AI VIETNAM
All-in-One Course

Style Transfer

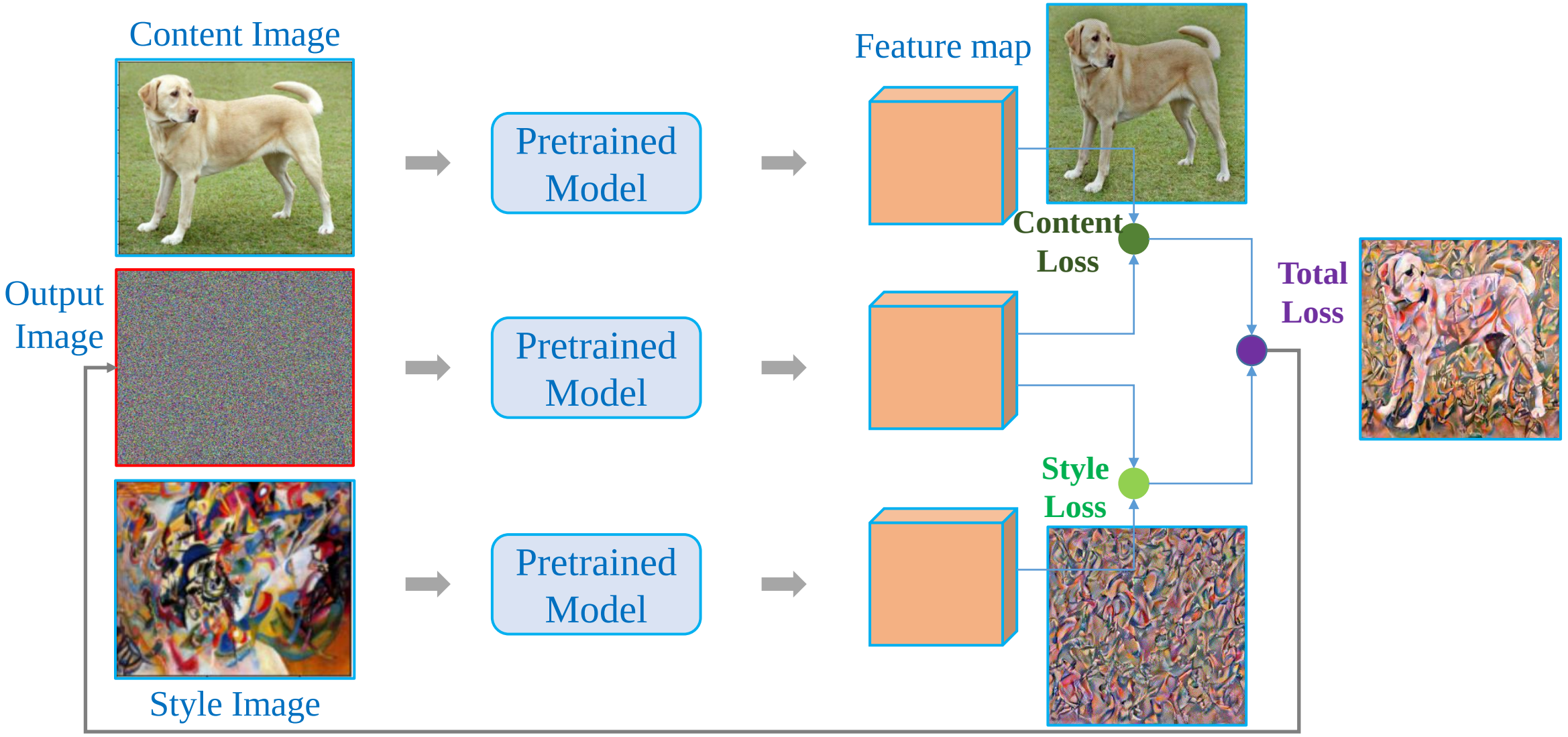
Beyond the content and style losses

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2024

Objectives

- ✓ Study about content loss + style loss
- ✓ Implementation of a simple method



Outline

- Quick Review
- Content Loss+Style Loss
- Implementation
- Einstein Summation
- Extensions

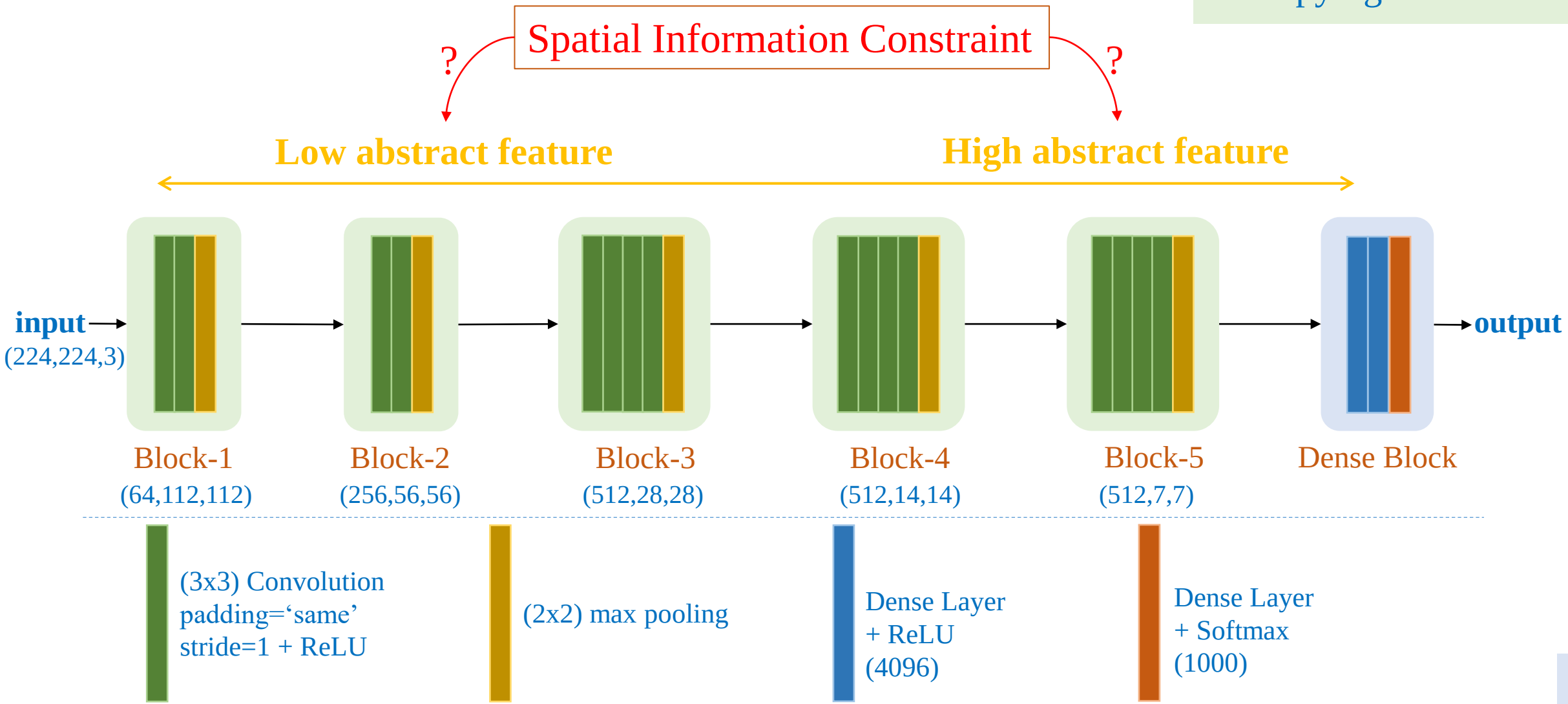
Review

Local and Global

Detection Problem

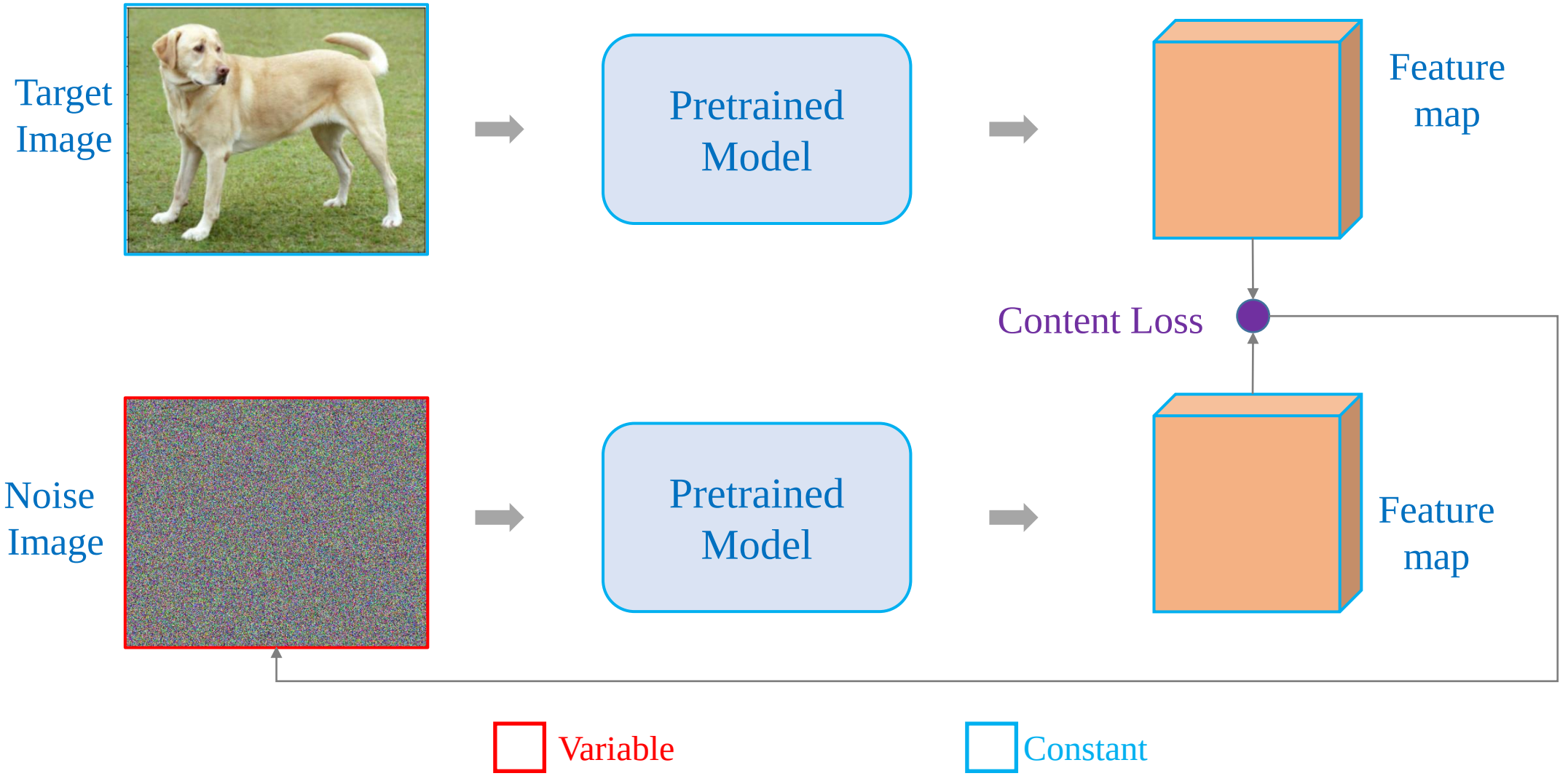
Copying Problem

❖ Pretrained model (VGG19)



Review

❖ Content Loss



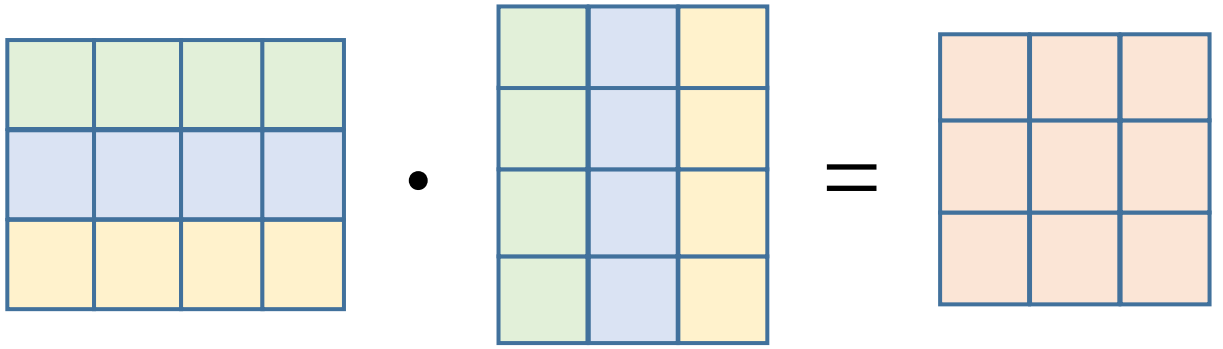
Review

❖ Gram Matrix

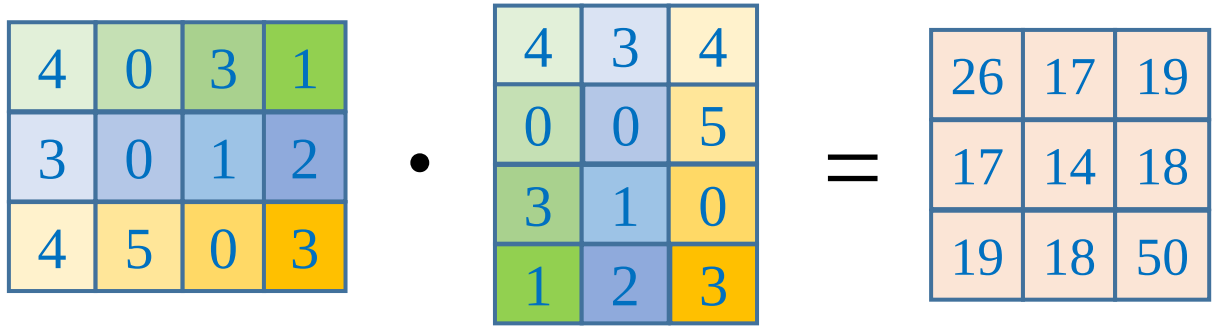
$$G_{ij} = \sum_{k} V_{ik} V_{jk}$$

```
import torch

x = torch.Tensor([[4, 0, 3, 1],
                  [3, 0, 1, 2],
                  [4, 5, 0, 3]])
G = torch.mm(x, x.t())
print(G)
```

```
tensor([[26., 17., 19.],
        [17., 14., 18.],
        [19., 18., 50.]])
```



V V^T Gram matrix

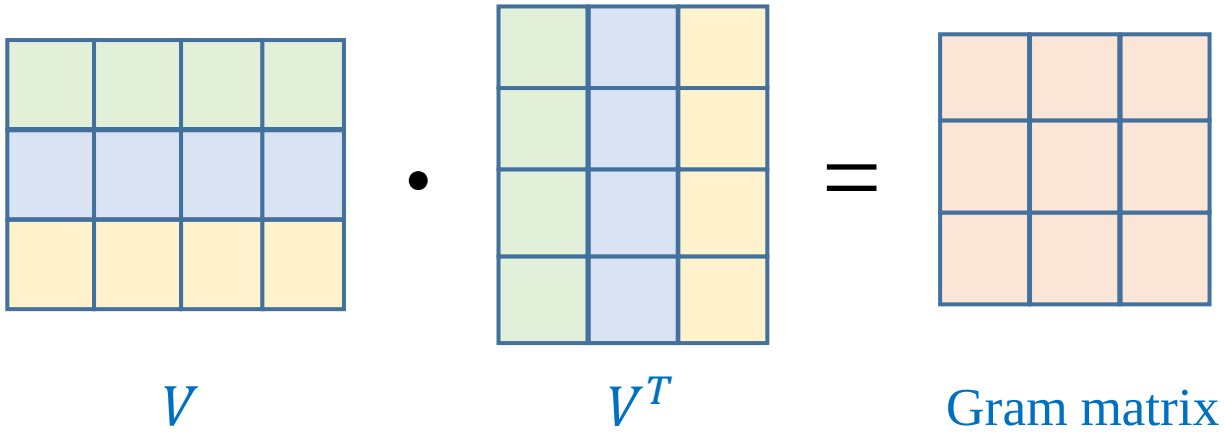


V V^T Gram matrix

Review

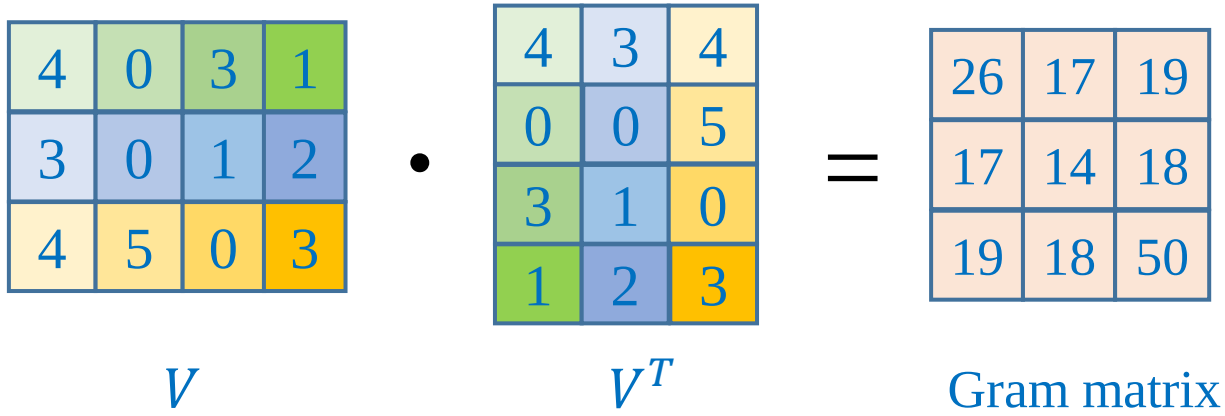
❖ Gram Matrix

$$G_{ij} = \sum_k V_{ik} V_{jk}$$




```
import torch

x = torch.Tensor([[4, 0, 3, 1],
                  [3, 0, 1, 2],
                  [4, 5, 0, 3]])
G = torch.einsum('ij,jk->ik', x, x.t())
print(G)
```

```
tensor([[26., 17., 19.],
        [17., 14., 18.],
        [19., 18., 50.]])
```



Gram Matrix

$$G_{ij} = \sum_k V_{ik} V_{jk}$$

4	0	3	1
3	0	1	2
4	5	0	3

V

$\bullet$

4	3	4
0	0	5
3	1	0
1	2	3

V^T

$=$

26	17	19
17	14	18
19	18	50

Gram matrix

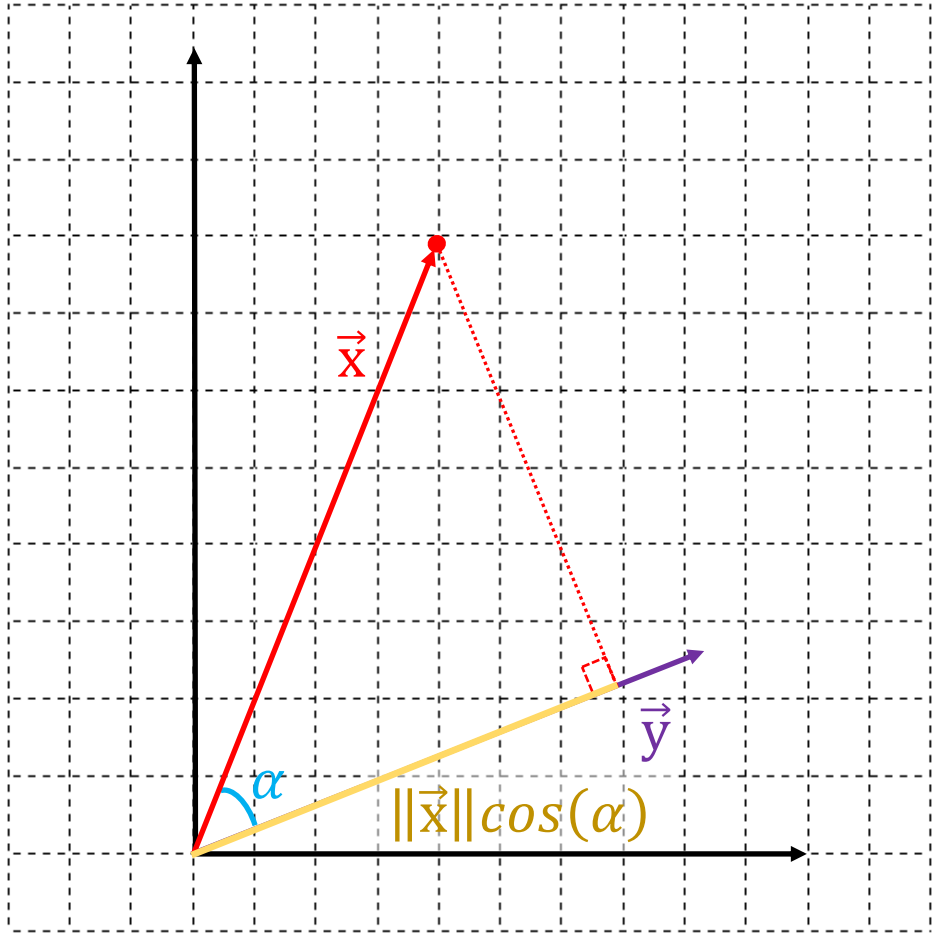
Inner Product

Algebra	Geometry
$\vec{x}.\vec{y} = \sum_1^n x_i y_i$	$\vec{x}.\vec{y} = \|\vec{x}\|\|\vec{y}\|cos(\alpha)$

4	0	3	1	$\bullet$	4	0	3	1	=	26

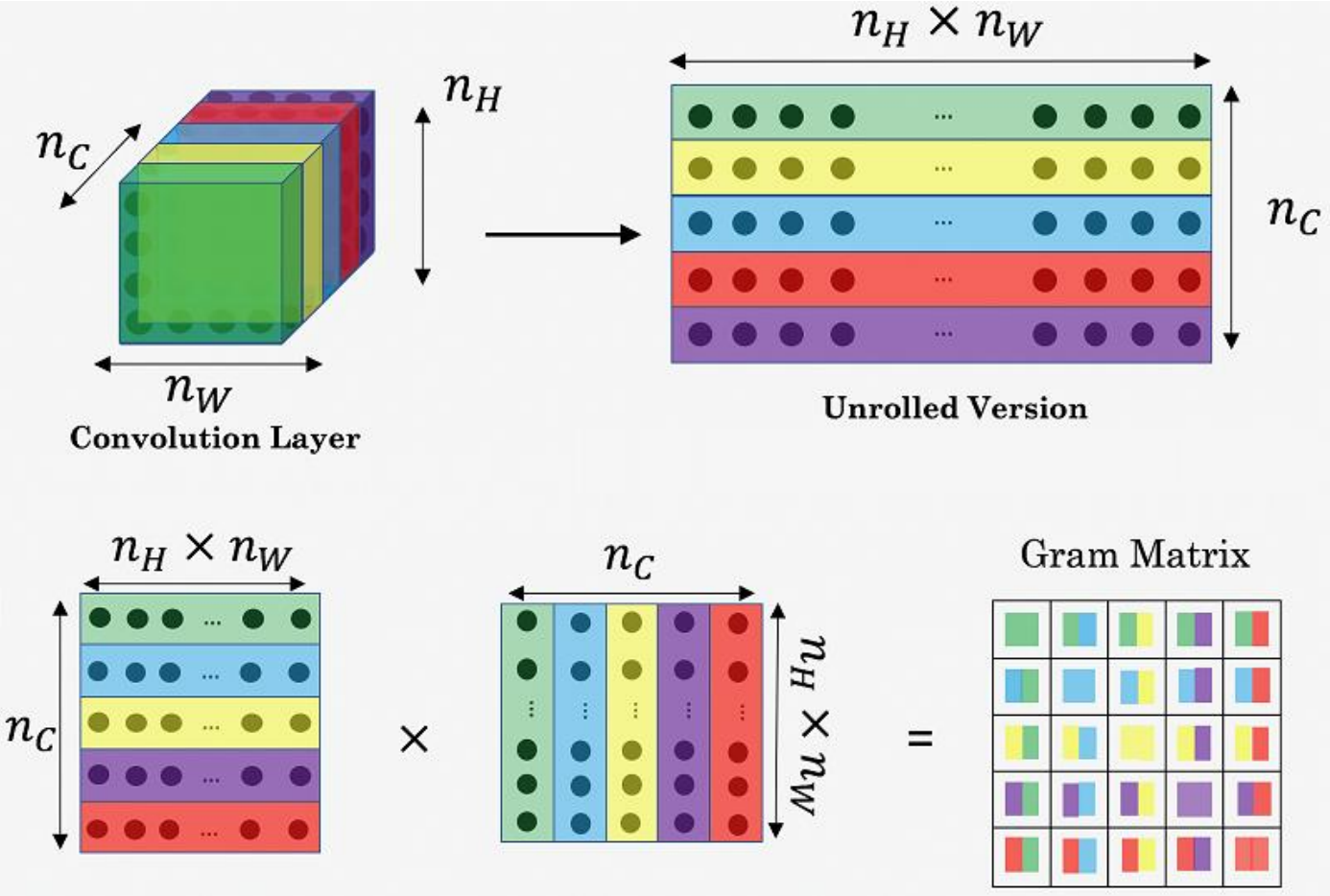
3	0	4	1	$\bullet$	3	0	4	1	=	26

Position-invariant measure

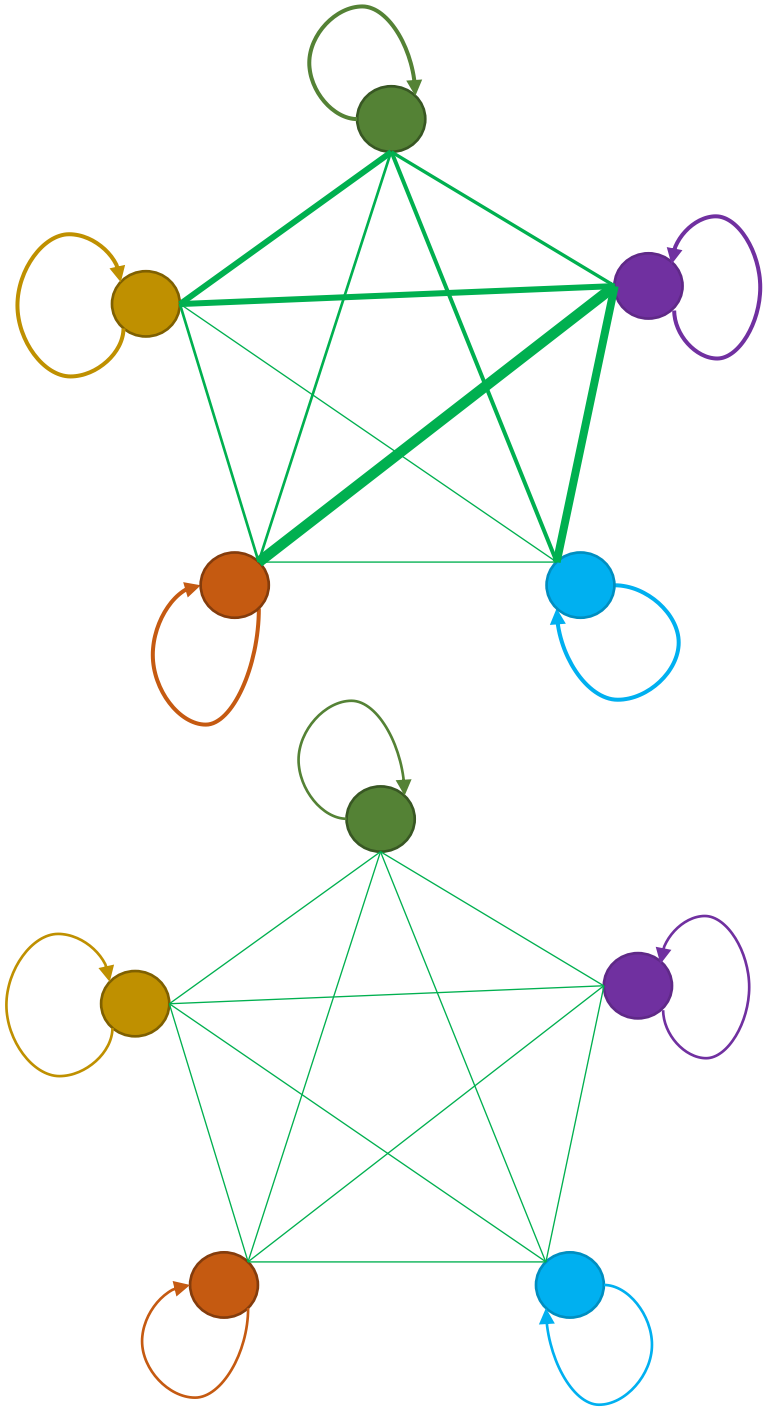


Review

❖ Gram matrix

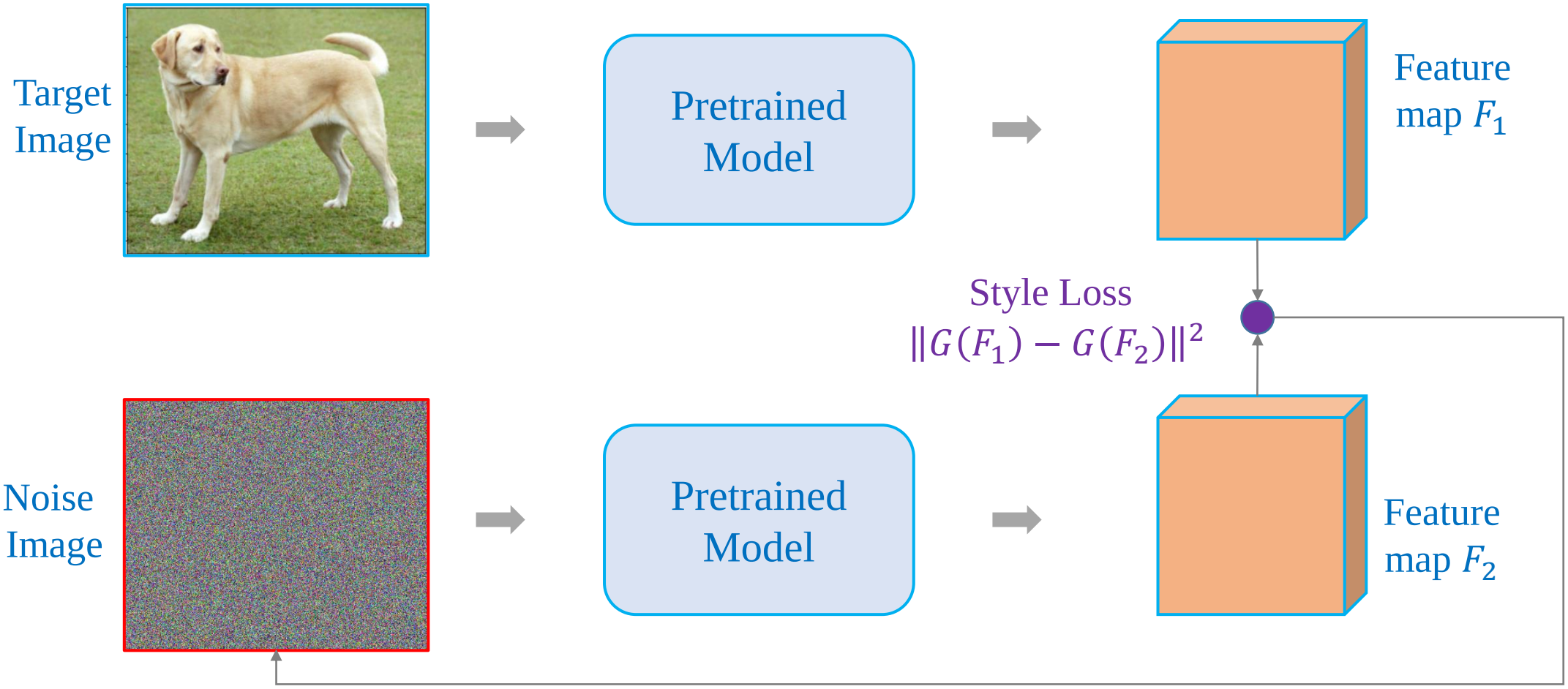


https://pytorchtaipei.github.io/articles/PyTorchTP-Style-Transfer/



Review

❖ Style Loss

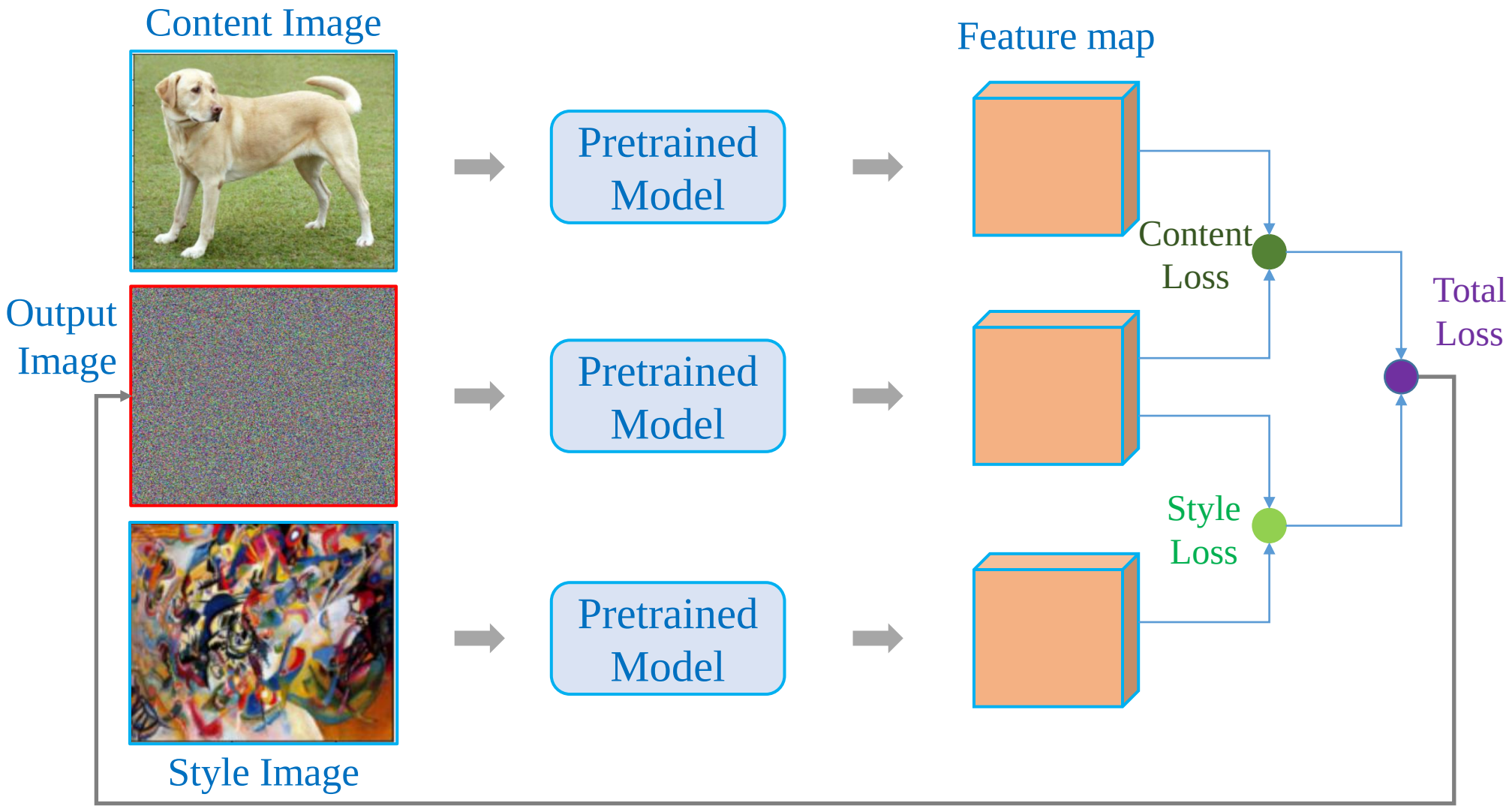


Outline

- Quick Review
- Content Loss+Style Loss
- Implementation
- Einstein Summation
- Extensions

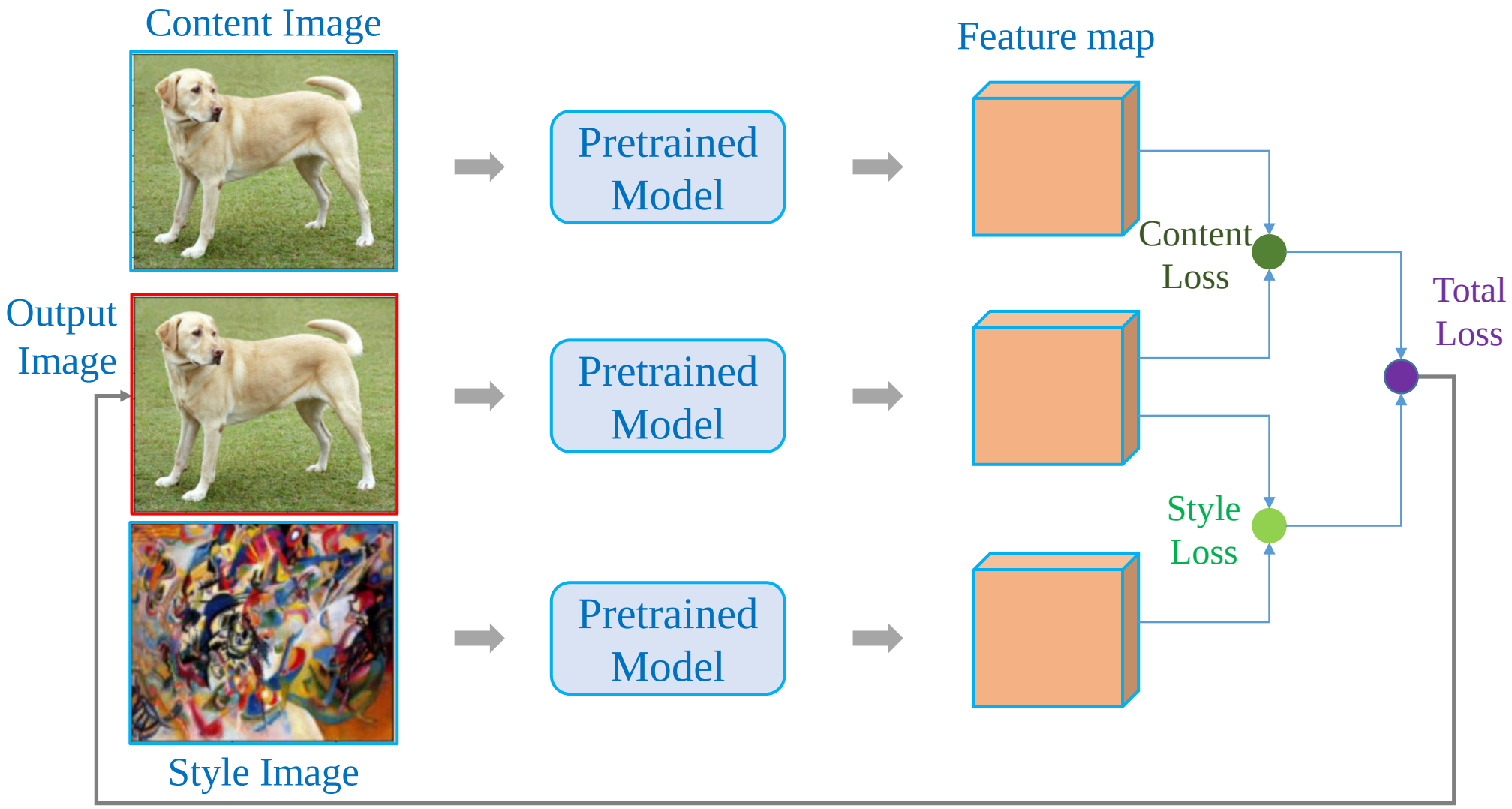
Style Transfer

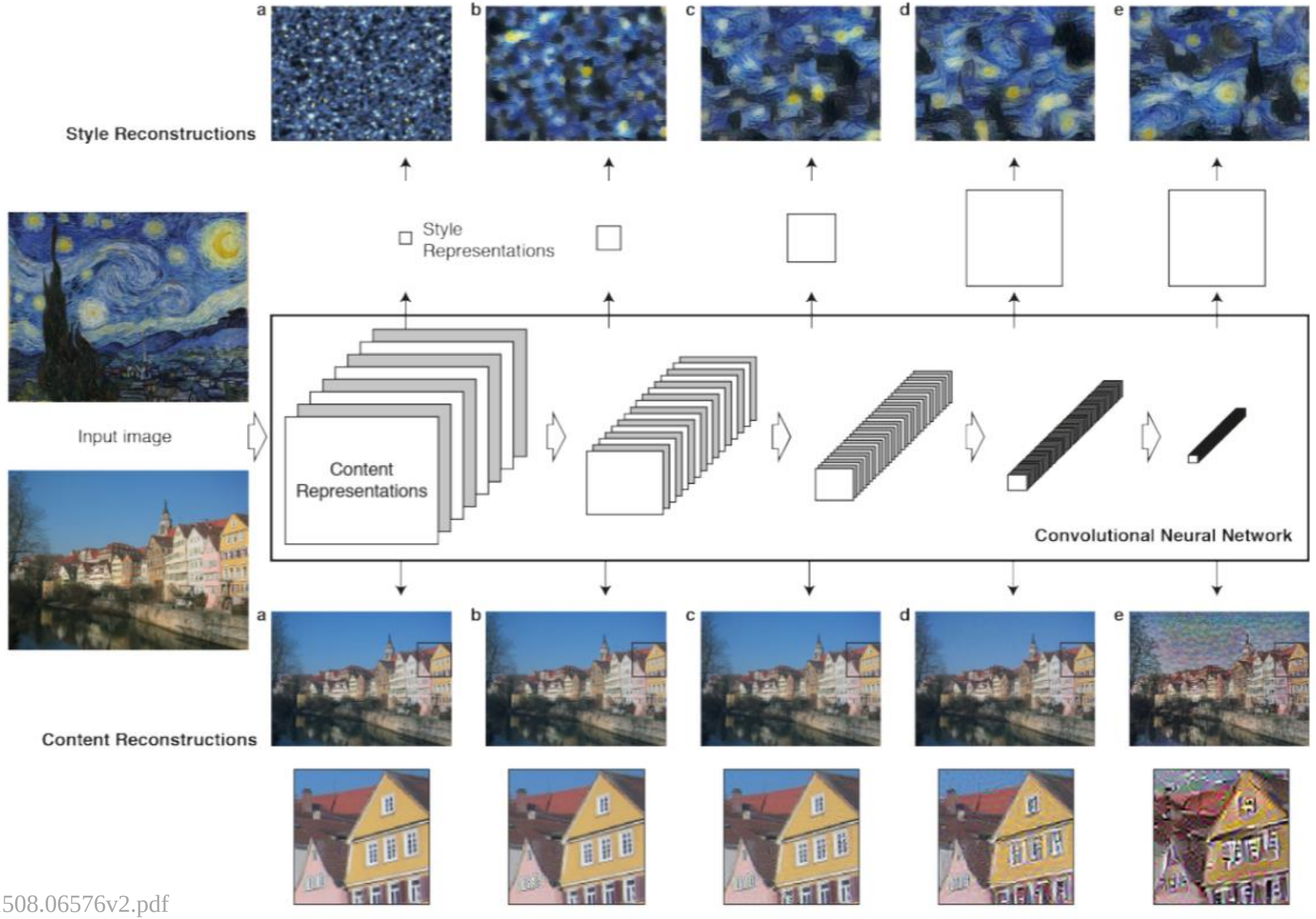
❖ Content Loss + Style Loss



Style Transfer

❖ **Using the content image as initialization for the output image**



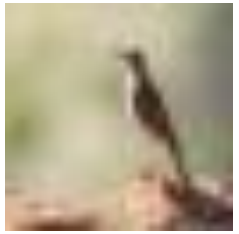


https://arxiv.org/pdf/1508.06576v2.pdf

Style Transfer

❖ Example

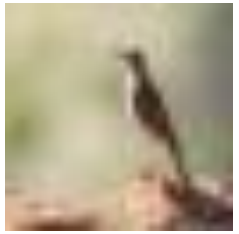
Content image



Shape=(3, 32, 32)

$$\begin{bmatrix} 212 & 215 & \cdots & 128 & 120 \\ 214 & 217 & \cdots & 123 & 116 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 223 & 214 & \cdots & 66 & 79 \\ 232 & 195 & \cdots & 102 & 94 \end{bmatrix} \begin{bmatrix} 204 & 207 & \cdots & 131 & 123 \\ 206 & 209 & \cdots & 126 & 119 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 182 & 173 & \cdots & 45 & 55 \\ 189 & 152 & \cdots & 82 & 72 \end{bmatrix} \begin{bmatrix} 168 & 171 & \cdots & 112 & 104 \\ 170 & 172 & \cdots & 107 & 100 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 150 & 141 & \cdots & 44 & 55 \\ 155 & 118 & \cdots & 83 & 74 \end{bmatrix}$$

Output image



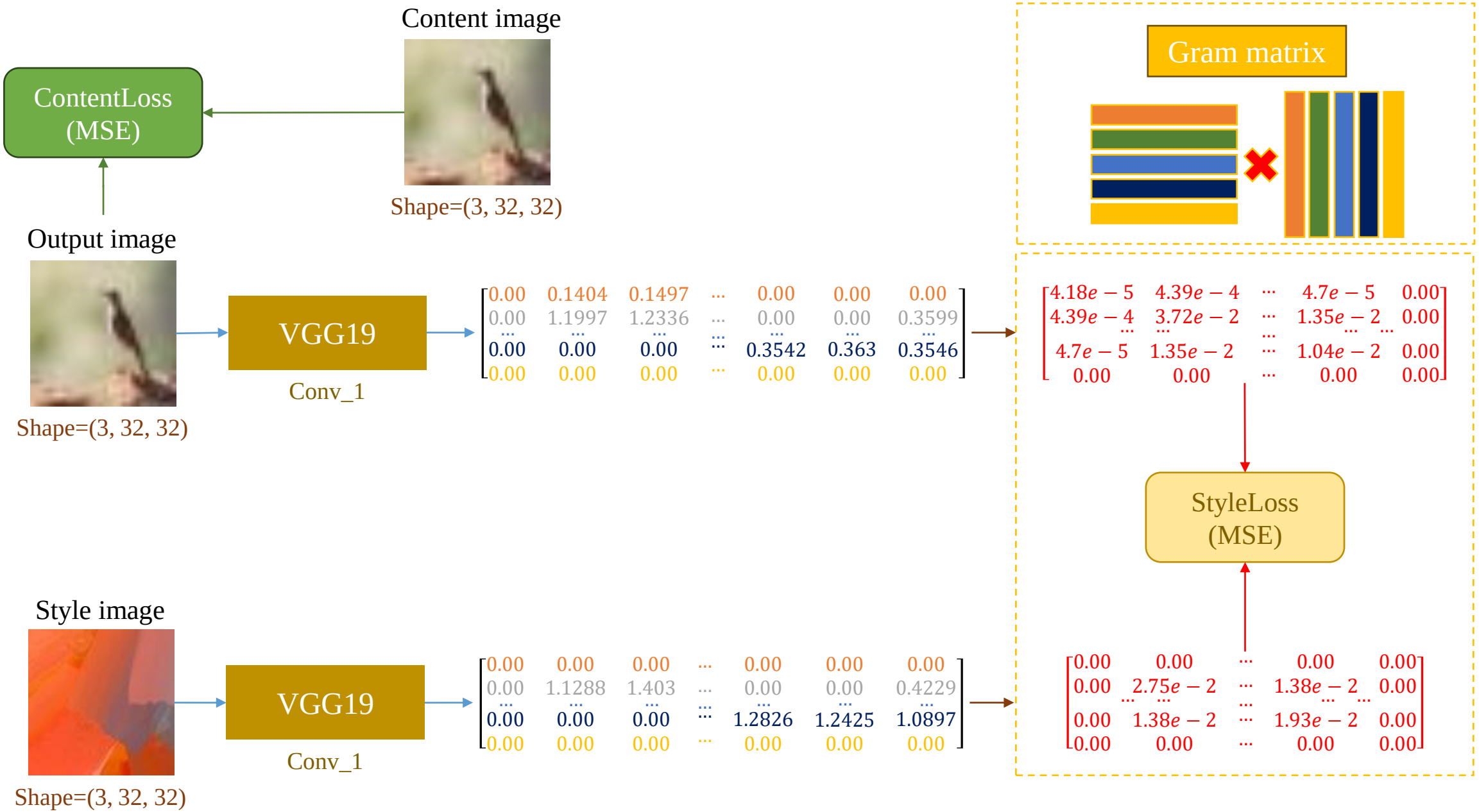
Shape=(3, 32, 32)

$$\begin{bmatrix} 212 & 215 & \cdots & 128 & 120 \\ 214 & 217 & \cdots & 123 & 116 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 223 & 214 & \cdots & 66 & 79 \\ 232 & 195 & \cdots & 102 & 94 \end{bmatrix} \begin{bmatrix} 204 & 207 & \cdots & 131 & 123 \\ 206 & 209 & \cdots & 126 & 119 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 182 & 173 & \cdots & 45 & 55 \\ 189 & 152 & \cdots & 82 & 72 \end{bmatrix} \begin{bmatrix} 168 & 171 & \cdots & 112 & 104 \\ 170 & 172 & \cdots & 107 & 100 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 150 & 141 & \cdots & 44 & 55 \\ 155 & 118 & \cdots & 83 & 74 \end{bmatrix}$$

Style image

Shape=(3, 32, 32)

$$\begin{bmatrix} 251 & 253 & \cdots & 197 & 193 \\ 253 & 254 & \cdots & 197 & 192 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 252 & 254 & \cdots & 183 & 181 \\ 253 & 253 & \cdots & 184 & 182 \end{bmatrix} \begin{bmatrix} 123 & 124 & \cdots & 111 & 113 \\ 121 & 123 & \cdots & 111 & 113 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 85 & 87 & \cdots & 65 & 66 \\ 86 & 86 & \cdots & 64 & 65 \end{bmatrix} \begin{bmatrix} 72 & 75 & \cdots & 102 & 106 \\ 72 & 75 & \cdots & 104 & 107 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 43 & 45 & \cdots & 24 & 28 \\ 44 & 44 & \cdots & 22 & 26 \end{bmatrix}$$


Content image
ContentLoss (MSE)
Shape=(3, 32, 32)
Output image
Shape=(3, 32, 32)
VGG19
Conv_1
Gram matrix
StyleLoss (MSE)
Style image
Shape=(3, 32, 32)
VGG19
Conv_1

ContentLoss (MSE) × ContentWeight (1)

StyleLoss (MSE) × StyleWeight (1e6)

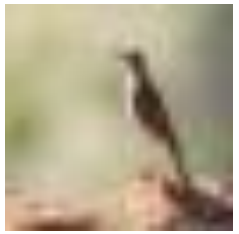
→ TotalLoss (L)

Note: the pixel value of the ouput image when training will be normalized, not in uint8 type.

Update Parameters ($lr = 0.05$)

$Target = \text{Target} - lr \times dL$

Output image

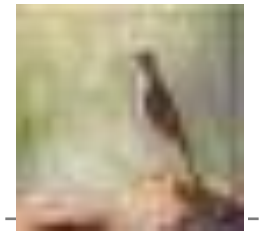


Shape=(3, 32, 32)

$$\begin{bmatrix} 212 & 215 & \cdots & 128 & 120 \\ 214 & 217 & \cdots & 123 & 116 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 223 & 214 & \cdots & 66 & 79 \\ 232 & 195 & \cdots & 102 & 94 \end{bmatrix} - 0.05 \times \begin{bmatrix} 0.0231 & -0.0083 & \cdots & -0.037 & -0.1309 \\ 0.0434 & 0.0678 & \cdots & 0.0996 & -0.0534 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -0.017 & -0.0192 & \cdots & 0.0358 & -0.0135 \\ 0.0175 & 0.0347 & \cdots & 0.1084 & 0.0578 \end{bmatrix}$$

$$\begin{bmatrix} 204 & 207 & \cdots & 131 & 123 \\ 206 & 209 & \cdots & 126 & 119 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 182 & 173 & \cdots & 45 & 55 \\ 189 & 152 & \cdots & 82 & 72 \end{bmatrix} - 0.05 \times \begin{bmatrix} 0.0407 & 0.0107 & \cdots & -0.0027 & -0.1198 \\ 0.0652 & 0.01011 & \cdots & 0.1118 & -0.0578 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.022 & 0.0262 & \cdots & 0.0299 & -0.0281 \\ 0.0164 & 0.0203 & \cdots & 0.0662 & 0.0242 \end{bmatrix}$$

$$\begin{bmatrix} 168 & 171 & \cdots & 112 & 104 \\ 170 & 172 & \cdots & 107 & 100 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 150 & 141 & \cdots & 44 & 55 \\ 155 & 118 & \cdots & 83 & 74 \end{bmatrix} - 0.05 \times \begin{bmatrix} 0.0245 & 0.0201 & \cdots & 0.0312 & -0.0136 \\ 0.0283 & 0.0521 & \cdots & 0.04 & -0.003 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0.0382 & 0.0378 & \cdots & 0.0021 & -0.0126 \\ -0.0006 & -0.009 & \cdots & -0.021 & -0.0168 \end{bmatrix}$$



$$\begin{bmatrix} 199 & 227 & \cdots & 140 & 132 \\ 201 & 204 & \cdots & 110 & 128 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 235 & 226 & \cdots & 53 & 91 \\ 219 & 182 & \cdots & 89 & 81 \end{bmatrix}$$

$$\begin{bmatrix} 191 & 194 & \cdots & 143 & 135 \\ 193 & 196 & \cdots & 113 & 131 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 169 & 160 & \cdots & 32 & 67 \\ 176 & 139 & \cdots & 69 & 59 \end{bmatrix}$$

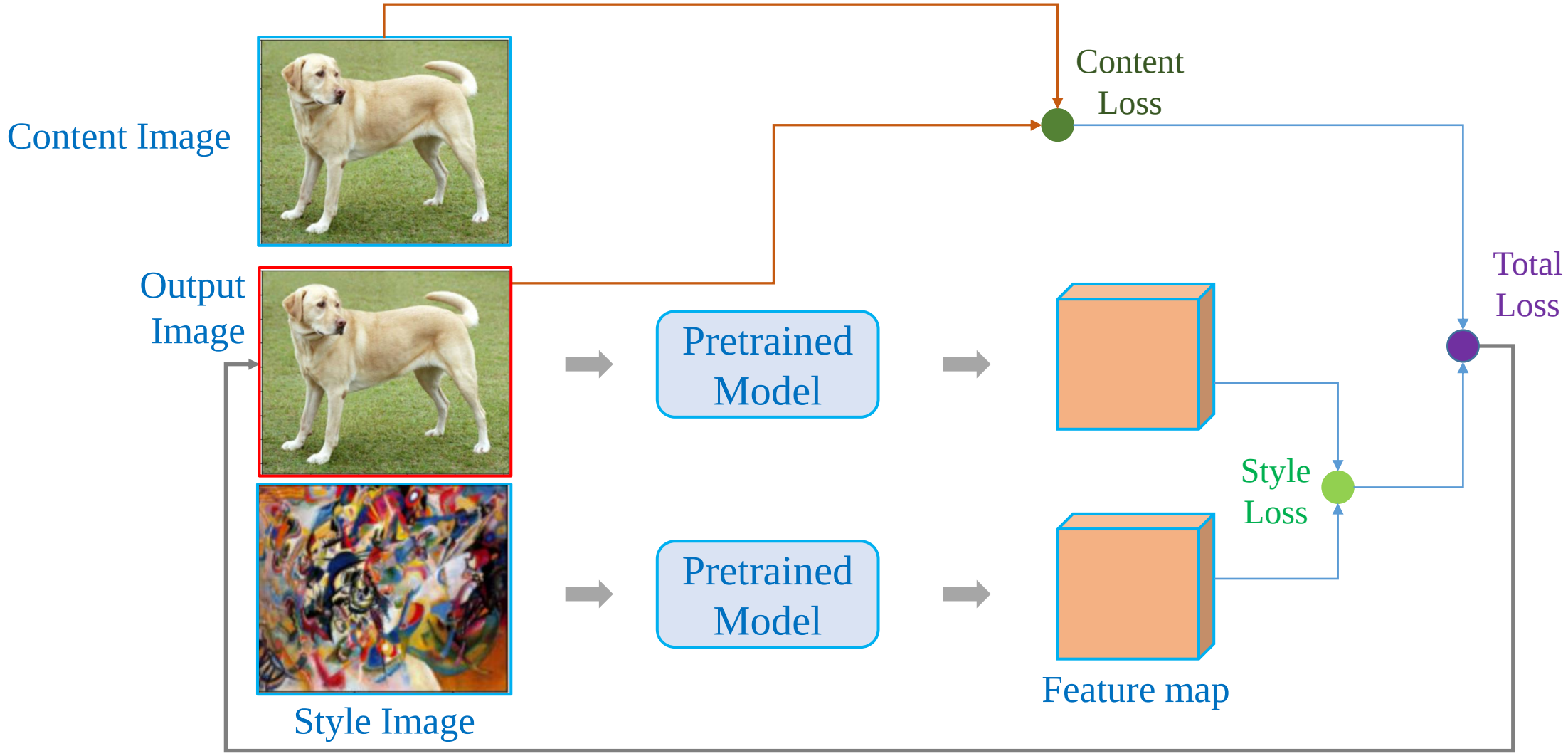
$$\begin{bmatrix} 155 & 158 & \cdots & 99 & 116 \\ 157 & 159 & \cdots & 94 & 112 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 137 & 128 & \cdots & 31 & 67 \\ 167 & 130 & \cdots & 95 & 86 \end{bmatrix}$$

Outline

- Quick Review
- Content Loss+Style Loss
- Implementation
- Einstein Summation
- Extensions

Style Transfer

❖ Content Loss + Style Loss



Style Transfer

(1) Data Preparation

Content Image

Output Image

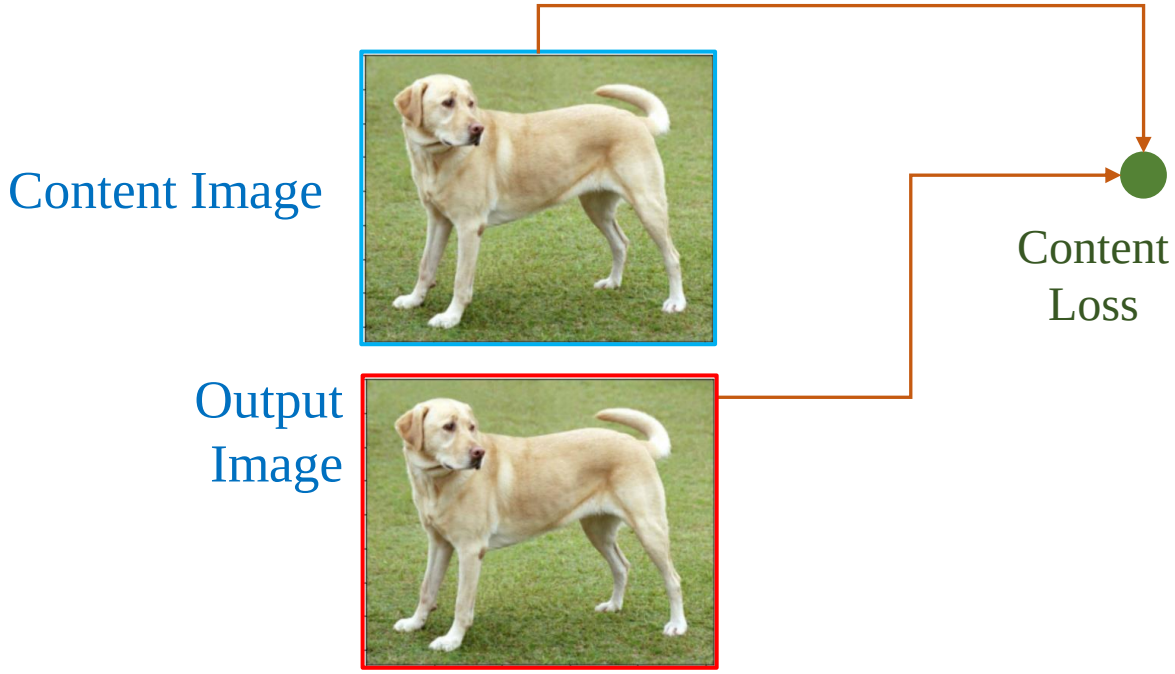
Style Image

```python
from PIL import Image
import torchvision.transforms as transforms

imsize = 256
img_transforms = transforms.Compose([
    transforms.Resize((imsize, imsize)),
    transforms.ToTensor()
])
```

```python
def image_loader(image_name):
    image = Image.open(image_name)
    image = img_transforms(image).unsqueeze(0)
    return image.to(device, torch.float)

style_img = image_loader("style_img.jpg")
content_img = image_loader("content_img.jpg")

output_img = content_img.clone().requires_grad_(True).to(device)
```

Style Transfer

(2) Content Loss



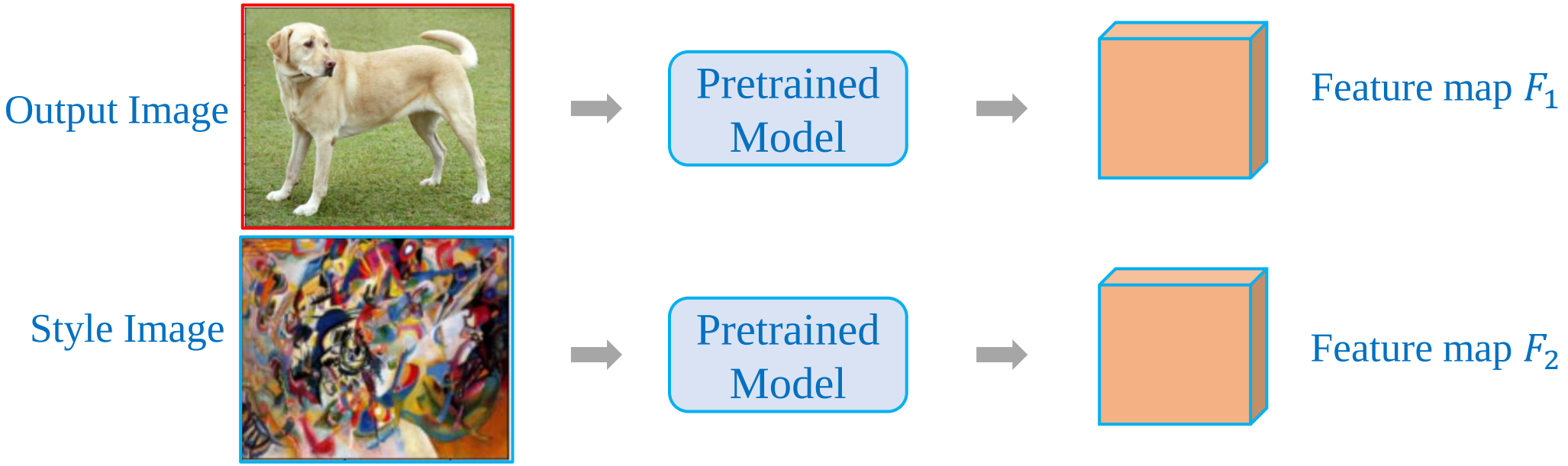

```python
# Content loss
ContentLoss = nn.MSELoss()

STEPS = 2000
for step in range(STEPS):
    optimizer.zero_grad()
    content_loss = ContentLoss(content_img,
                               output_img)

    # ...
    total_loss.backward()
    optimizer.step()
```

Style Transfer

❖ **Content Loss + Style Loss**

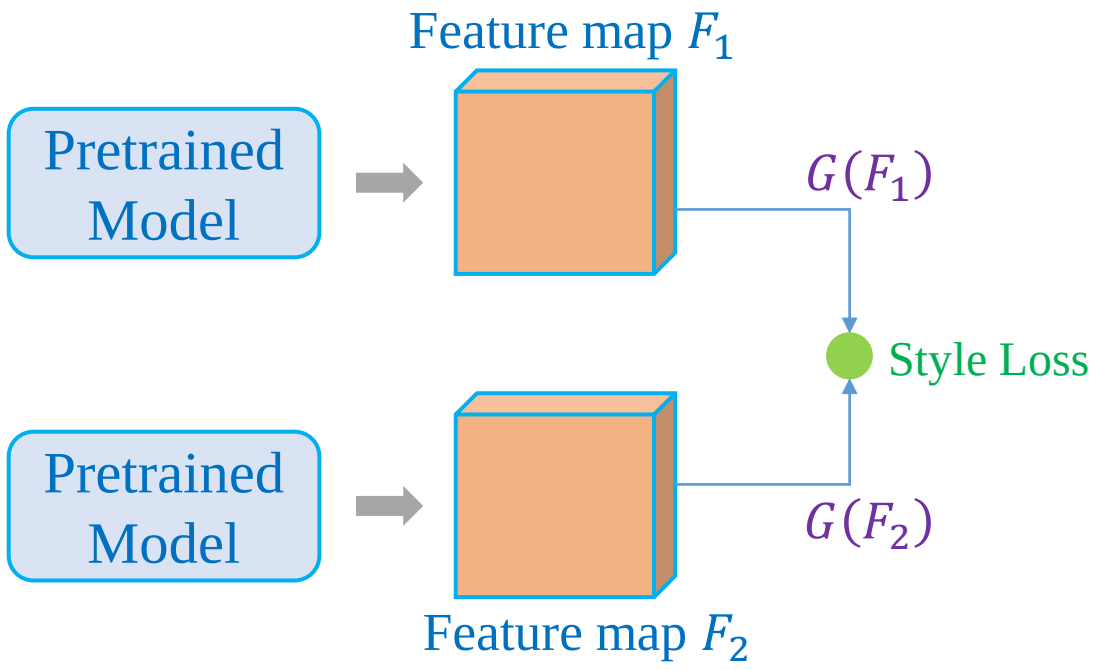
```python
VGG19_pretrained = vgg19(weights=VGG19_Weights.DEFAULT).features.eval()
for param in VGG19_pretrained.parameters():
    param.requires_grad_(False)

# get outputs from the conv1 of the block 3
model = VGG19_pretrained[:11]
model.to(device)
```

```python
style_features = model(style_img)
output_features = model(output_img)
```



Style Transfer

❖ **Content Loss + Style Loss**

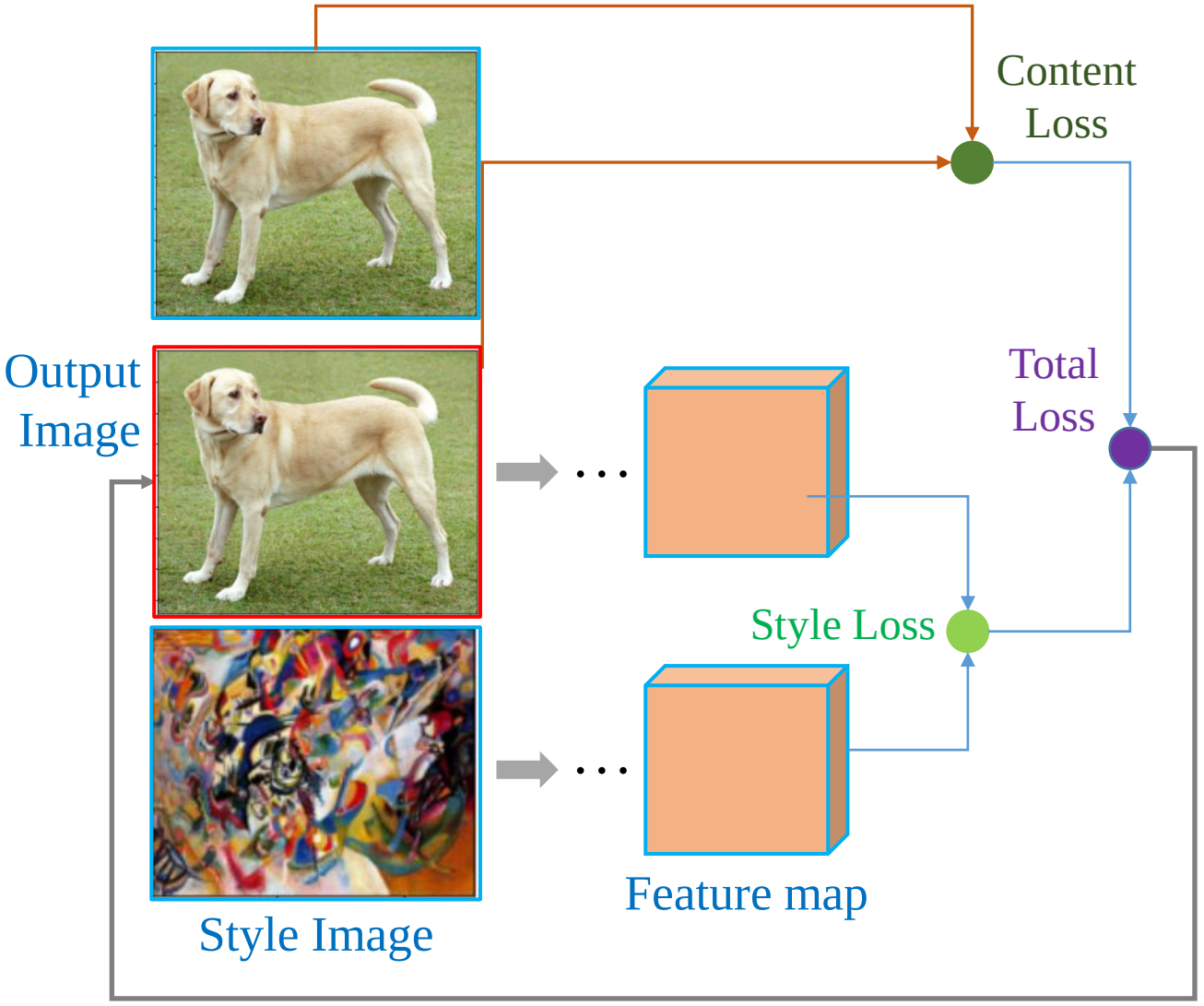



```python
def gram_matrix(tensor):
    a, b, c, d = tensor.size()
    tensor = tensor.view(a * b, c * d)
    G = torch.mm(tensor, tensor.t())
    return G.div(a * b * c * d)
```

```python
# Style loss
StyleLoss = nn.MSELoss()

# comppute Gram for the style image
style_features = model(style_img)
style_gram = gram_matrix(style_features)

STEPS = 2000
for step in range(STEPS):
    output_features = model(output_img)
    output_gram = gram_matrix(output_features)
    style_loss = StyleLoss(style_gram,
                           output_gram)
    # ...
```

Style Transfer

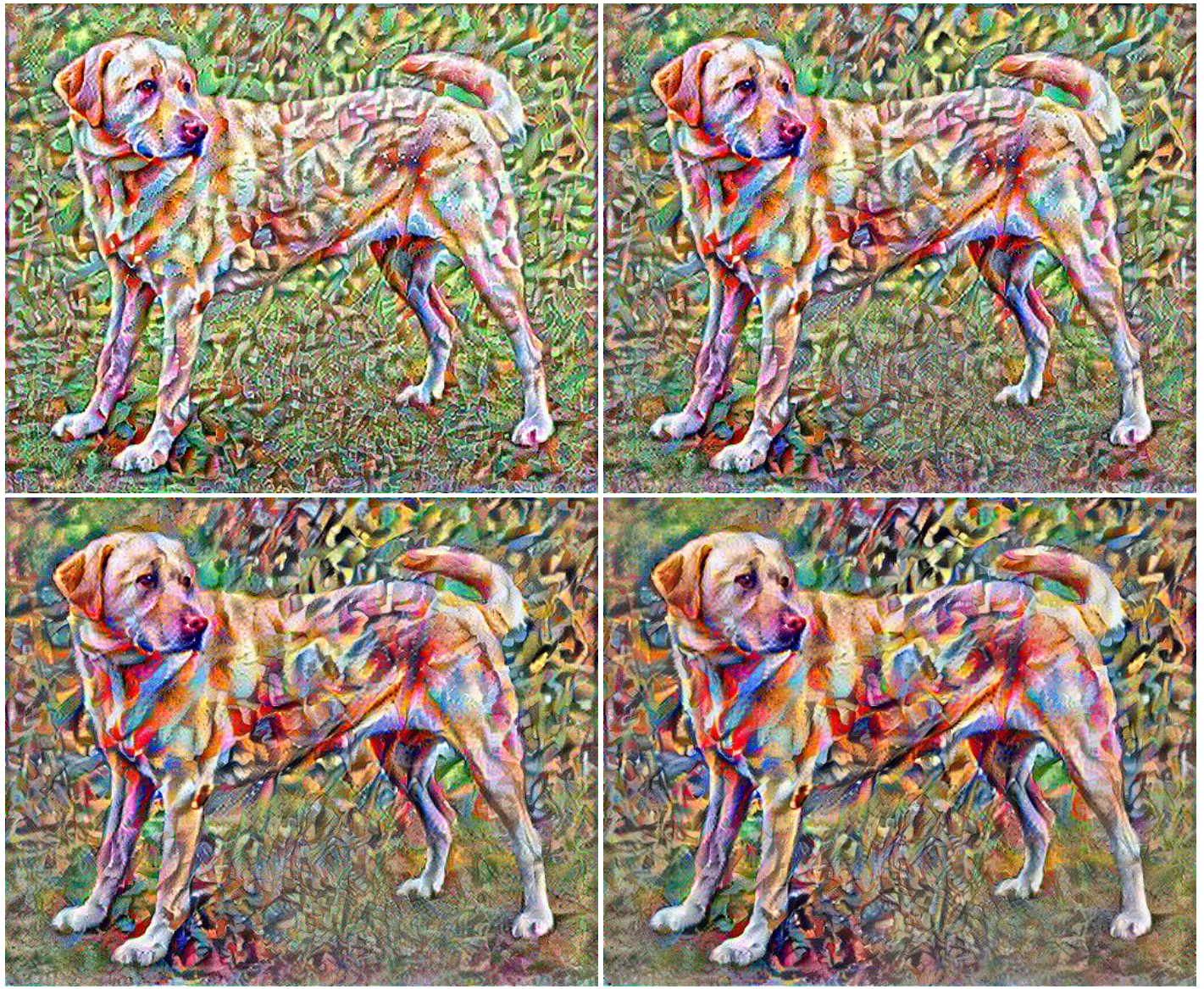
❖ **Content Loss + Style Loss**




```
style_features = model(style_img)
style_gram = gram_matrix(style_features)

STEPS = 2000
for step in range(STEPS):
    optimizer.zero_grad()

    # content loss
    content_loss = ContentLoss(content_img,
                               output_img)


    # style loss
    output_features = model(output_img)
    output_gram = gram_matrix(output_features)
    style_loss = StyleLoss(style_gram,
                           output_gram)


    # total loss
    total_loss = content_loss + style_loss*weight
    total_loss.backward()
    optimizer.step()

    with torch.no_grad():
        output_img.clamp_(0, 1)
```

Style Transfer

❖ Content Loss + Style Loss

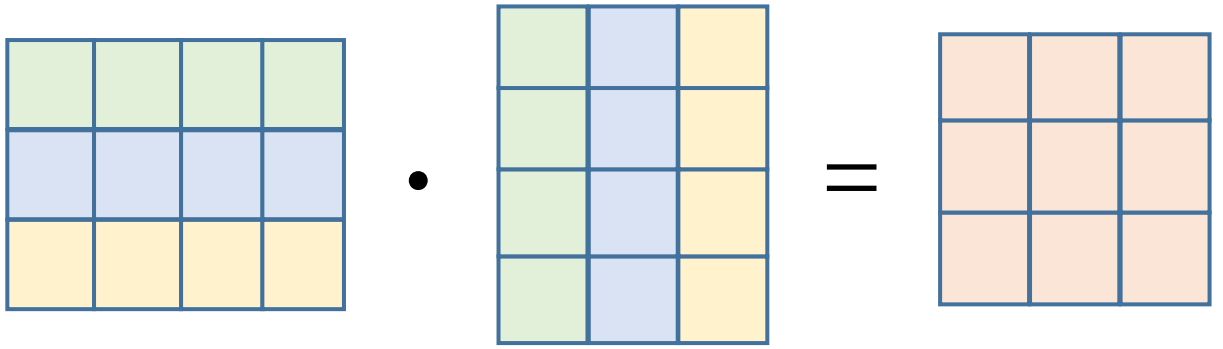


Outline

- Quick Review
- Content Loss+Style Loss
- Implementation
- Einstein Summation
- Extensions

Gram Matrix

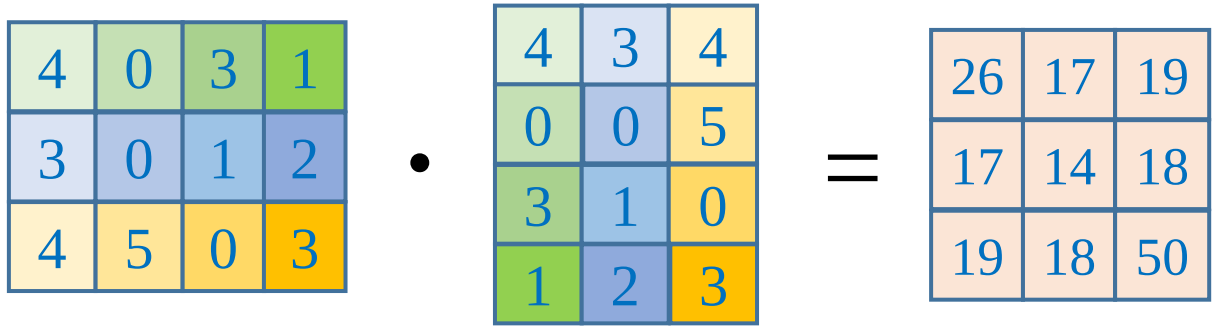
$$G_{ij} = \sum_k V_{ik} V_{jk}$$



V V^T Gram matrix

```
import torch

x = torch.Tensor([[4, 0, 3, 1],
                  [3, 0, 1, 2],
                  [4, 5, 0, 3]])
G = torch.einsum('ij,jk->ik', x, x.t())
print(G)
```

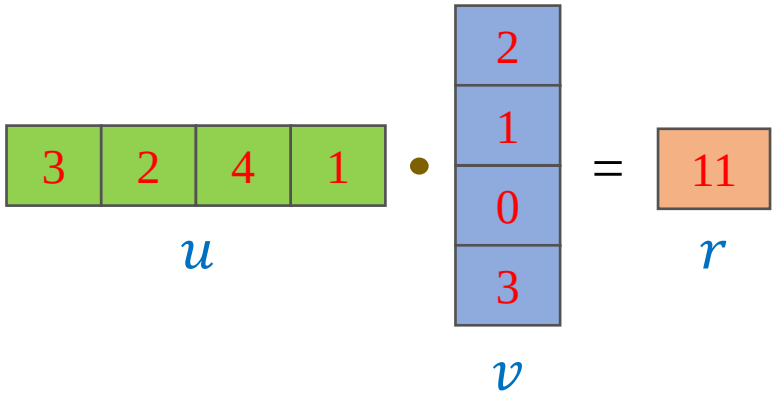
```
tensor([[26., 17., 19.],
        [17., 14., 18.],
        [19., 18., 50.]])
```



V V^T Gram matrix

Einstein Summation

Dot product



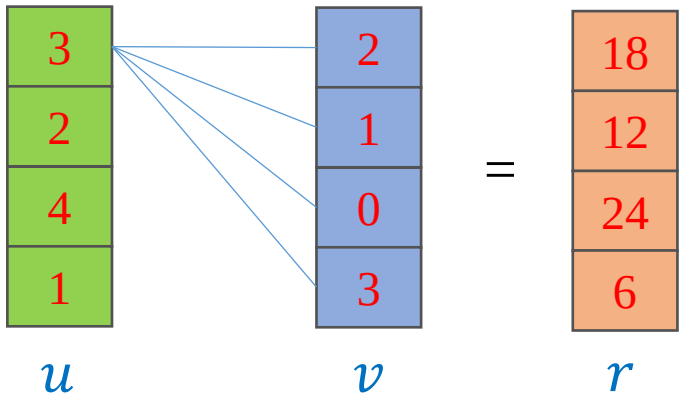

```
import torch

u = torch.Tensor([3, 2, 4, 1])
v = torch.Tensor([2, 1, 0, 3])

r = torch.einsum('i,i->', u, v)
print(r)
```

```
tensor(11.)
```

$$i, i \rightarrow$$

$$r = \sum_i u_i v_i = u_1 v_1 + u_2 v_2 + \cdots$$

Einstein Summation



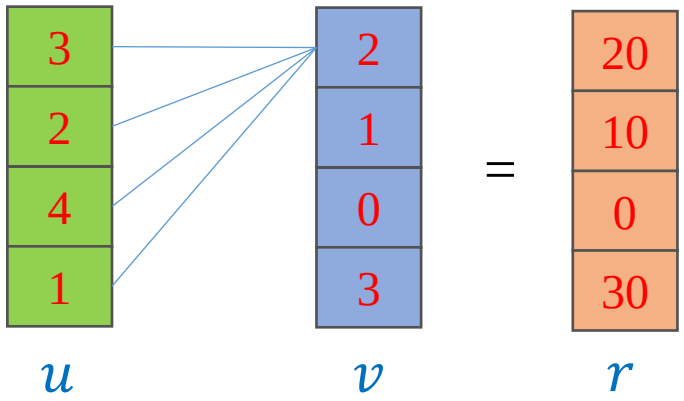

```
import torch

u = torch.Tensor([3, 2, 4, 1])
v = torch.Tensor([2, 1, 0, 3])

r = torch.einsum('i,j->i', u, v)
print(r)
```

```
tensor([18., 12., 24.,  6.])
```

$$i, j \rightarrow i$$

$$r_i = \sum_j u_i v_j = u_i v_1 + u_i v_2 + \cdots$$

Einstein Summation




```
import torch

u = torch.Tensor([3, 2, 4, 1])
v = torch.Tensor([2, 1, 0, 3])

r = torch.einsum('i,j->j', u, v)
print(r)
```

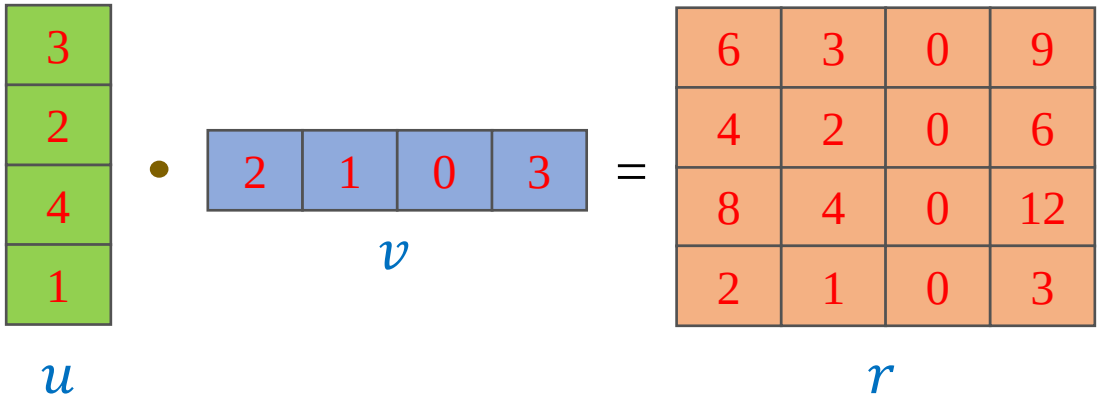
```
tensor([20., 10.,  0., 30.])
```

$$i, j \rightarrow j$$

$$r_j = \sum_i u_i v_j = u_1 v_j + u_1 v_j + \cdots$$

Einstein Summation

Outer product




```
u = torch.Tensor([3, 2, 4, 1])
v = torch.Tensor([2, 1, 0, 3])

r = torch.einsum('i,j->ij', u, v)
print(r)
```

```
tensor([[ 6.,  3.,  0.,  9.],
        [ 4.,  2.,  0.,  6.],
        [ 8.,  4.,  0., 12.],
        [ 2.,  1.,  0.,  3.]])
```

$$i, j \rightarrow ij$$

$$r_{ij} = u_i v_j$$

Einstein Summation

Transpose

0	2	1
3	1	4

A

transpose

0	3
2	1
1	4

A^T

```
import torch

A = torch.Tensor([[0, 2, 1],
                  [3, 1, 4]])
T = torch.einsum('ij->ji', A)
print(T)
```

```
tensor([[0., 3.],
        [2., 1.],
        [1., 4.]])
```

$$ij \rightarrow ji$$

j ↓

$i \rightarrow$

0	2	1
3	1	4

A

k ↓

$j \rightarrow$

2	5
0	1
3	4

A^T

$$A^T{}_{ji} = A_{ij}$$

Einstein Summation

Diagonal

5	2	1
3	1	6
2	4	3

A

diagonal

5
1
3

r

```
import torch

A = torch.Tensor([[5, 2, 1],
                  [3, 1, 6],
                  [2, 4, 3]])
r = torch.einsum('ii->i', A)
print(r)
```

```
tensor([5., 1., 3.])
```

$$ii \rightarrow i$$

$i \rightarrow$

i ↓

5	2	1
3	1	6
2	4	3

$i \rightarrow$

5
1
3

$$r_i = A_{ii}$$

Einstein Summation

Trace

5	2	1
3	1	6
2	4	3

A

trace →

9

r

```
import torch

A = torch.Tensor([[5, 2, 1],
                  [3, 1, 6],
                  [2, 4, 3]])
r = torch.einsum('ii->', A)
print(r)
```

```
tensor(9.)
```

$$ii \rightarrow$$

$i \rightarrow$

$i \downarrow$

5	2	1
3	1	6
2	4	3

$$r = \sum_i A_{ii}$$

Einstein Summation

Element-wise multiplication

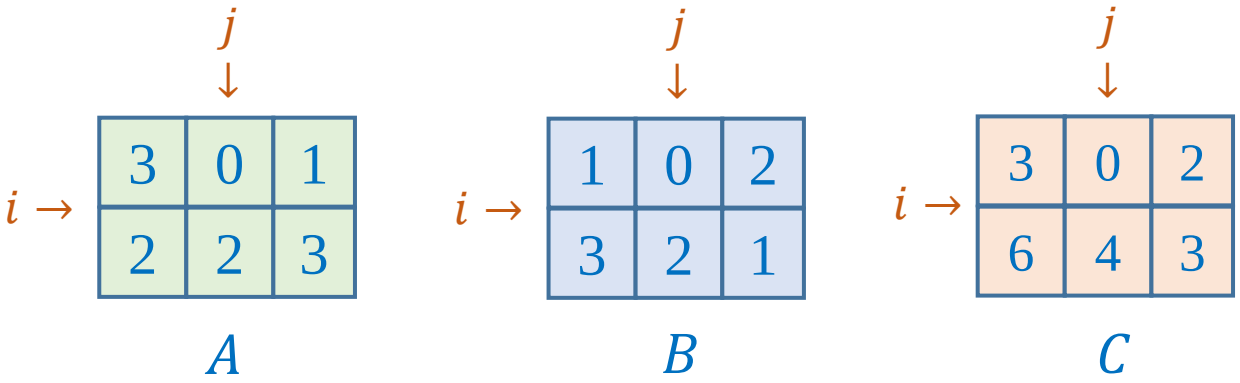
$$\begin{bmatrix} 3 & 0 & 1 \\ 2 & 2 & 3 \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 2 \\ 3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 & 2 \\ 6 & 4 & 3 \end{bmatrix}$$

A B C

```
import torch

A = torch.Tensor([[3, 0, 1],
                  [2, 2, 3]])
B = torch.Tensor([[1, 0, 2],
                  [3, 2, 1]])
r = torch.einsum('ij,ij->ij', A, B)
print(r)
```

```
tensor([[3., 0., 2.],
        [6., 4., 3.]])
```

$$ij, ij \rightarrow ij$$



$$C_{ij} = A_{ij} B_{ij}$$

Einstein Summation

Matrix multiplication

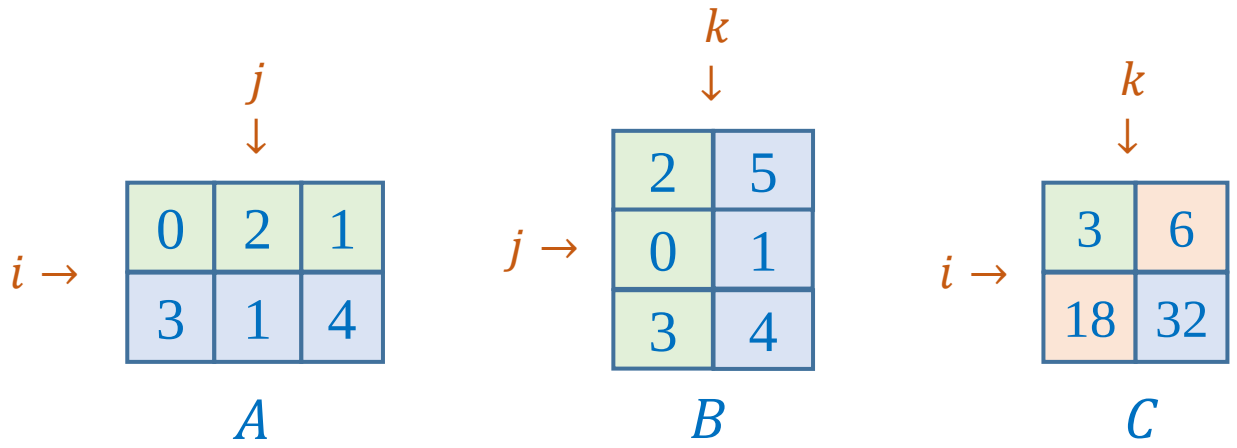
$$\begin{bmatrix} 0 & 2 & 1 \\ 3 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} 2 & 5 \\ 0 & 1 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 18 & 32 \end{bmatrix}$$

A B C

```
import torch

A = torch.Tensor([[0, 2, 1],
                  [3, 1, 4]])
B = torch.Tensor([[2, 5],
                  [0, 1],
                  [3, 4]])
C = torch.einsum('ij,jk->ik', A, B)
print(C)
```

```
tensor([[ 3.,  6.],
        [18., 32.]])
```

$$ij, jk \rightarrow ik$$



$$C_{ik} = \sum_j A_{ij} B_{jk} = A_{i1} B_{1k} + A_{i2} B_{2k} + \cdots$$

Einstein Summation

Matrix multiplication

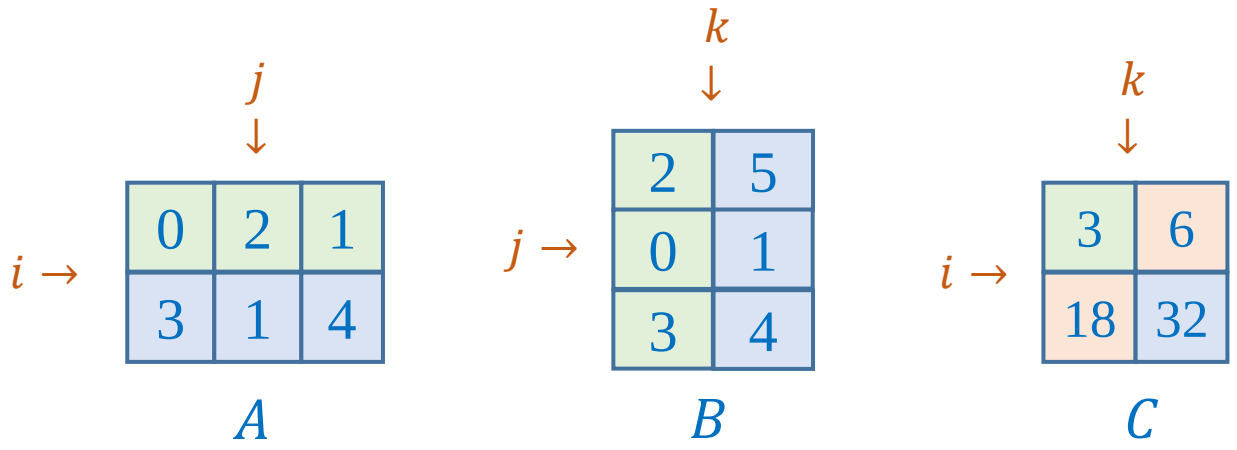
$$\begin{bmatrix} 0 & 2 & 1 \\ 3 & 1 & 4 \end{bmatrix} \cdot \begin{bmatrix} 2 & 5 \\ 0 & 1 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 18 & 32 \end{bmatrix}$$

A B C

```python
import torch

A = torch.Tensor([[0, 2, 1],
                  [3, 1, 4]])
B = torch.Tensor([[2, 5],
                  [0, 1],
                  [3, 4]])
C = torch.einsum('ij,jk', A, B)
print(C)
```

```
tensor([[ 3.,  6.],
        [18., 32.]])
```

$$ij, jk \rightarrow$$



$$C_{ik} = \sum_j A_{ij} B_{jk} = A_{i1} B_{1k} + A_{i2} B_{2k} + \cdots$$

Repeated indices are summed if the output indices are not specified.

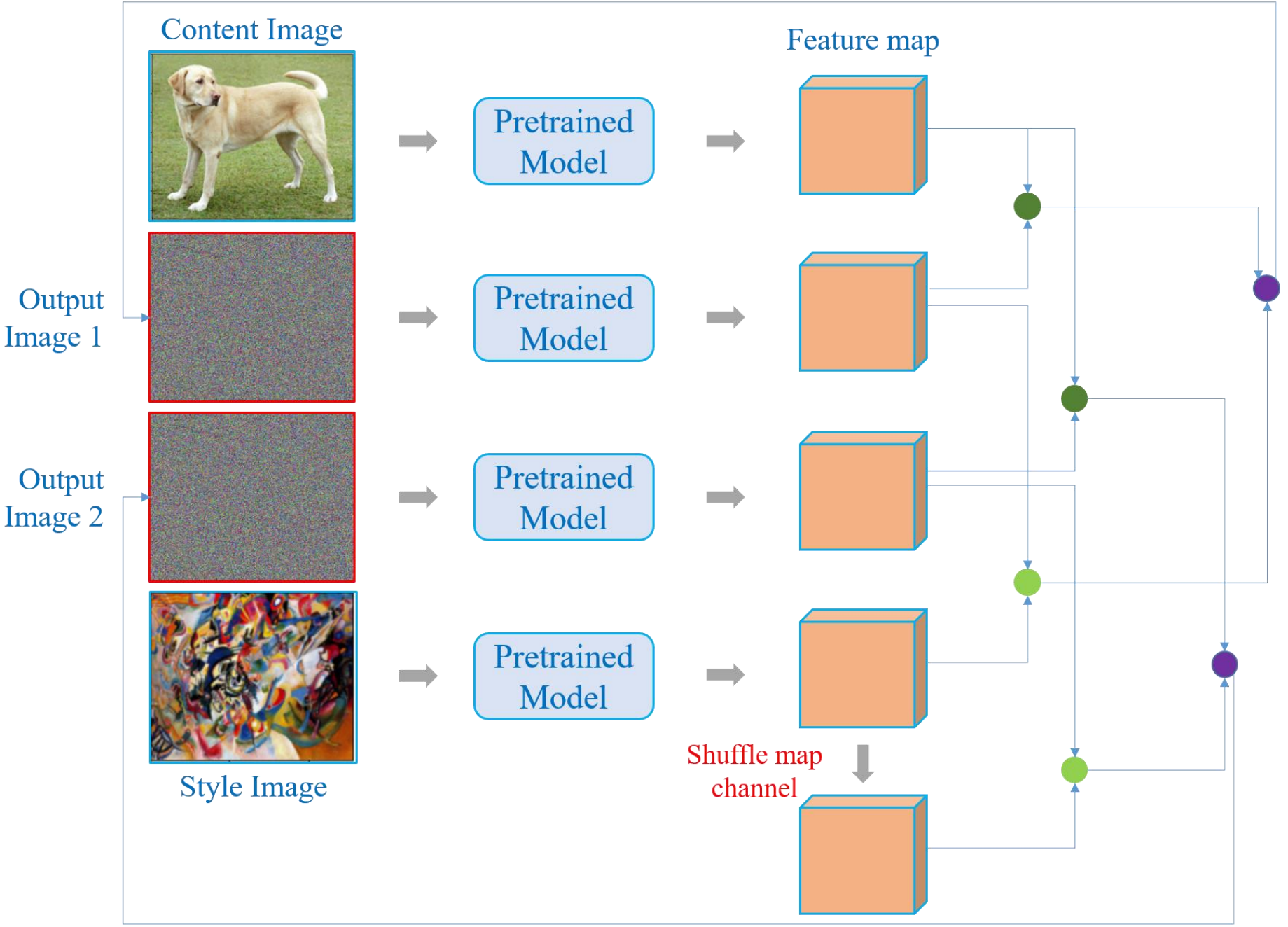

QUIZ TIME

Outline

- Quick Review
- Content Loss+Style Loss
- Implementation
- Einstein Summation
- Extensions

Multi-modal Style Transfer

Multi-modal
Style Transfer

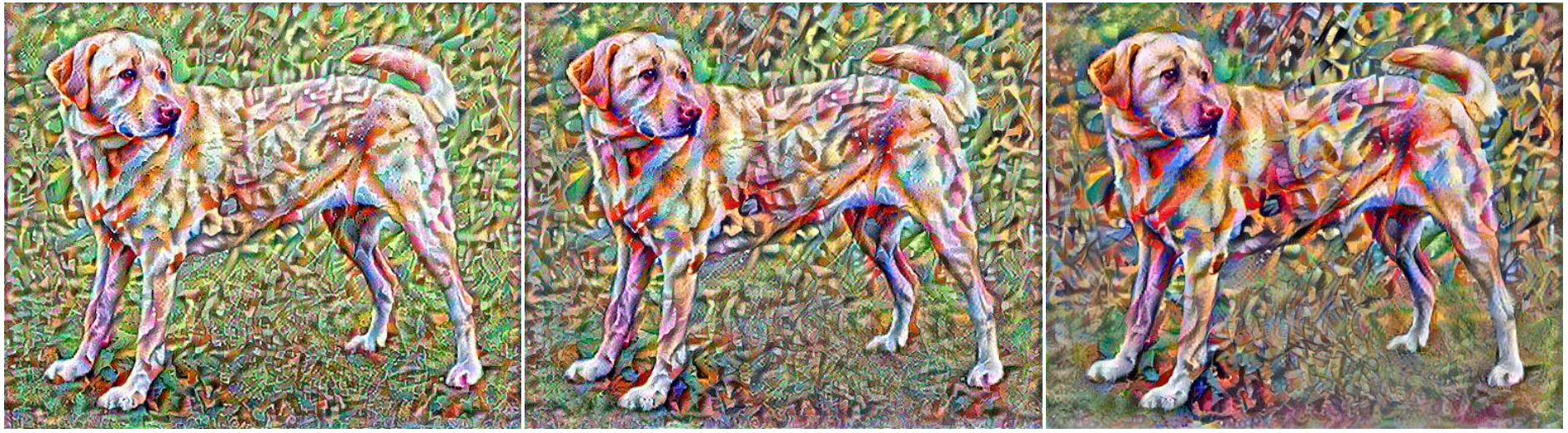

Content Image
Feature map
Pretrained Model
Pretrained Model
Pretrained Model
Pretrained Model
Output Image 1
Output Image 2
Style Image
Shuffle map channel
Style Loss
Content Loss
Total Loss

Multi-modal Style Transfer

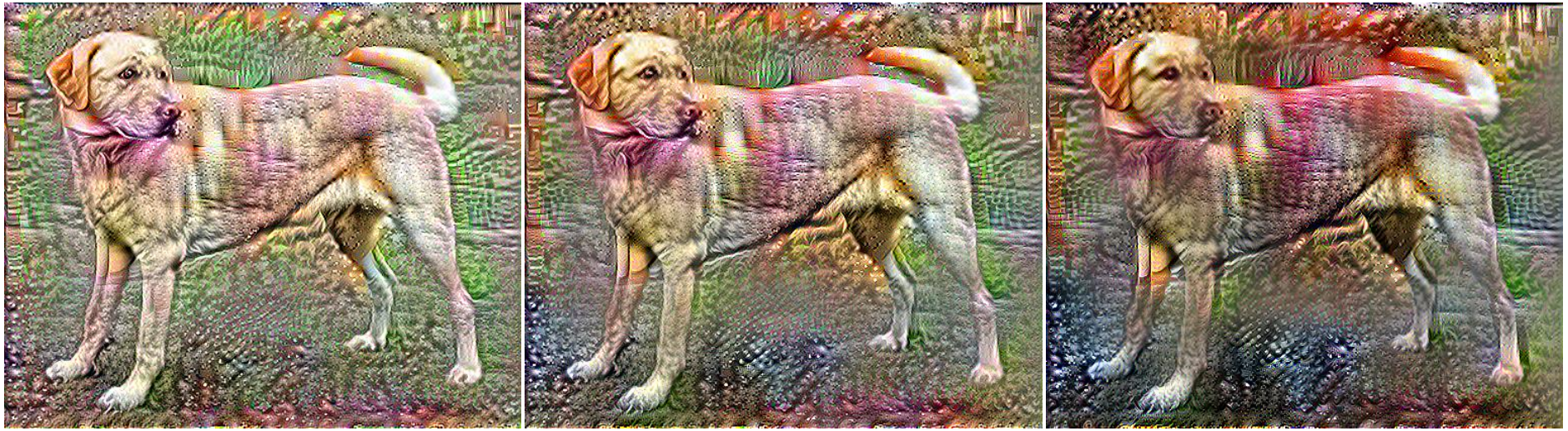
❖ **Shuffle feature maps**

```
style_outputs_1 = self.vgg(preprocessed_input)
style_outputs_2 = [tf.roll(style_output,
                           shift=[style_output.shape[1]//2,
                                  style_output.shape[2]//2,
                                  style_output.shape[3]//2],
                           axis=[1,2,3])
                   for style_output in style_outputs_1]
```

Output Image 1

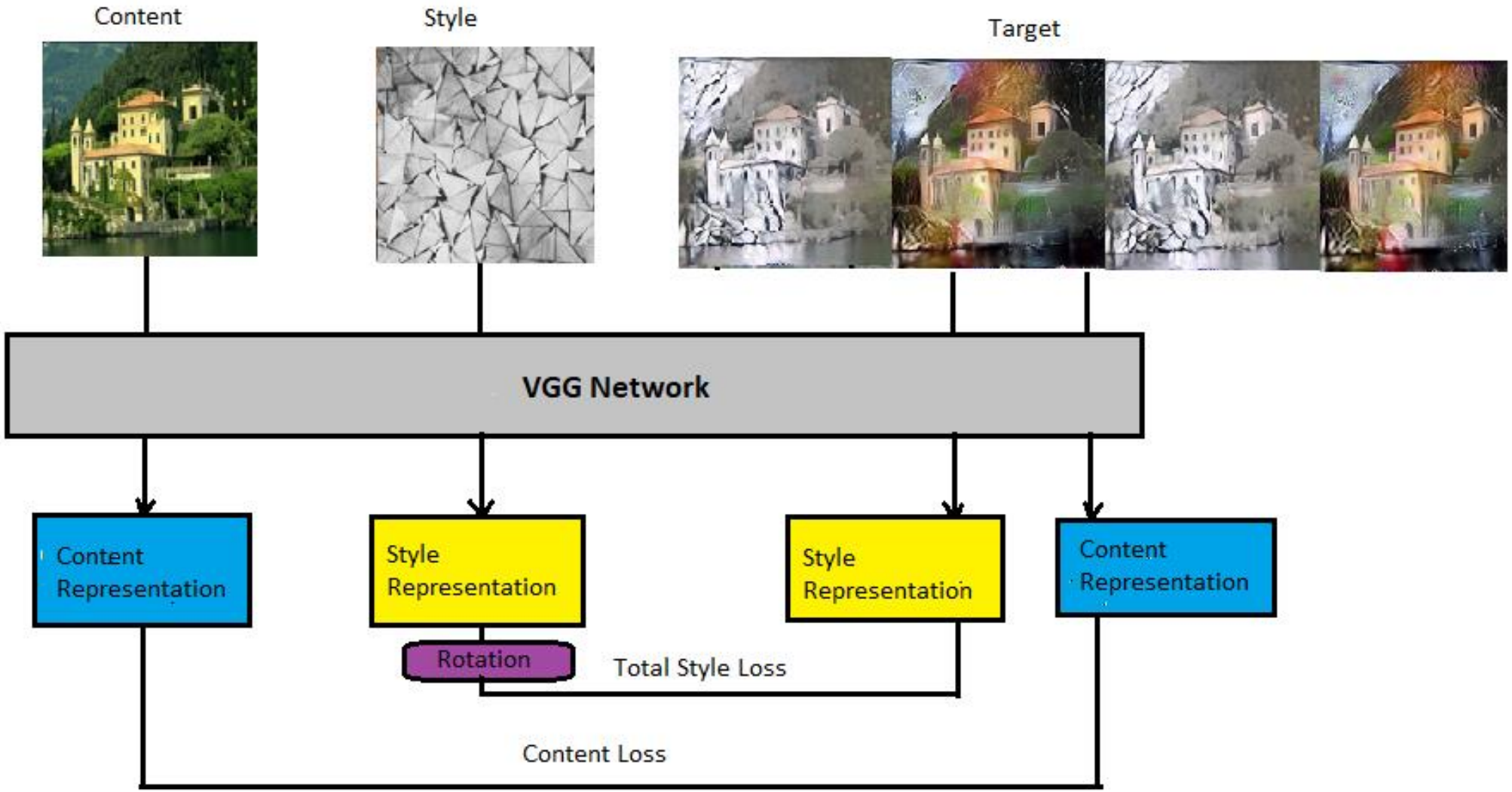


Output Image 2



Multi-modal Style Transfer

❖ Deep feature rotation



Multi-modal Style Transfer

0	0	0	0
0	0	0	0
0	0	0	0
1	1	1	1

(a) 0

0	0	0	1
0	0	0	1
0	0	0	1
0	0	0	1

(b) 90

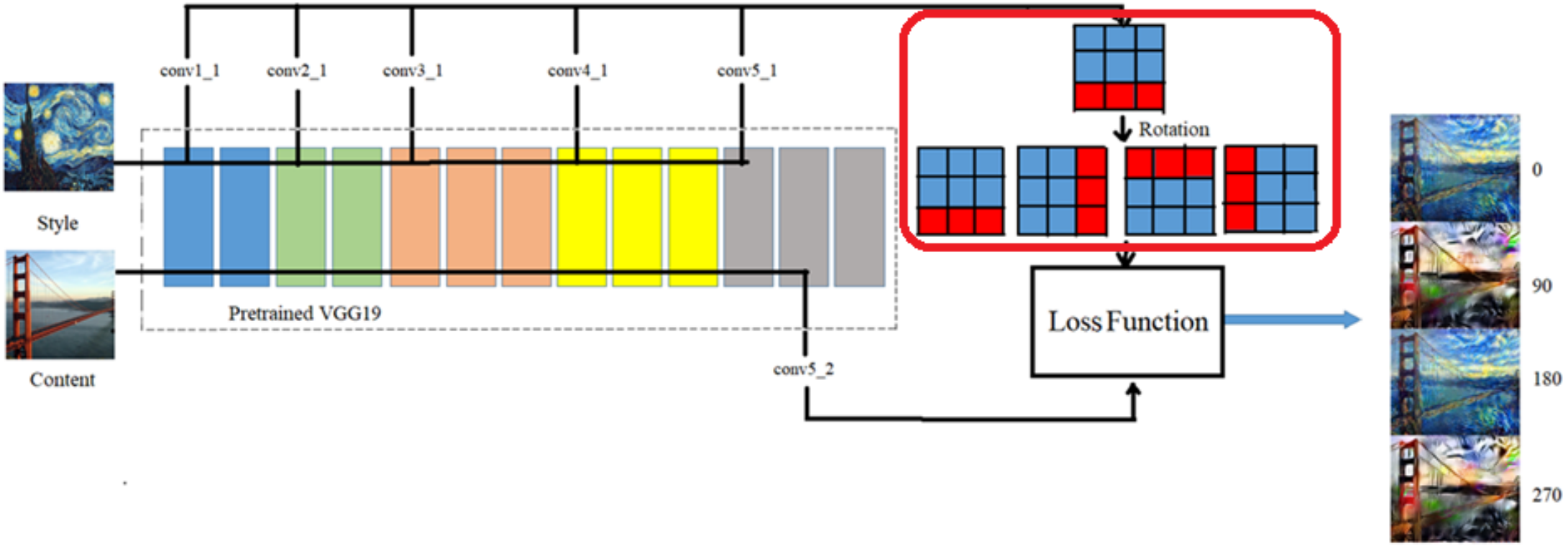
1	1	1	1
0	0	0	0
0	0	0	0
0	0	0	0

(c) 180

1	0	0	0
1	0	0	0
1	0	0	0
1	0	0	0

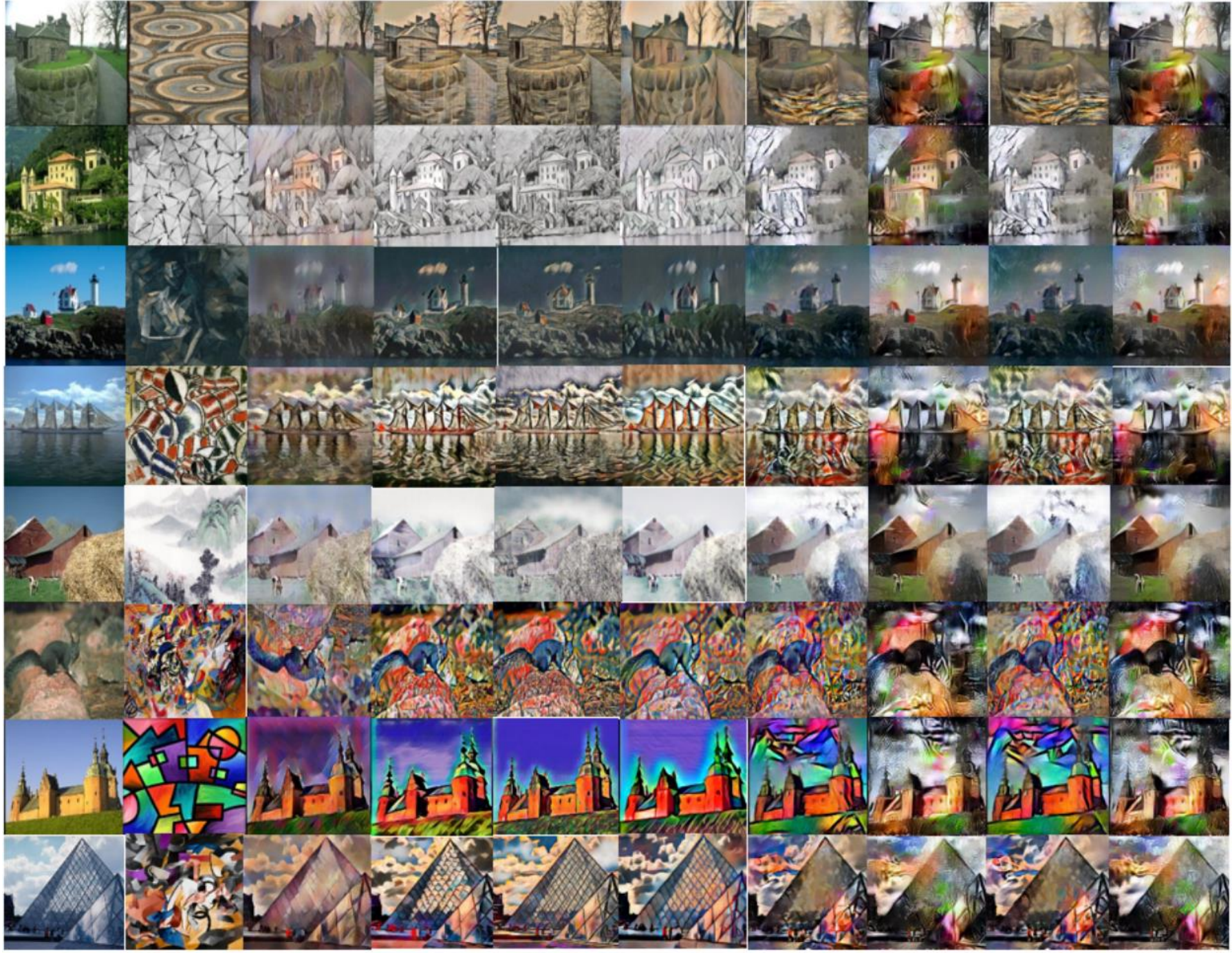
(d) 270

❖ Deep feature rotation



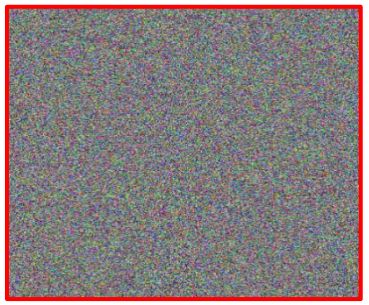
Multi-modal Style Transfer

- **Results**



❖ Two style images

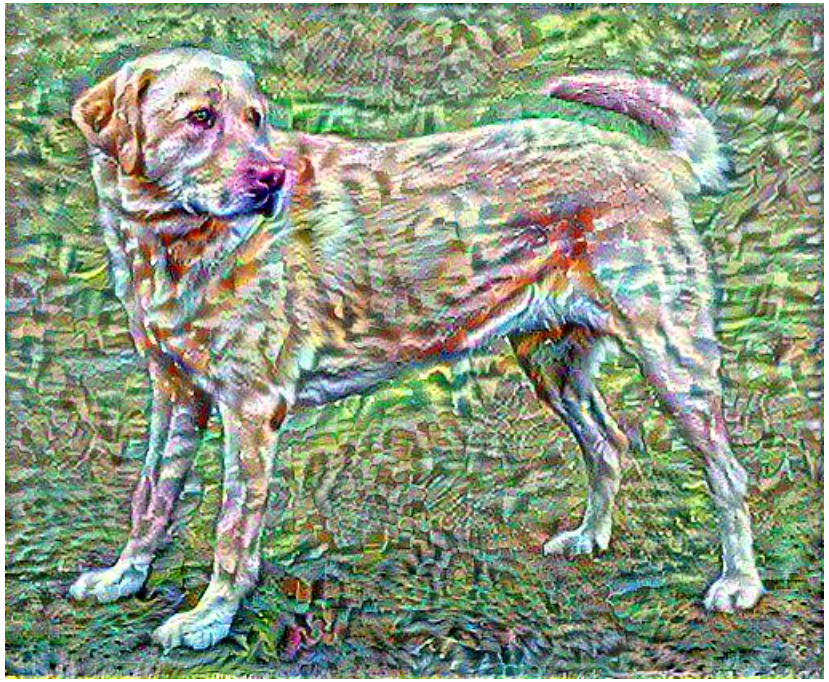
7.2.Style_transfer_1content_2style.ipynb



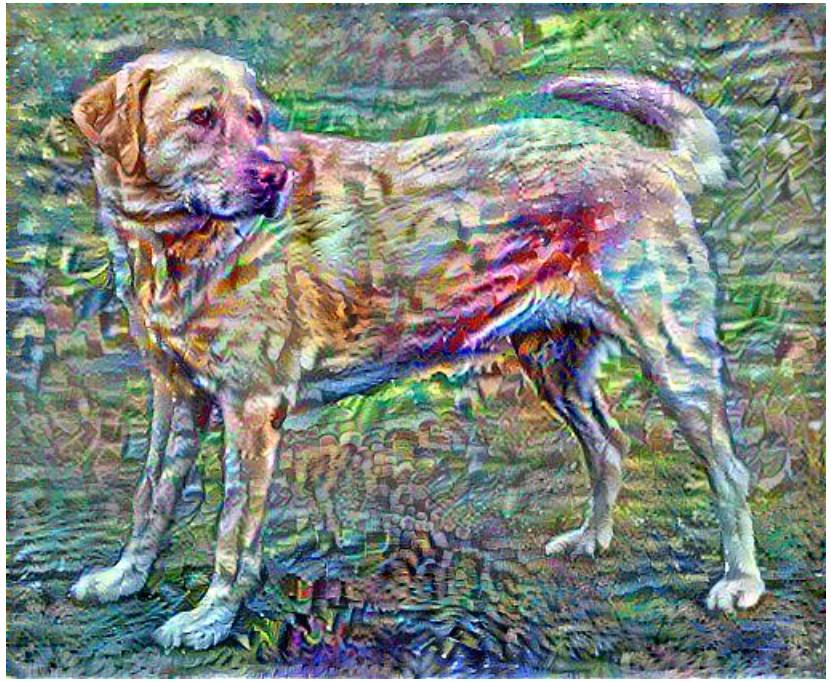
Multi-modal Style Transfer

❖ Two style images

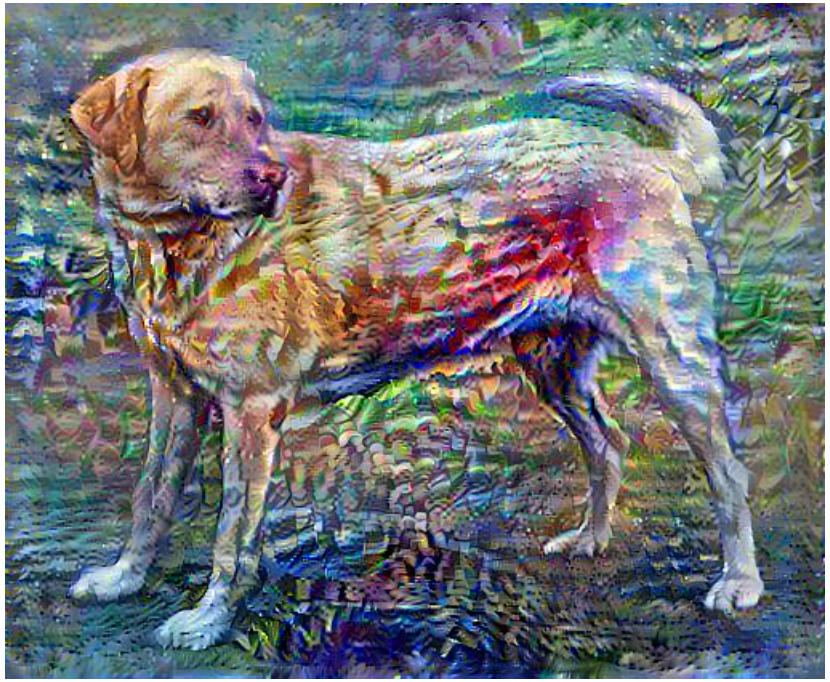
7.3.Style_transfer_1content_2style.ipynb



Epoch 1



Epoch 3

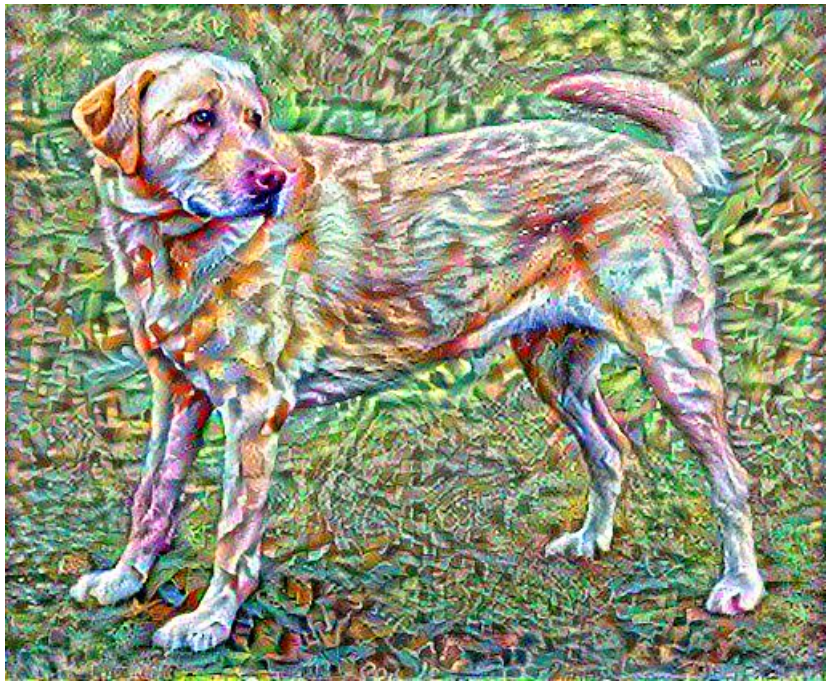


Epoch 10

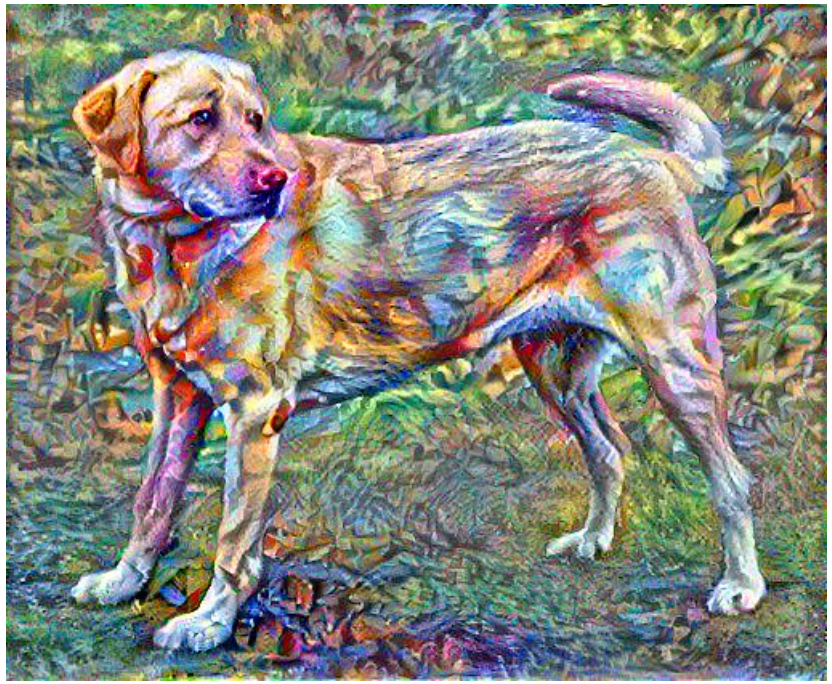
Multi-modal Style Transfer

❖ Two style images

7.4.Style_transfer_1content_2style.ipynb



Epoch 1



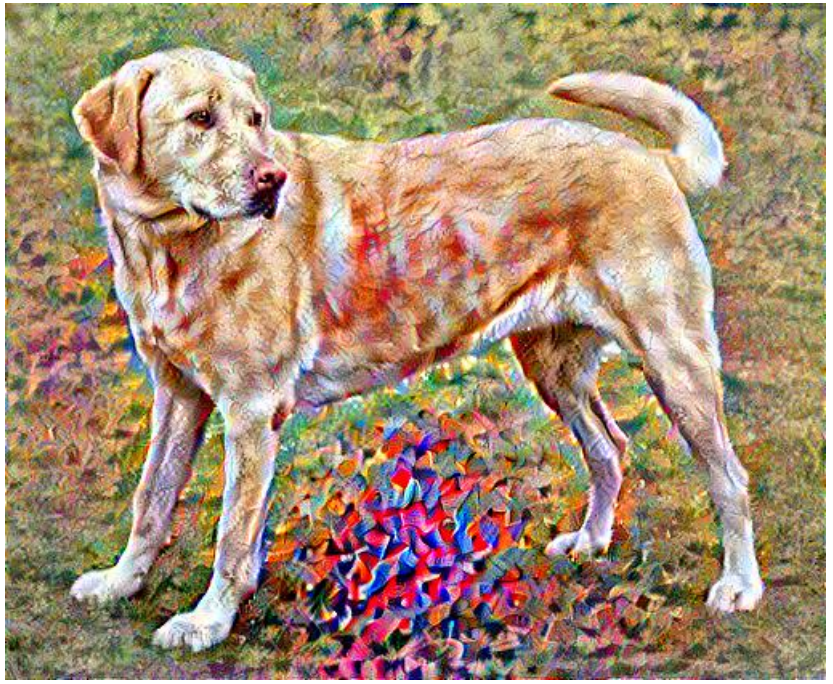
Epoch 3

Epoch 10

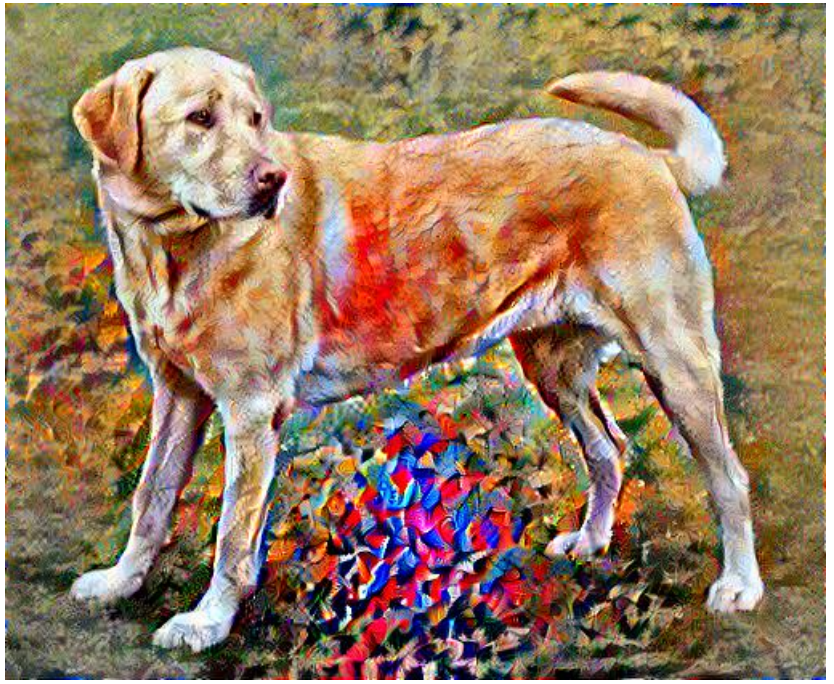
Multi-modal Style Transfer

❖ Using the Euclidian Distance

Epoch 1



Epoch 3



Epoch 10

Multi-modal Style Transfer

- **Different measures**

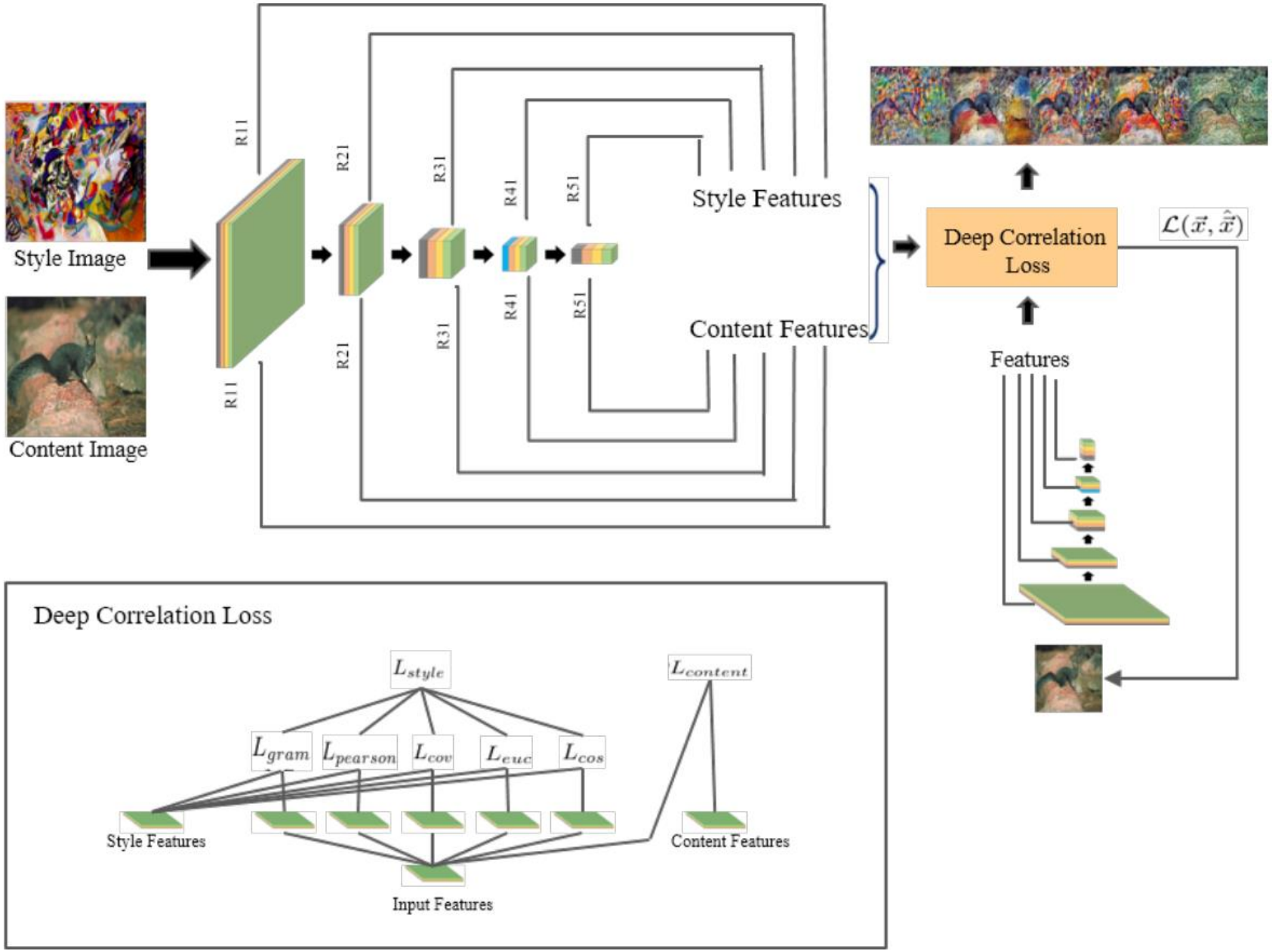


FIGURE 1: Overview of DeCorMST and details of its sub-networks. The architecture consists of a backbone of VGG extracting features from style, content, and generated images. These feature maps go through a deep correlation loss layer that integrates losses based on different correlations.

Multi-modal Style Transfer

❖ Results

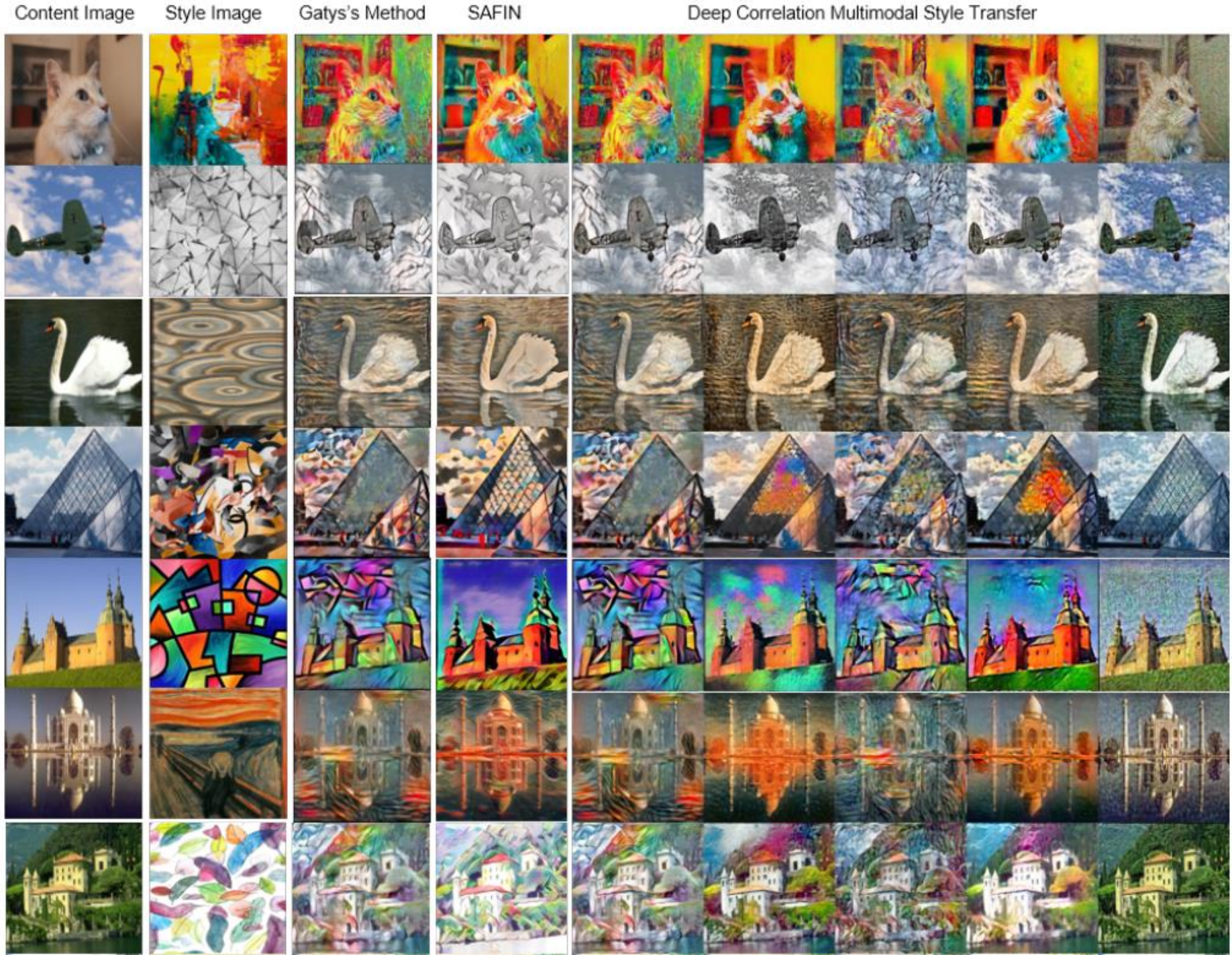
(a) Content Image (b) Style Image (c) Gatys's method (d) SAFIN

Deep Correlation Multimodal Style Transfer

(e) Output 1 (f) Output 2 (g) Output 3 (h) Output 4 (i) Output 5

Multi-modal Style Transfer

- **Results**



Summary

- Quick Review
- Content Loss+Style Loss
- Implementation
- Einstein Summation
- Extensions

Summary

- ✓ Studied about content loss + style loss
- ✓ Implemented simple methods
- ✓ Studied the Einsum and applied to style transfer
- ✓ Discussed ideas beyond the traditional style transfer

```python
import torch

x = torch.Tensor([[4, 0, 3, 1],
                  [3, 0, 1, 2],
                  [4, 5, 0, 3]])
G = torch.einsum('ij,jk->ik', x, x.t())
print(G)
```

```
tensor([[26., 17., 19.],
        [17., 14., 18.],
        [19., 18., 50.]])
```