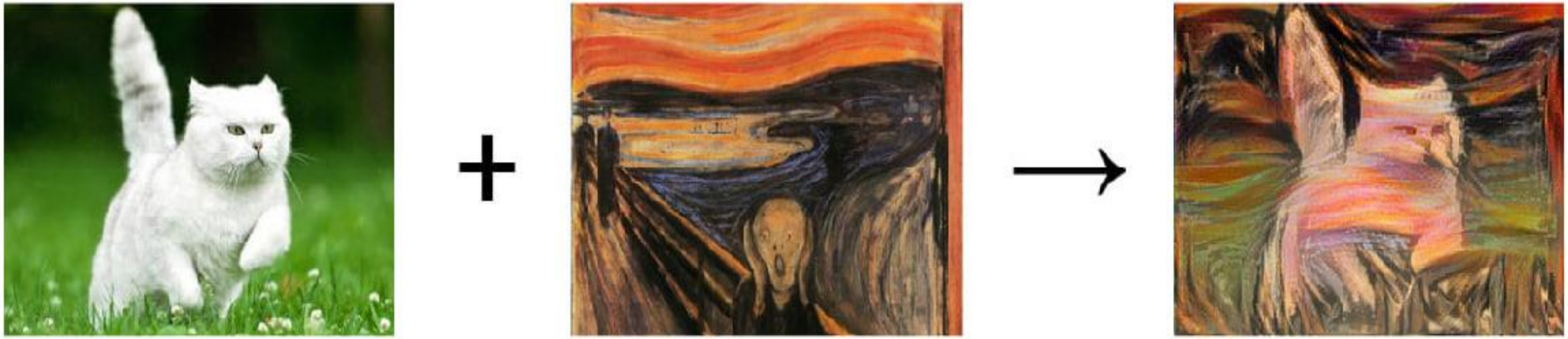
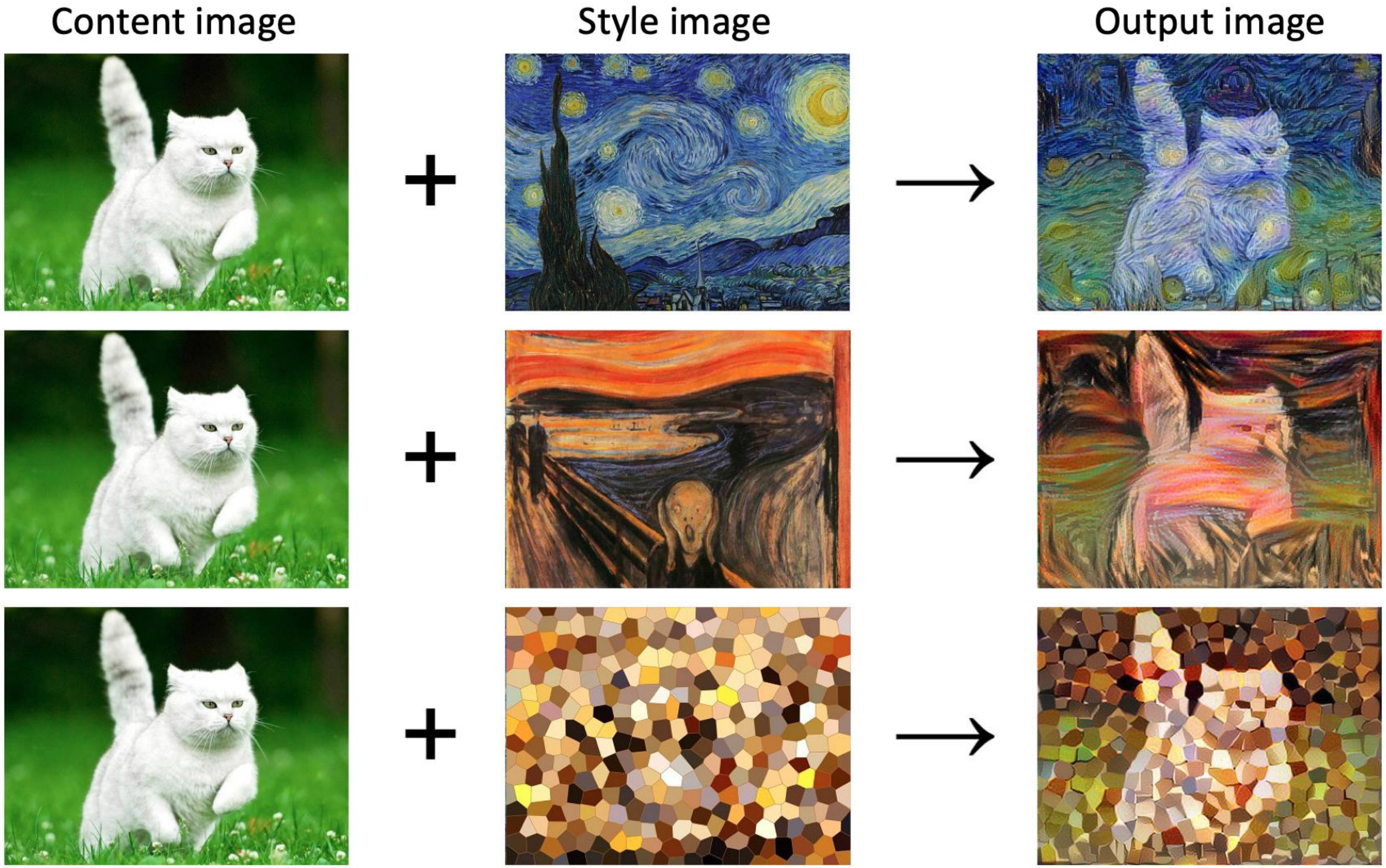


Style Transfer

By Mehrab Kalantary And Anita Soroush

For Neural Network Course

Style Transfer



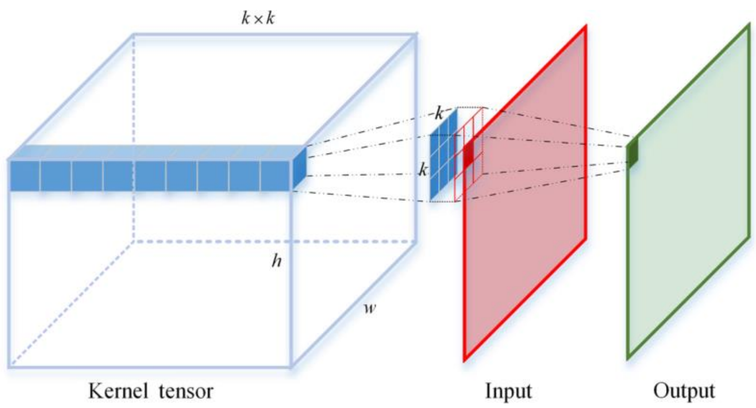
Style Transfer

- Style of content image is transferred onto style image.
- Style consists:
  - Statistical properties
  - Geometric structure

Style Transfer Methods

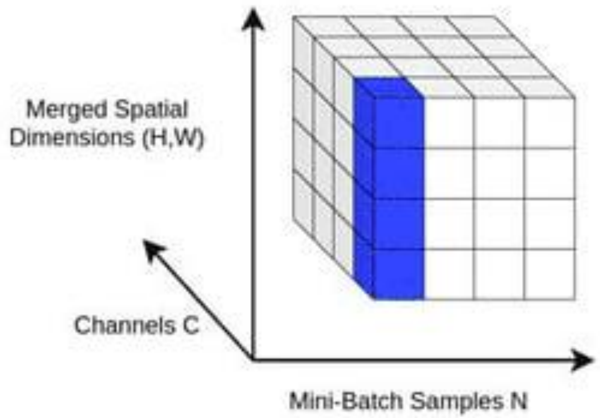
Adaptive Convolutions

- A better method
- Both statistical properties and geometric structure
- Based on kernel prediction

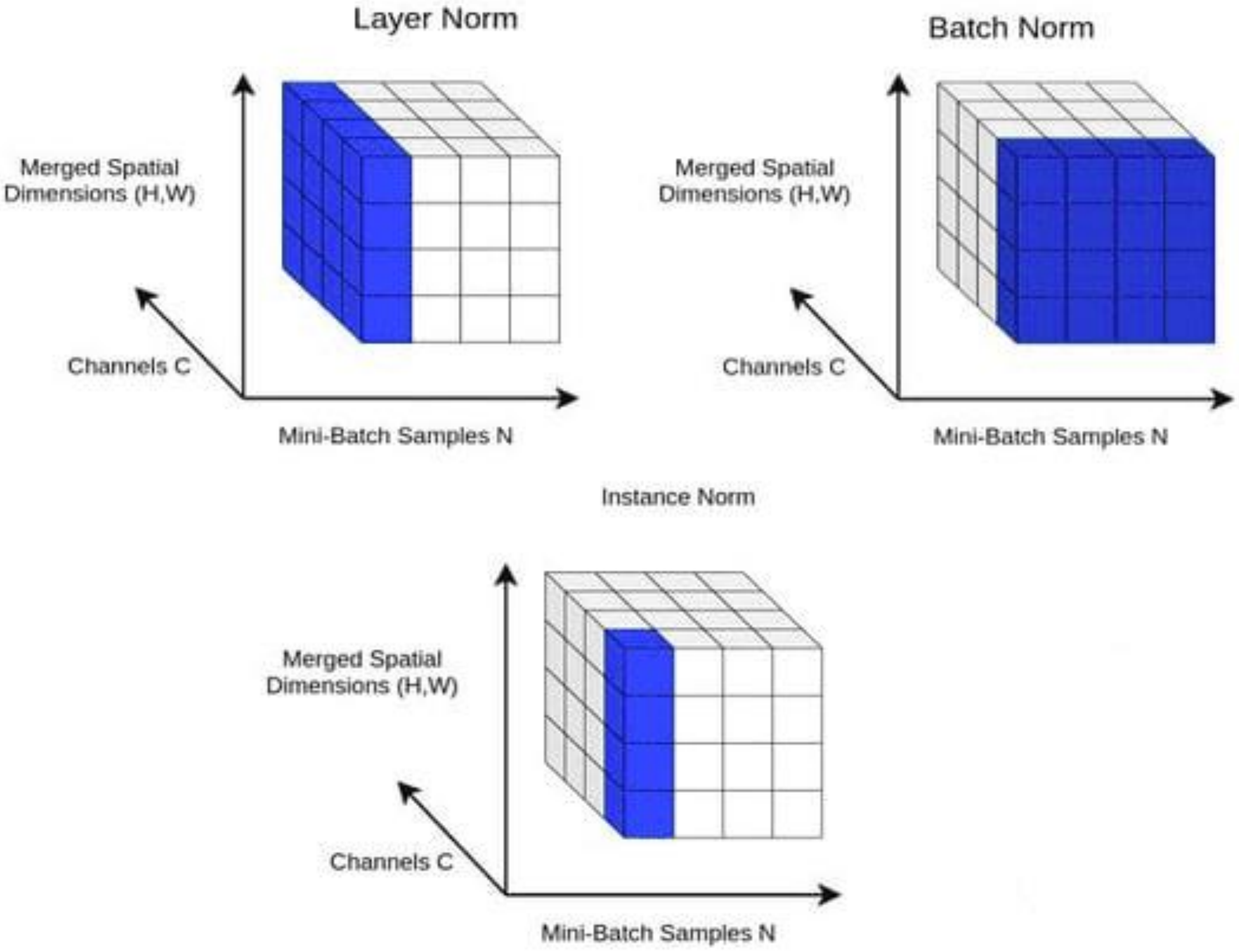


Adaptive Instance Normalization

- Frist method
- Just statistical properties
- Based on instance normalization



Instance Normalization



Instance Normalization

- Batch normalization normalizes activations in a network across the mini-batch of definite size.
- Layer normalization normalizes input across the features instead of normalizing input features across the batch dimension in batch normalization.
- Instance normalization normalizes across each channel in each training example.
- Instance normalization formula for instance i:

$$\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)}$$

Adaptive Instance Normalization

- By extending IN formula we can achieve AdaIN formula.

$$\mathrm{AdaIN}(x; a, b) = a\left(\frac{x - \mu_x}{\sigma_x}\right) + b$$

- Where a and b represent the style as scale and bias terms.
- For style transfer, a and b are the mean and standard deviation of the style image features.

Adaptive Instance Normalization

- For style transfer, a and b are the mean and standard deviation of the style image features.
- So we can say:

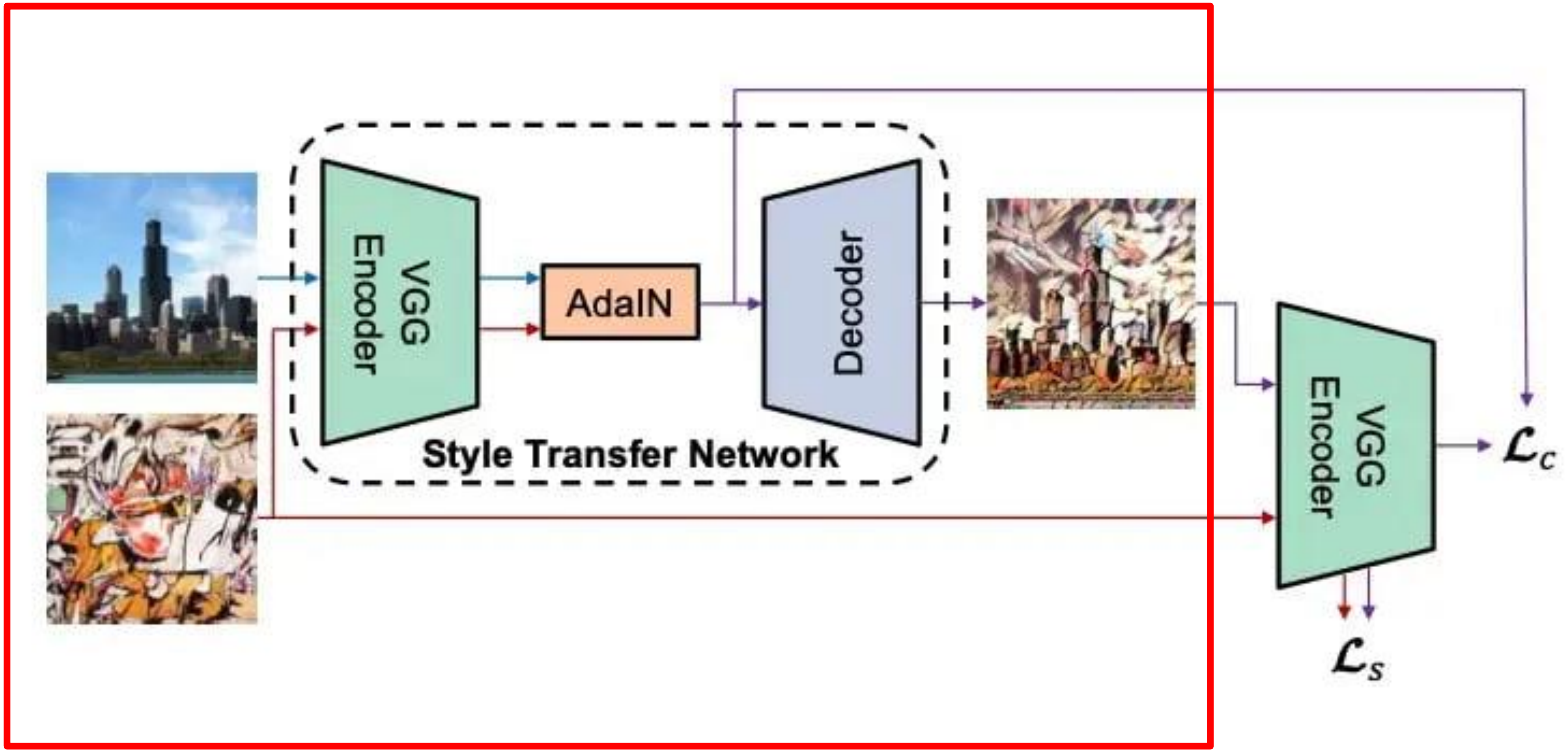
$$\mathrm{AdaIN}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y)$$

- Where x is content image and y is style image.

Implementation!

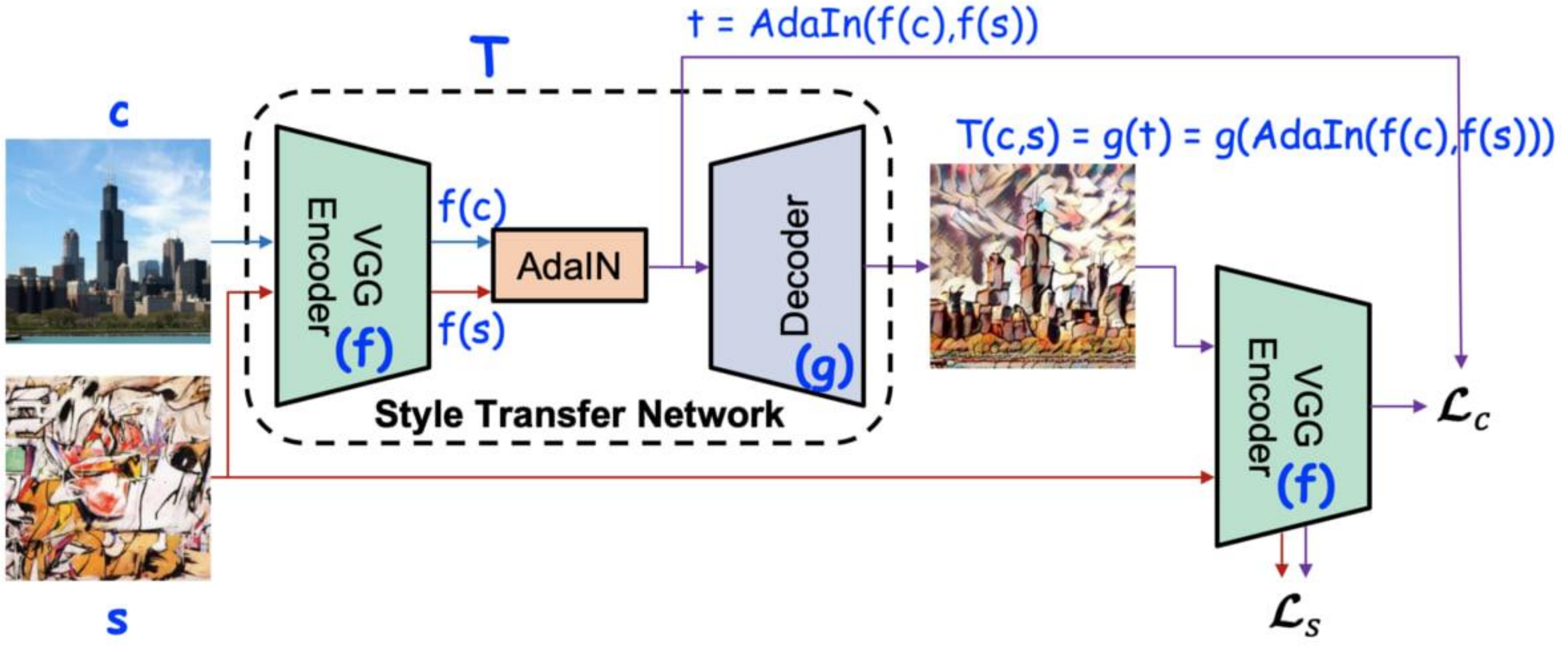
Let’s see python code

AdaIN Style Transfer Network

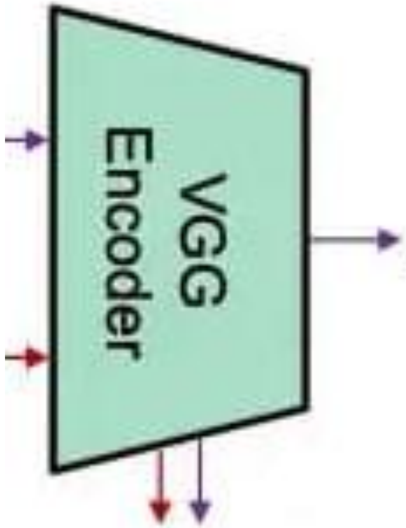


AdaIN Style Transfer Network

- The AdaIn StyleNet follows an Encoder-Decoder architecture.
- The encoder is the first few pre-trained layers of the VGG-19 network.
  (<u>The encoder is fixed and not trained</u>)
- The decoder is initialized with random weights and its weights are learned.



AdaIN Loss Function



$$\mathcal{L}_c = \|f(g(t)) - t\|_2$$

$$\ell_S = \sum_{i=1}^{L} ||\mu(\phi_i(g(t))) - \mu(\phi_i(t))||_2 + ||\sigma(\phi_i(g(t))) - \sigma(\phi_i(t))||_2$$

$$\mathcal{L} = \ell_C + \lambda \ell_S$$

AdaIN Review

$$\mathrm{AdaIN}(x; a, b) = a\left(\frac{x - \mu_x}{\sigma_x}\right) + b$$

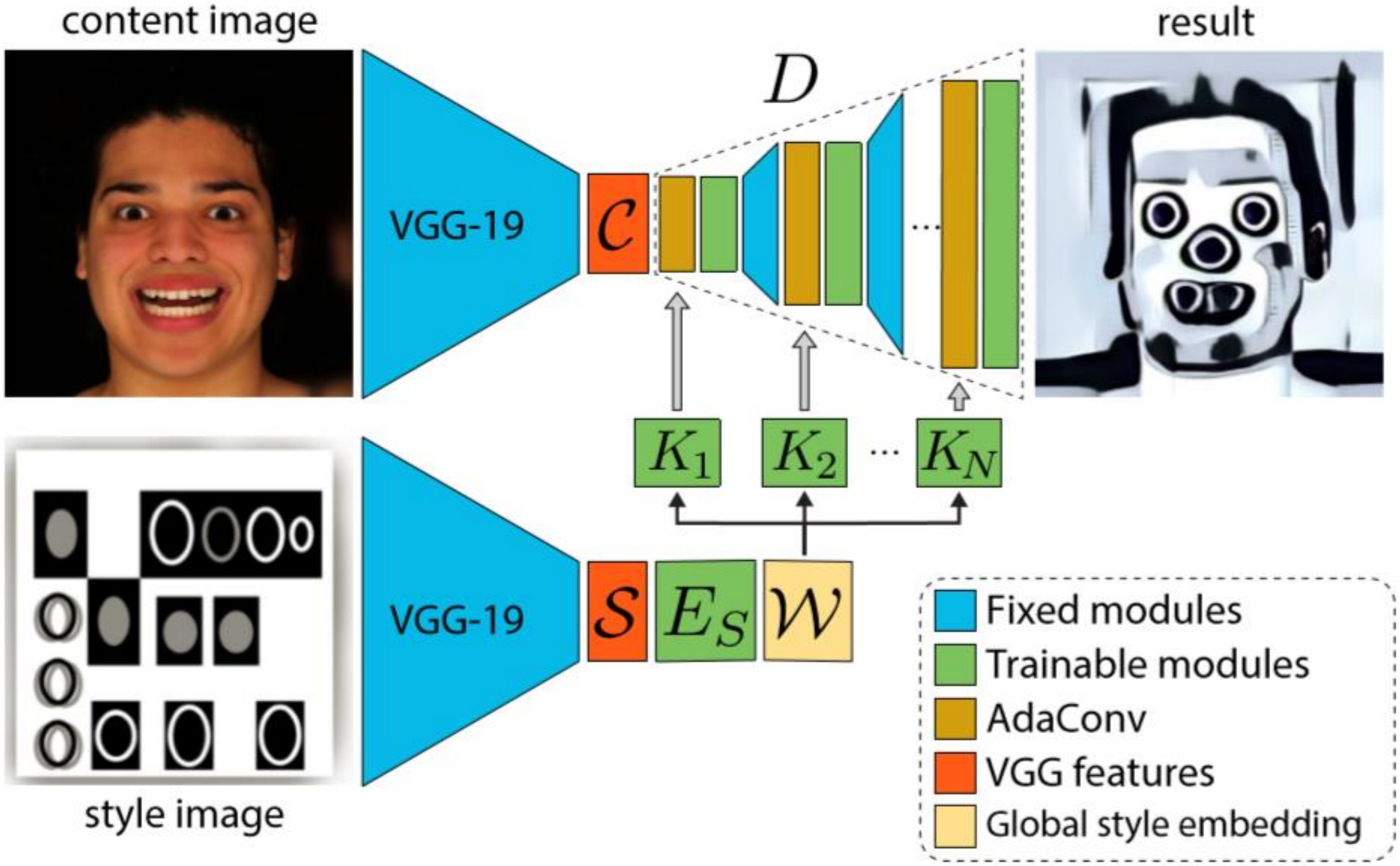
AdaConv: Step 1

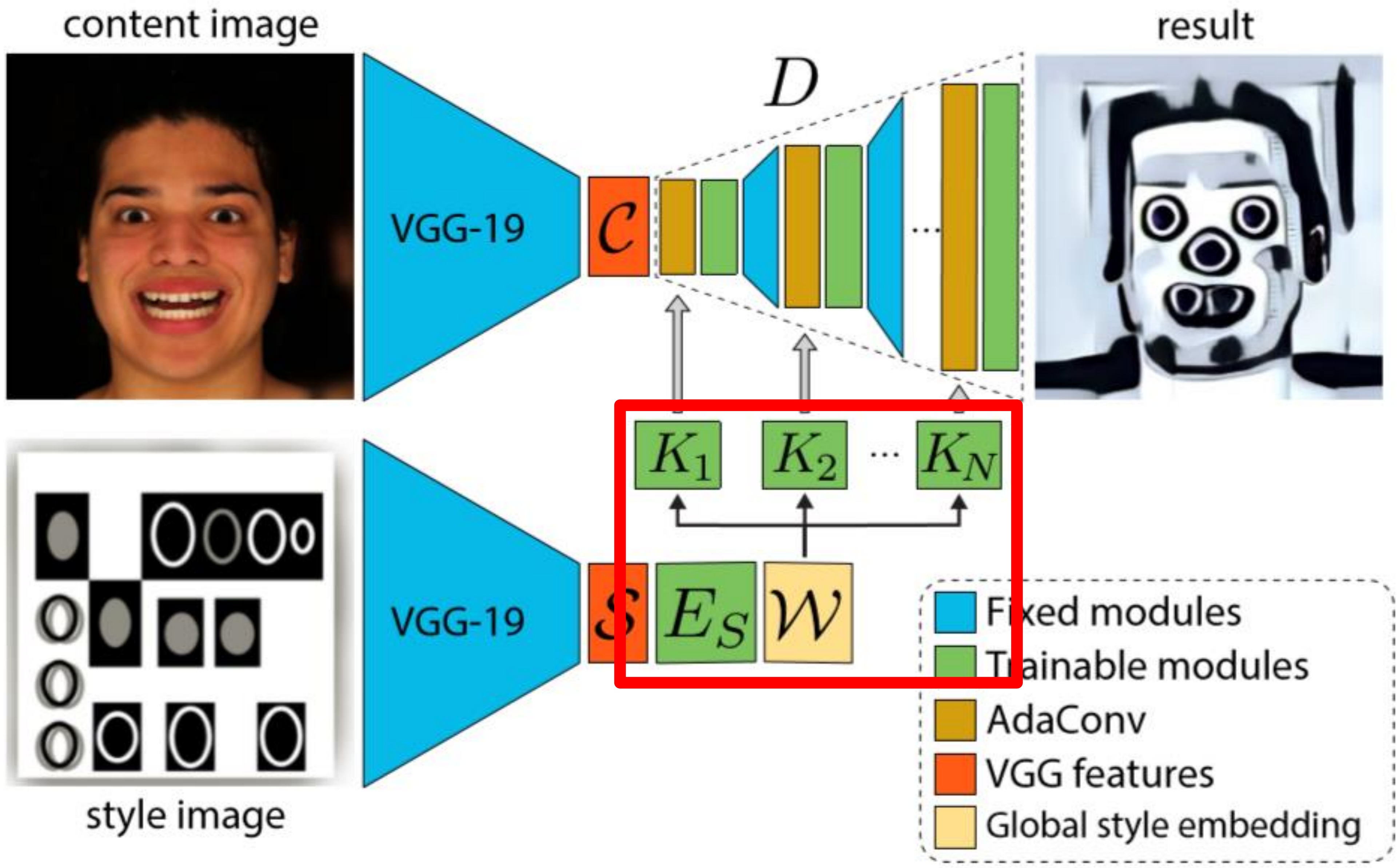
$$\begin{aligned}\mathrm{AdaConv}_{\mathrm{dw}}(x; \mathbf{f}, b) &= \sum_{x_i \in \mathcal{N}(x)} f_i \left(\frac{x_i - \mu_x}{\sigma_x} \right) + b \\ &= \sum_{x_i \in \mathcal{N}(x)} \mathrm{AdaIN}(x; f_i, b)\,.\end{aligned}$$

AdaConv: Step 2

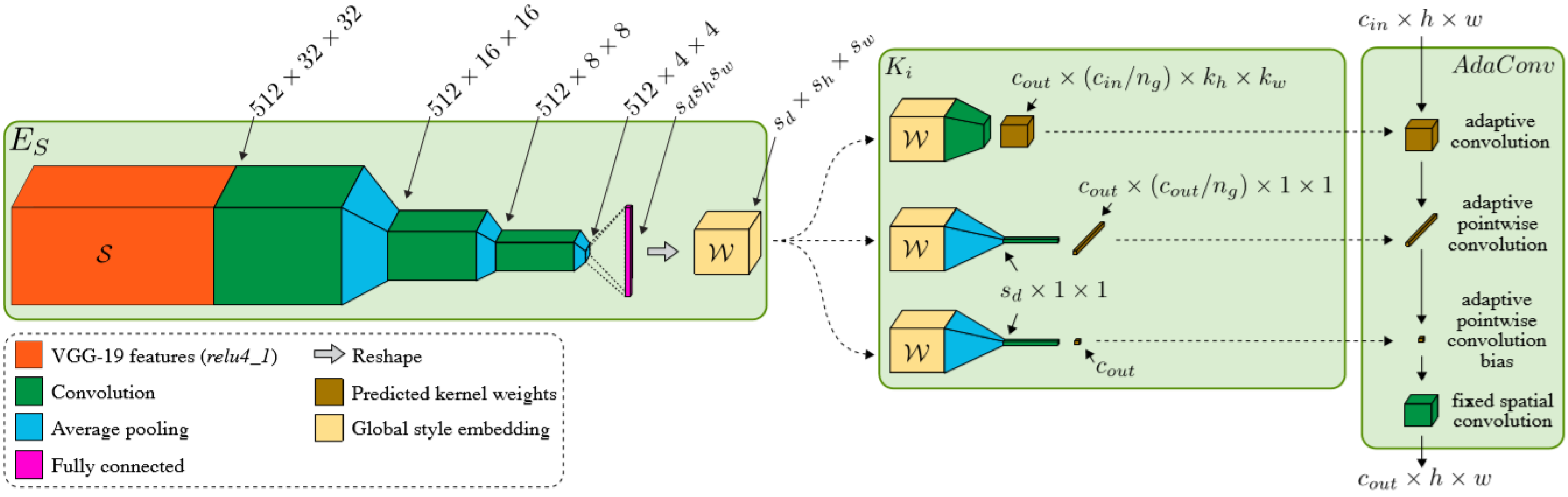
$$\mathrm{AdaConv}(x; \mathbf{p}, \mathbf{f}, \mathbf{b}) = \sum_{c} p_c \, \mathrm{AdaConv}_{\mathrm{dw}}(x_c; \mathbf{f}_c, b_c)$$

AdaConv Structure




content image
result
VGG-19
C
D
VGG-19
S
K1
K2
KN
ES
W
style image
Fixed modules
Trainable modules
AdaConv
VGG features
Global style embedding

AdaConv Structure



Implementation!

Let's see python code