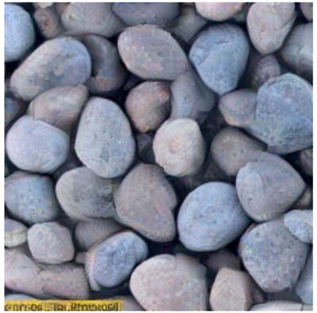
Review from deepsystems.io

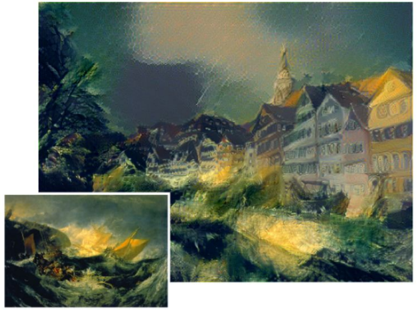
Texture synthesis, Style transfer

~852k parameters

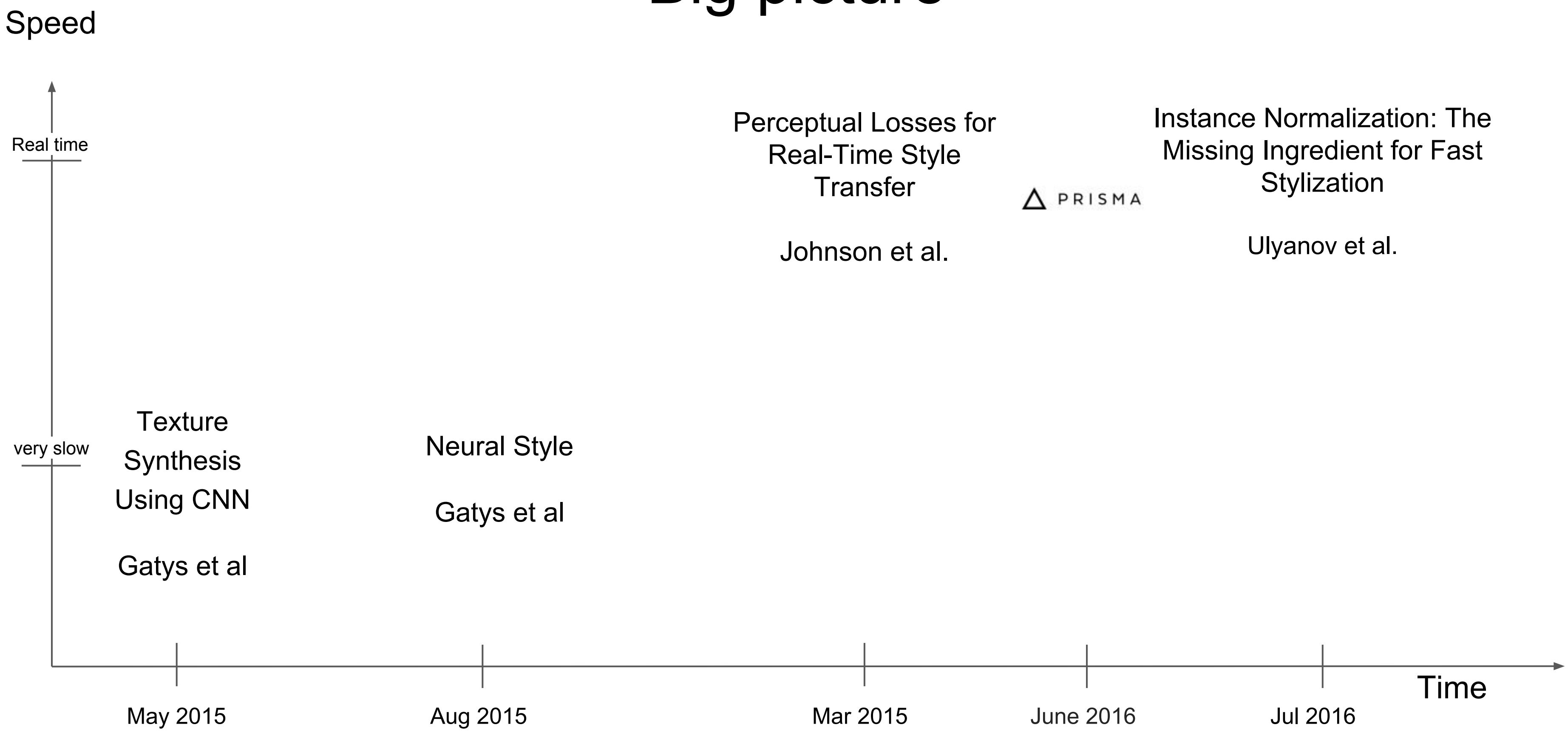


original

source: Simoncelli



Big picture



Texture Synthesis Using CNN

Gatys et al

Neural Style

Gatys et al

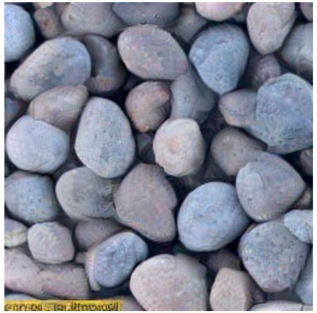
Perceptual Losses for Real-Time Style Transfer

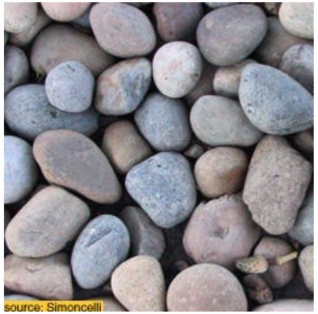
Johnson et al.

Instance Normalization: The Missing Ingredient for Fast Stylization

Ulyanov et al.

Texture Synthesis Using CNN



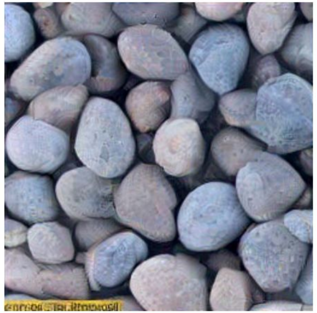


source: Simoncelli

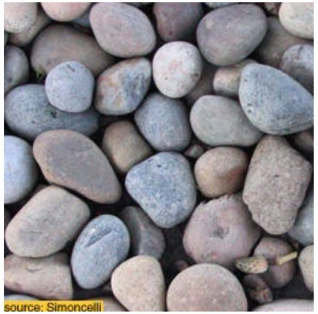
Which one is
generated?

Texture Synthesis Using CNN

~852k parameters

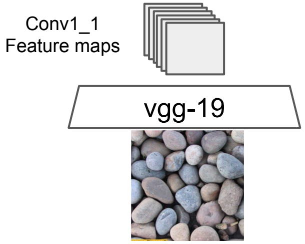


original

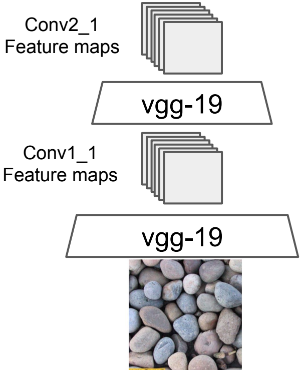


source: Simoncelli

Which one is generated?



Original texture



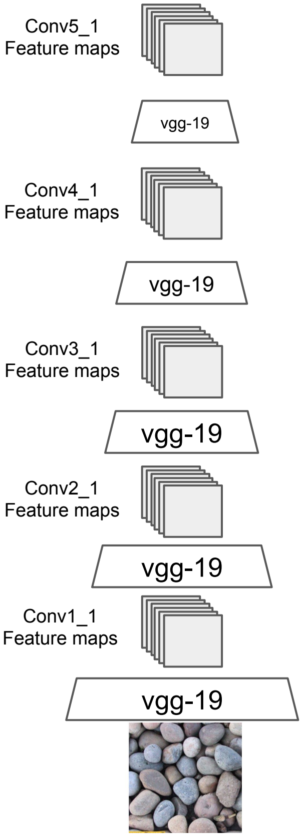
Conv2_1
Feature maps

vgg-19

Conv1_1
Feature maps

vgg-19

Original texture



Conv5_1
Feature maps

vgg-19

Conv4_1
Feature maps

vgg-19

Conv3_1
Feature maps

vgg-19

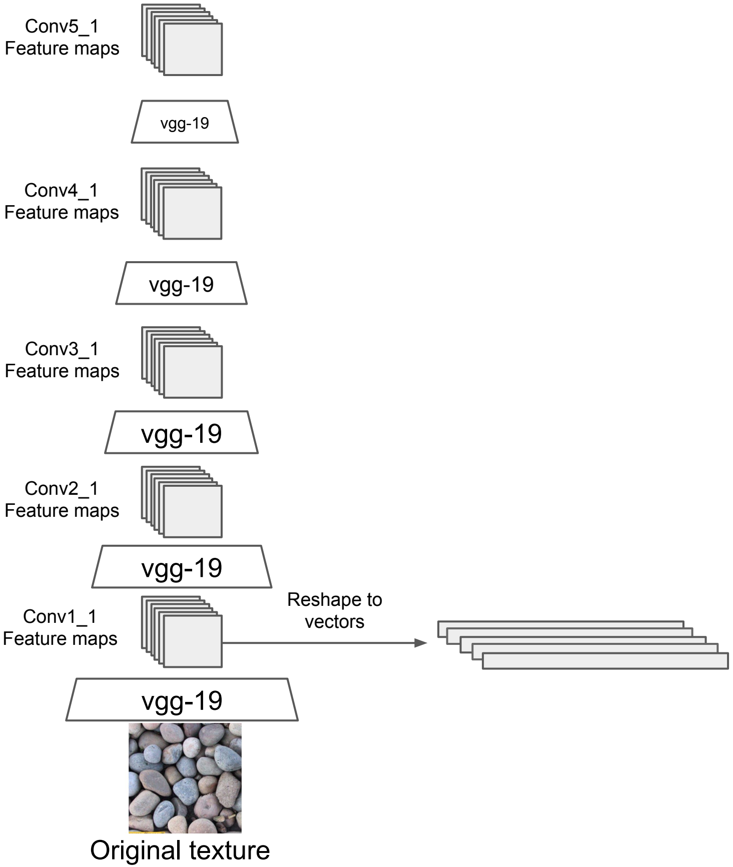
Conv2_1
Feature maps

vgg-19

Conv1_1
Feature maps

vgg-19

Original texture



Conv5_1
Feature maps

vgg-19

Conv4_1
Feature maps

vgg-19

Conv3_1
Feature maps

vgg-19

Conv2_1
Feature maps

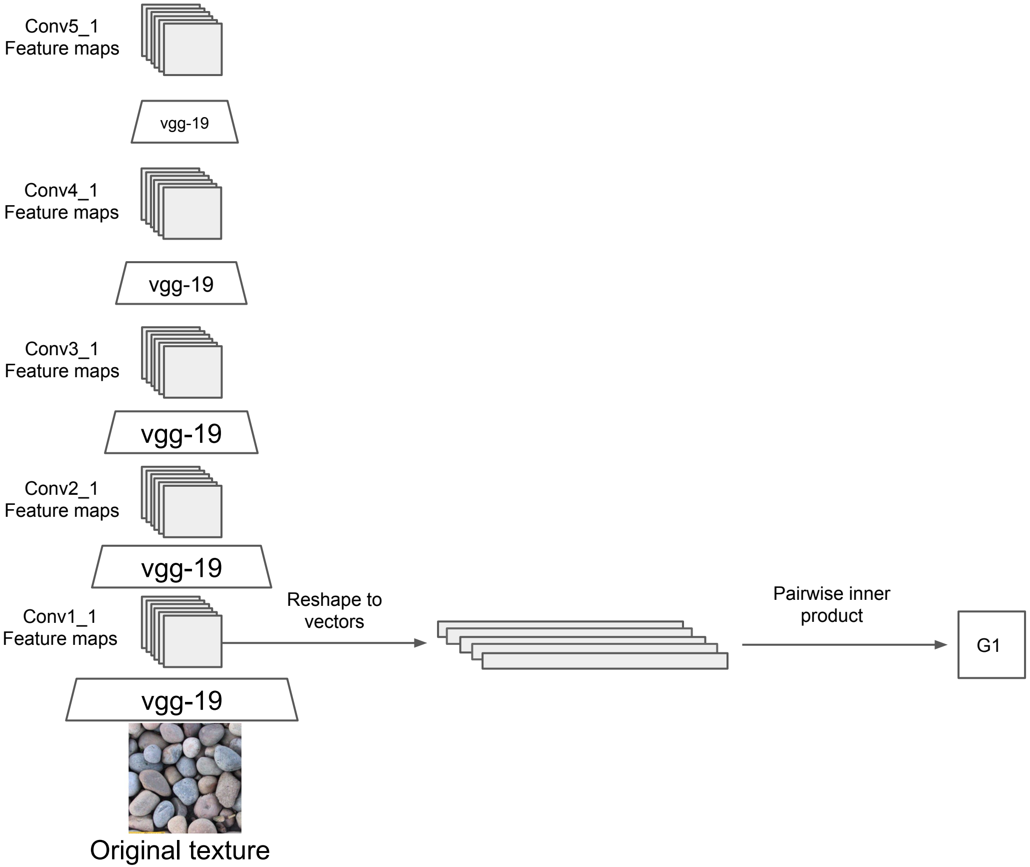
vgg-19

Conv1_1
Feature maps

Reshape to
vectors

vgg-19

Original texture



Conv5_1
Feature maps

vgg-19

Conv4_1
Feature maps

vgg-19

Conv3_1
Feature maps

vgg-19

Conv2_1
Feature maps

vgg-19

Conv1_1
Feature maps

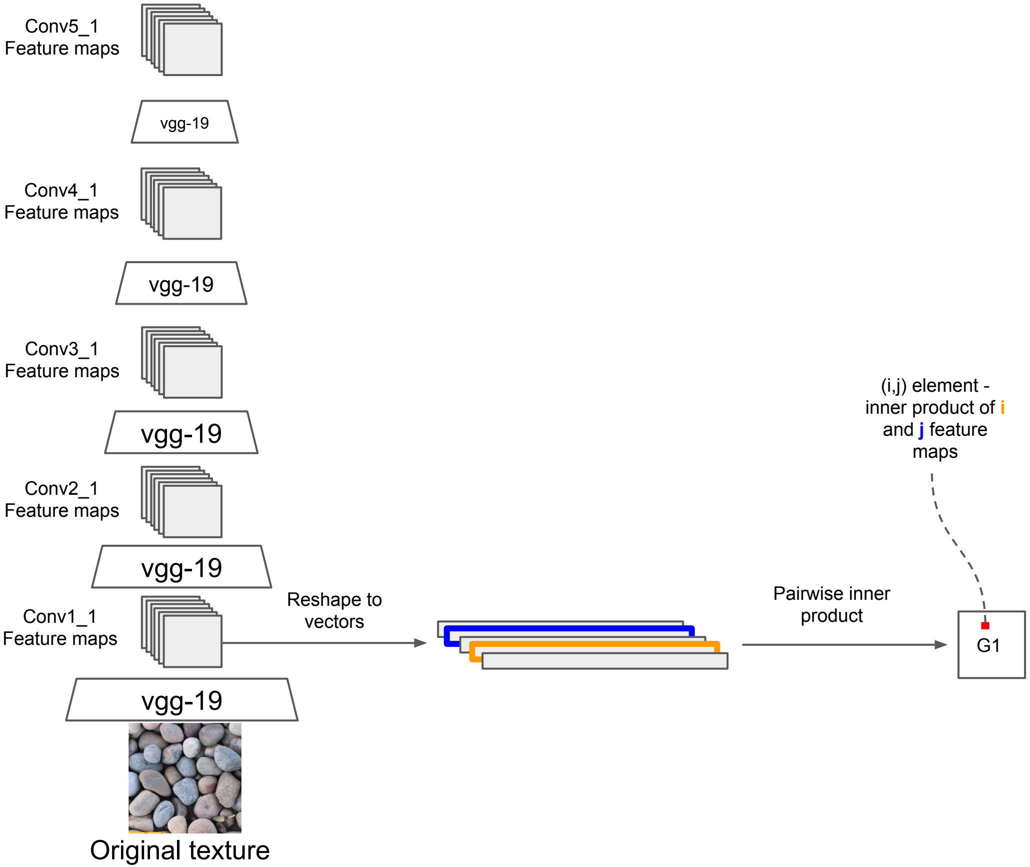
Reshape to vectors

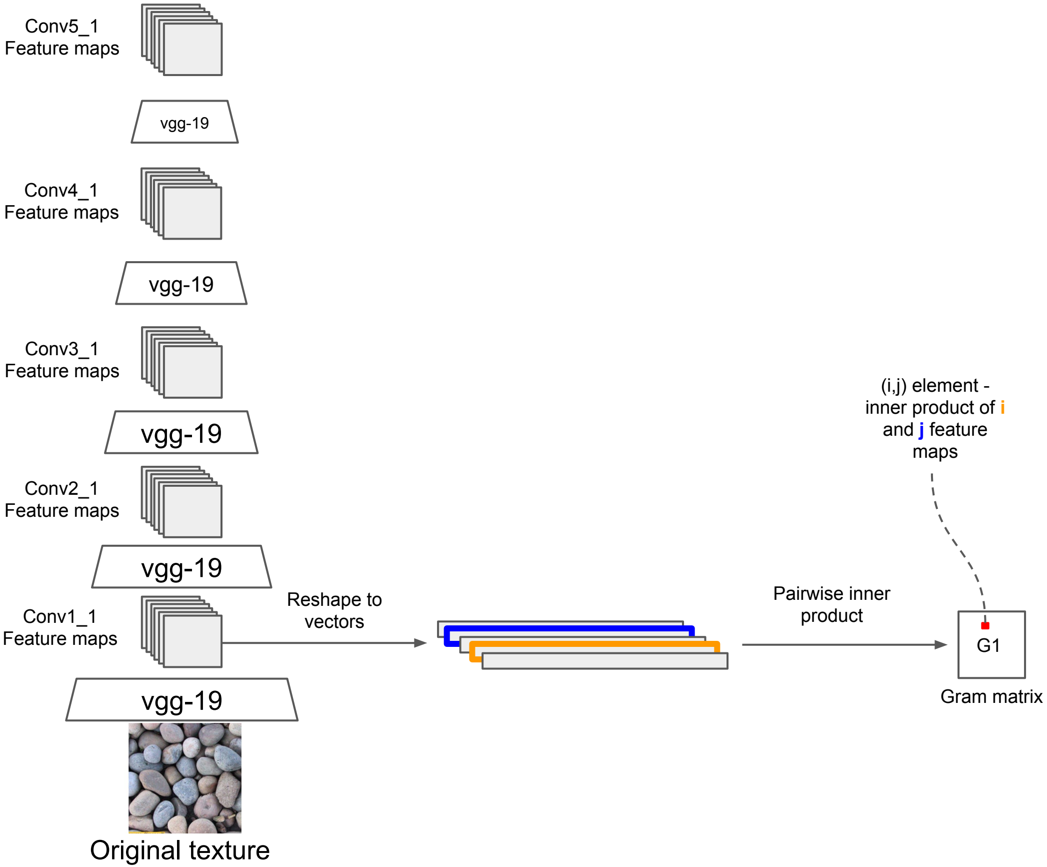
Pairwise inner product

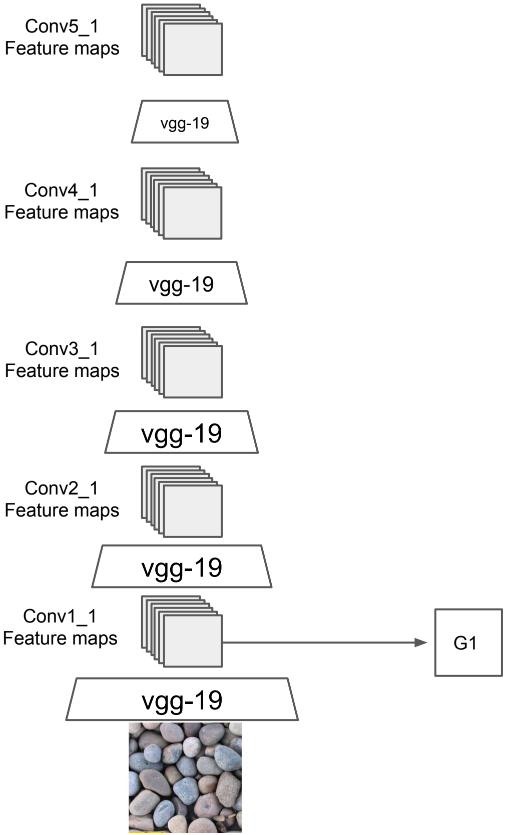
G1

vgg-19

Original texture


Conv5_1
Feature maps
vgg-19
Conv4_1
Feature maps
vgg-19
Conv3_1
Feature maps
vgg-19
Conv2_1
Feature maps
vgg-19
Conv1_1
Feature maps
vgg-19
Original texture
Reshape to vectors
Pairwise inner product
(i,j) element - inner product of i and j feature maps
G1


Conv5_1
Feature maps
vgg-19
Conv4_1
Feature maps
vgg-19
Conv3_1
Feature maps
vgg-19
Conv2_1
Feature maps
vgg-19
Conv1_1
Feature maps
vgg-19
Original texture
Reshape to vectors
Pairwise inner product
(i,j) element - inner product of i and j feature maps
G1
Gram matrix



Conv5_1
Feature maps

vgg-19

Conv4_1
Feature maps

vgg-19

Conv3_1
Feature maps

vgg-19

Conv2_1
Feature maps

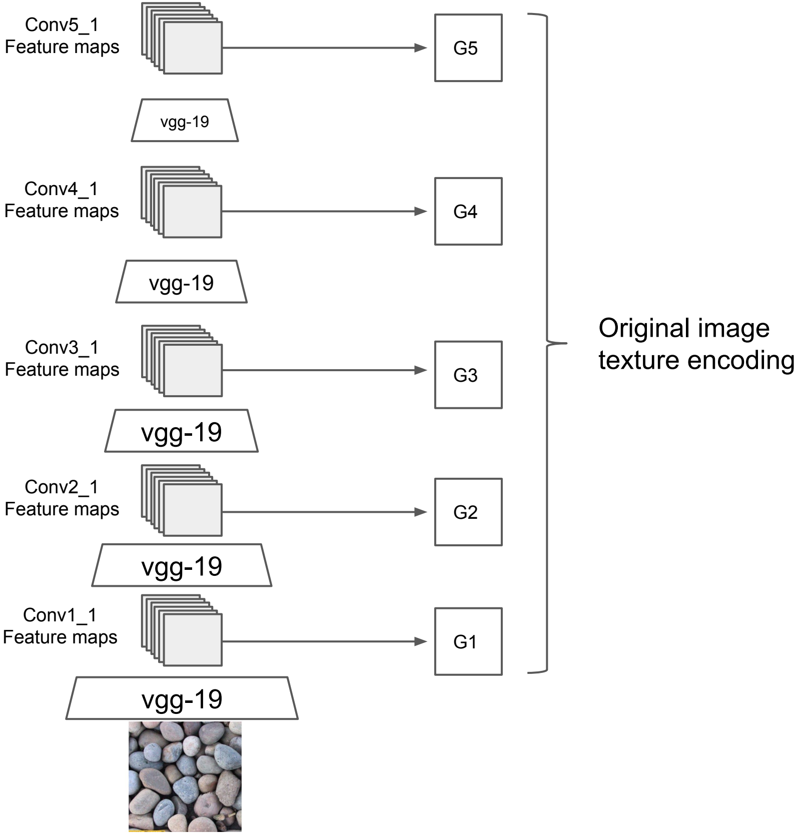
vgg-19

Conv1_1
Feature maps

G1

vgg-19

Original texture



Conv5_1
Feature maps

G5

vgg-19

Conv4_1
Feature maps

G4

vgg-19

Original image
texture encoding

Conv3_1
Feature maps

G3

vgg-19

Conv2_1
Feature maps

G2

vgg-19

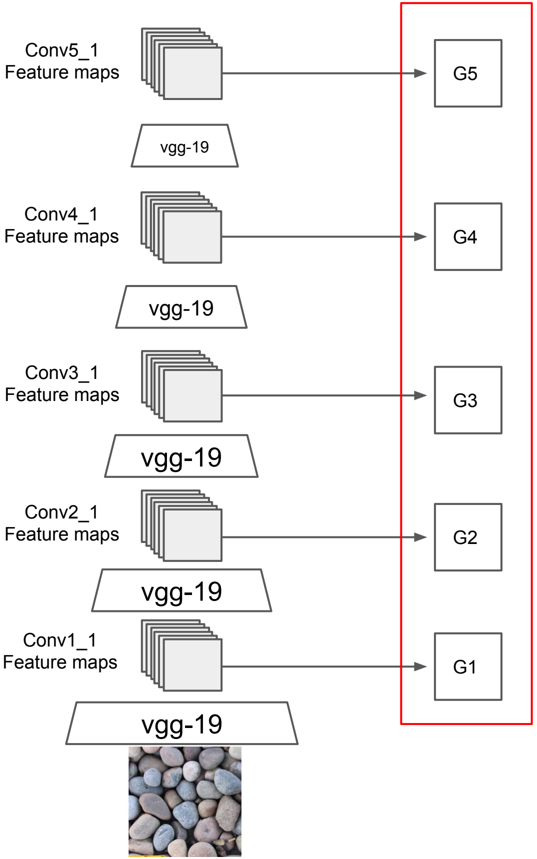
Conv1_1
Feature maps

G1

vgg-19

Original texture

Save this



Conv5_1
Feature maps

G5

vgg-19

Conv4_1
Feature maps

G4

vgg-19

Conv3_1
Feature maps

G3

vgg-19

Conv2_1
Feature maps

G2

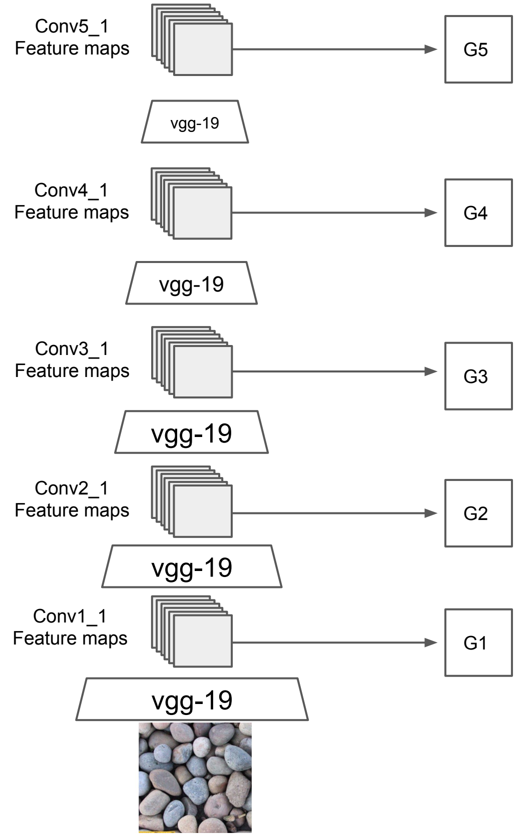
vgg-19

Conv1_1
Feature maps

G1

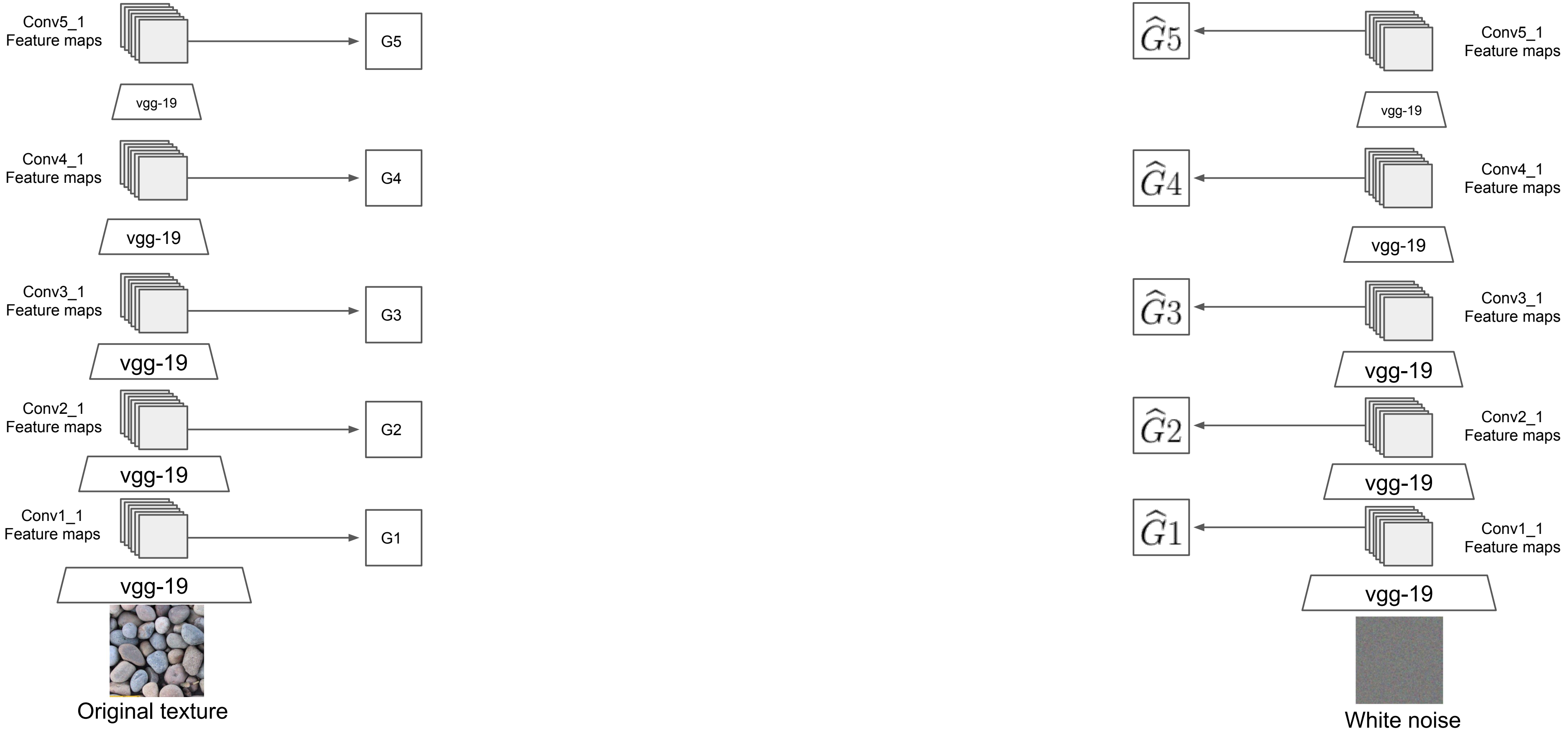
vgg-19

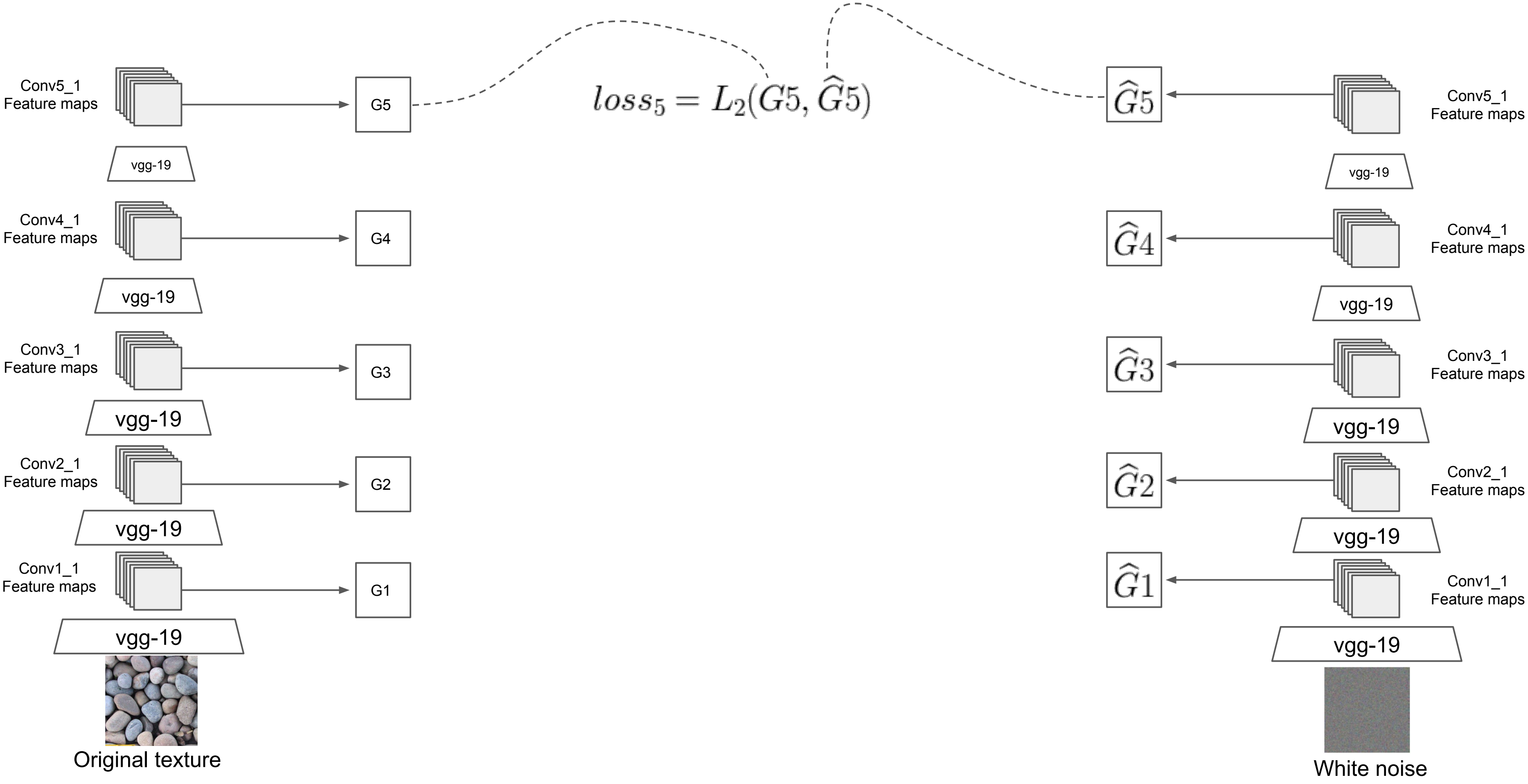
Original texture



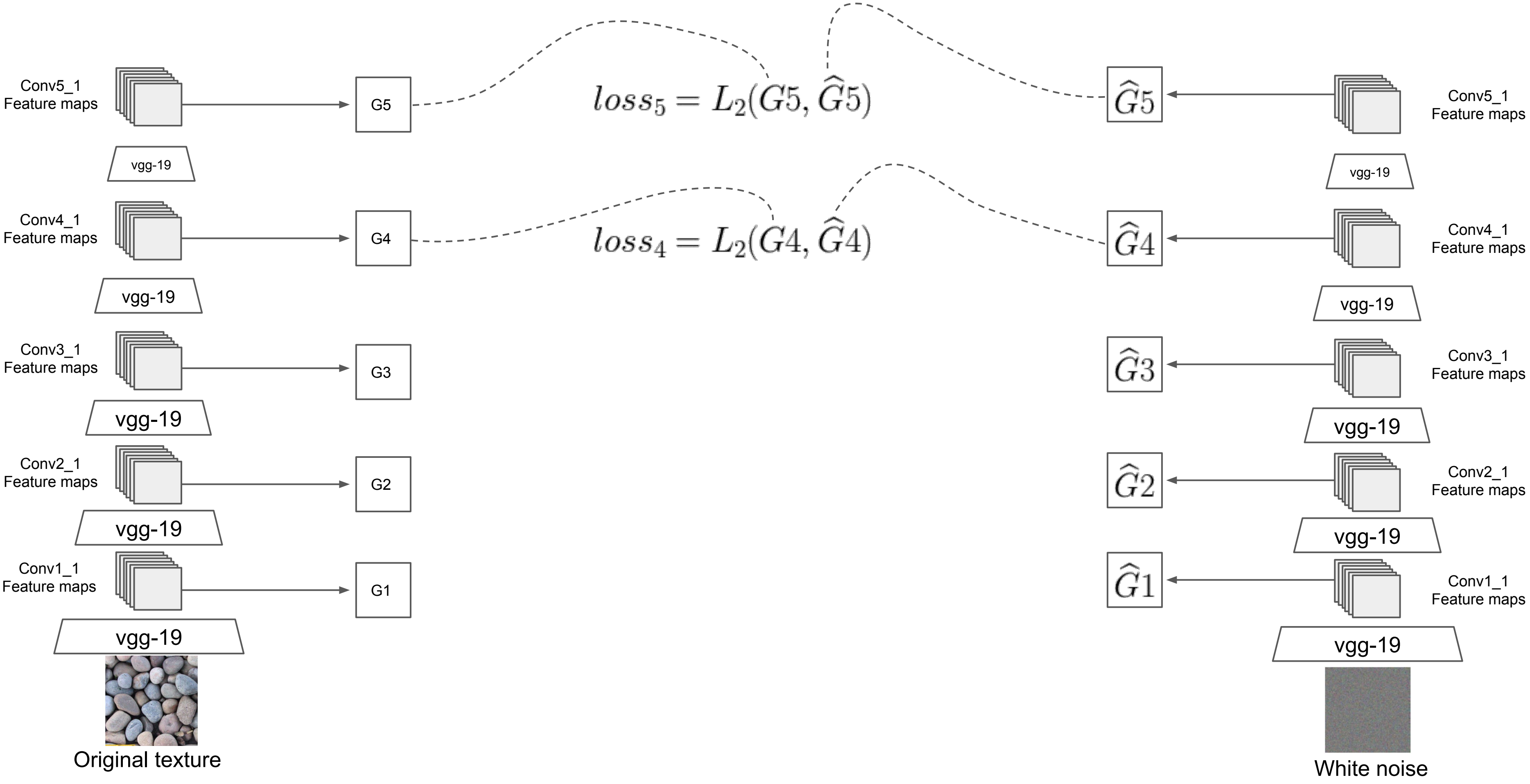
Original texture

White noise


Conv5_1 Feature maps
G5
vgg-19
Conv4_1 Feature maps
G4
vgg-19
Conv3_1 Feature maps
G3
vgg-19
Conv2_1 Feature maps
G2
vgg-19
Conv1_1 Feature maps
G1
vgg-19
Original texture
$\widehat{G}5$
Conv5_1 Feature maps
vgg-19
$\widehat{G}4$
Conv4_1 Feature maps
vgg-19
$\widehat{G}3$
Conv3_1 Feature maps
vgg-19
$\widehat{G}2$
Conv2_1 Feature maps
vgg-19
$\widehat{G}1$
Conv1_1 Feature maps
vgg-19
White noise



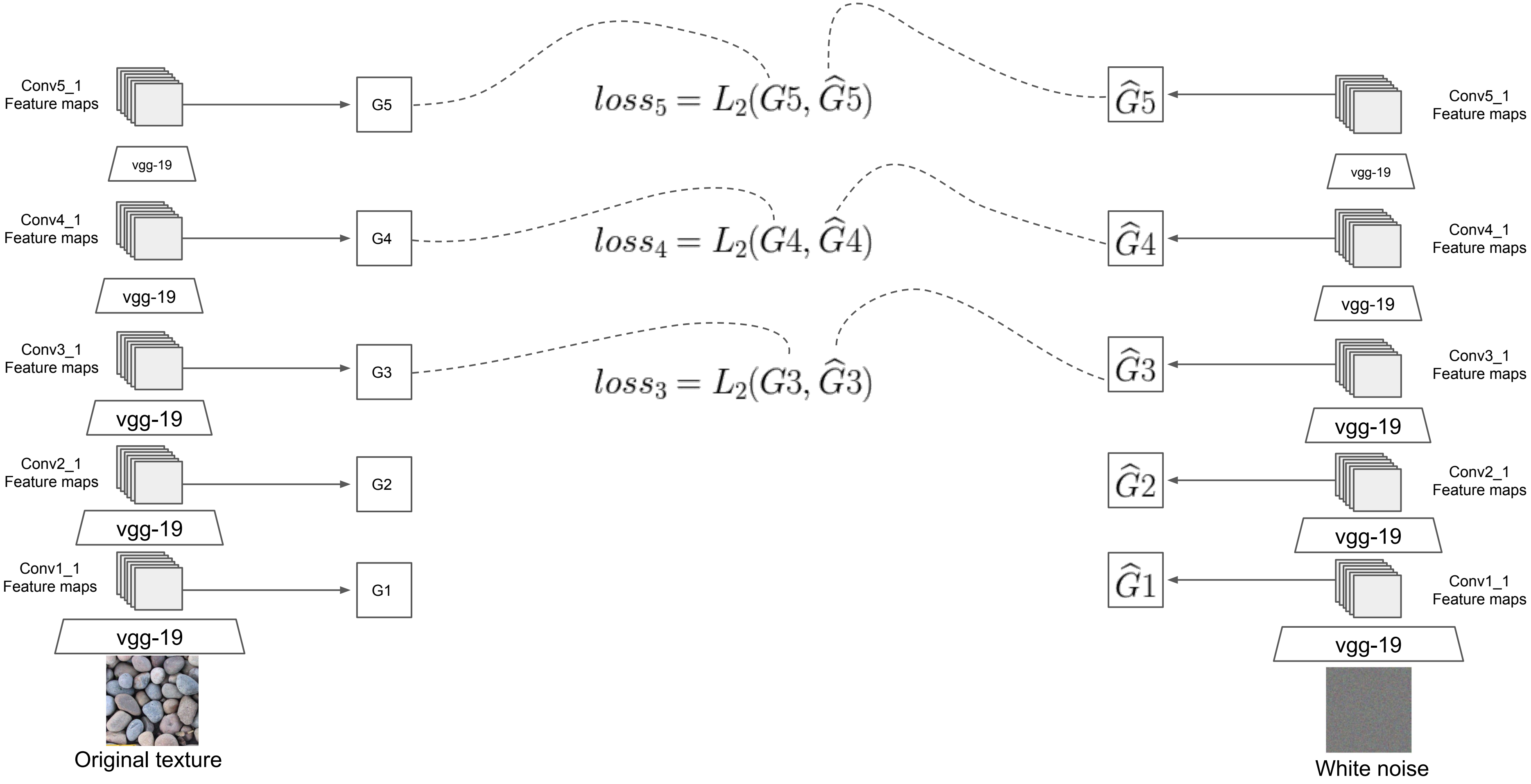
$$loss_5 = L_2(G5, \widehat{G}5)$$



$$loss_5 = L_2(G5, \widehat{G}5)$$

$$loss_4 = L_2(G4, \widehat{G}4)$$

deepsystems.io



Conv5_1 Feature maps

vgg-19

Conv4_1 Feature maps

vgg-19

Conv3_1 Feature maps

vgg-19

Conv2_1 Feature maps

vgg-19

Conv1_1 Feature maps

vgg-19

Original texture

G5

G4

G3

G2

G1

$$loss_5 = L_2(G5, \widehat{G}5)$$

$$loss_4 = L_2(G4, \widehat{G}4)$$

$$loss_3 = L_2(G3, \widehat{G}3)$$

$\widehat{G}5$

$\widehat{G}4$

$\widehat{G}3$

$\widehat{G}2$

$\widehat{G}1$

Conv5_1 Feature maps

vgg-19

Conv4_1 Feature maps

vgg-19

Conv3_1 Feature maps

vgg-19

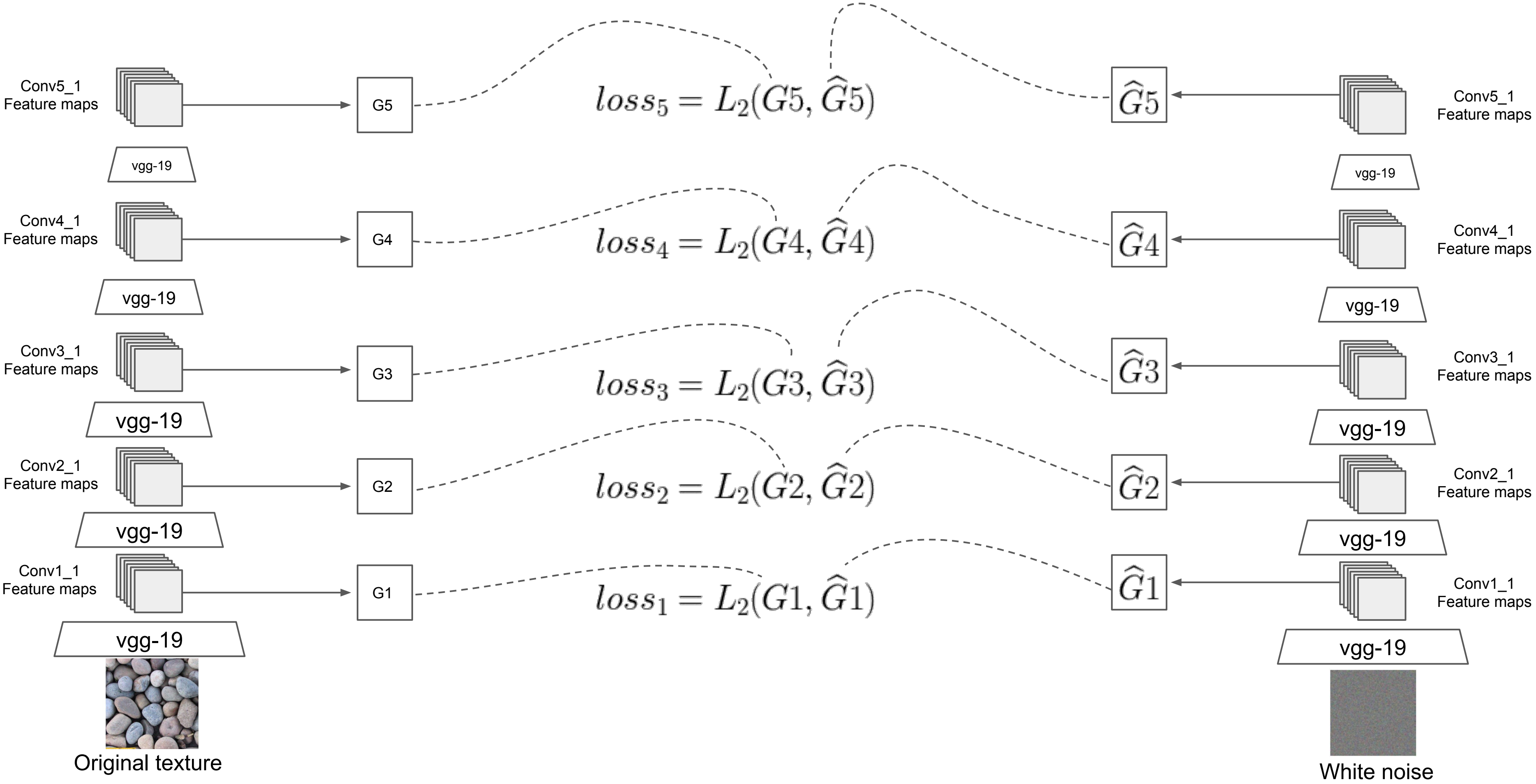
Conv2_1 Feature maps

vgg-19

Conv1_1 Feature maps

vgg-19

White noise



Conv5_1 Feature maps → G5

vgg-19

Conv4_1 Feature maps → G4

vgg-19

Conv3_1 Feature maps → G3

vgg-19

Conv2_1 Feature maps → G2

vgg-19

Conv1_1 Feature maps → G1

vgg-19

Original texture

$$loss_5 = L_2(G5, \widehat{G}5)$$

$$loss_4 = L_2(G4, \widehat{G}4)$$

$$loss_3 = L_2(G3, \widehat{G}3)$$

$$loss_2 = L_2(G2, \widehat{G}2)$$

$$loss_1 = L_2(G1, \widehat{G}1)$$

$\widehat{G}5$ ← Conv5_1 Feature maps

vgg-19

$\widehat{G}4$ ← Conv4_1 Feature maps

vgg-19

$\widehat{G}3$ ← Conv3_1 Feature maps

vgg-19

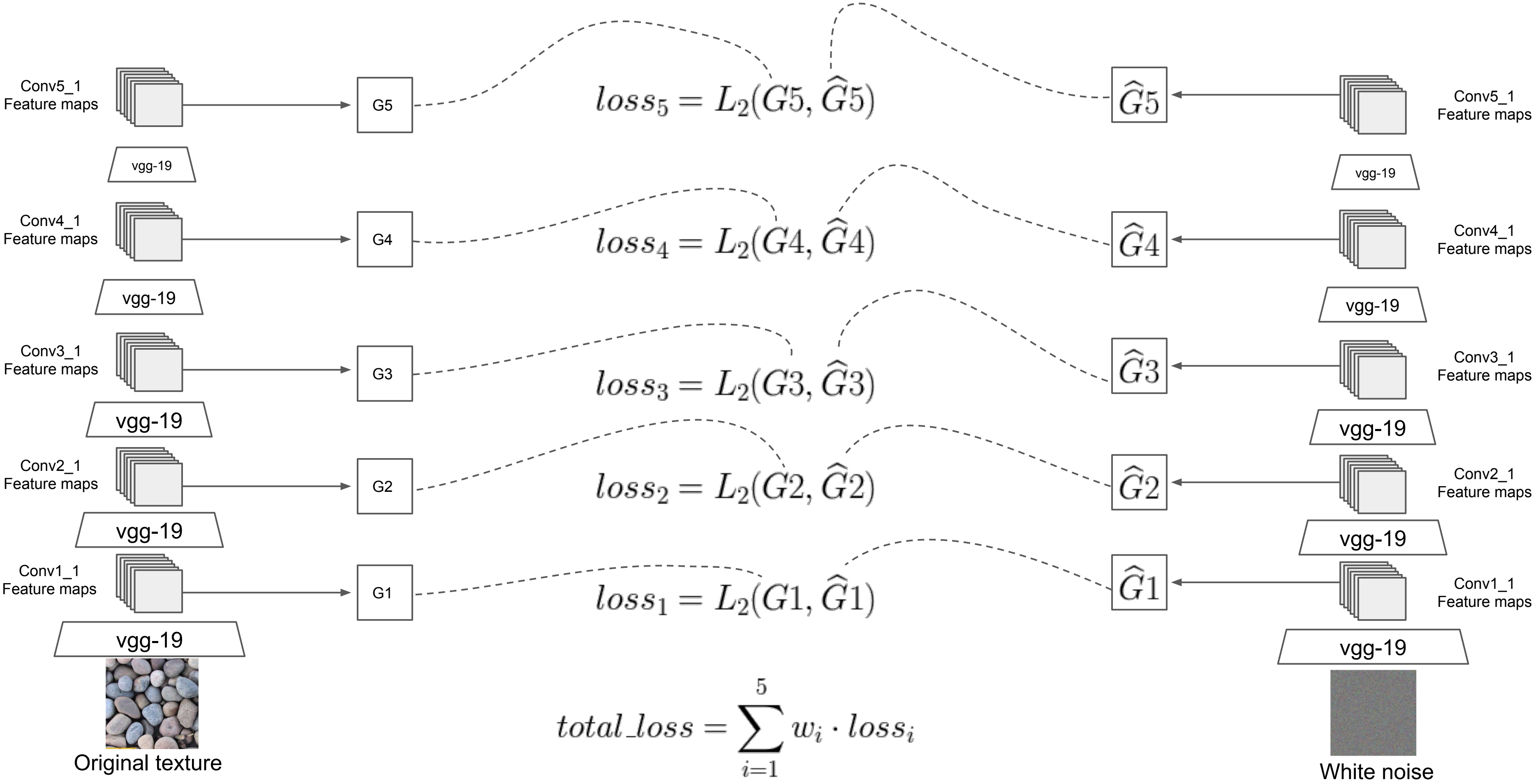
$\widehat{G}2$ ← Conv2_1 Feature maps

vgg-19

$\widehat{G}1$ ← Conv1_1 Feature maps

vgg-19

White noise



Conv5_1
Feature maps

vgg-19

Conv4_1
Feature maps

vgg-19

Conv3_1
Feature maps

vgg-19

Conv2_1
Feature maps

vgg-19

Conv1_1
Feature maps

vgg-19

Original texture

G5

G4

G3

G2

G1

$$loss_5 = L_2(G5, \widehat{G}5)$$

$$loss_4 = L_2(G4, \widehat{G}4)$$

$$loss_3 = L_2(G3, \widehat{G}3)$$

$$loss_2 = L_2(G2, \widehat{G}2)$$

$$loss_1 = L_2(G1, \widehat{G}1)$$

$$total\_loss = \sum_{i=1}^{5} w_i \cdot loss_i$$

$\widehat{G}5$

$\widehat{G}4$

$\widehat{G}3$

$\widehat{G}2$

$\widehat{G}1$

Conv5_1
Feature maps

vgg-19

Conv4_1
Feature maps

vgg-19

Conv3_1
Feature maps

vgg-19

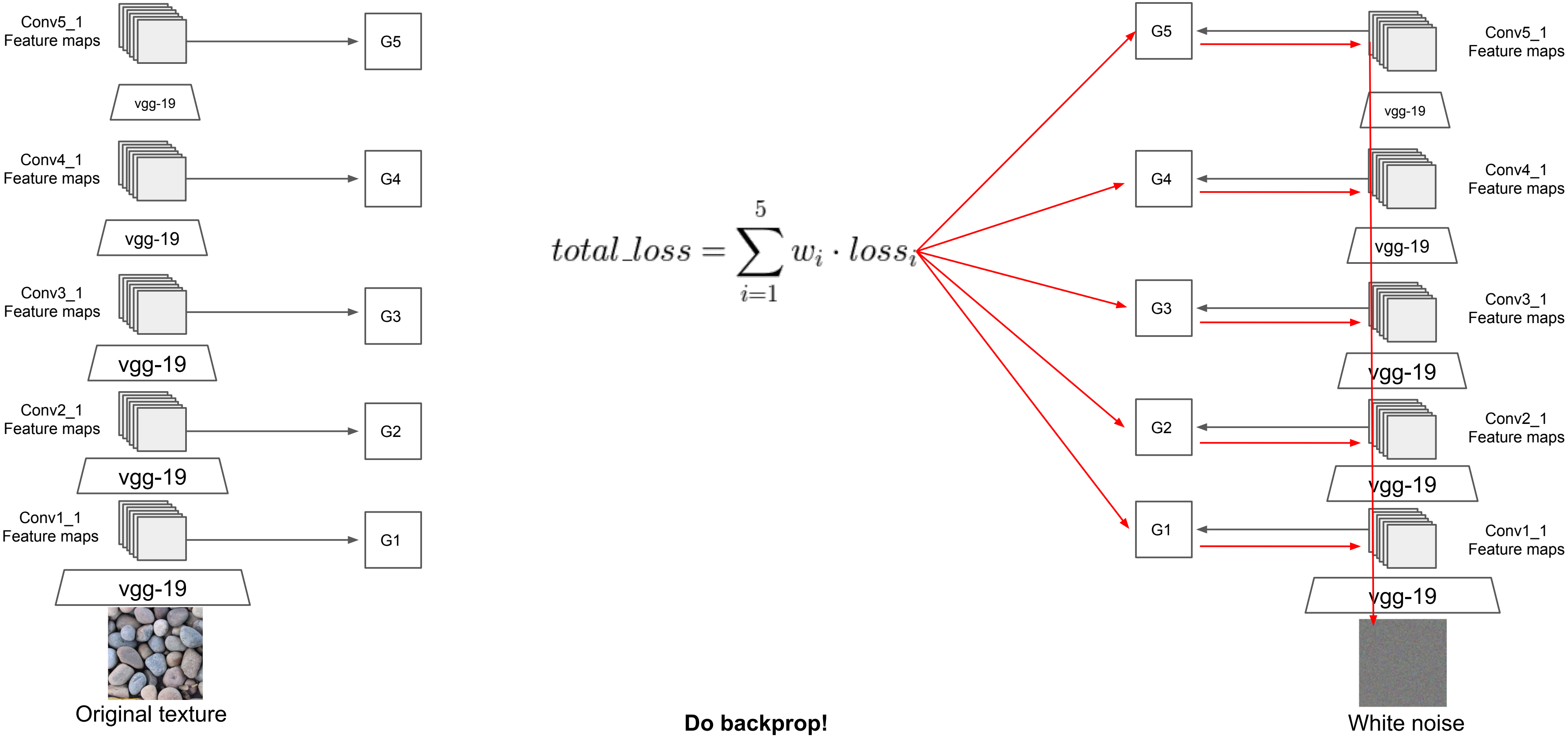
Conv2_1
Feature maps

vgg-19

Conv1_1
Feature maps

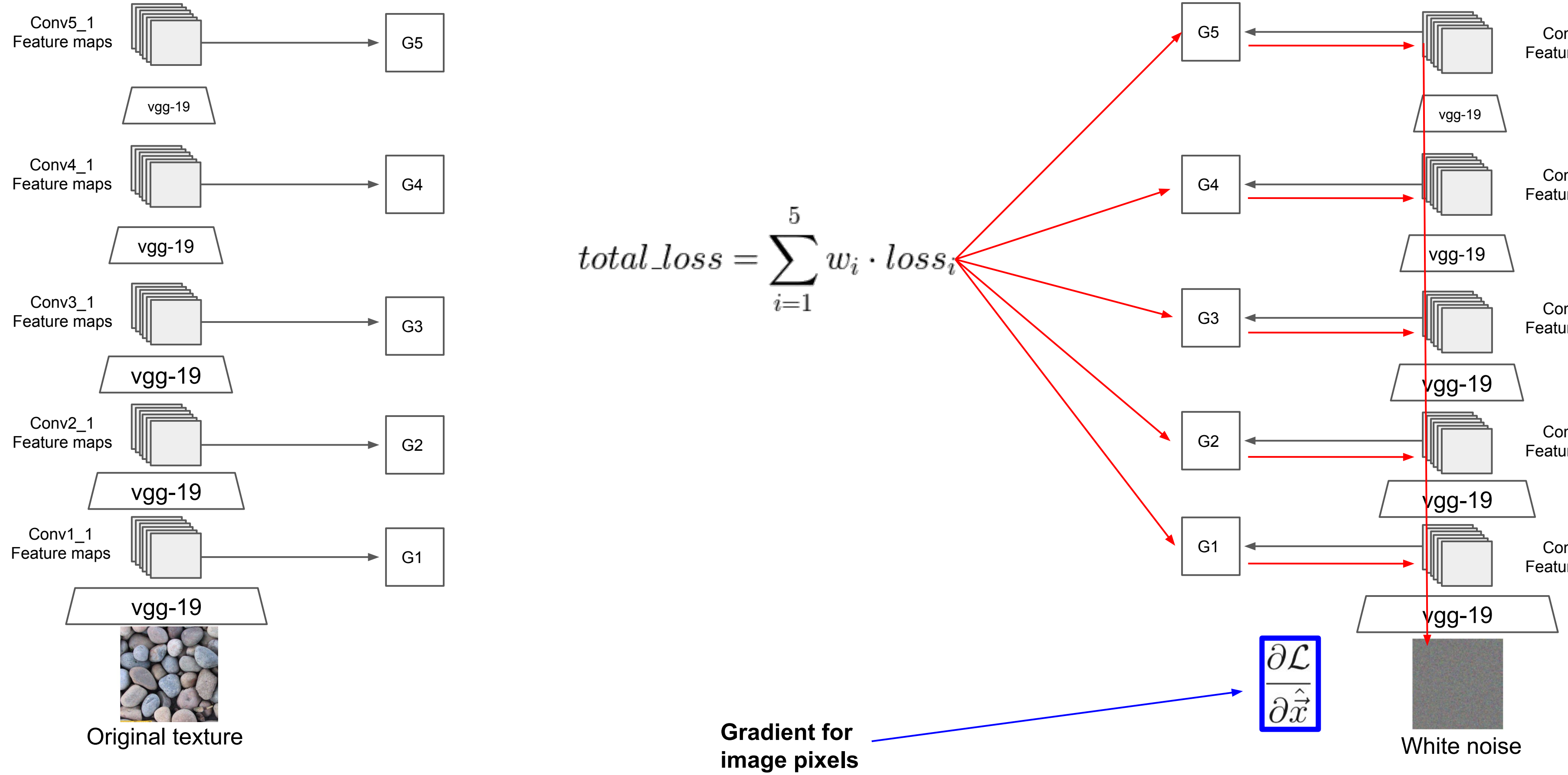
vgg-19

White noise



$$total\_loss = \sum_{i=1}^{5} w_i \cdot loss_i$$

Do backprop!

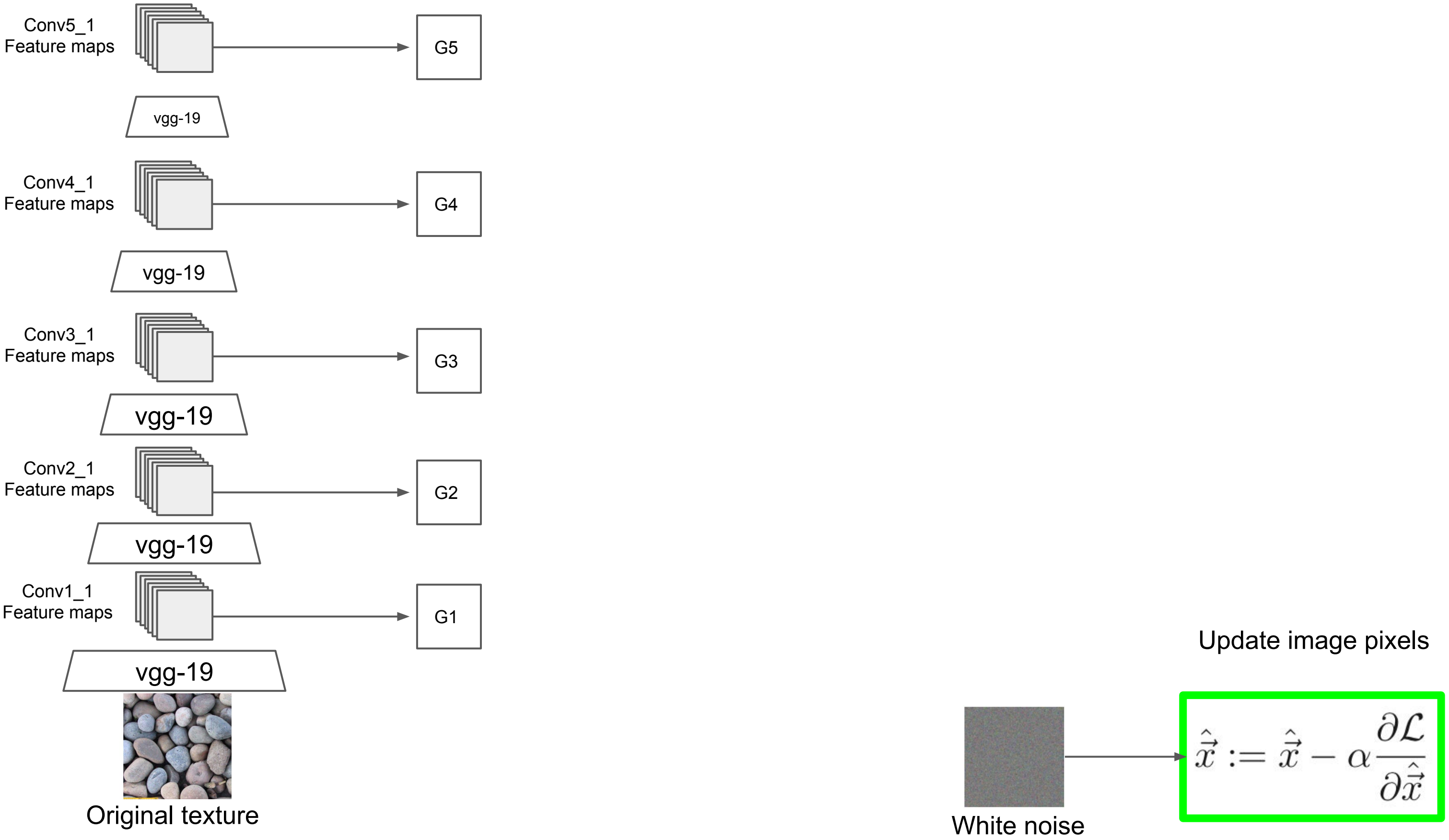


$$total\_loss = \sum_{i=1}^{5} w_i \cdot loss_i$$

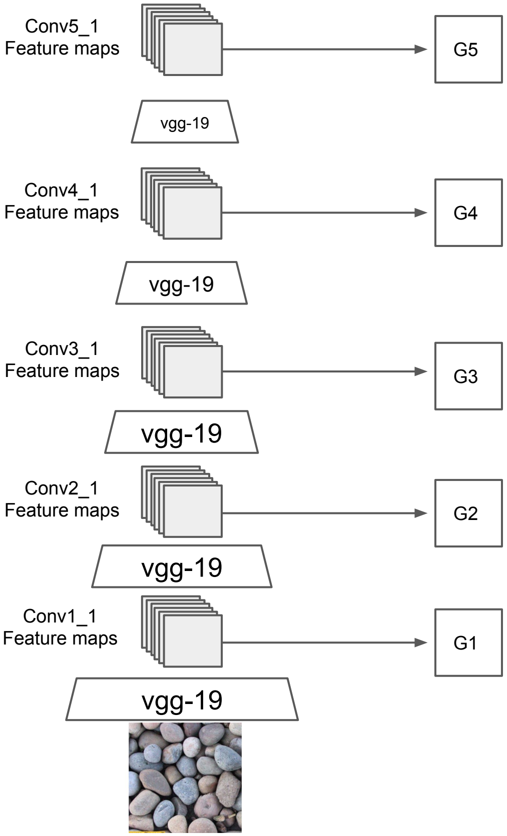
$$\boxed{\frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}}$$

Gradient for image pixels

deepsystems.io



$$\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$$



Conv5_1
Feature maps

G5

vgg-19

Conv4_1
Feature maps

G4

vgg-19

Conv3_1
Feature maps

G3

vgg-19

Conv2_1
Feature maps

G2

vgg-19

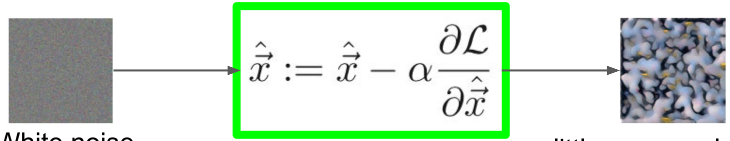
Conv1_1
Feature maps

G1

vgg-19

Original texture

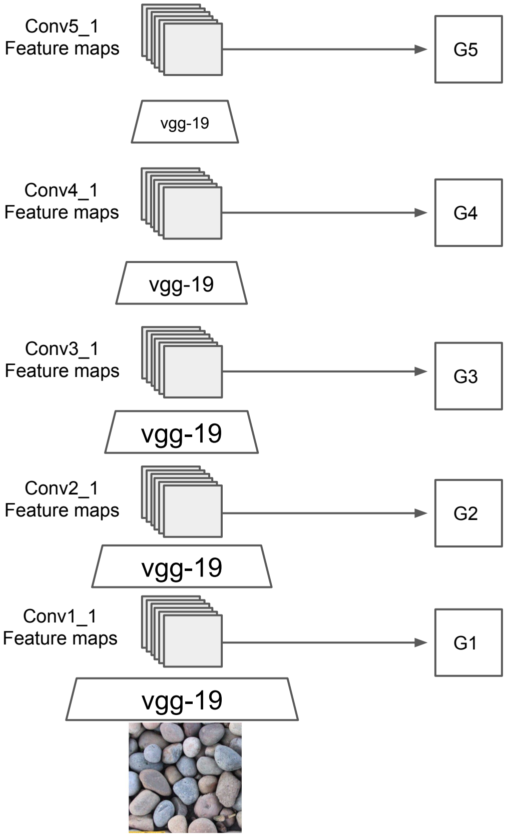
Update image pixels



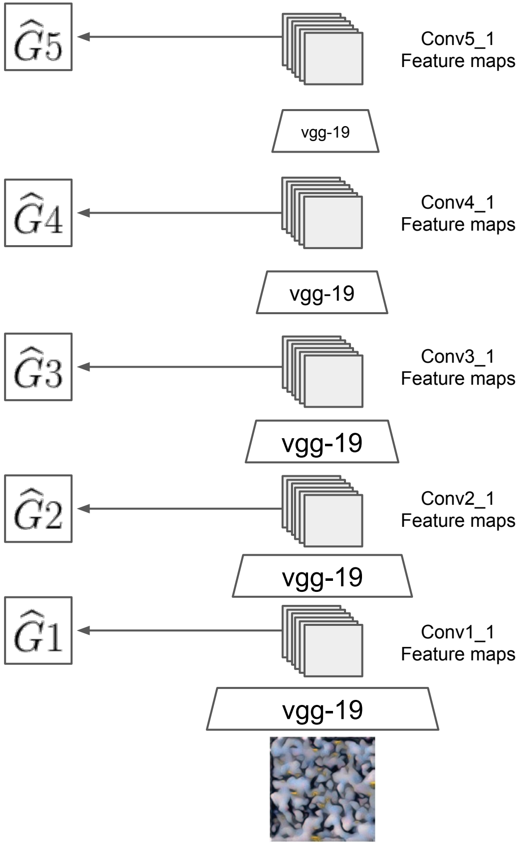
$$\hat{\vec{x}} := \hat{\vec{x}} - \alpha \frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}$$

White noise

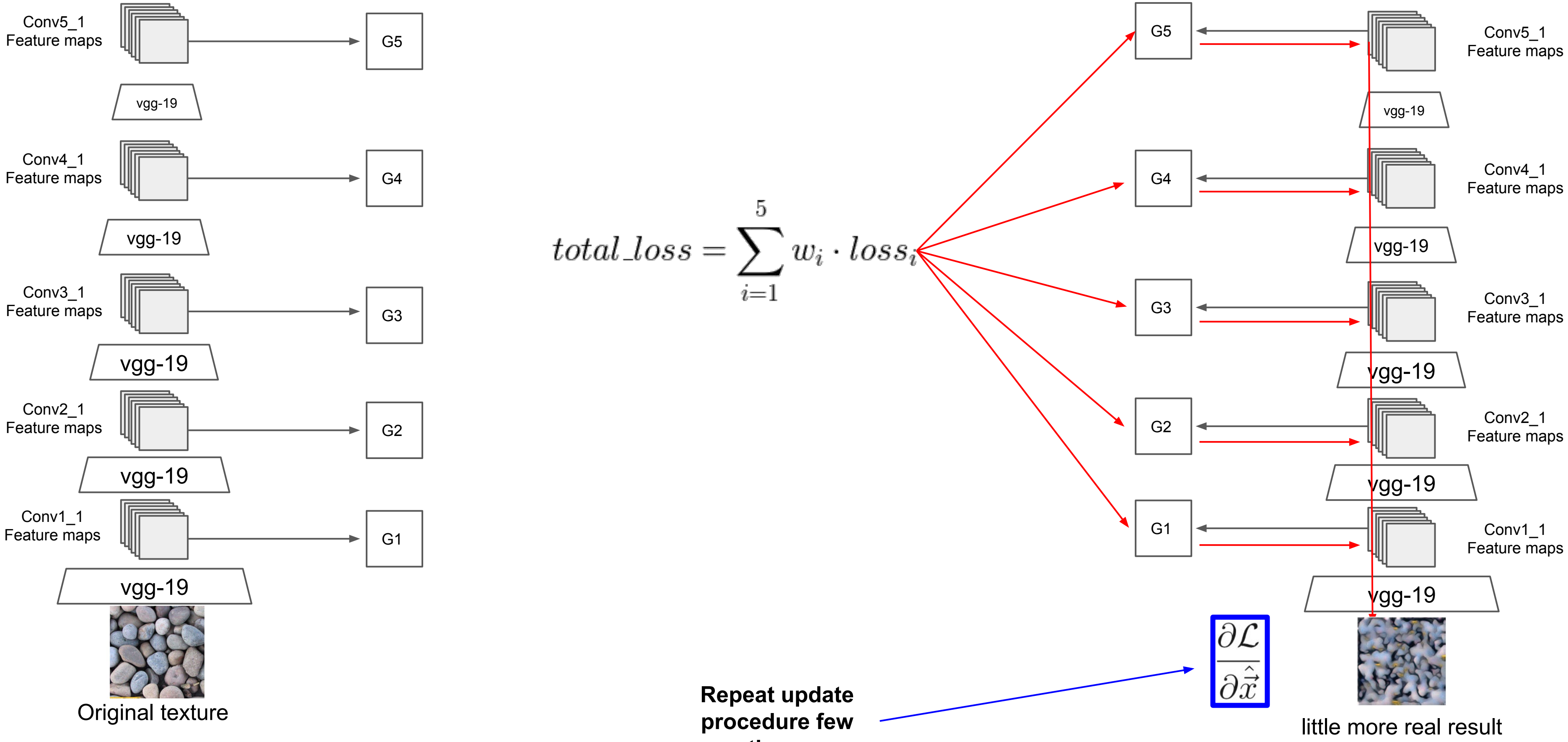
little more real result



Original texture



little more real result



Original texture

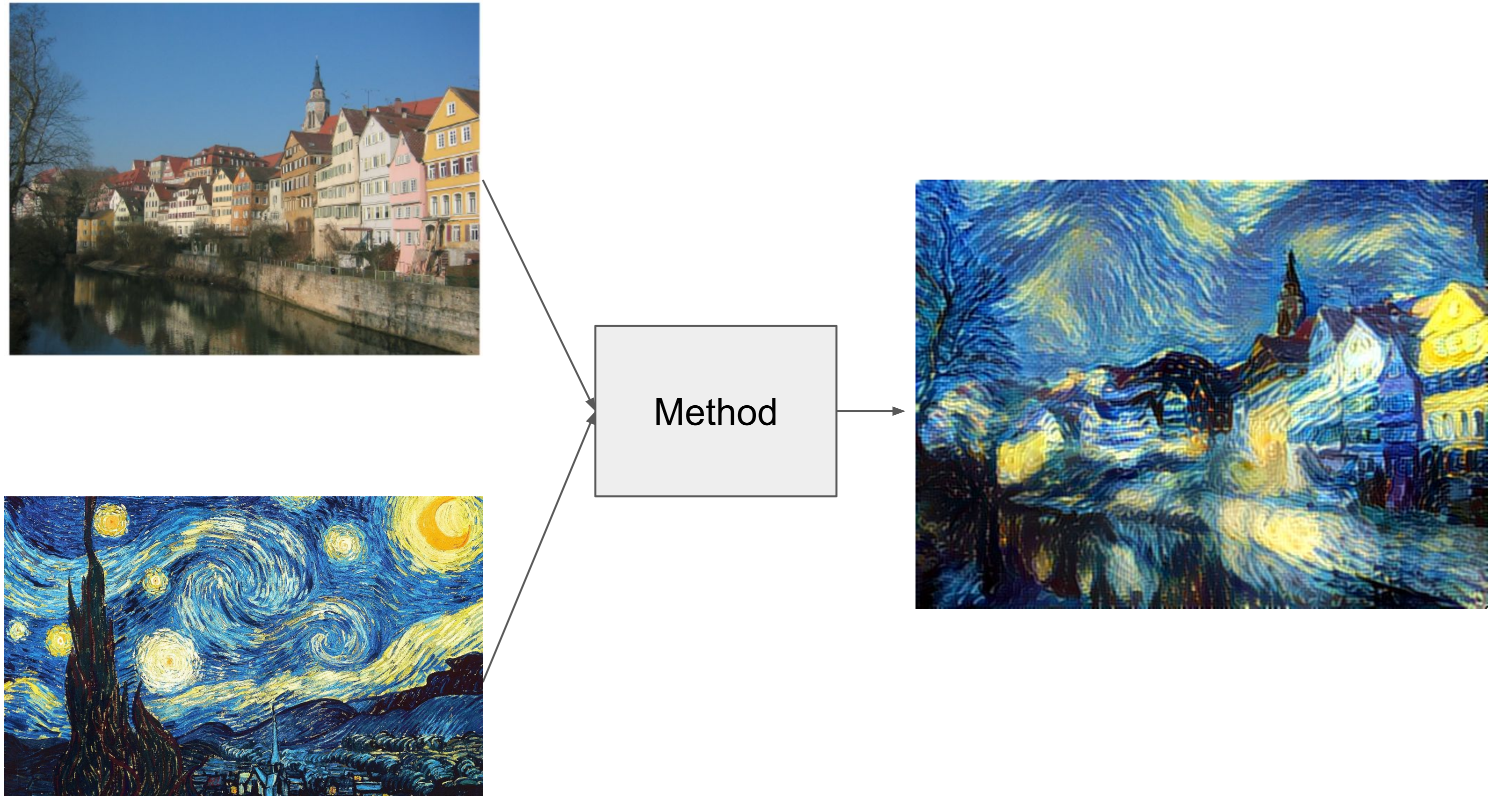
$$total\_loss = \sum_{i=1}^{5} w_i \cdot loss_i$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial \hat{\vec{x}}}}$$

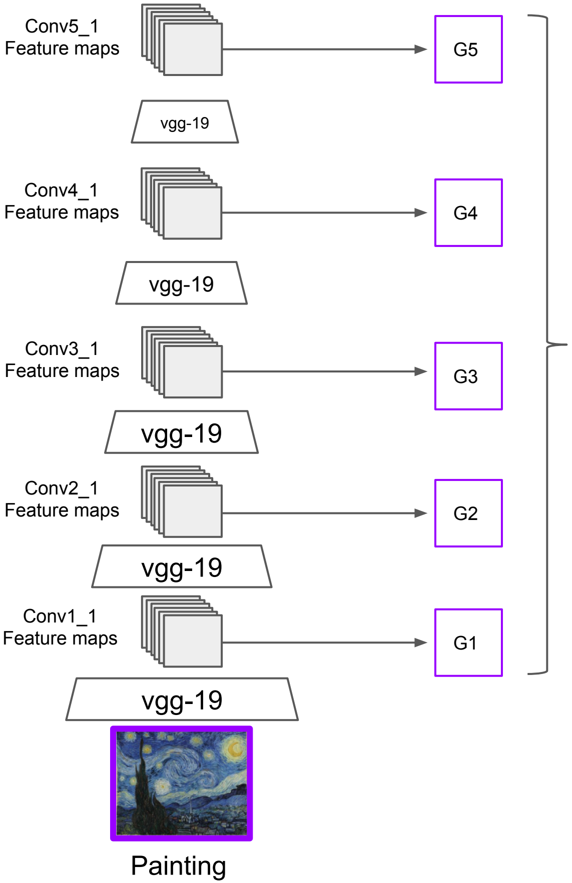
Repeat update procedure few times

little more real result

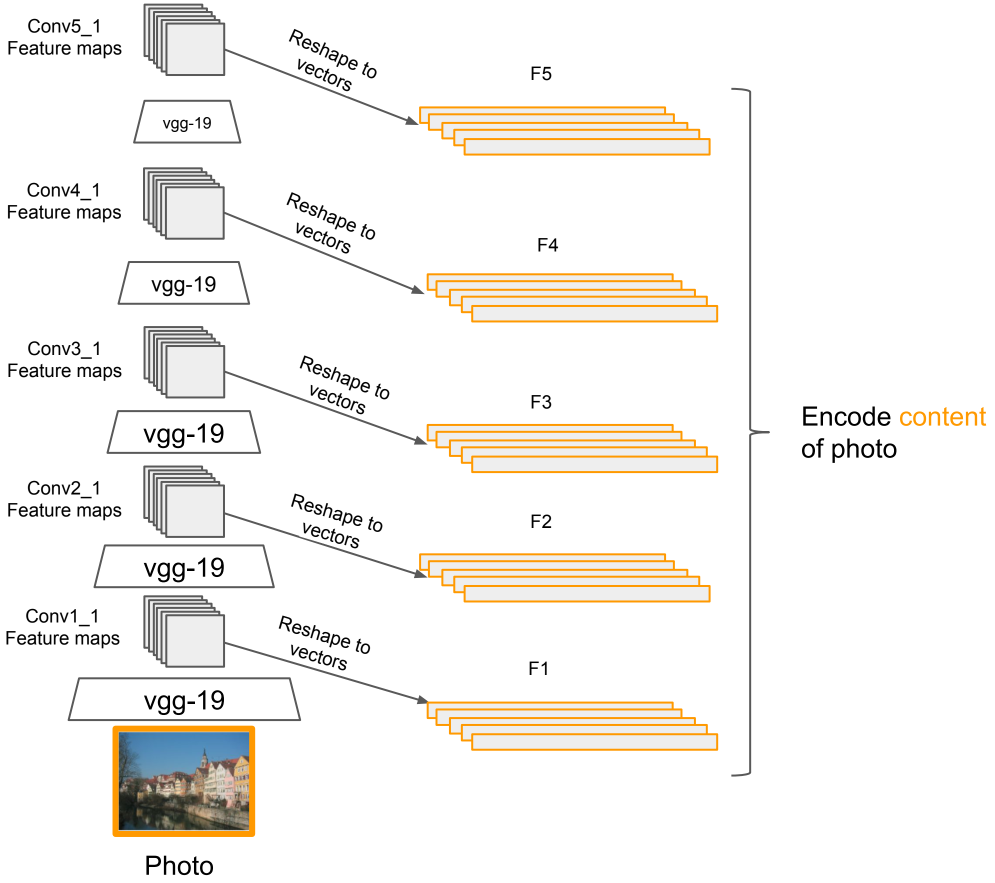
A neural algorithm for artistic style

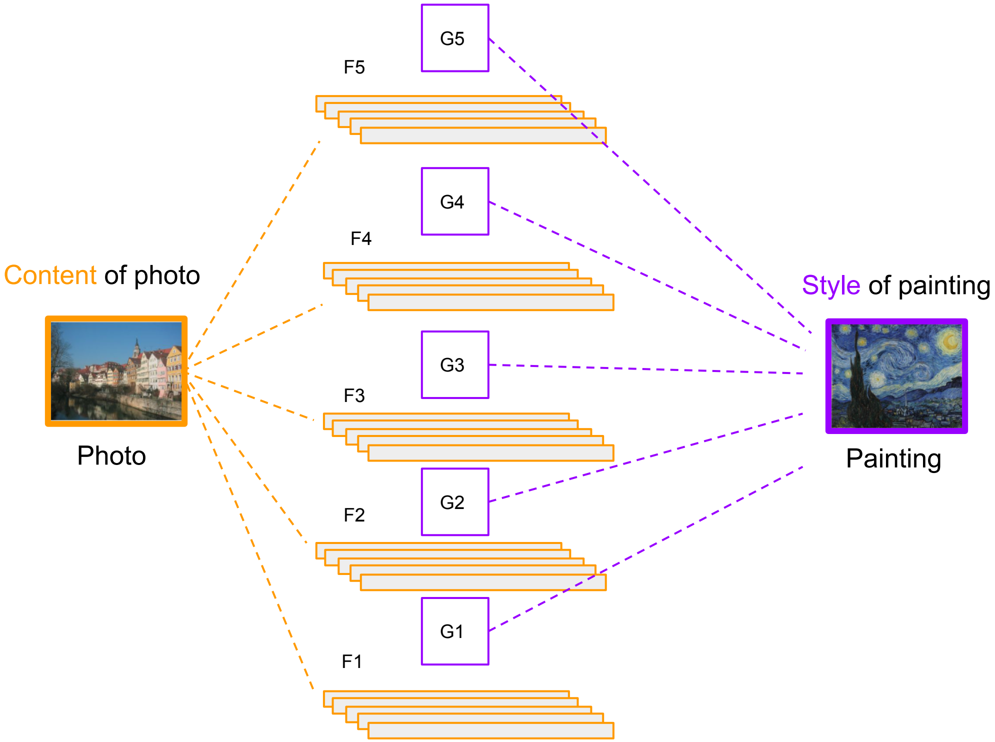


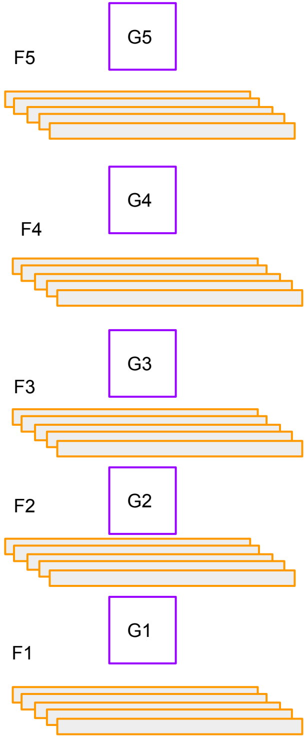
Method



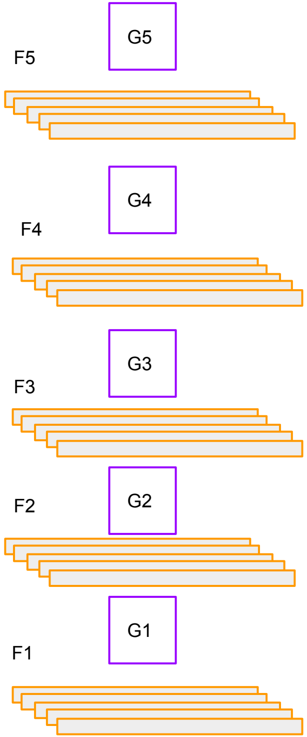
Encode style of painting

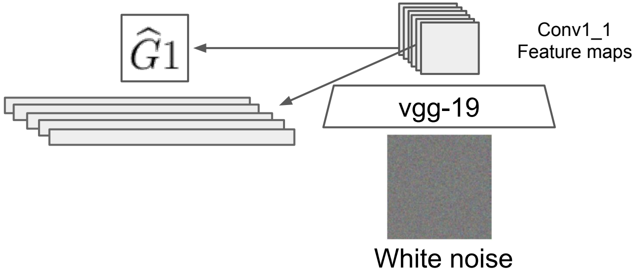

Conv5_1
Feature maps
Reshape to vectors
F5
vgg-19
Conv4_1
Feature maps
Reshape to vectors
F4
vgg-19
Conv3_1
Feature maps
Reshape to vectors
F3
vgg-19
Conv2_1
Feature maps
Reshape to vectors
F2
vgg-19
Conv1_1
Feature maps
Reshape to vectors
F1
vgg-19
Photo
Encode content
of photo

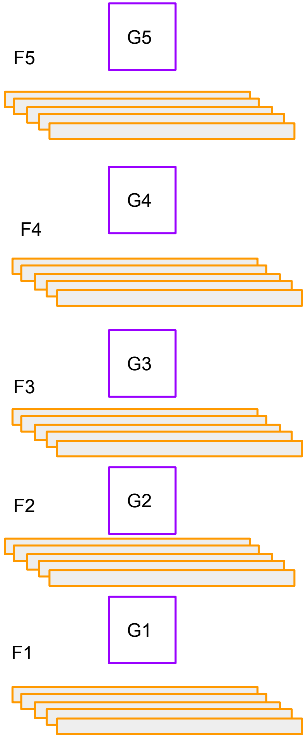

G5
F5
G4
F4
Content of photo
Style of painting
G3
F3
Photo
Painting
G2
F2
G1
F1

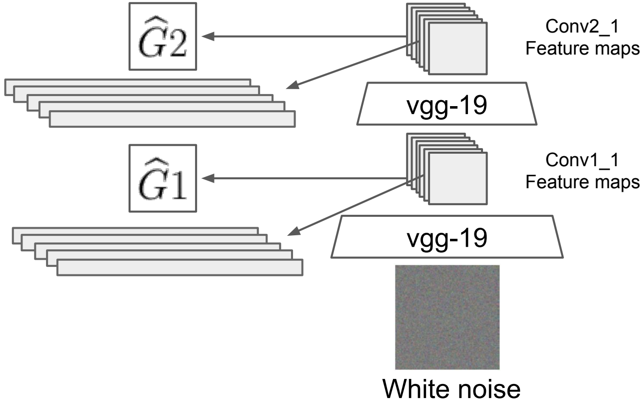


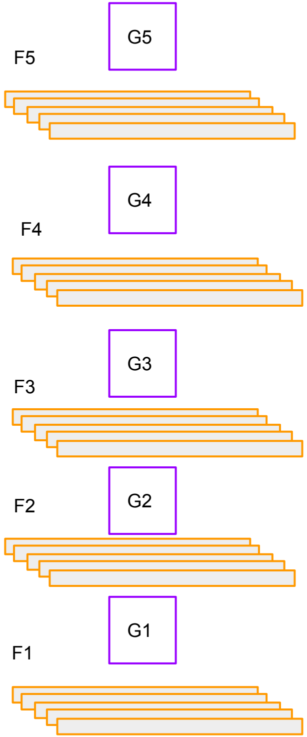
White noise

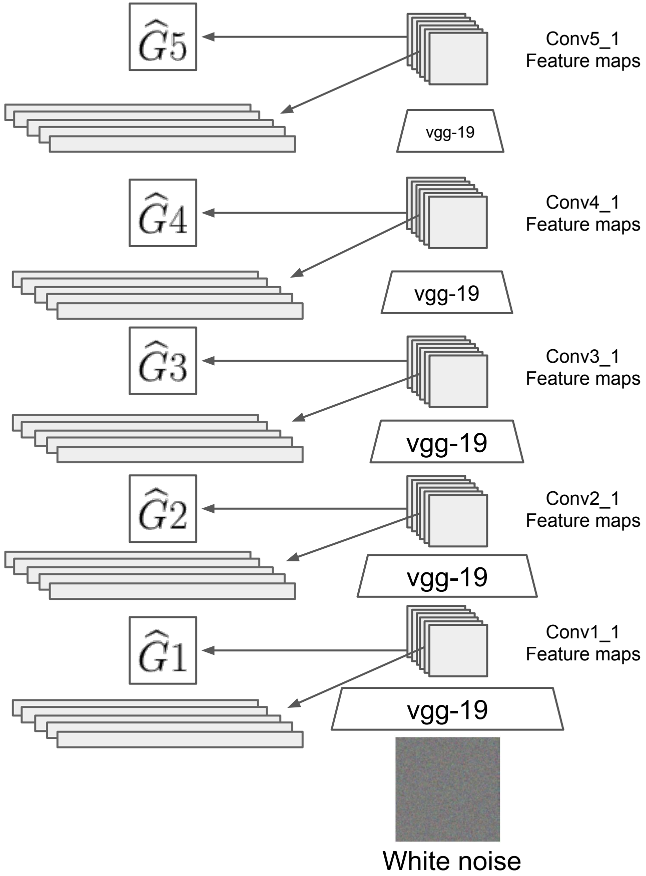

G5
F5
G4
F4
G3
F3
G2
F2
G1
F1

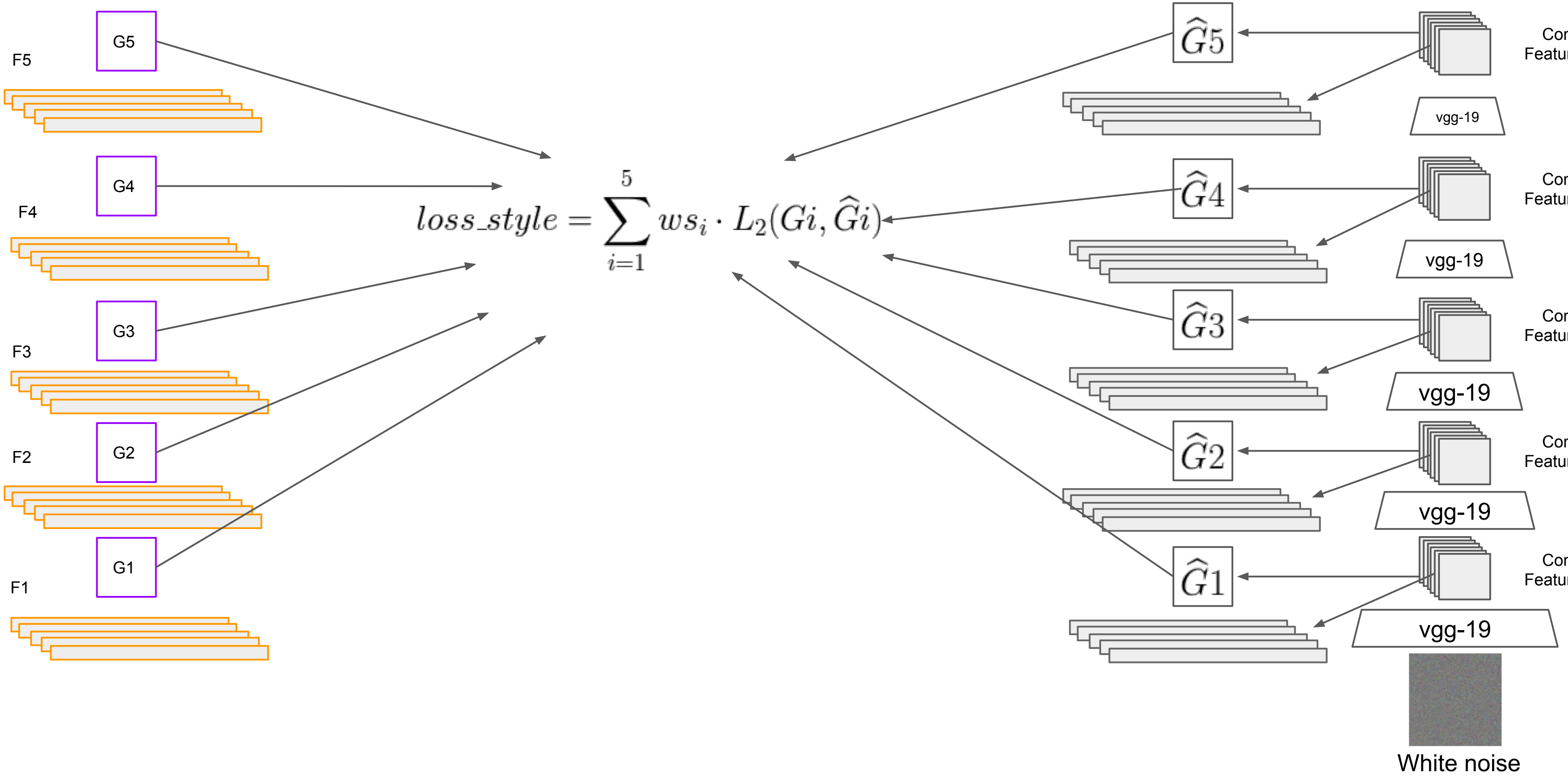

$\widehat{G}1$
Conv1_1
Feature maps
vgg-19
White noise


G5
F5
G4
F4
G3
F3
G2
F2
G1
F1

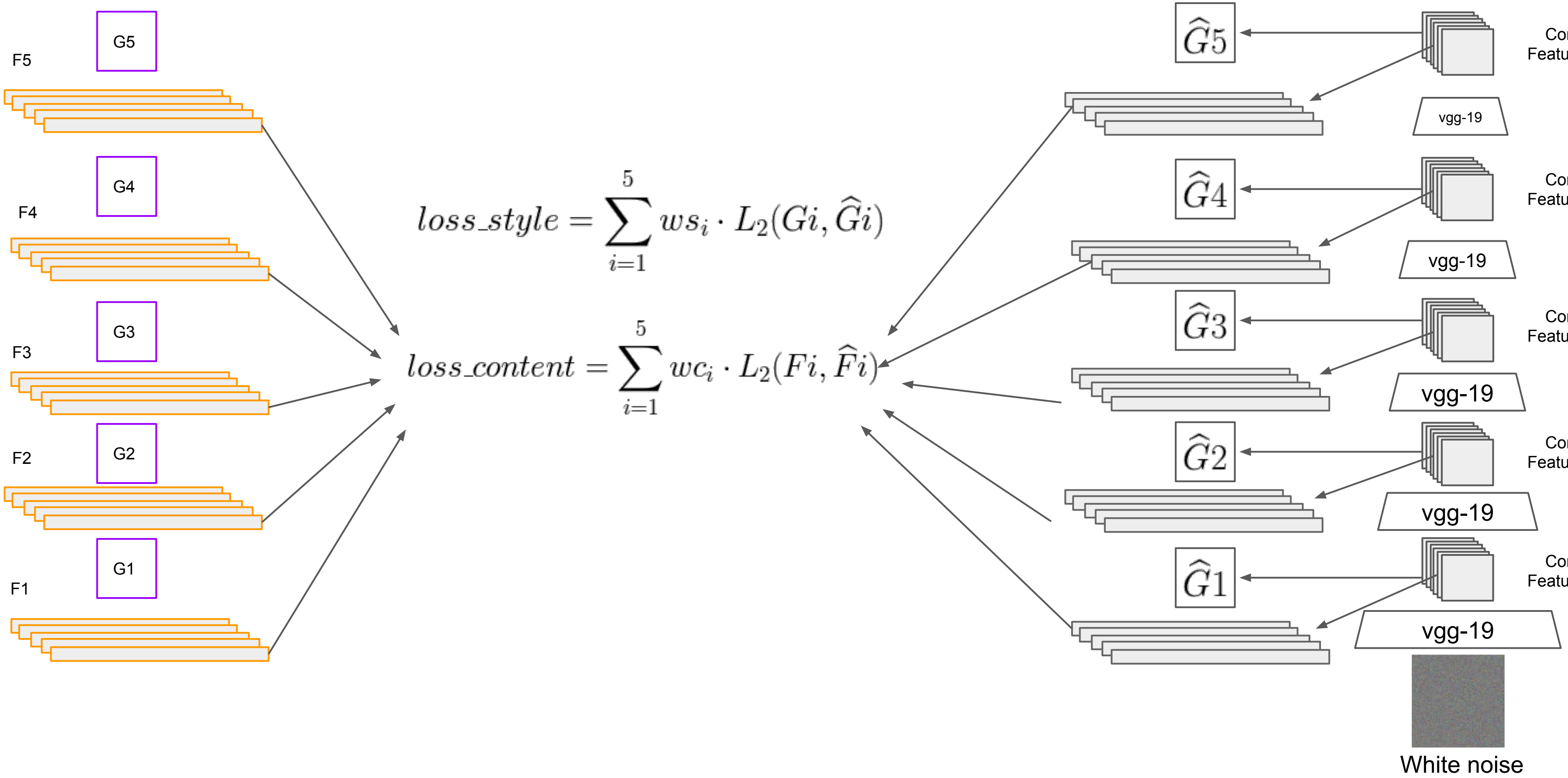

$\widehat{G}2$
Conv2_1
Feature maps
vgg-19
$\widehat{G}1$
Conv1_1
Feature maps
vgg-19
White noise


G5
F5
G4
F4
G3
F3
G2
F2
G1
F1


$\widehat{G}5$
Conv5_1
Feature maps
vgg-19
$\widehat{G}4$
Conv4_1
Feature maps
vgg-19
$\widehat{G}3$
Conv3_1
Feature maps
vgg-19
$\widehat{G}2$
Conv2_1
Feature maps
vgg-19
$\widehat{G}1$
Conv1_1
Feature maps
vgg-19
White noise

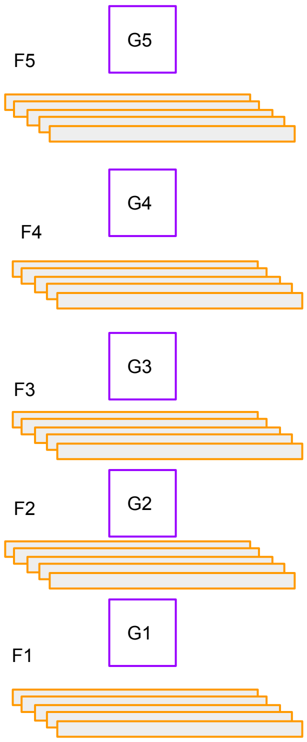


$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{G}i)$$



$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{Gi})$$

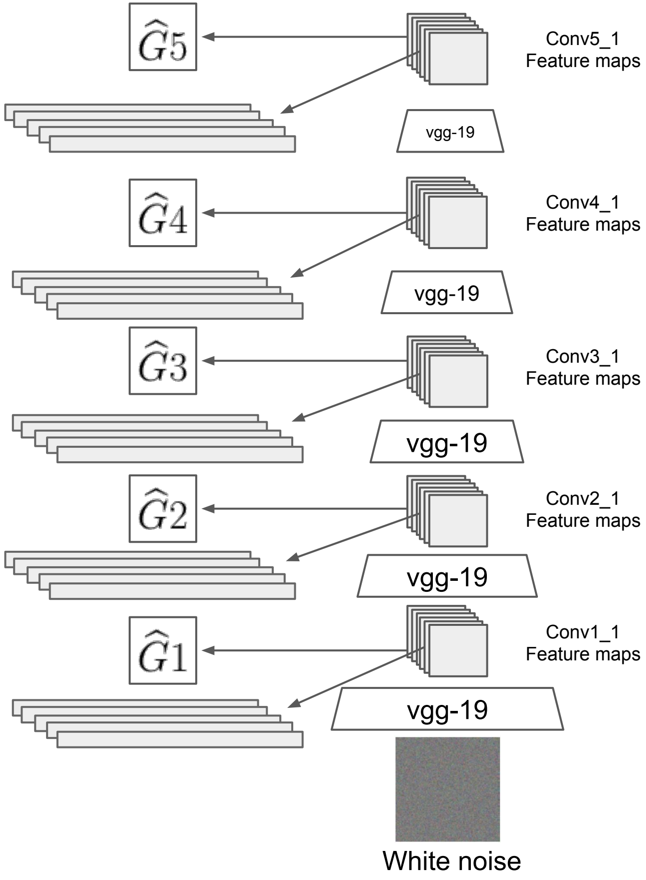
$$loss\_content = \sum_{i=1}^{5} wc_i \cdot L_2(Fi, \widehat{Fi})$$

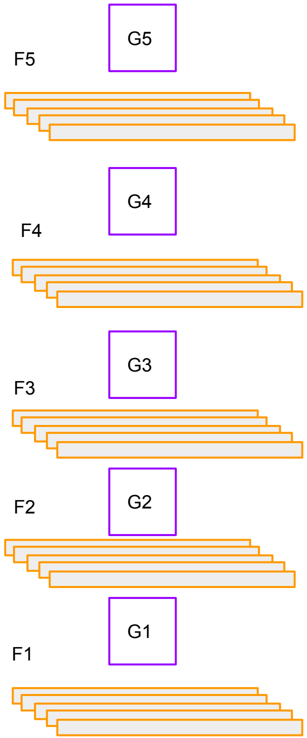


$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{G}i)$$

$$loss\_content = \sum_{i=1}^{5} wc_i \cdot L_2(Fi, \widehat{F}i)$$

$$total\_loss = \alpha \cdot loss\_content + \beta \cdot loss\_style$$

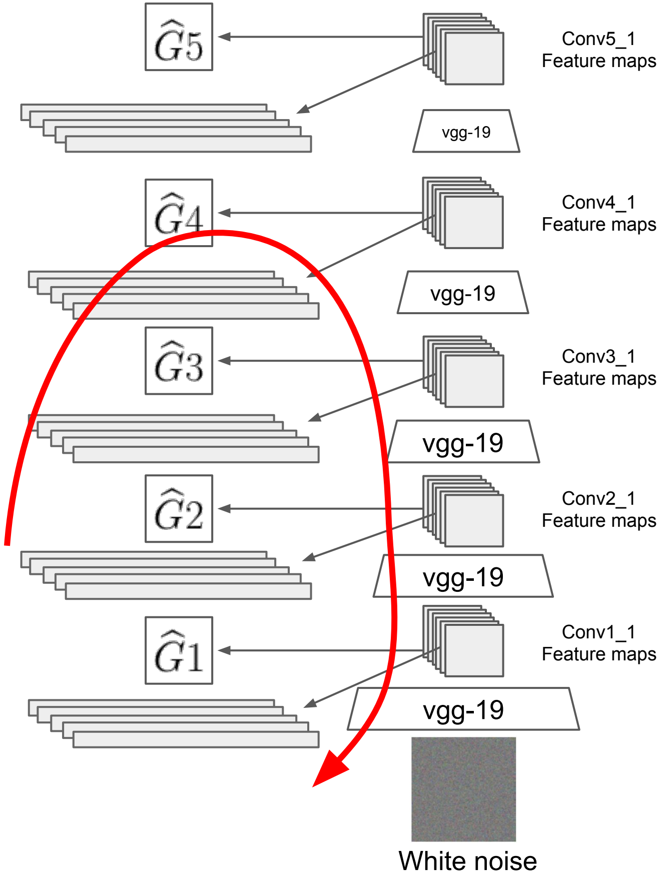




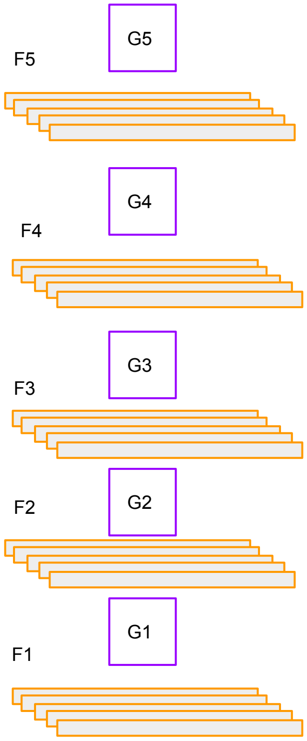
$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{Gi})$$

$$loss\_content = \sum_{i=1}^{5} wc_i \cdot L_2(Fi, \widehat{Fi})$$

$$total\_loss = \alpha \cdot loss\_content + \beta \cdot loss\_style$$



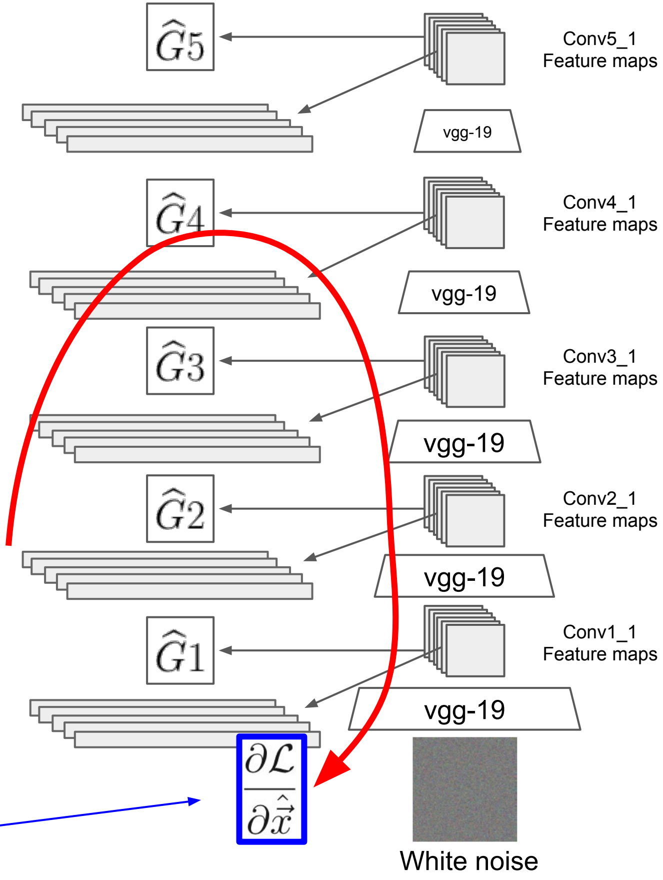
Do backprop!



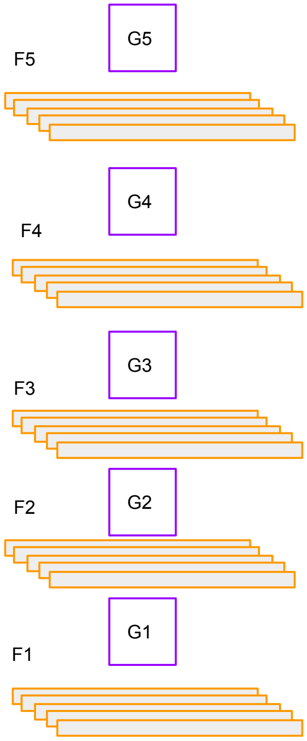
$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{Gi})$$

$$loss\_content = \sum_{i=1}^{5} wc_i \cdot L_2(Fi, \widehat{Fi})$$

$$total\_loss = \alpha \cdot loss\_content + \beta \cdot loss\_style$$



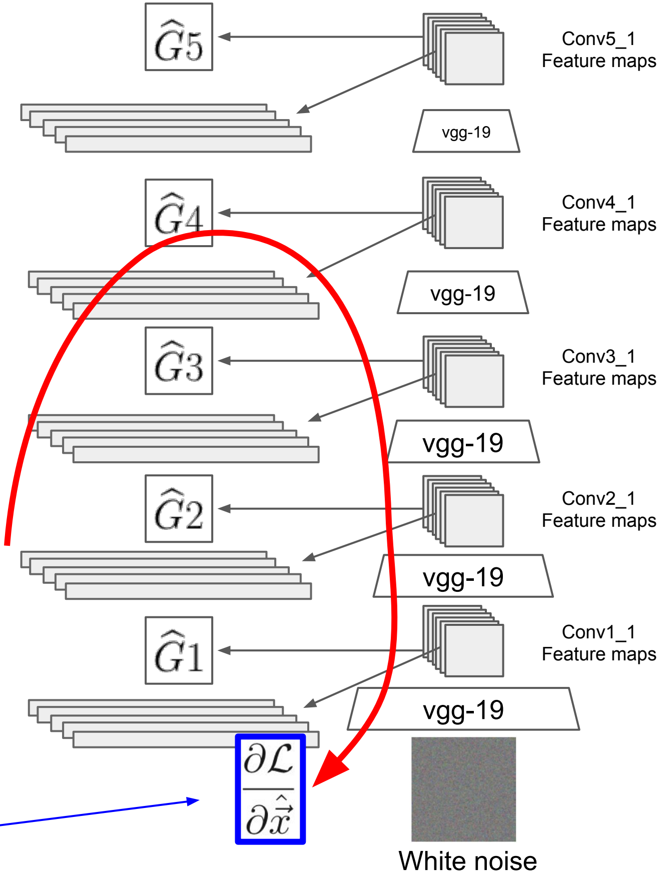
Gradient for image pixels



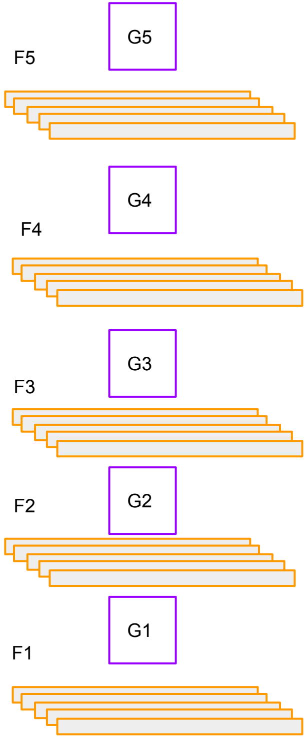
$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{Gi})$$

$$loss\_content = \sum_{i=1}^{5} wc_i \cdot L_2(Fi, \widehat{Fi})$$

$$total\_loss = \alpha \cdot loss\_content + \beta \cdot loss\_style$$



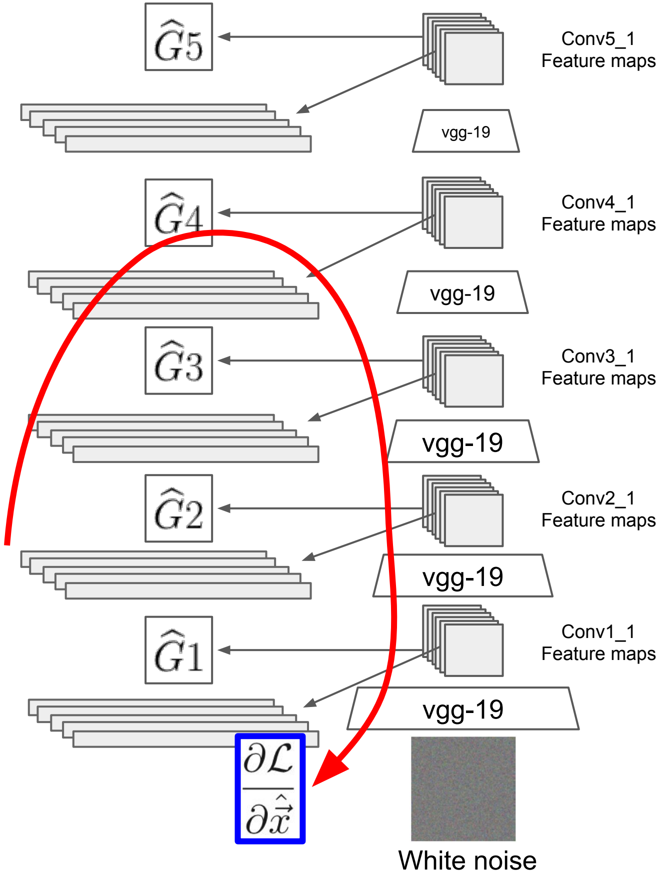
Update only image pixels



$$loss\_style = \sum_{i=1}^{5} ws_i \cdot L_2(Gi, \widehat{G}i)$$

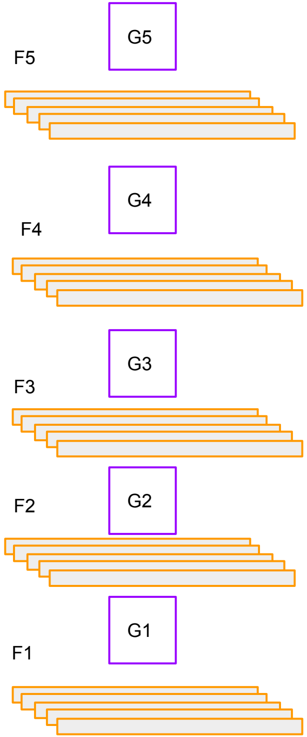
$$loss\_content = \sum_{i=1}^{5} wc_i \cdot L_2(Fi, \widehat{F}i)$$

$$total\_loss = \alpha \cdot loss\_content + \beta \cdot loss\_style$$

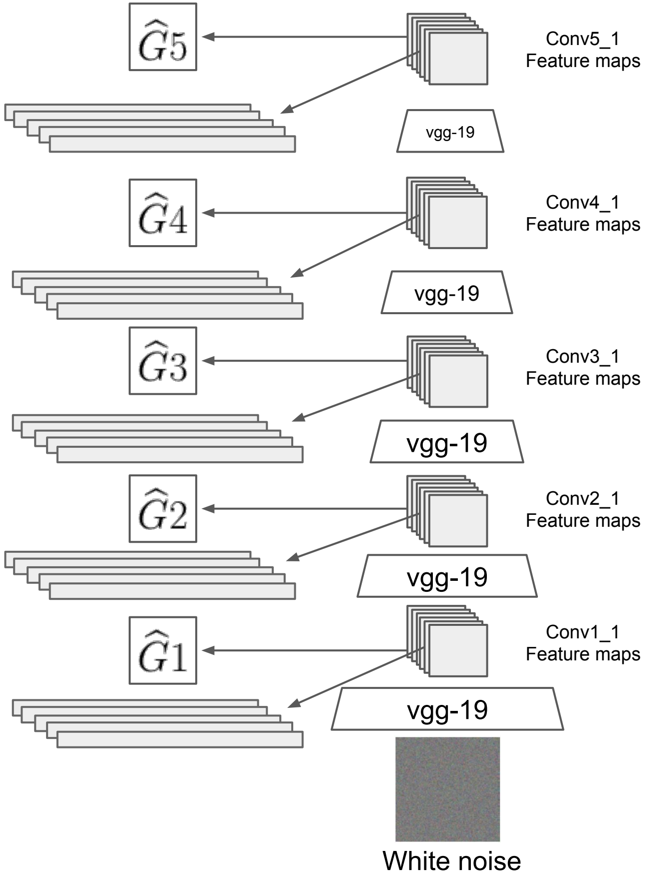


Repeat several times

Intuition



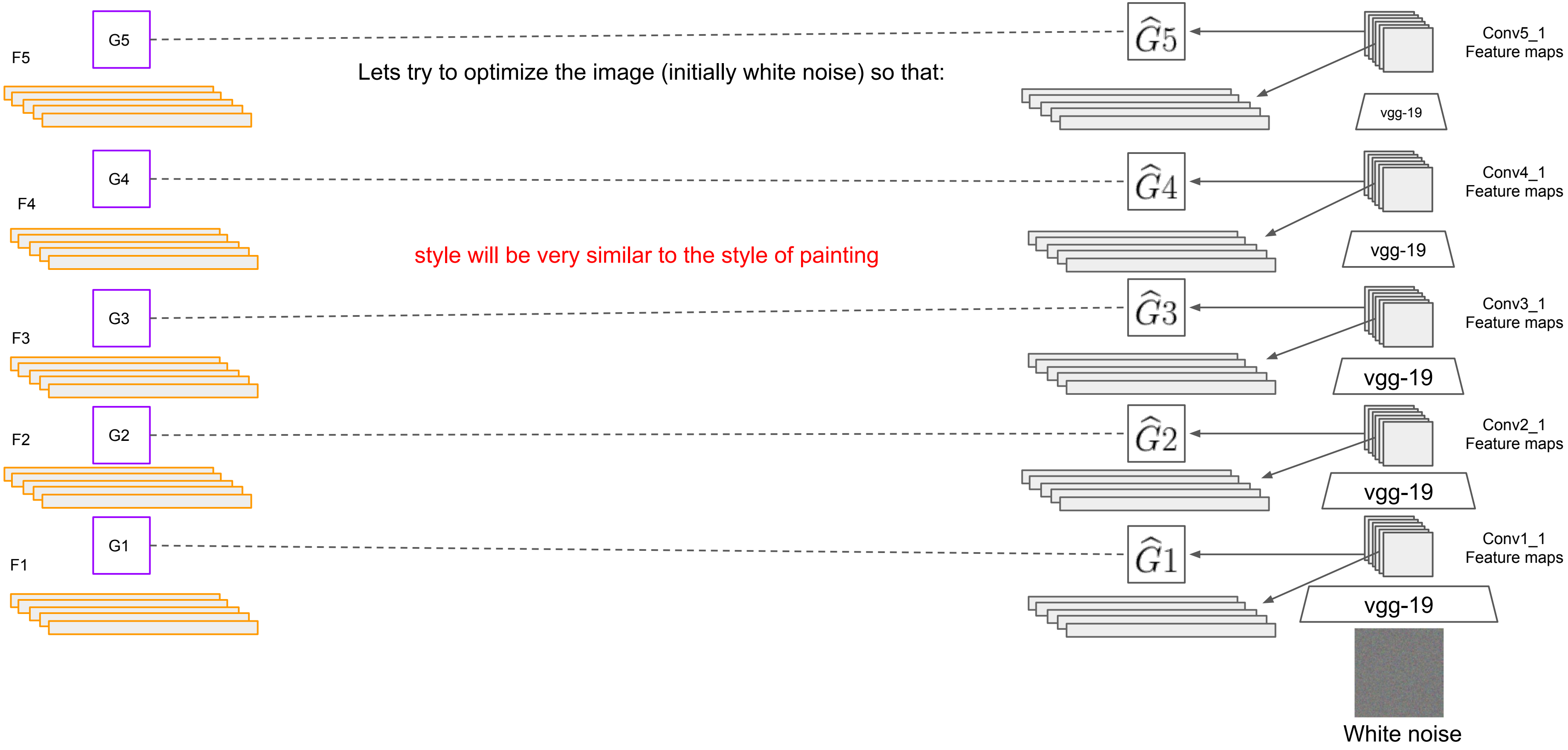
Lets try to optimize the image (initially white noise) so that:



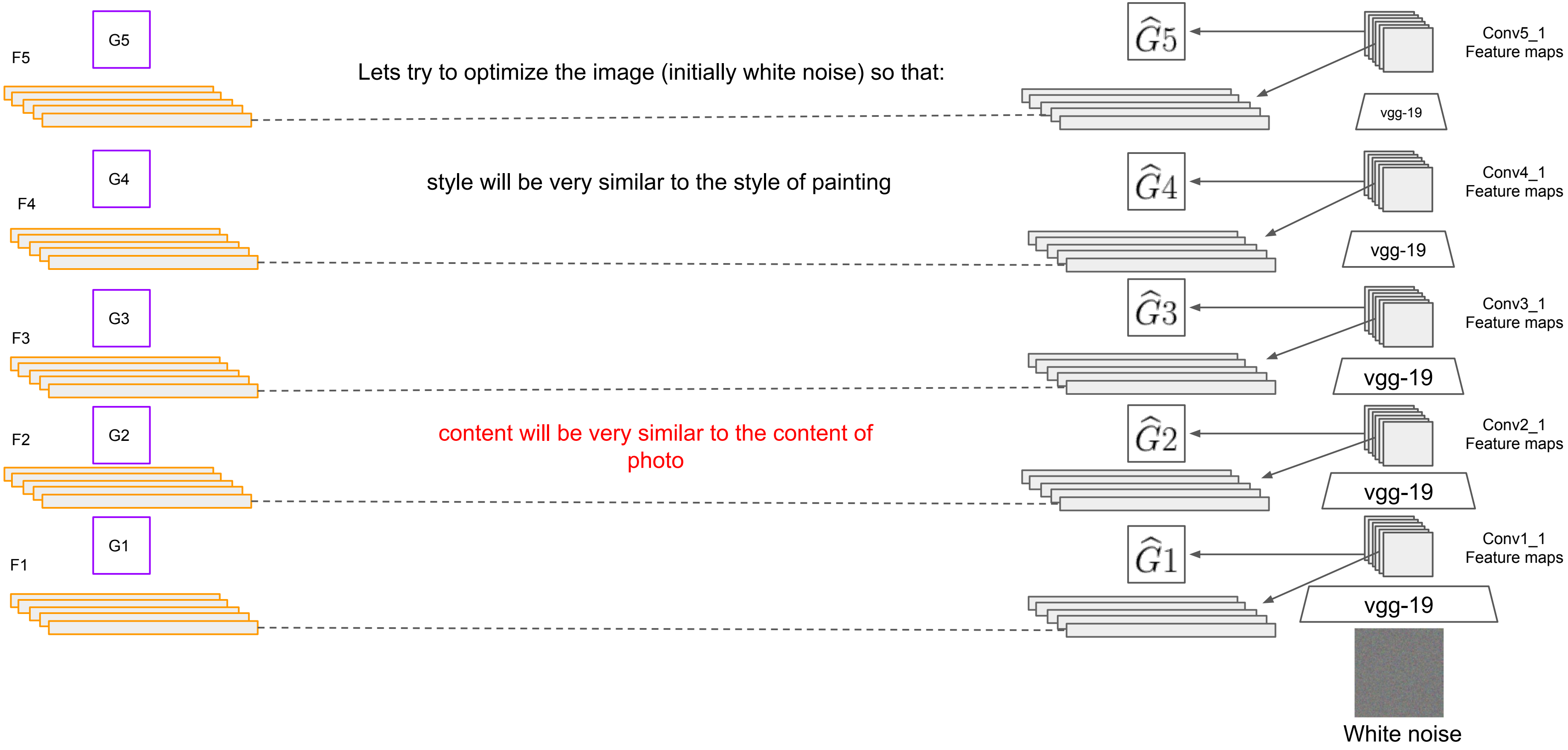
Intuition

Lets try to optimize the image (initially white noise) so that:

style will be very similar to the style of painting



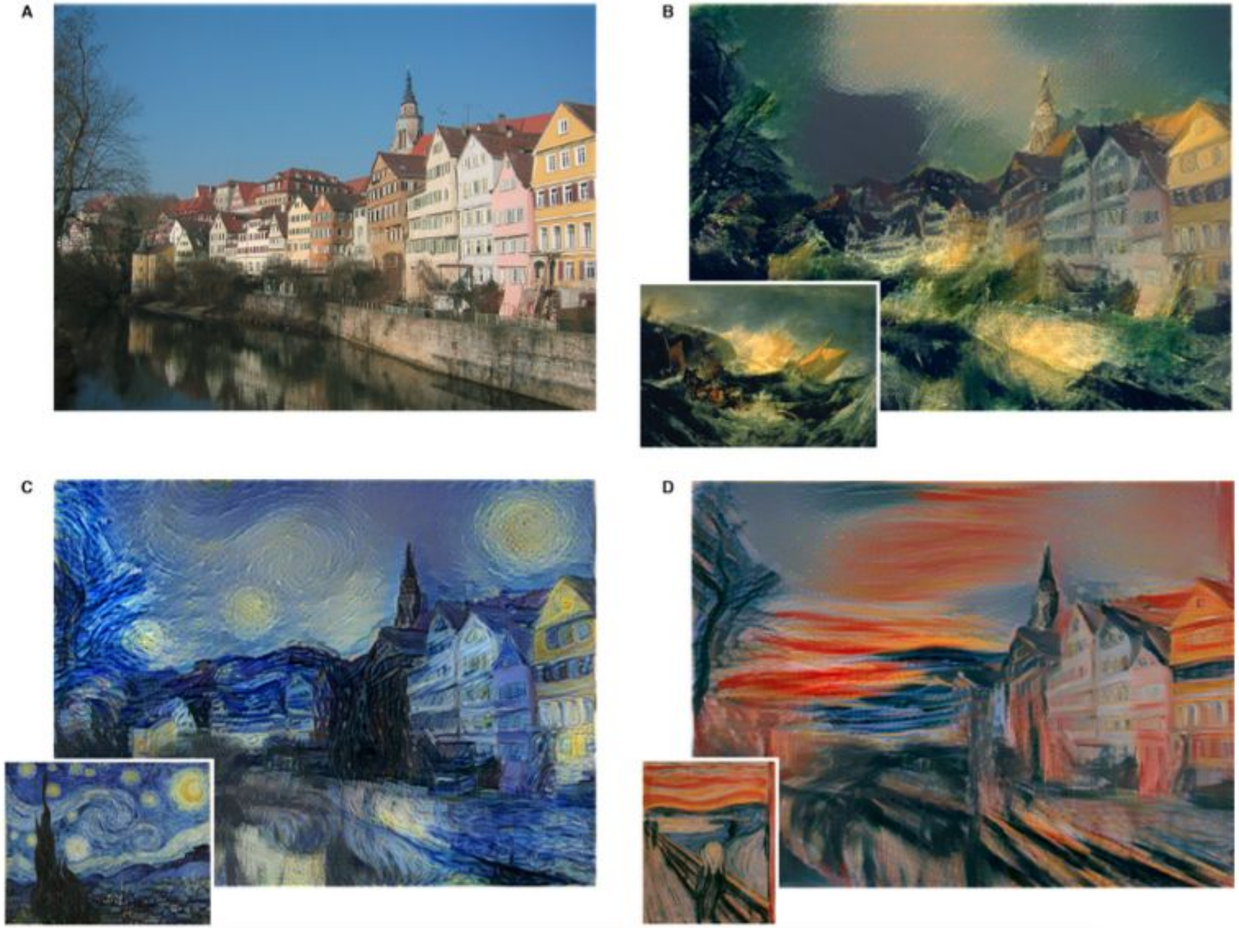
Intuition



Lets try to optimize the image (initially white noise) so that:

style will be very similar to the style of painting

content will be very similar to the content of photo


A
B
C
D

Key points

1. NeuralStyle was a start point for many research and commercial applications (prisma)
2. Non trivial example of transfer learning (deep learning is more than just classification or regression)
3. Object recognition model (VGG) is invariant to different object representations (style, shape, background, etc) => content and style can be divided

Links

- Arxiv (texture synthesis): https://arxiv.org/abs/1505.07376
- Arxiv (neural style): https://arxiv.org/abs/1508.06576
- Torch (neural style): https://github.com/jcjohnson/neural-style
- Torch (fast neural style): https://github.com/jcjohnson/fast-neural-style

Thank you!

Our website: deepsystems.io

Our team is looking for business partners to make exciting deep learning solutions.