IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Computational Imaging Project

Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis

Team 5-b

SOMMAIRE

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

CHAPITRE 1
Methodology Recall

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

- **Data-driven 2D Images Synthesis**

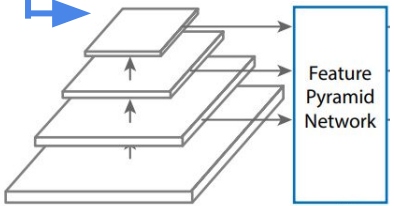
Input: example image

- Style image: describe the building blocks of the image
- Content image: constraint the layout

Markov Random Field (MRF) Models

Deep Convolutional Neural Networks (dCNNs) - VGG19

control the image layout at an abstract level



- **MRF + dCNN**

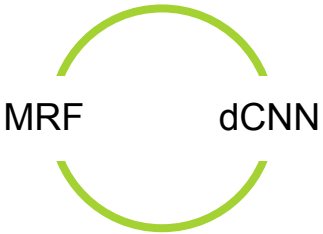
Locally correlated information

Translational invariance

[In this paper]

★ Use MRF regularizer.
★ An additional energy term to model the Markovian consistency in the upper layer of dCNN.
★ Use the EM algorithm for MRF optimization.

Improve local plausibility of the feature layouts



Match semantically related image portions without user annotations

Match and adapt local features with considerable variability

CHAPITRE 2
Changes

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

$$\mathbf{x} = \arg\min_{x} E_s(\Phi(\mathbf{x}), \Phi(\mathbf{x}_s)) + \alpha_1 E_c(\Phi(\mathbf{x}), \Phi(\mathbf{x}_c)) + \alpha_2 \Upsilon(\mathbf{x}) \quad (1)$$

```
class CNNMRF(nn.Module):
    def __init__(self, style_image, content_image, device, content_weight,\
     style_weight, tv_weight, gpu_chunck_size=256, mrf_style_stride=2,
                  mrf_synthesis_stride=2):
        super(CNNMRF, self).__init__()
        # fine tune alpha_content to interpolate between the content and the style
        self.content_weight = content_weight
        self.style_weight = style_weight
        self.tv_weight = tv_weight
        self.patch_size = 3     #Define the patch size
        self.device = device
        self.gpu_chunck_size = gpu_chunck_size
        self.mrf_style_stride = mrf_style_stride
        self.mrf_synthesis_stride = mrf_synthesis_stride
        self.style_layers = [11, 20] #Define the style layer
        self.content_layers = [22]  #Define the style layers
        self.model, self.content_losses, self.style_losses, self.tv_loss = \
            self.get_model_and_losses(style_image=style_image, content_image=content_image)
```

```python
def get_model_and_losses(self, style_image, content_image):
    """
    create network model by intermediate layer of vgg19 and some customized\
    layer(style loss, content loss and tv loss)
    :param style_image:
    :param content_image:
    :return:
    """
    vgg = models.vgg19(pretrained=True).to(self.device)
    # inception = models.inception_v3(pretrained=True).to(self.device)
    model = nn.Sequential()
    content_losses = []
    style_losses = []
    # add tv loss layer
    tv_loss = TVLoss()
    model.add_module('tv_loss', tv_loss)

    next_content_idx = 0
    next_style_idx = 0
```

```python
class TVLoss(nn.Module):
    def __init__(self):
        """
        tv loss layer
        """
        super(TVLoss, self).__init__()
        self.loss = None

    def forward(self, input):
        image = input.squeeze().permute([1, 2, 0])
        r = (image[:, :, 0] + 2.12) / 4.37
        g = (image[:, :, 1] + 2.04) / 4.46
        b = (image[:, :, 2] + 1.80) / 4.44

        temp = torch.cat([r.unsqueeze(2), g.unsqueeze(2), b.unsqueeze(2)], dim=2)
        gx = torch.cat((temp[1:, :, :], temp[-1, :, :].unsqueeze(0)), dim=0)
        gx = gx - temp

        gy = torch.cat((temp[:, 1:, :], temp[:, -1, :].unsqueeze(1)), dim=1)
        gy = gy - temp

        self.loss = torch.mean(torch.pow(gx, 2)) + torch.mean(torch.pow(gy, 2))
        return input
```

2.1 Code architecture

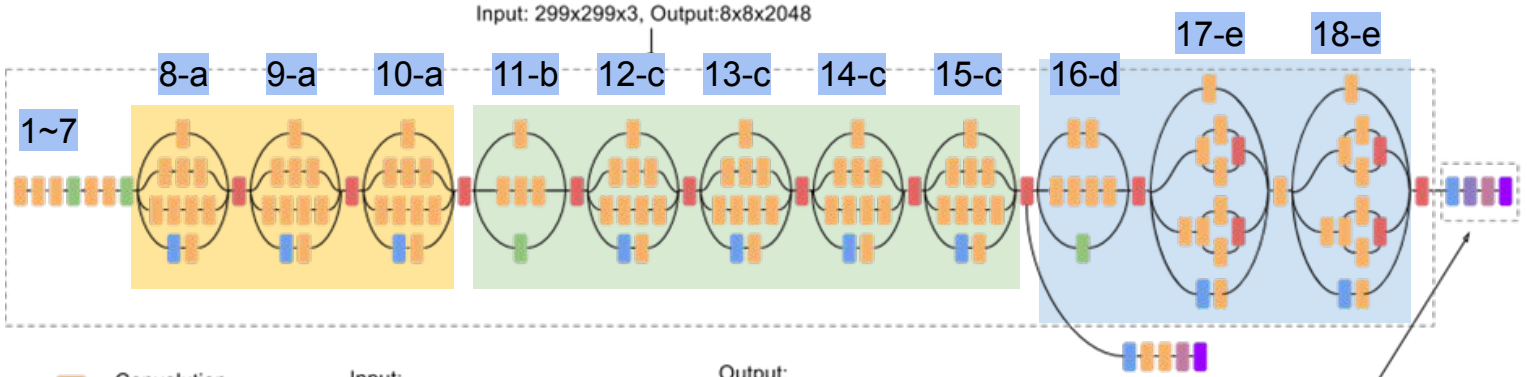
$$E_s(\Phi(\mathbf{x}), \Phi(\mathbf{x}_s)) = \sum_{i=1}^{m} ||\Psi_i(\Phi(\mathbf{x})) - \Psi_{NN(i)}(\Phi(\mathbf{x}_s))||^2$$

$$NN(i) := \arg\min_{j=1,\dots,m_s} \frac{\Psi_i(\Phi(\mathbf{x})) \cdot \Psi_j(\Phi(\mathbf{x}_s))}{|\Psi_i(\Phi(\mathbf{x}))| \cdot |\Psi_j(\Phi(\mathbf{x}_s))|} \quad (3)$$

```
    for i in range(len(vgg.features)):
        if next_content_idx >= len(self.content_layers) and next_style_idx >= len(self.style_layers):
            break
        # add layer of vgg19
        layer = vgg.features[i]
        name = str(i)
        model.add_module(name, layer)

        # add content loss layer
        if i in self.content_layers:
            target = model(content_image).detach()
            content_loss = ContentLoss(target)
            model.add_module("content_loss_{}".format(next_content_idx), content_loss)
            content_losses.append(content_loss)
            next_content_idx += 1

        # add style loss layer
        if i in self.style_layers:
            target_feature = model(style_image).detach()
            style_loss = StyleLoss(target_feature, patch_size=self.patch_size, mrf_style_stride=self.mrf_style_stride,
                                   mrf_synthesis_stride=self.mrf_synthesis_stride, gpu_chunck_size=self.gpu_chunck_size

            model.add_module("style_loss_{}".format(next_style_idx), style_loss)
            style_losses.append(style_loss)
            next_style_idx += 1

    return model, content_losses, style_losses, tv_loss
```

Extraction of features

```python
if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    parser.add_argument('--content_path', type=str, default='./data/content1.jpg')
    parser.add_argument('--style_path', type=str, default='./data/style1.jpg')
    parser.add_argument('--max_iter', type=int, default=100)
    parser.add_argument('--sample_step', type=int, default=50)
    parser.add_argument('--content_weight', type=float, default=1)
    parser.add_argument('--style_weight', type=float, default=0.4)
    parser.add_argument('--tv_weight', type=float, default=0.1)
    parser.add_argument('--num_res', type=int, default=3)
    parser.add_argument('--gpu_chunck_size', type=int, default=512)
    parser.add_argument('--mrf_style_stride', type=int, default=2)
    parser.add_argument('--mrf_synthesis_stride', type=int, default=2)
    config = parser.parse_args()
    print(config)
    main(config)
```



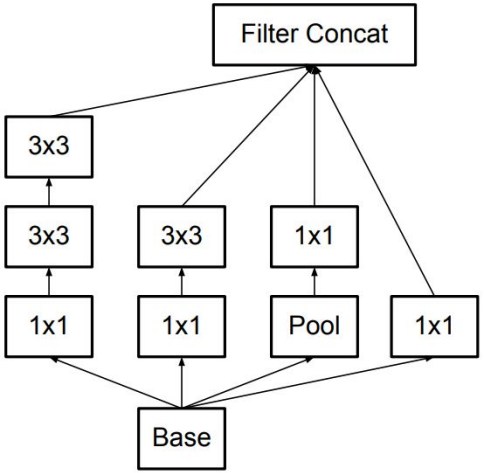
Mixed_5 Mixed_6 Mixed_7

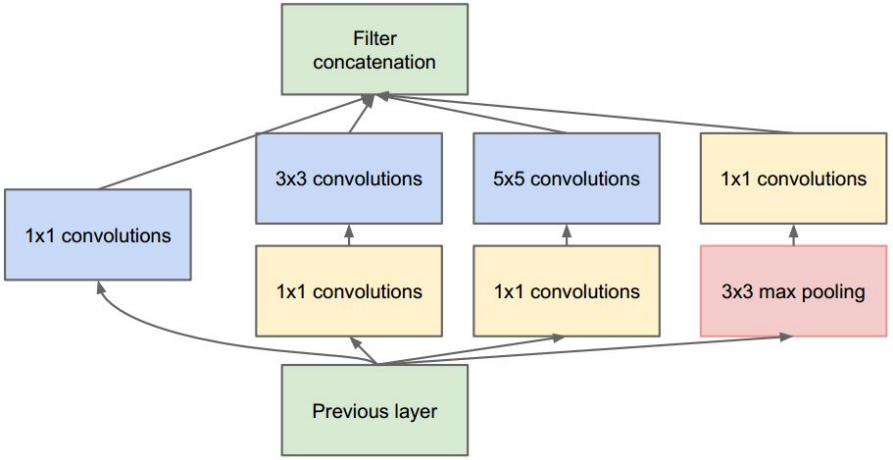
	type	patch size/stride or remarks	input size
1	conv	3×3/2	299×299×3
2	conv	3×3/1	149×149×32
3	conv padded	3×3/1	147×147×32
4	pool	3×3/2	147×147×64
5	conv	3×3/1	73×73×64
6	conv	3×3/2	71×71×80
7	conv	3×3/1	35×35×192
8~10	3×Inception	As in figure 5	35×35×288
11~15	5×Inception	As in figure 6	17×17×768
16~17	2×Inception	As in figure 7	8×8×1280
18	pool	8 × 8	8 × 8 × 2048
19	linear	logits	1 × 1 × 2048
20	softmax	classifier	1 × 1 × 1000

Inception v2

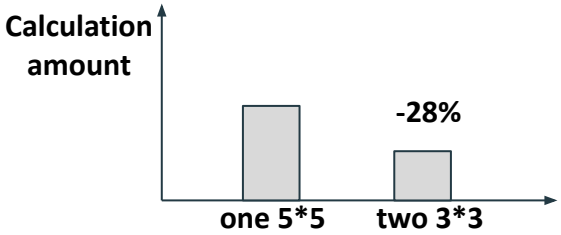
```
### outputs ###

# inception_a
[branch1x1, branch5x5, branch3x3dbl,
branch_pool]
# inception_b
outputs = [branch3x3, branch3x3dbl,
branch_pool]
# inception_c
outputs = [branch1x1, branch7x7, branch7x7dbl,
branch_pool]
# inception_d
outputs = [branch3x3, branch7x7x3, branch_pool]
# inception_e
outputs = [branch1x1, branch3x3, branch3x3dbl,
branch_pool]
```

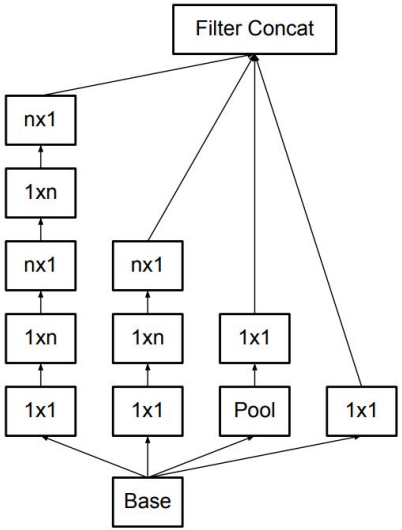
```
self.Mixed_5b = inception_a(192,
pool_features=32)
self.Mixed_5c = inception_a(256,
pool_features=64)
self.Mixed_5d = inception_a(288,
pool_features=64)

self.Mixed_6a = inception_b(288)
self.Mixed_6b = inception_c(768,
channels_7x7=128)
self.Mixed_6c = inception_c(768,
channels_7x7=160)
self.Mixed_6d = inception_c(768,
channels_7x7=160)
self.Mixed_6e = inception_c(768,
channels_7x7=192)

self.Mixed_7a = inception_d(768)
self.Mixed_7b = inception_e(1280)
self.Mixed_7c = inception_e(2048)
```


Filter Concat
3x3
3x3
3x3
1x1
1x1
1x1
Pool
1x1
Base

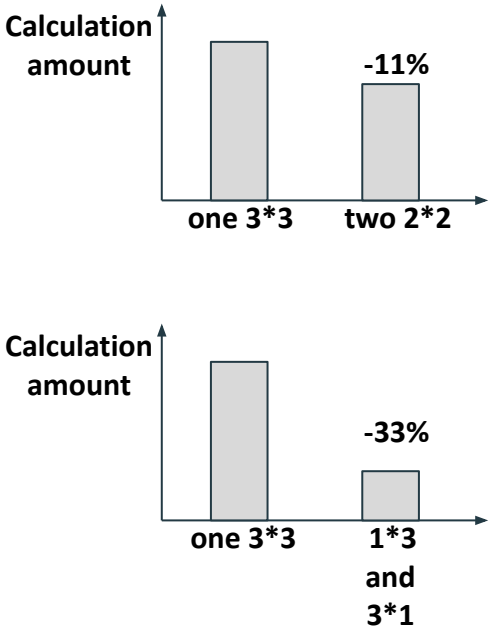

Filter concatenation
1x1 convolutions
3x3 convolutions
5x5 convolutions
1x1 convolutions
1x1 convolutions
1x1 convolutions
3x3 max pooling
Previous layer

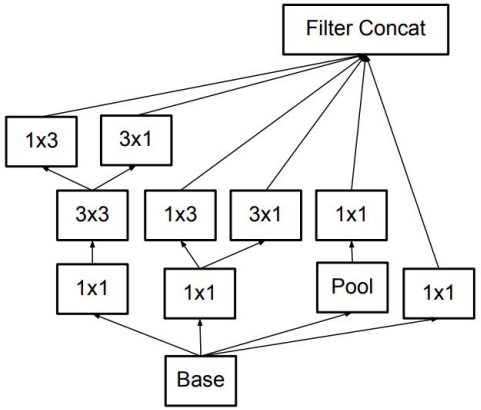
inception_a


Calculation amount
-28%
one 5*5
two 3*3



inception_c





inception_e

CHAPITRE 3
Results and Analysis

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

CHAPITRE 3 : Results and Analysis

3.1 Impact of the patch size

$$\mathbf{x} = \arg\min_{x} E_s(\Phi(\mathbf{x}), \Phi(\mathbf{x}_s)) + \alpha_1 E_c(\Phi(\mathbf{x}), \Phi(\mathbf{x}_c)) + \alpha_2 \Upsilon(\mathbf{x}) \quad (1)$$

$$E_s(\Phi(\mathbf{x}), \Phi(\mathbf{x}_s)) = \sum_{i=1}^{m} ||\Psi_i(\Phi(\mathbf{x})) - \Psi_{NN(i)}(\Phi(\mathbf{x}_s))||^2$$

$$NN(i) := \arg\min_{j=1,\dots,m_s} \frac{\Psi_i(\Phi(\mathbf{x})) \cdot \Psi_j(\Phi(\mathbf{x}_s))}{|\Psi_i(\Phi(\mathbf{x}))| \cdot |\Psi_j(\Phi(\mathbf{x}_s))|} \quad (3)$$

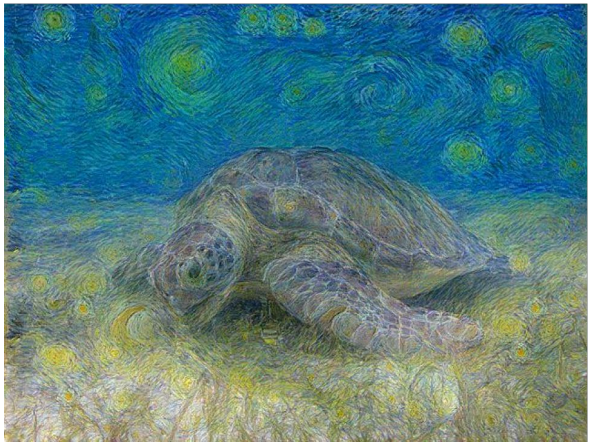
content image

style image

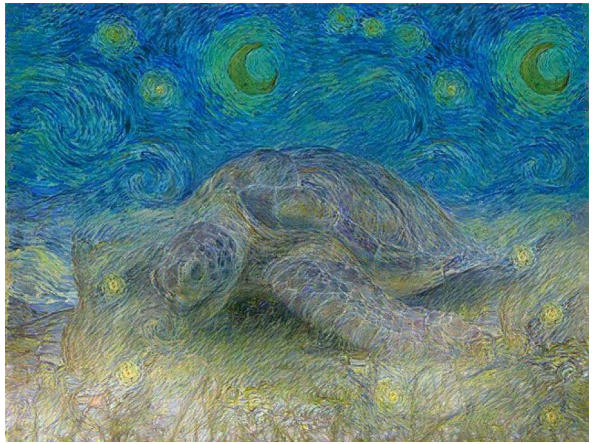
+VGG19

Parameters:
Style loss : Relu 3-1, Relu 4-1; weight =0.4
Content loss : Relu 4-2 ;weight = 1
200 iterations

3.1 Impact of the patch size



patch size = 3
style stride = 2
synthesis stride = 2



patch size = 5
style stride = 2
synthesis stride = 2

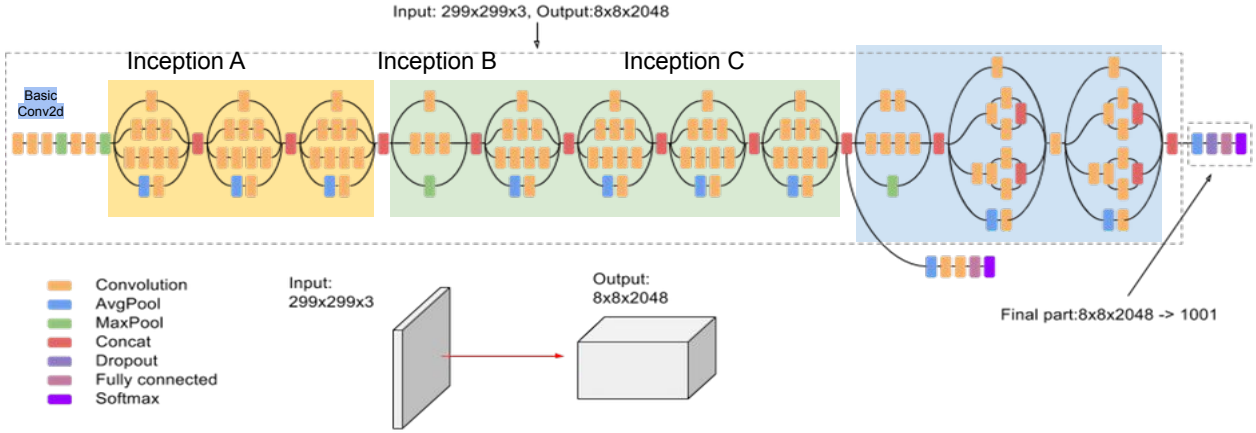
patch size = 7
style stride = 2
synthesis stride = 2

Style Layers:

Basic Conv2d + Inception A
Inception A + Inception A
Inception A + Inception B
Inception B + Inception C
Inception C + Inception C

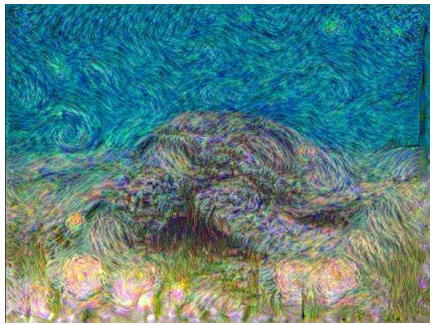
Content Layer:

The deeper layer

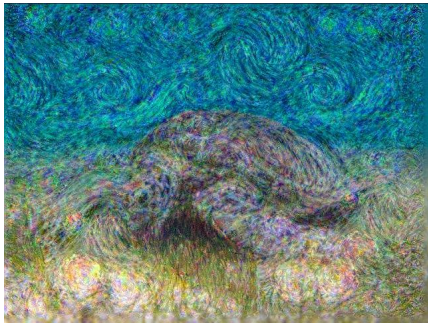


3.3 Results

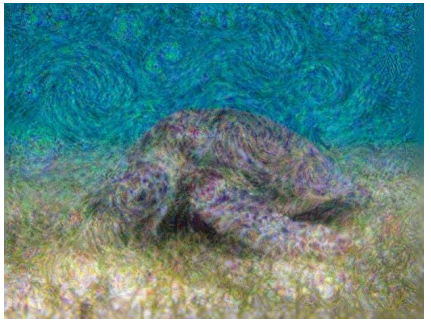
For fixed content layer = 14, content_weight = 1, style_weight = 0.3
Change style layers:



Basis Conv2d + Inception A（layer 5 & 7)



Inception A+Inception A（layer 7 & 9)



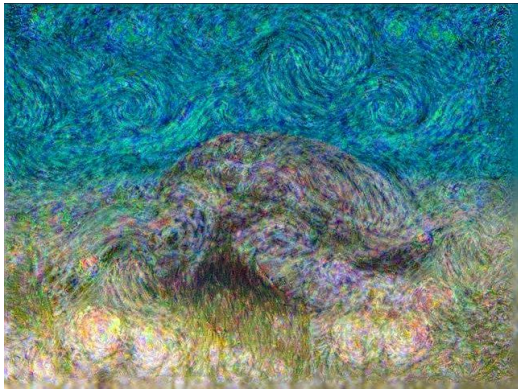
Inception A+Inception B(layer 9 & 10)

Inception B+Inception C(layer 10 & 11)

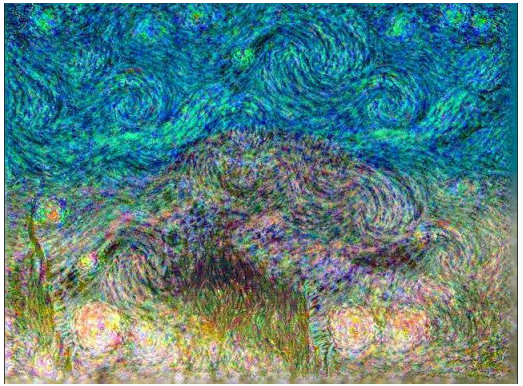
Inception B+Inception C(layer 11 & 13)

For fixed content layer = 14, content_weight = 1, style layer = 7 & 9
Change style_weight:

weight = 0.1



weight = 0.3

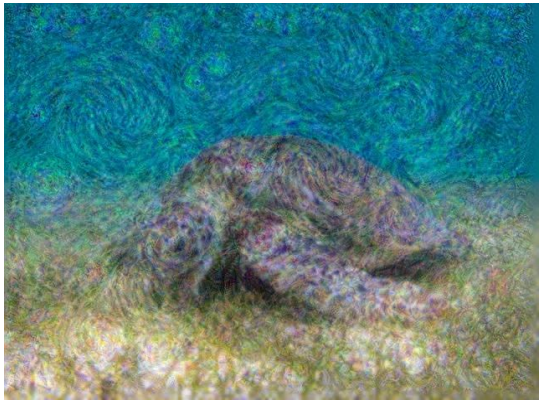


weight = 0.5

For fixed style layers = 7 & 9, style_weight = 0.3, content_weight = 1,
Change content layer:

Layer 12

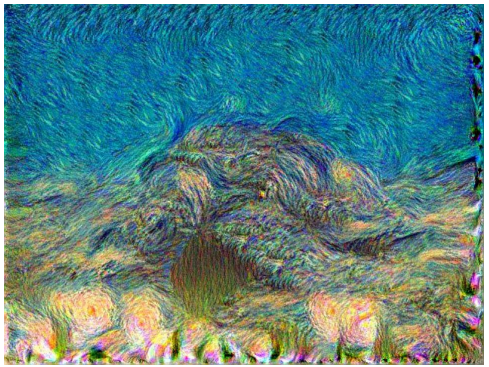
Layer 13



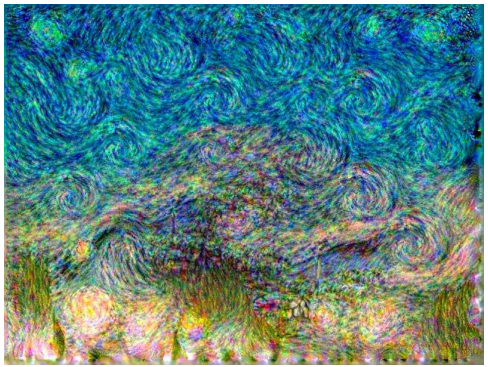
Layer 14

Changing the style layer to the shallow layers

Style loss :
layer 2+ layer 1
Content loss :
layer 14

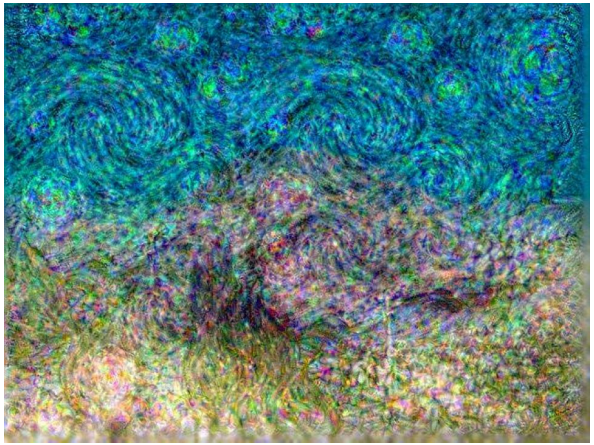


Style loss :
layer 5+ layer 4
Content loss :
layer 14

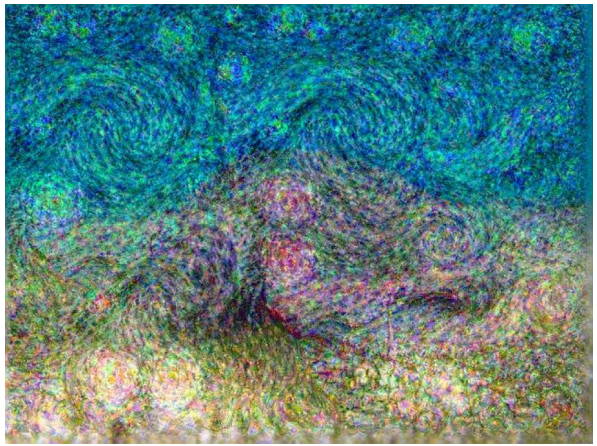


Style loss :
layer 7+ layer 8
Content loss :
layer 14

Use more (less) layers to calculate the style (content) loss



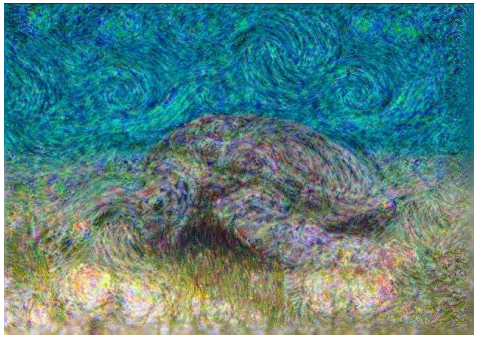
Style loss :
layer 9+ layer 10+layer11
Content loss :
layer 14



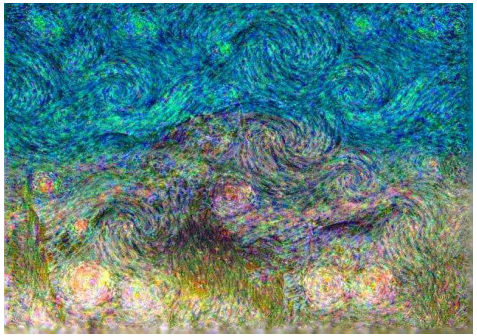
Style loss :
layer 10+layer11
Content loss :
layer 14

Style loss :
layer 9
Content loss :
layer 14

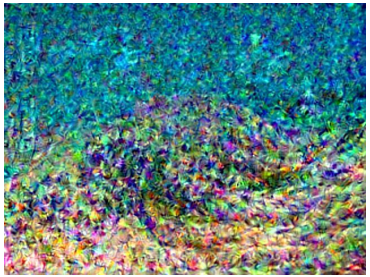
Change the number of iteration and number of resolution:



iter = 100, res = 3

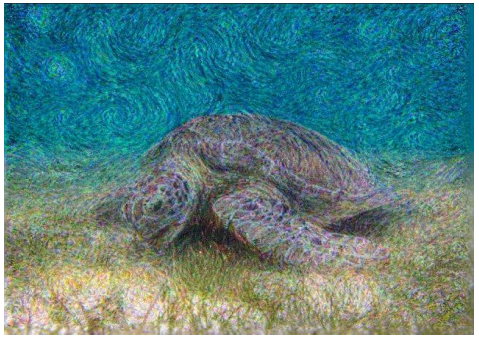


iter = 200, res = 3

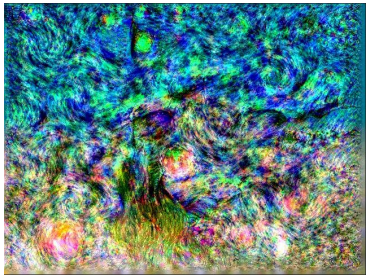


iter = 200 from group 5a

iter = 100, res = 2

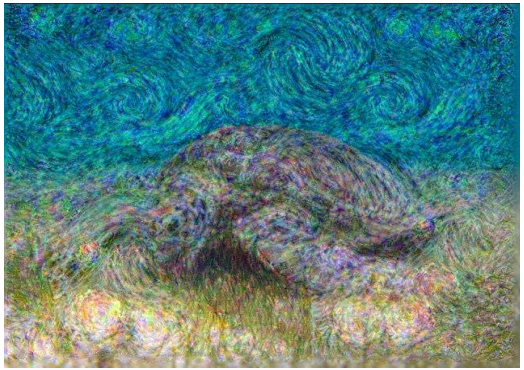


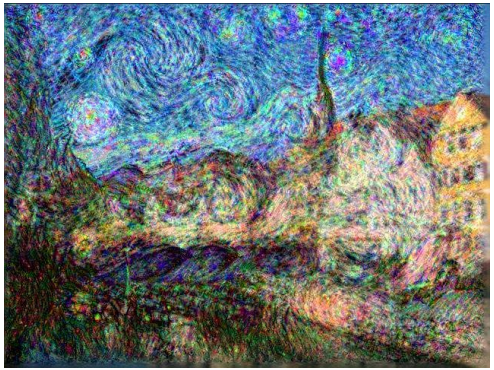
iter = 200, res = 2



iter = 100, res = 4

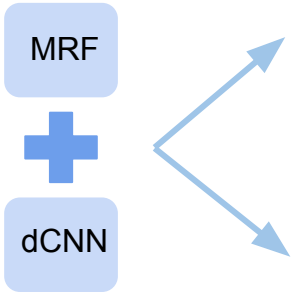
For different content images and different style images





CHAPITRE 4
Conclusion

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



VGG19 → Inception v3

Parameters → layers, weights, iterations, resolutions, …

❓ Method of evaluation

To go further

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Thank you for your attention

Team 5-b