Module):
    def __init__(self, target):
        super(ContentLoss, self).__init__()
        self.target = target.detach()

    def forward(self, input_img):
        self.loss = F.mse_loss(input_img, self.target)
        return input_img

class StyleLoss(nn.Module):
    def __init__(self, target_feature):
        super(StyleLoss, self).__init__()
        self.target = gram_matrix(target_feature).detach()

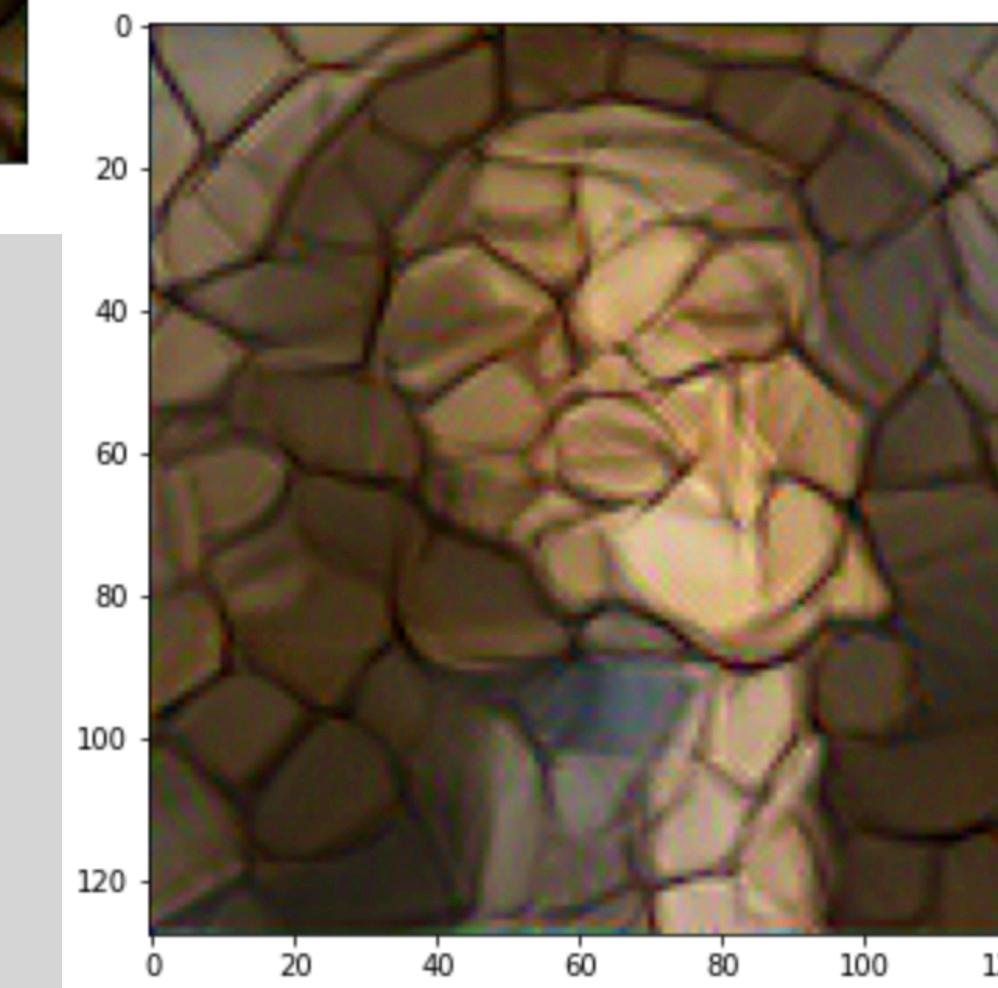
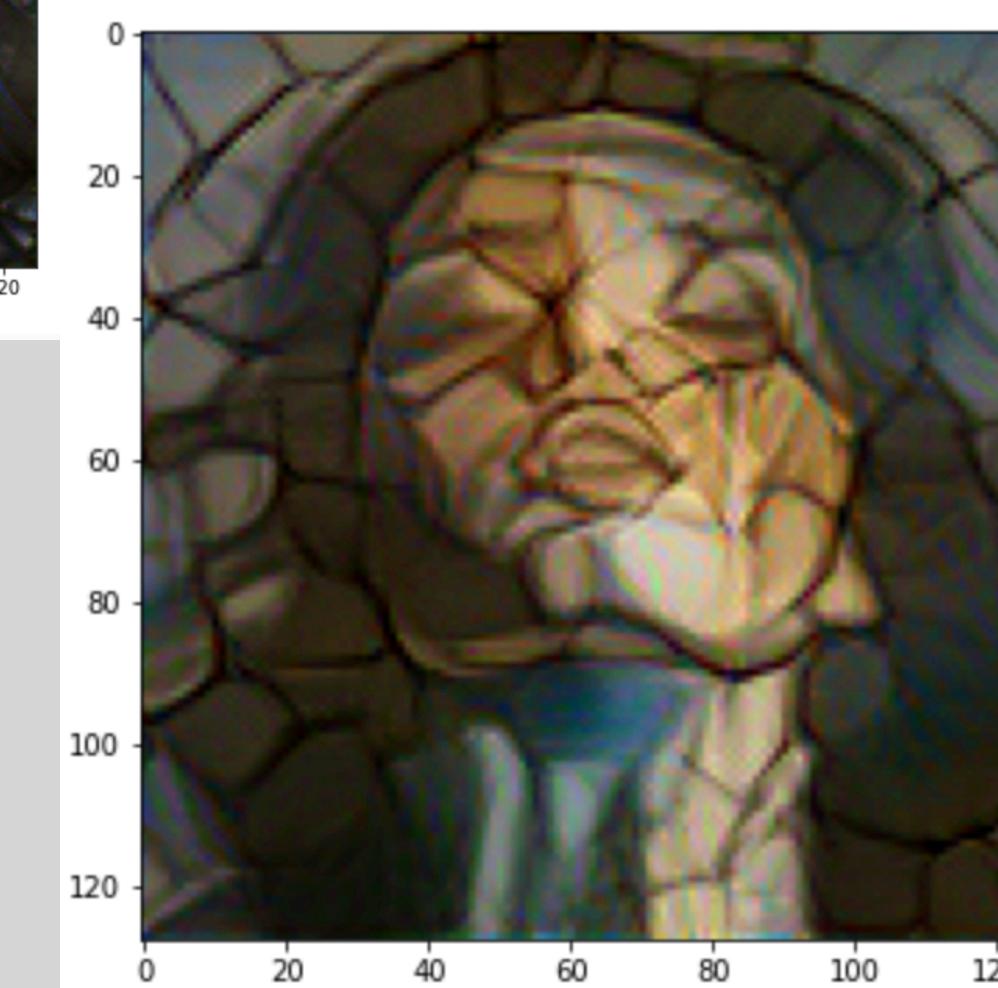
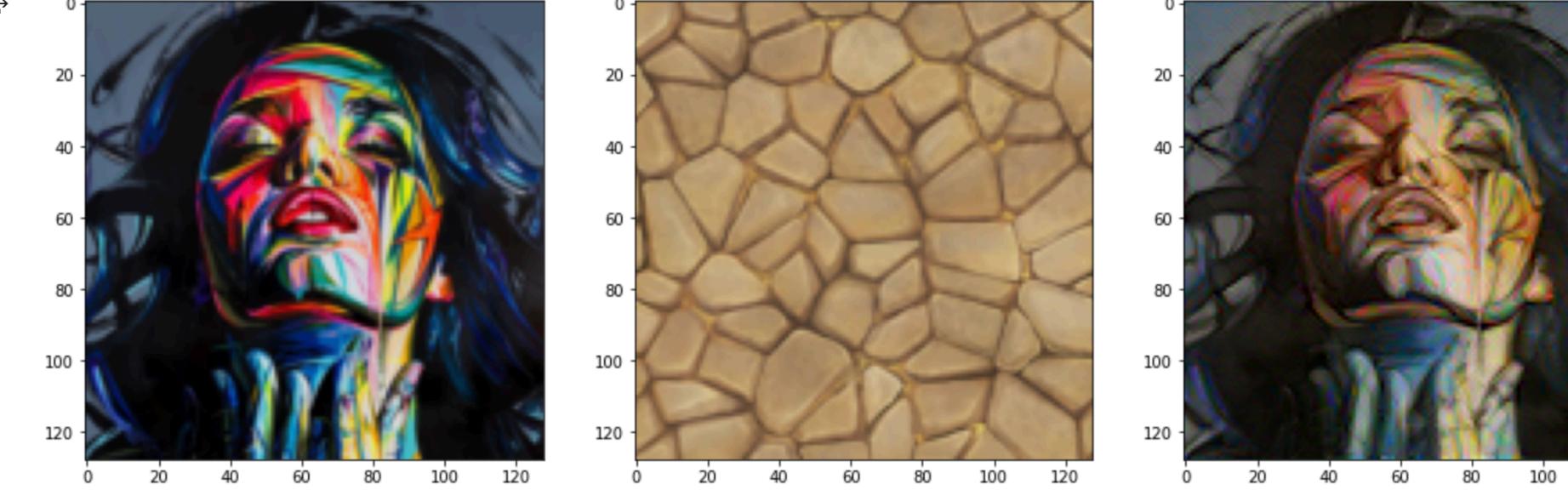
    def forward(self, input_img):
        G = gram_matrix(input_img)
        self.loss = F.mse_loss(G, self.target)
        return input_img
```

- Итоговая модель:

```
Sequential(
    (0): Normalization()
    (conv_1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (style_loss_1): StyleLoss()
    (relu_1): ReLU()
    (conv_2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (style_loss_2): StyleLoss()
    (relu_2): ReLU()
    (pool_2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (style_loss_3): StyleLoss()
    (relu_3): ReLU()
    (conv_4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (style_loss_4): StyleLoss()
    (relu_4): ReLU()
    (pool_4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (style_loss_5): StyleLoss()
    (relu_5): ReLU()
    (conv_6): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (style_loss_6): StyleLoss()
    (relu_6): ReLU()
    (conv_7): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (content_loss_7): ContentLoss()
    (relu_7): ReLU()
    (conv_8): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (content_loss_8): ContentLoss()
)
```

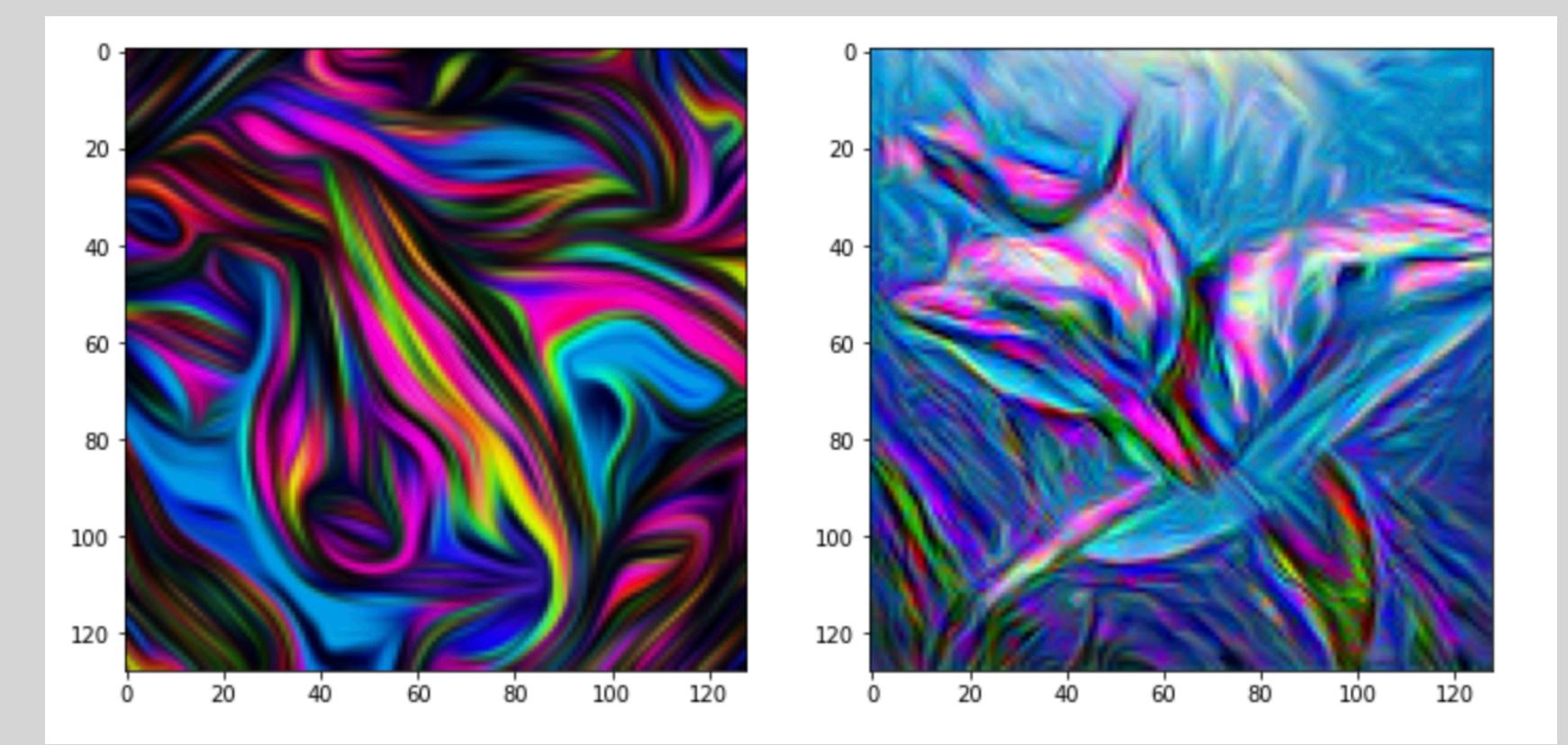
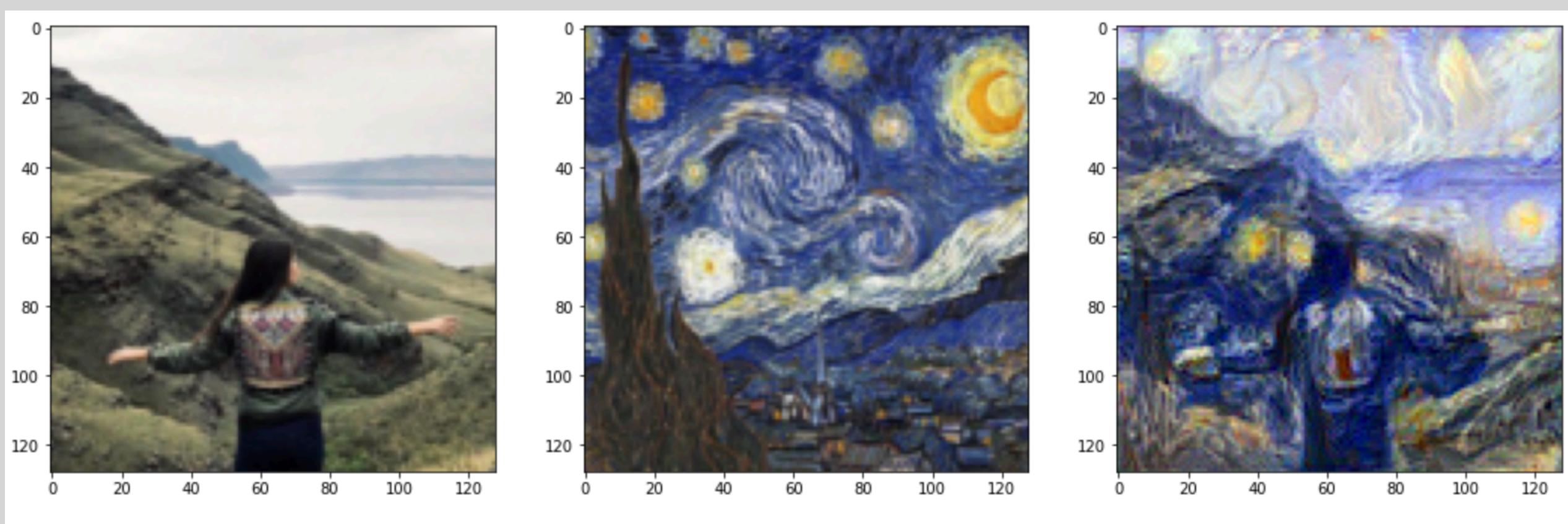
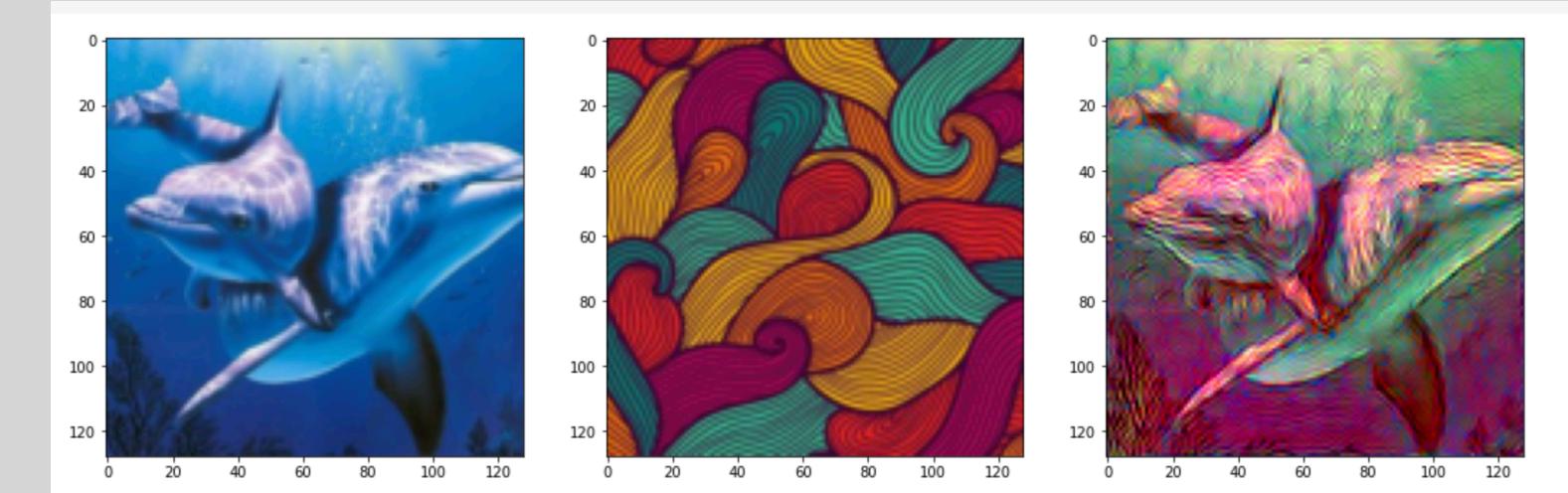
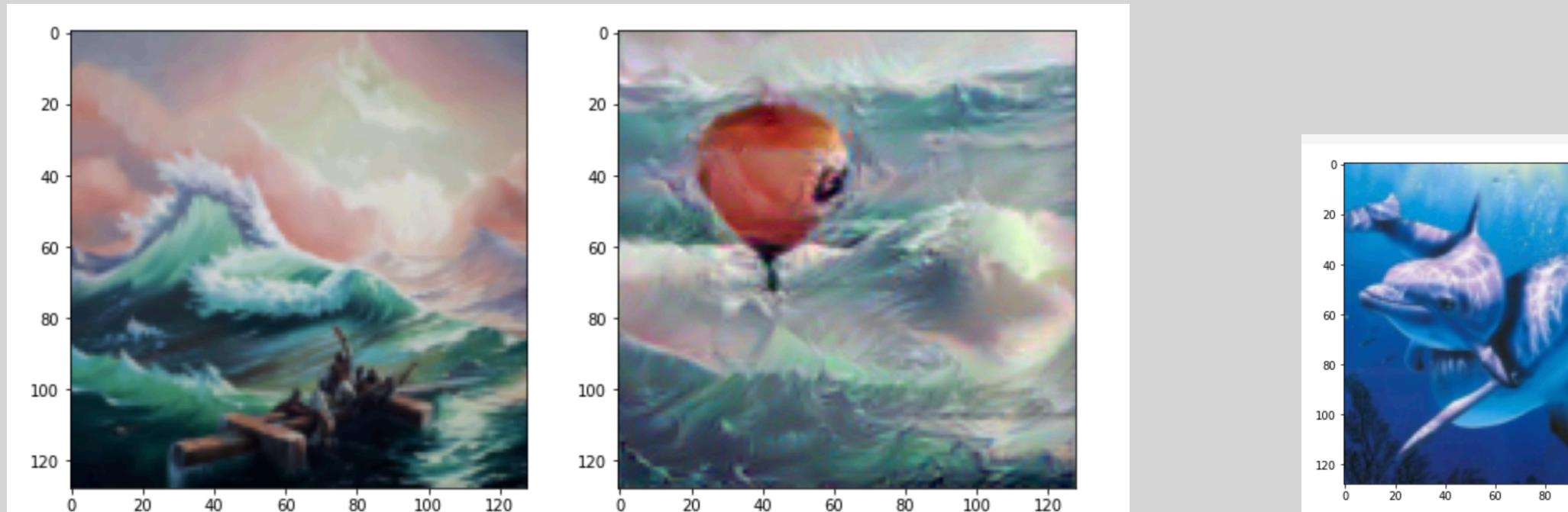
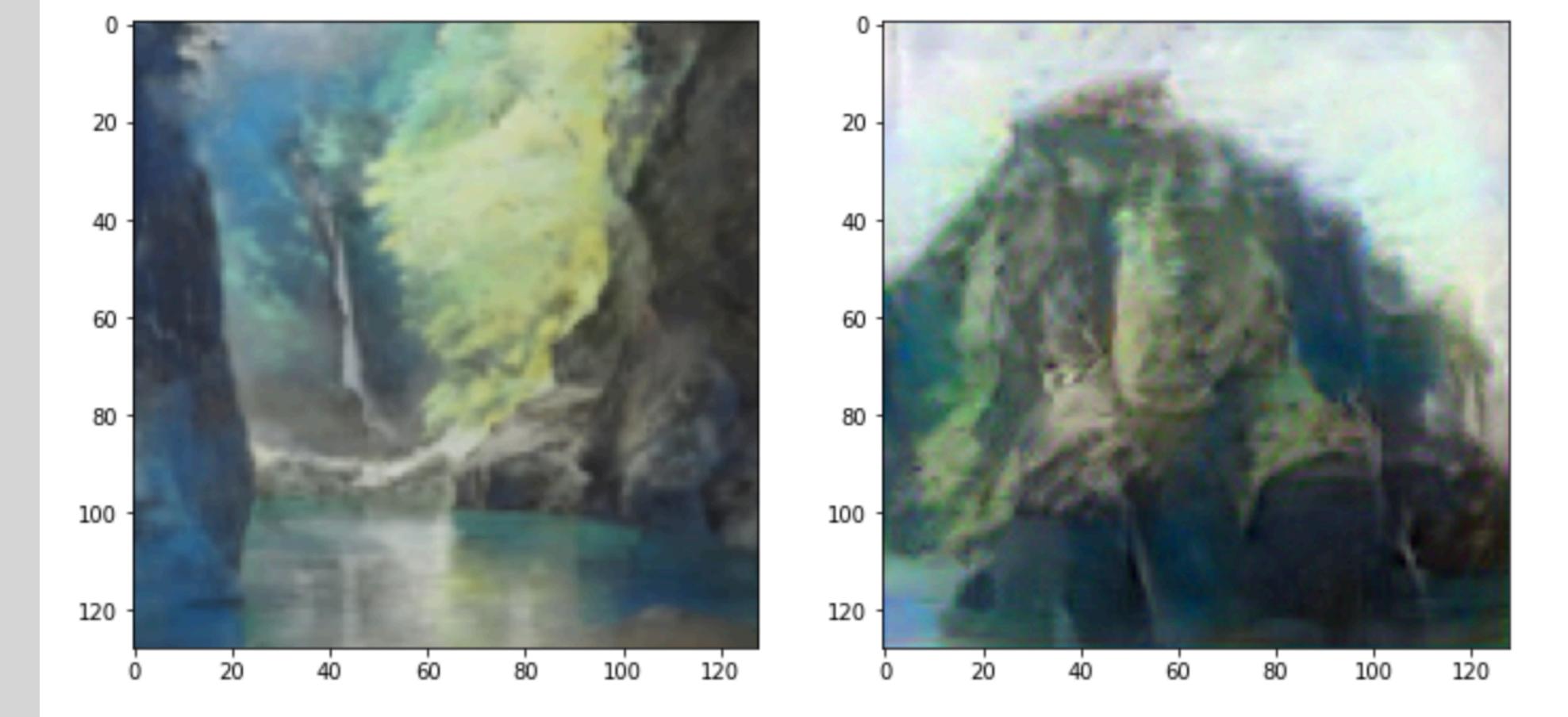
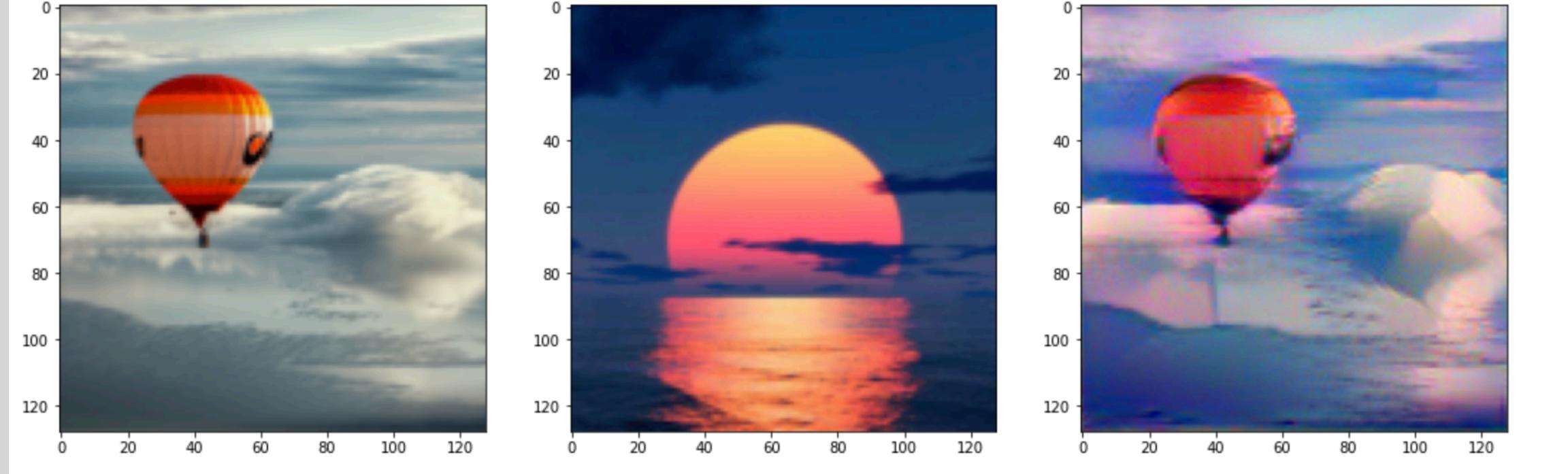
Веса и количество прогонов

```
▶ output_tensor = make_style_transfer('style7.jpg', 'style13.jpg',
                                     style_weight=10000, n_epoch=3
                                     )
```

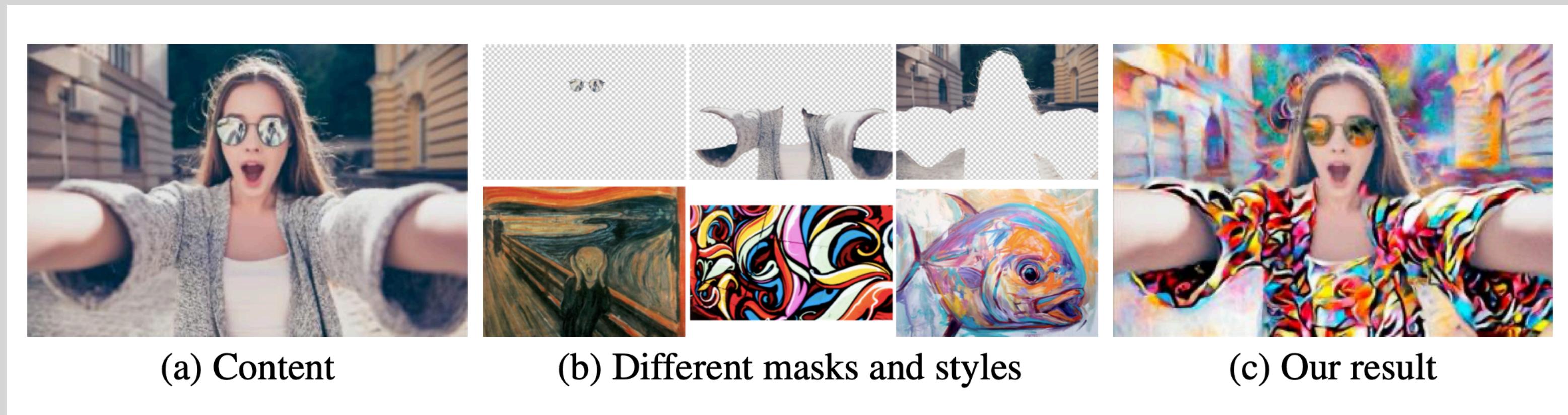


Фактически являются гиперпараметрами для каждого текстурного изображения

Я художник, я так вижу



Примеры улучшения



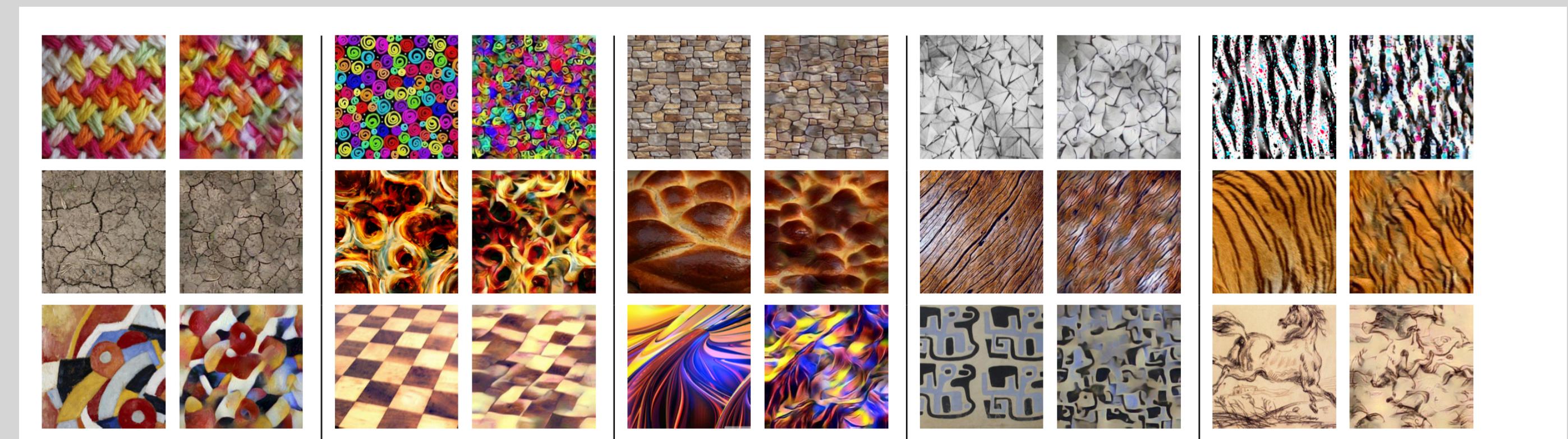
(a) Content

(b) Different masks and styles

(c) Our result

Применение нескольких текстур на разных участках изображения

Синтезирование новой текстуры из нескольких текстур



Universal Style Transfer via Feature Transforms @Adobe Research

<https://arxiv.org/abs/1705.08086v2>