Style Transfer with PyTorch

UNDERSTANDING NEURAL STYLE TRANSFER

Janani Ravi
CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Understand what style transfer is

Convolution layers in style transfer

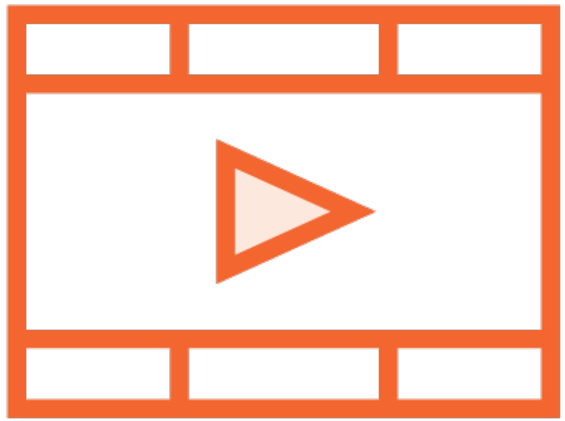
Content, style, and target images

Content loss and style loss

Feature maps and Gram matrices

Prerequisites and Course Outline

Prerequisites

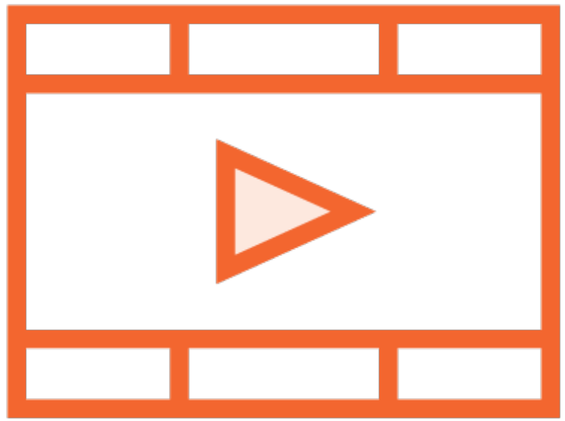


Comfortable programming in Python

Basic understanding of neural networks

Worked with PyTorch to build and train neural networks

Prerequisite Courses



Foundations of PyTorch

Building Your First PyTorch Solution

Image Classification with PyTorch

Course Outline

Style transfer in neural networks

Implementing style transfer in PyTorch

Generative Adversarial Networks (GANs) in PyTorch

Neural Style Transfer

Neural Style Transfer

Process of taking an image and reproducing it with a new artistic style using a pre-trained neural network.

Neural Style Transfer

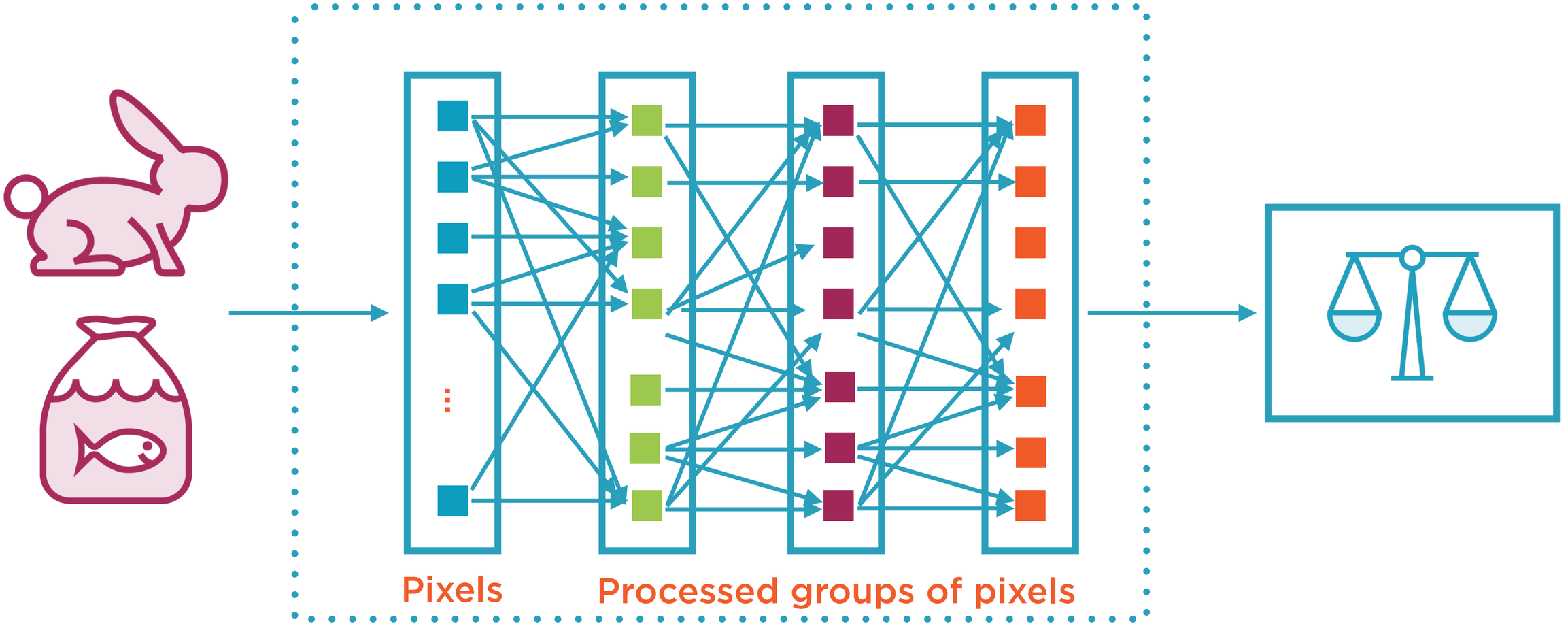
Process of taking an image and reproducing it with a new artistic style using a pre-trained neural network.

Neural Style Transfer

Process of taking an image and reproducing it with a new artistic style using a pre-trained neural network.

Images Involved

Neural Networks for Image Classification

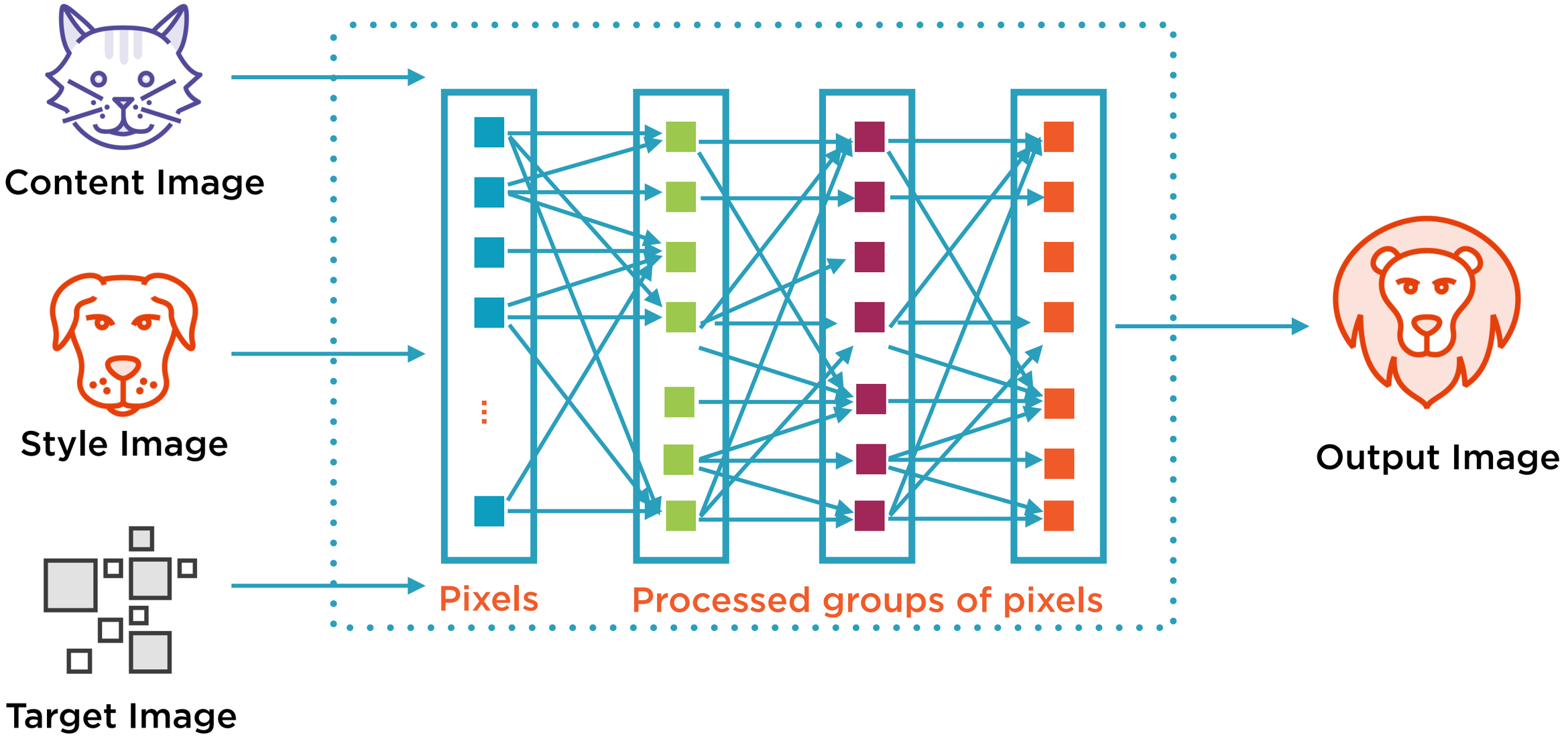


Corpus of Images

Each layer consists of individual interconnected neurons

ML-based Classifier

Neural Networks for Style Transfer



Images Involved

Images Involved

Content Image

Style Image

Target Image

Images Involved

Content Image

Style Image

Target Image

Images Involved

Content Image

Style Image

Target Image

Style transfer involves training the weights of the **target** image

The weights of the pre-trained model are **frozen**

Content Loss and Style Loss

Images Involved

Content Image

Style Image

Target Image

Loss Measures Involved

Two loss measures needed

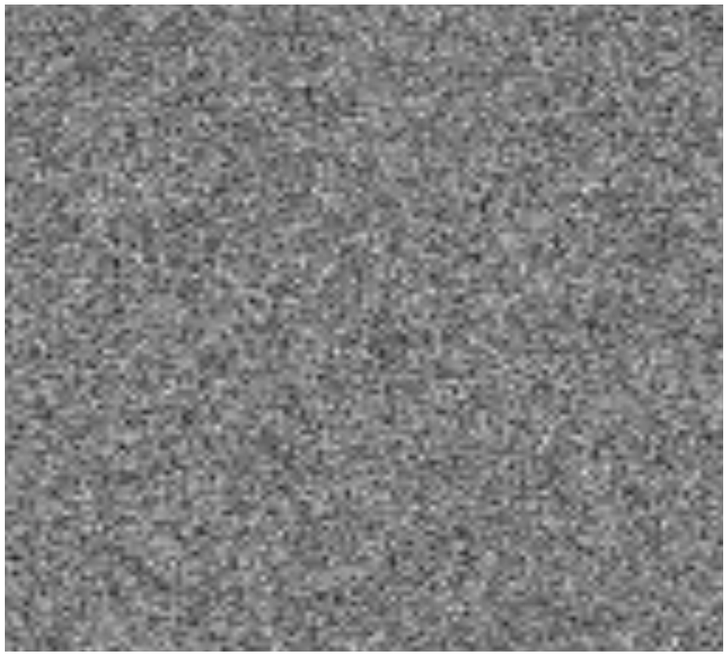
- Content loss
- Style loss

Minimize content loss with content image

Minimize style loss with style image

Minimize Content Loss

Content Image

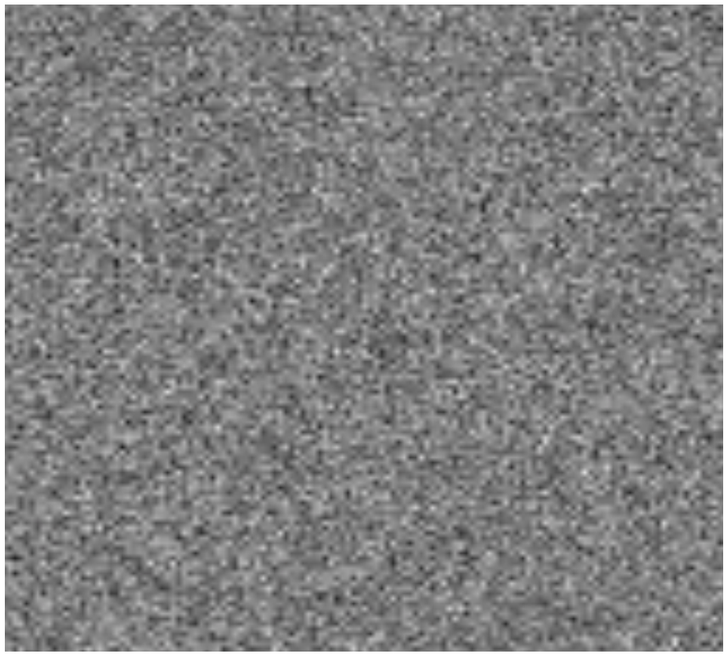


Initial Target Image

Extract Content

Minimize Style Loss

Style Image



Initial Target Image

Extract Style

Style Transfer

Content + Style = Style Transfer

Pre-trained Model

Actual style transfer is performed by a pre-trained neural network

Convolutional Neural Network (CNN) such as vgg19

CNN remains unchanged during process

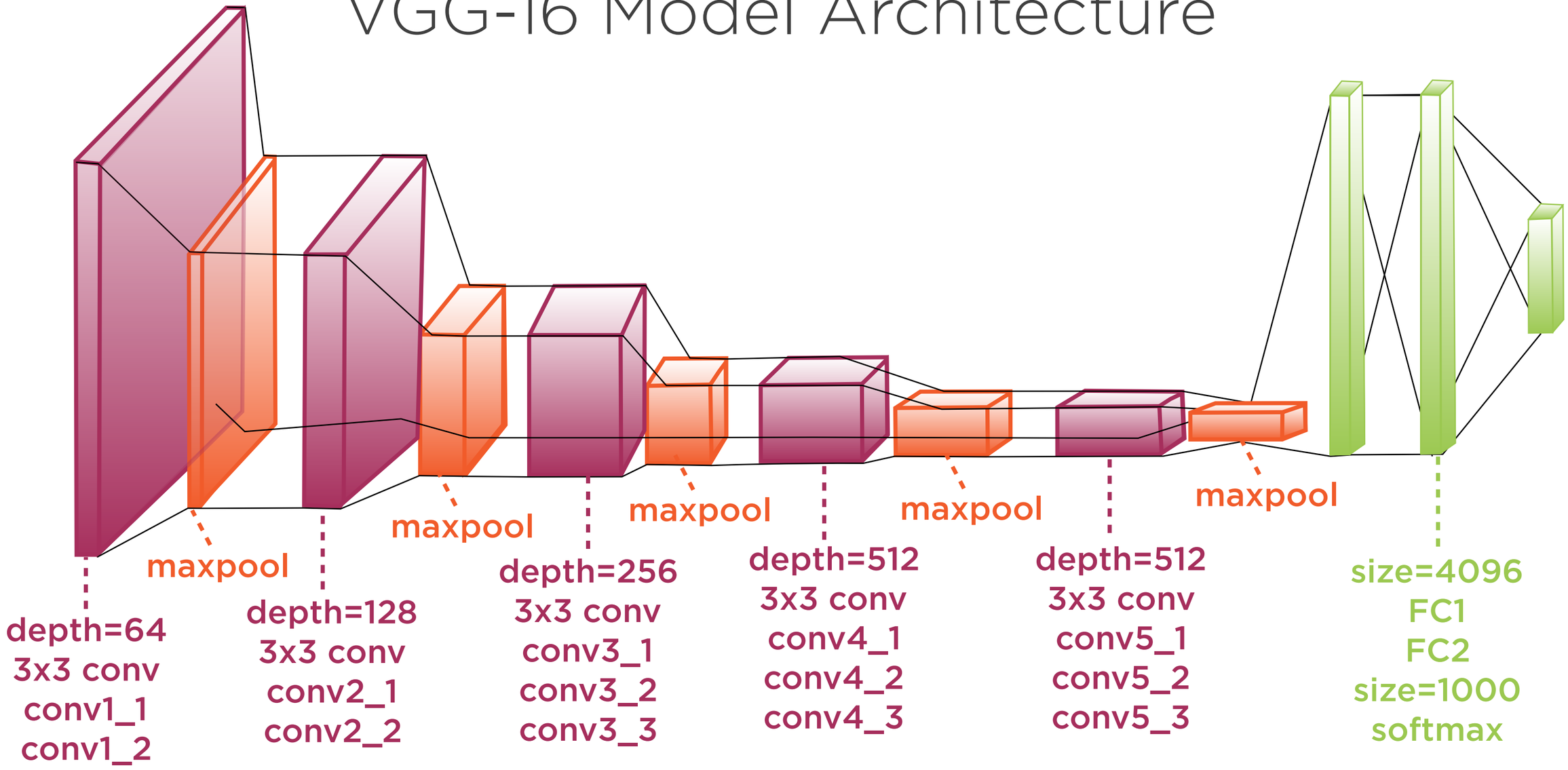
Only target image is modified

A Neural Algorithm of Artistic Style

https://arxiv.org/abs/1508.06576

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge

VGG-16 Model Architecture



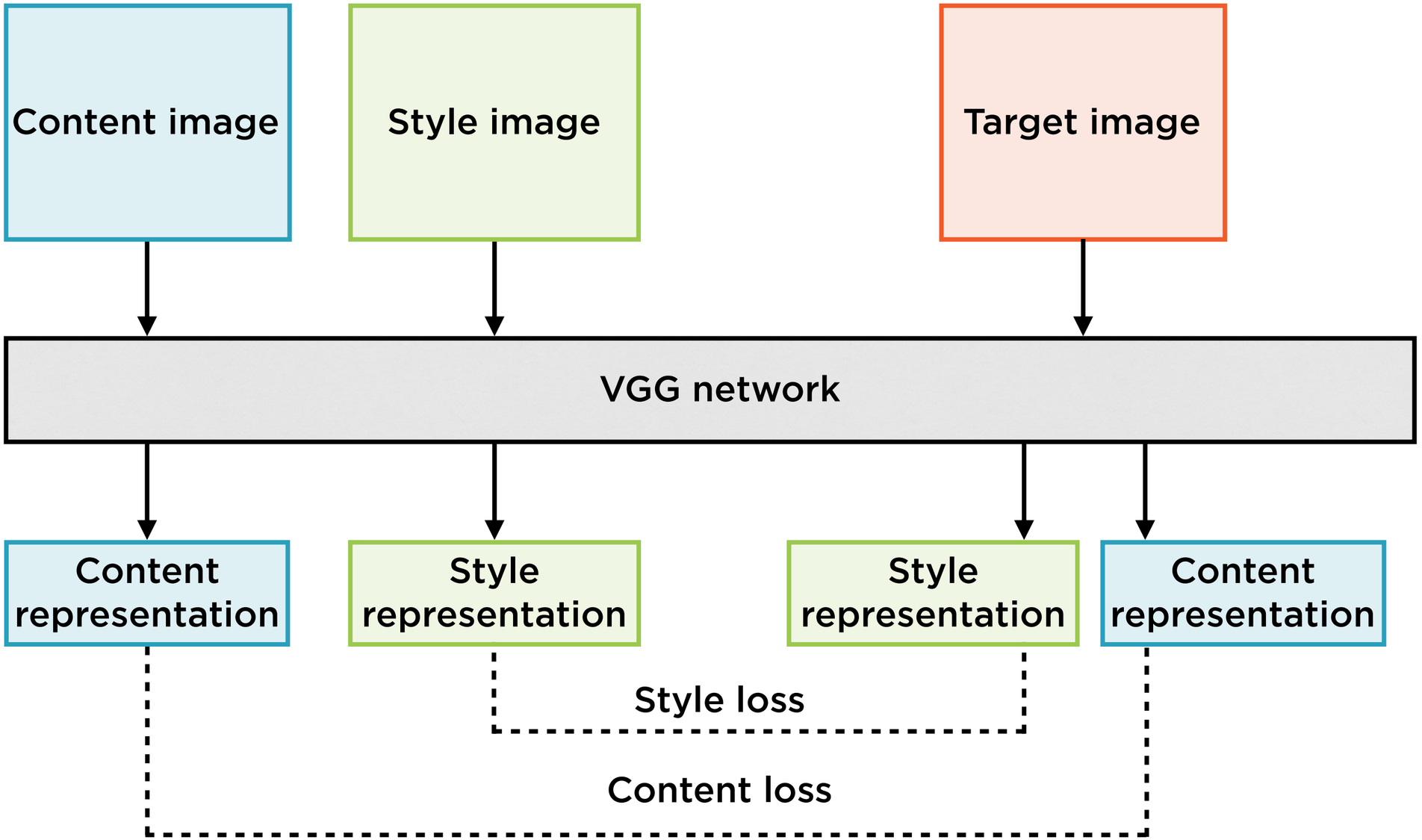
Back Propagation

Neural Style Transfer uses back propagation, but not for training

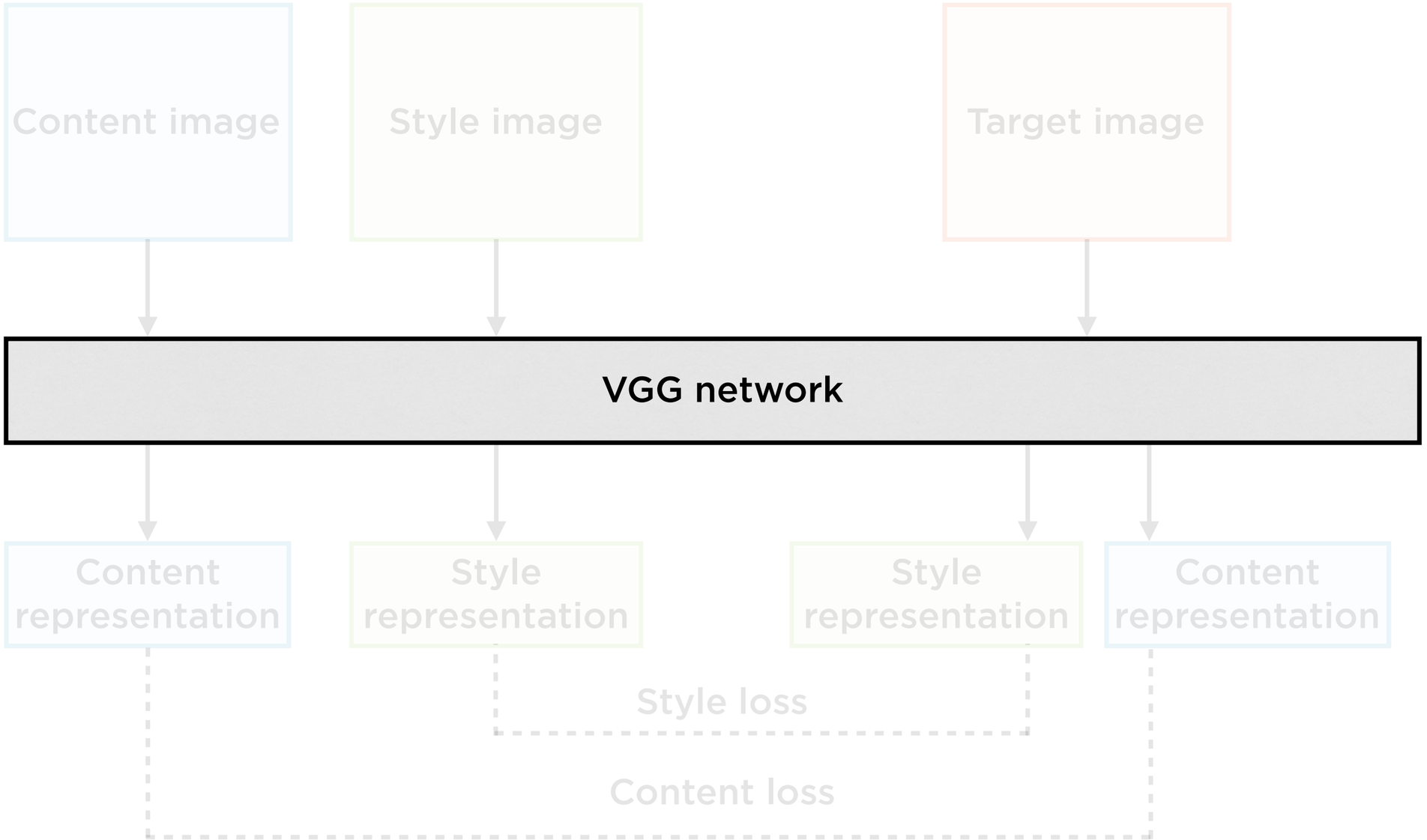
CNN is pre-trained and remains unchanged

Back propagation is used to modify the target image, not the model

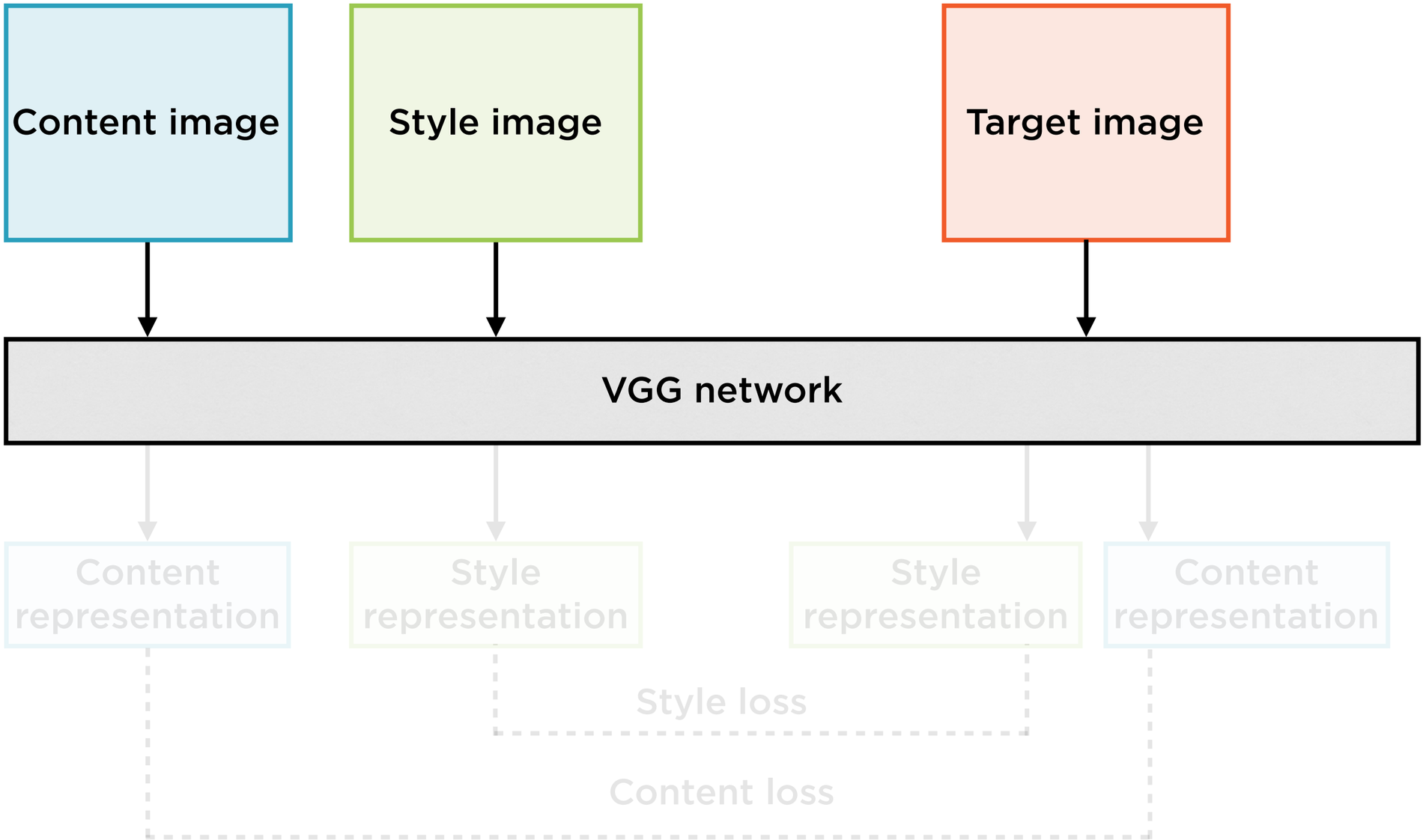
Style Transfer



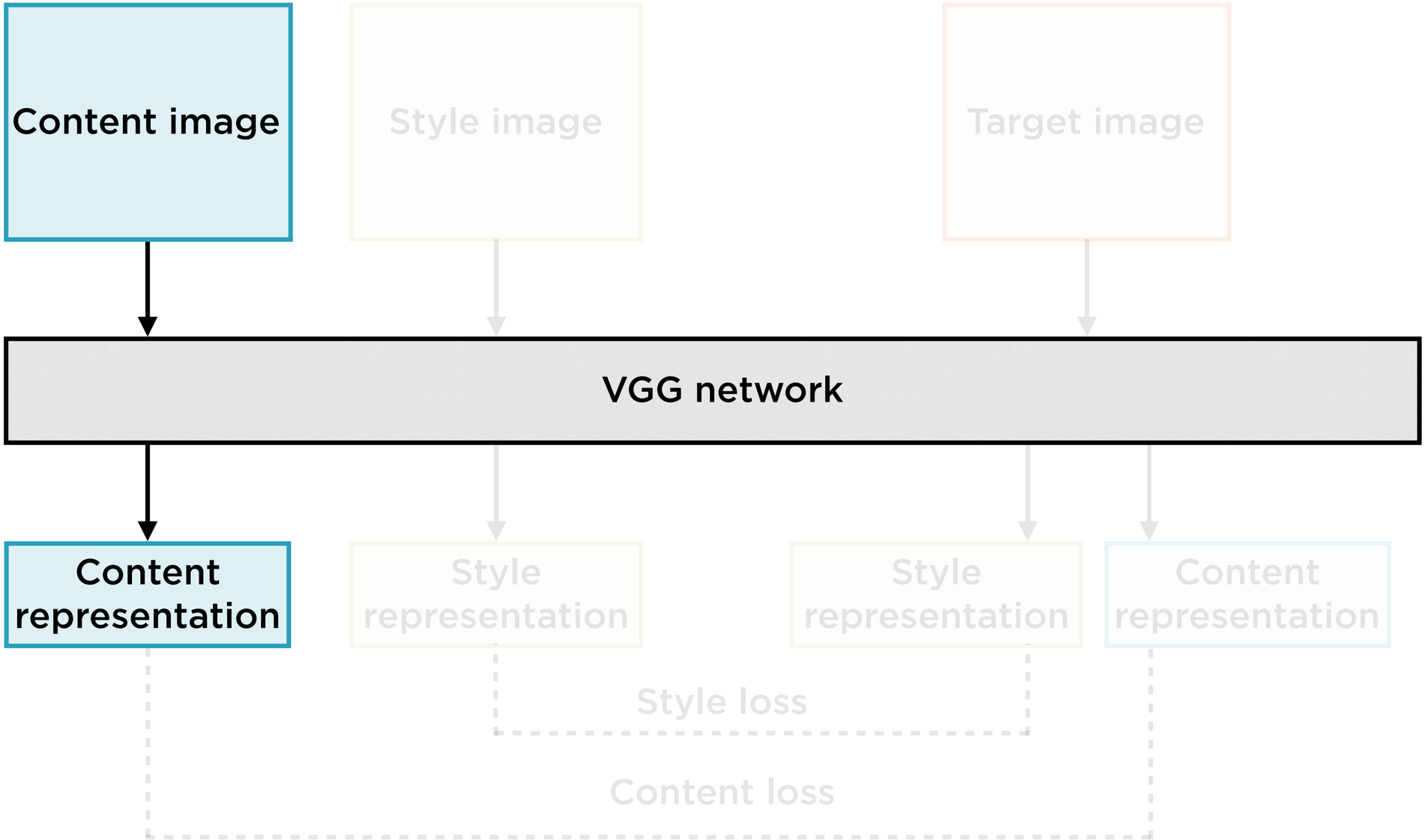
Pre-trained Model



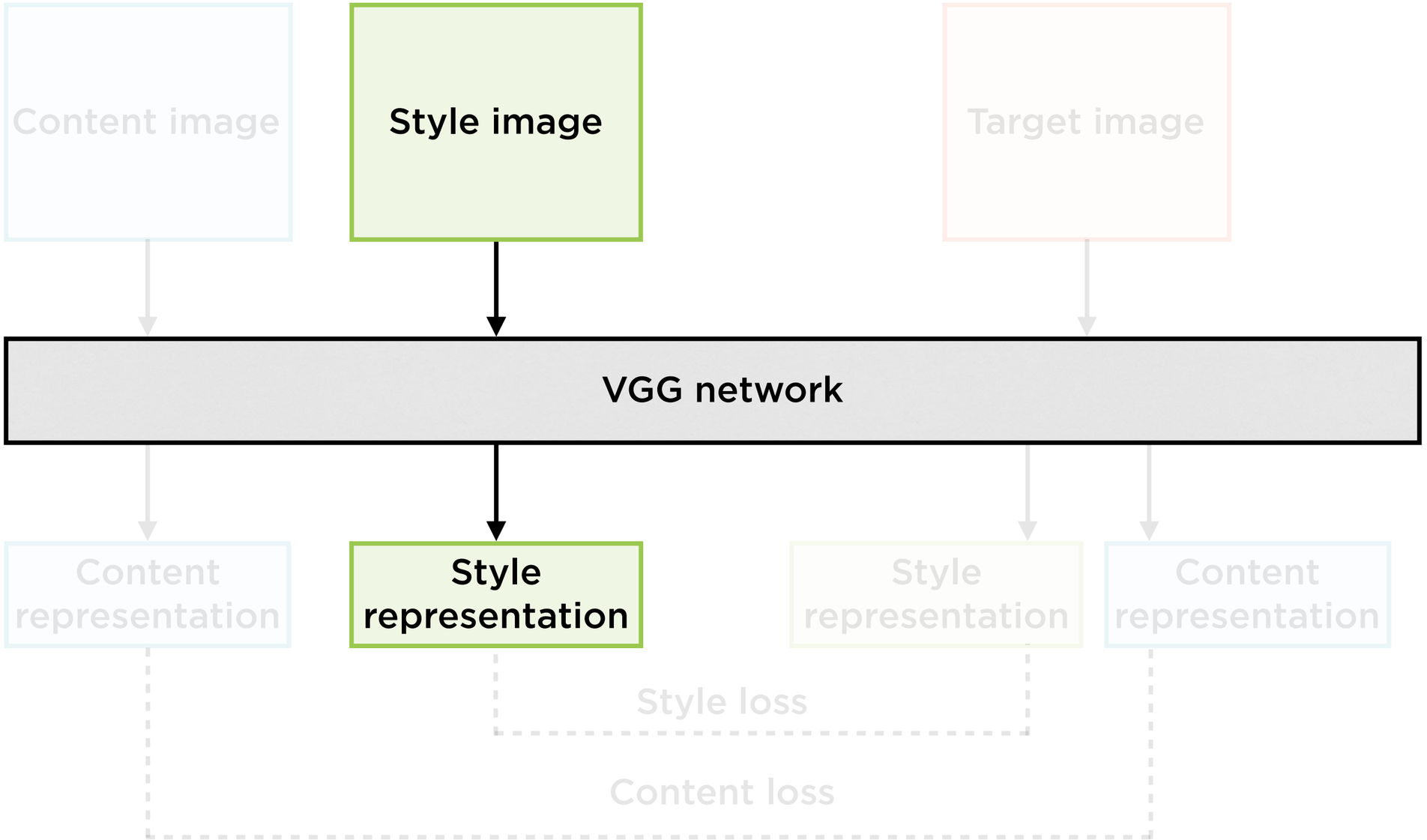
All Images Fed into Model



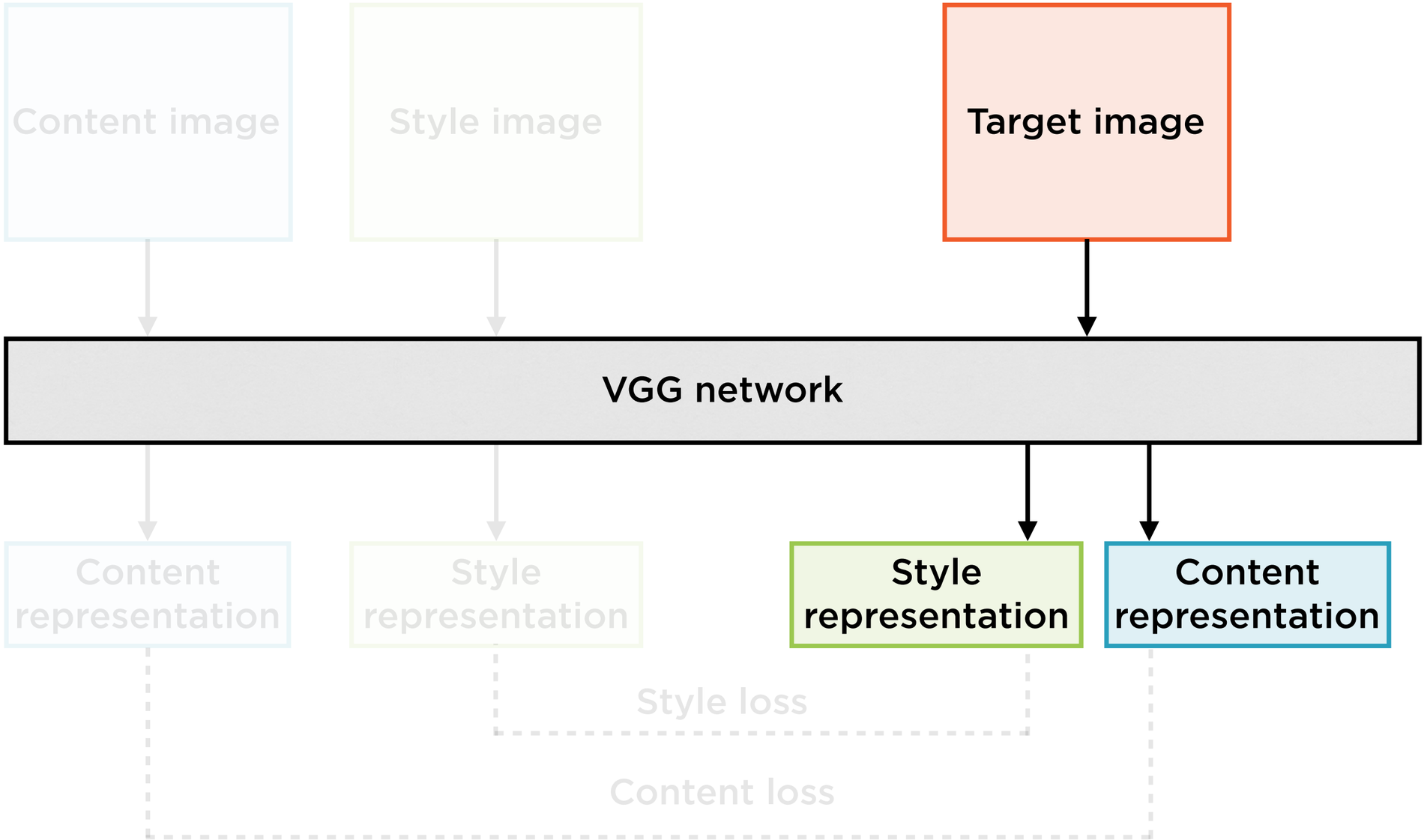
Content Representation



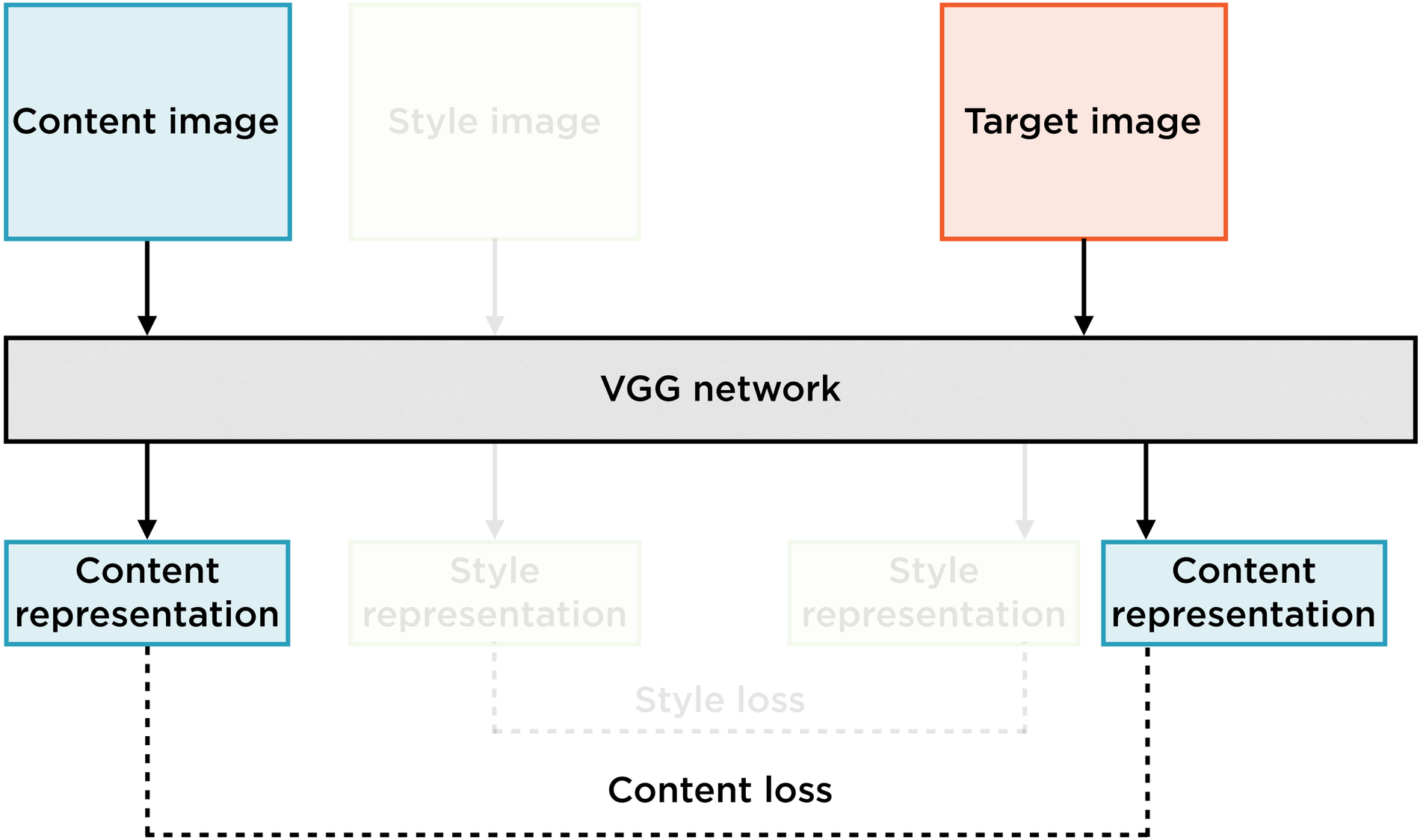
Style Representation



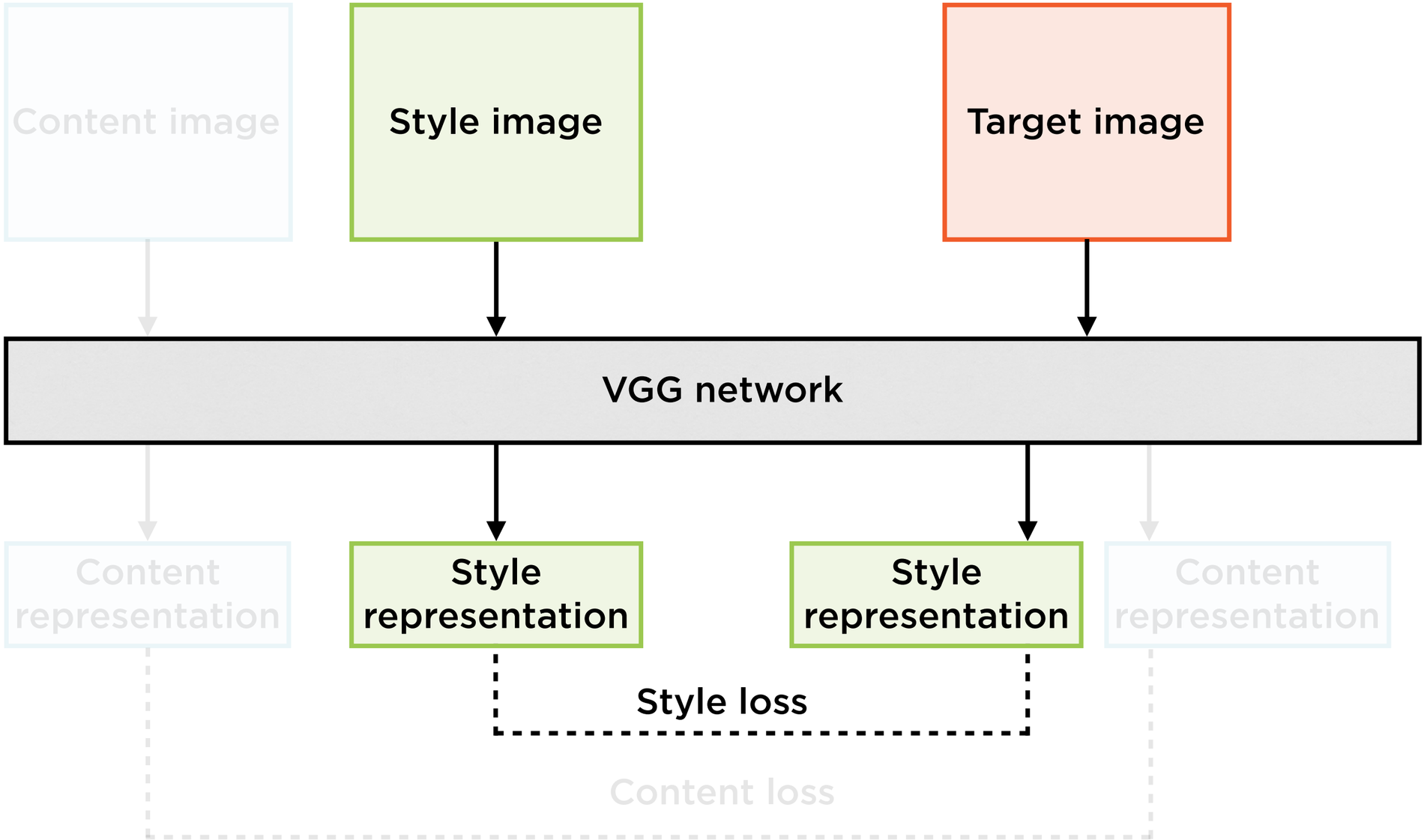
Train the Target Image



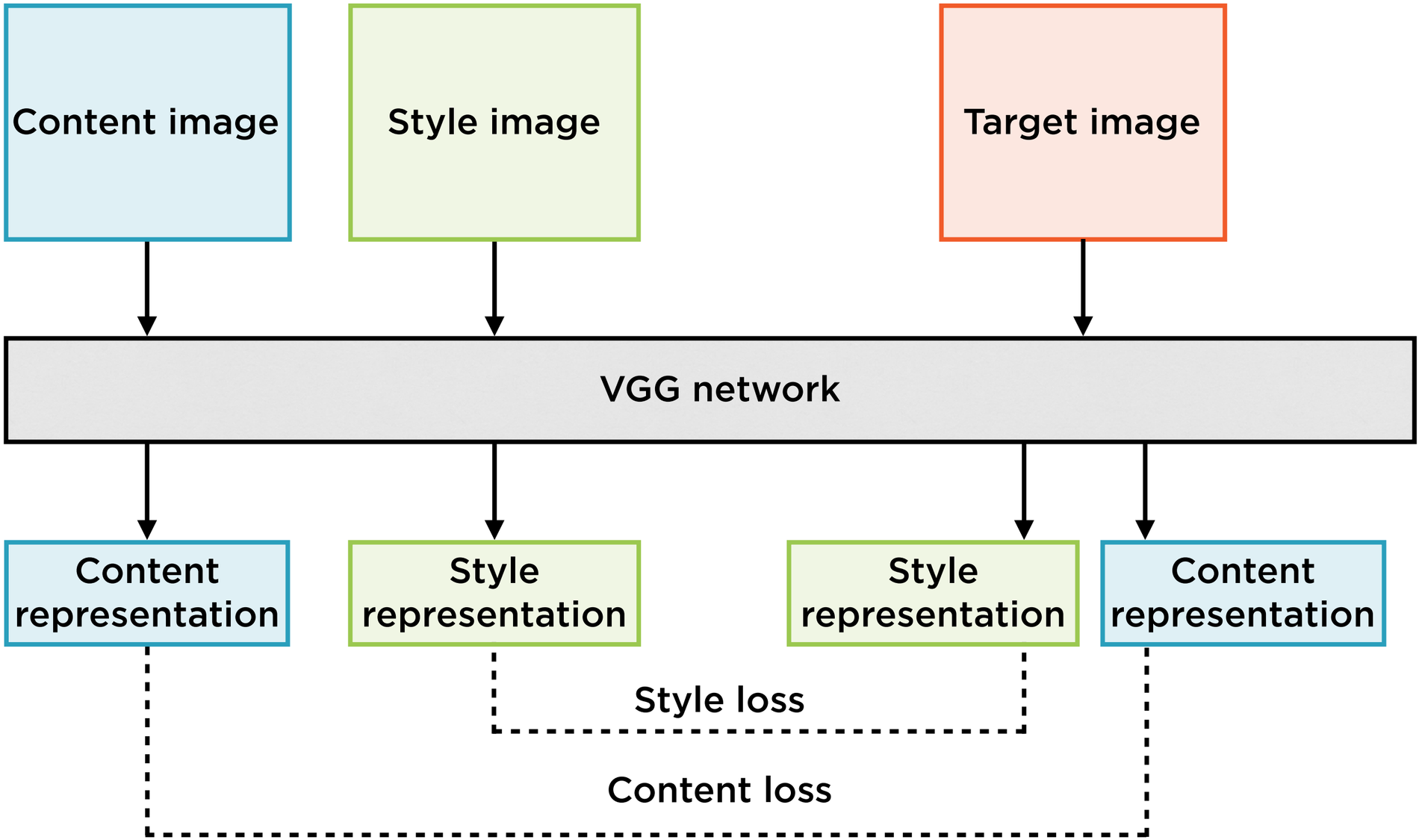
Minimize Content Loss



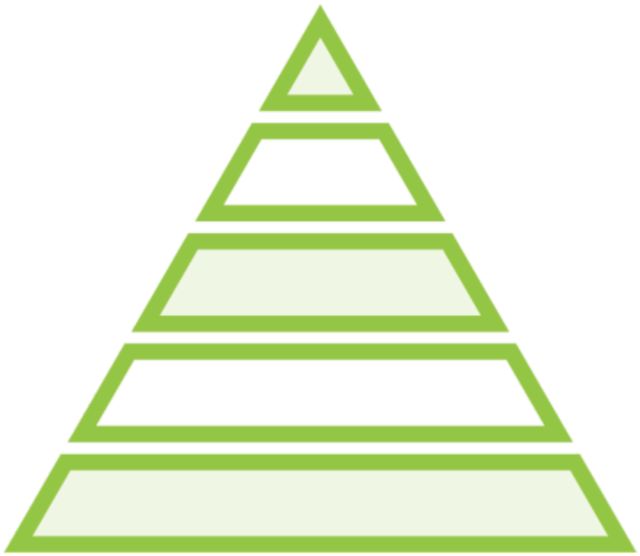
Minimize Style Loss



Style Transfer



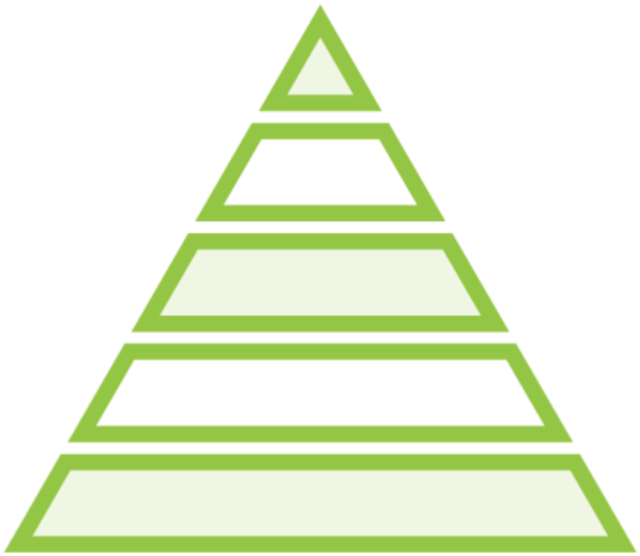
Layers of Interest



Designate specific layers of CNN as

- Content layers of interest
- Style layers of interest

Layers of Interest



Layers of interest are intermediate layers of the CNN

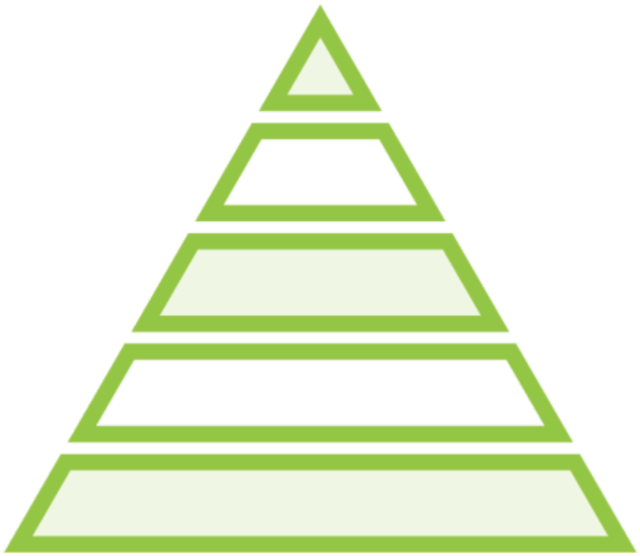
Outputs of these layers are simply feature maps

We need to extract content and style separately from these feature maps

Extracting Content Information

How do we extract **content** information from the content image?

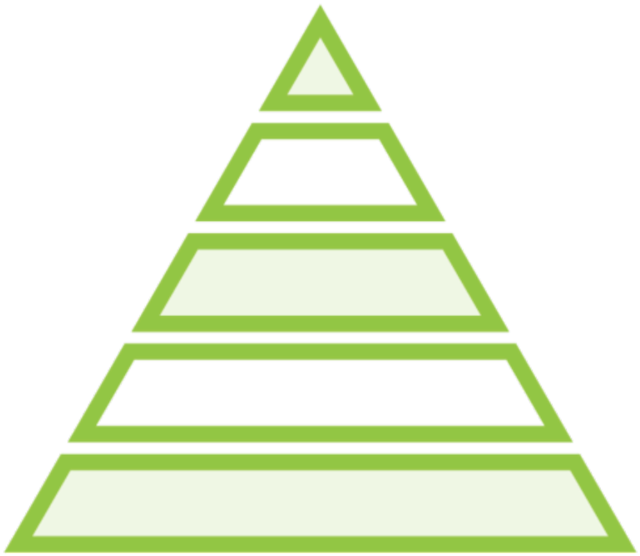
Layers of Interest



Designate specific layers of CNN as

- Content layers of interest
- Style layers of interest

Layers of Interest



Designate specific layers of CNN as

- Content layers of interest
- Style layers of interest

Two Kinds of Layers in CNNs

Convolution

Local receptive field

Pooling

Subsampling of inputs

Two Kinds of Layers in CNNs

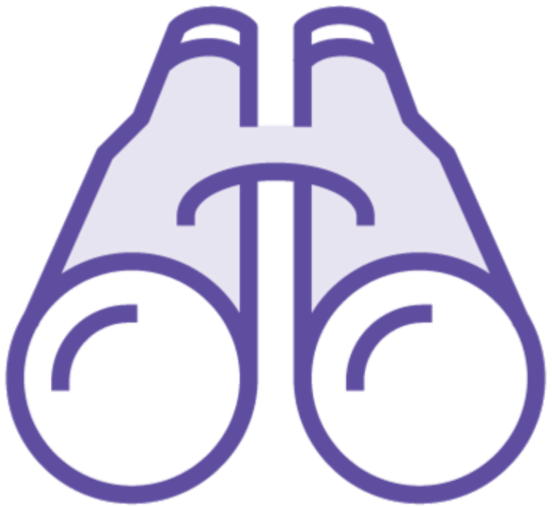
Convolution

Local receptive field

Pooling

Subsampling of inputs

Convolutional Layers



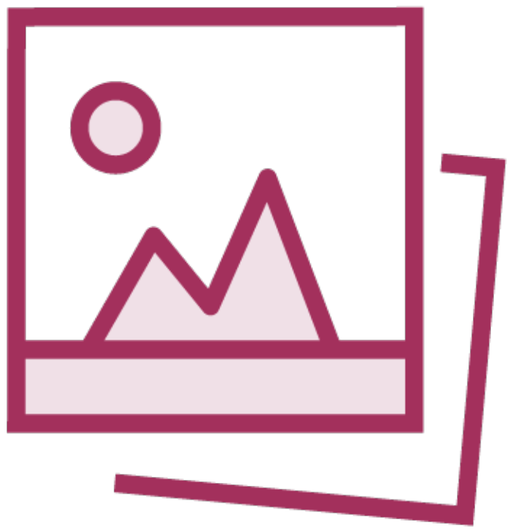
Convolution layers - zoom in on specific bits of input

Extract structure and features in the input image

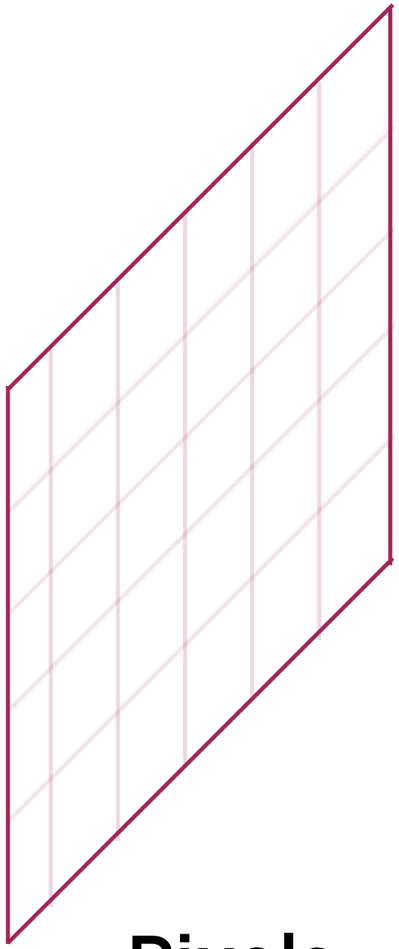
Successive layers aggregate inputs into higher level features

Pixels >> Lines >> Edges >> Object

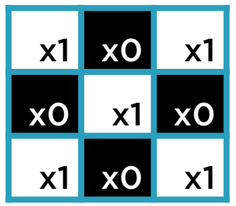
Feature Maps

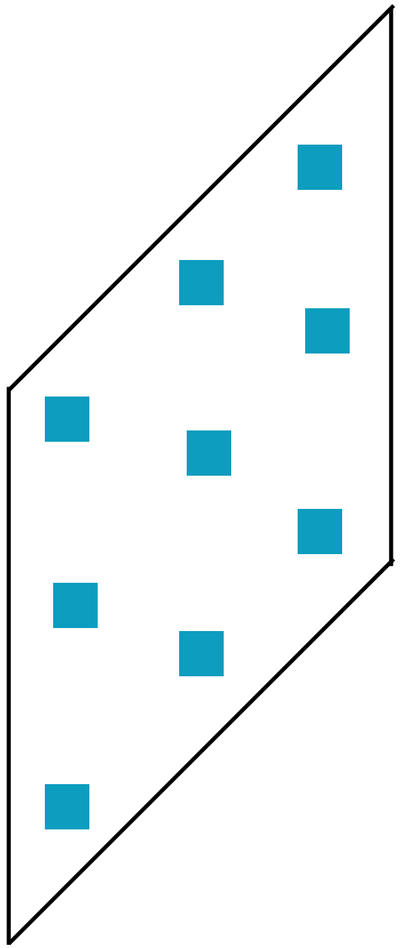


Image



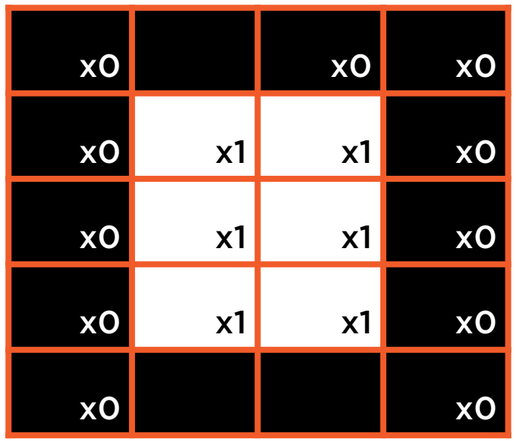
Pixels





Feature Map

Choice of Kernel Function



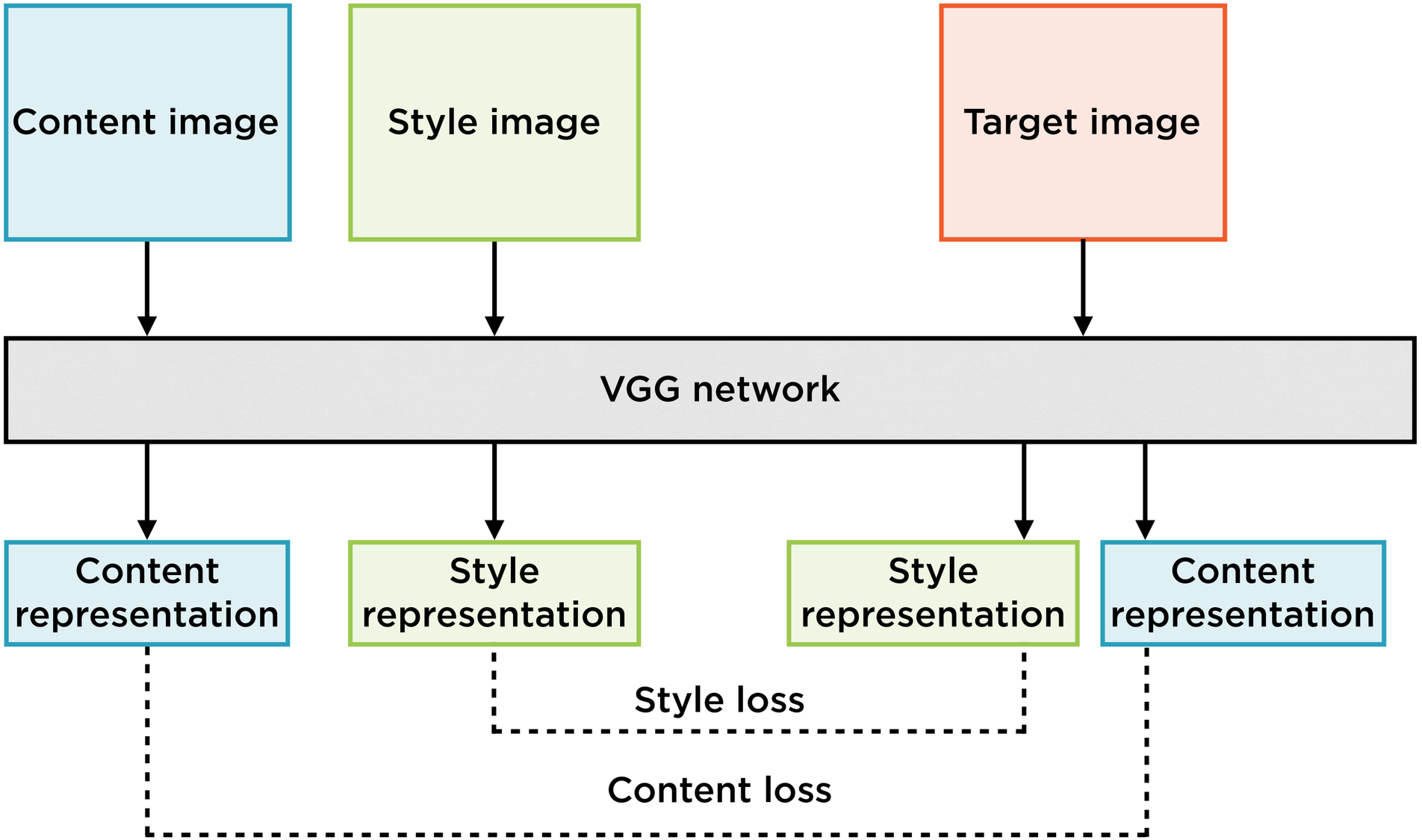
Averaging neighboring pixels ~
Blurring

Subtracting neighboring pixels ~
Edge detection

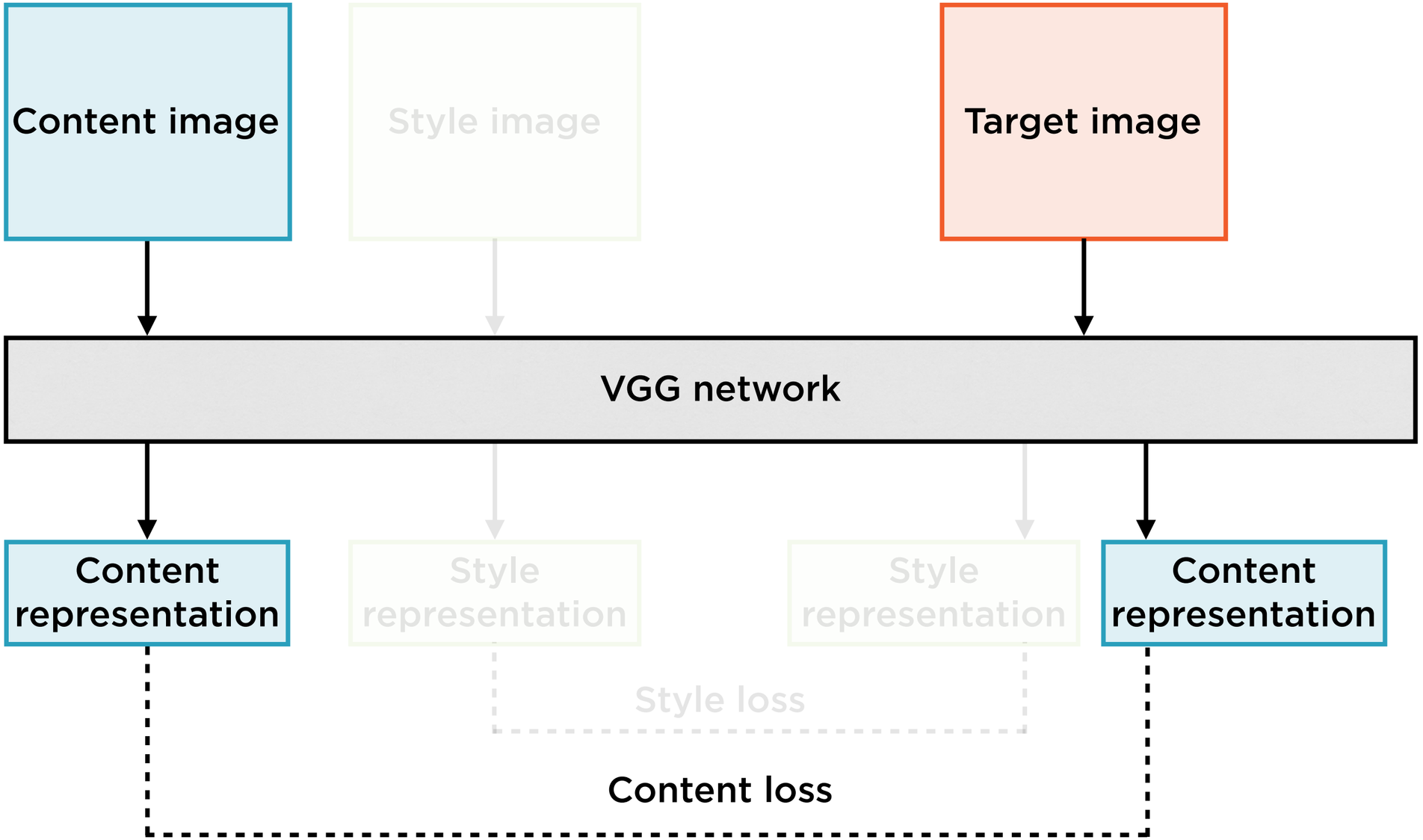
Positive middle, negative neighbors ~
Sharpen

Negative corners, zero elsewhere ~
Edge enhance

Style Transfer



Minimize Content Loss



Difference between the
feature maps of the target
image and the content image

Feature maps from the
content layers of interest

Minimize Content Loss

Content loss: Difference between feature maps of target image and content image

Use feature map representations from content layers of interest

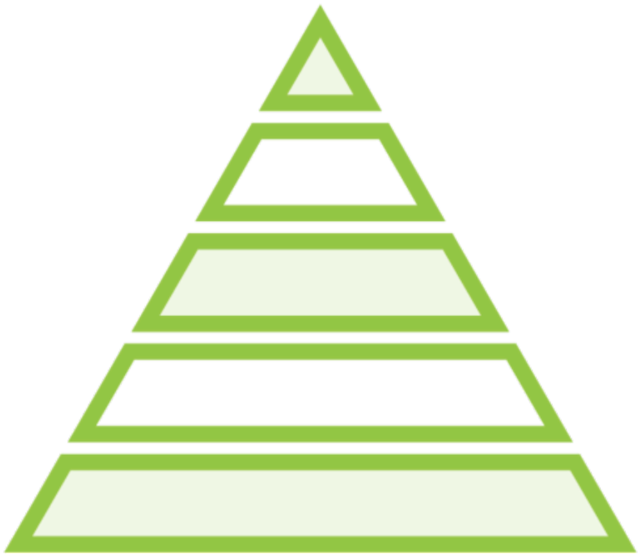
Earlier layers of the CNN contain more content information

MSE loss between content feature maps and target feature maps

Extracting Style Information

How do we extract **style** information from the style image?

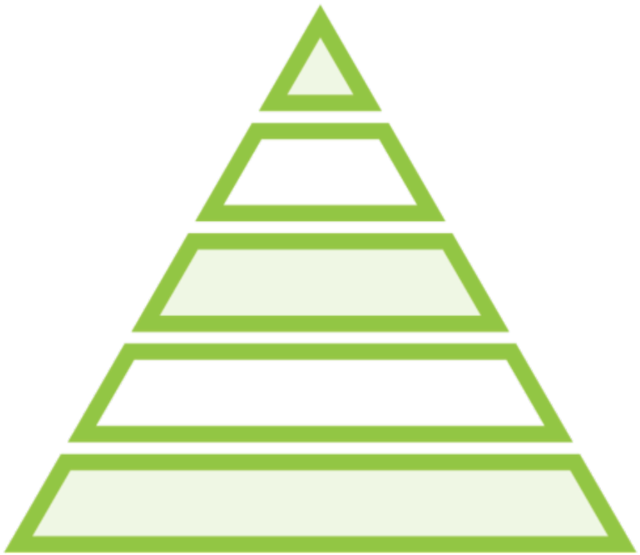
Layers of Interest



Designate specific layers of CNN as

- Content layers of interest
- Style layers of interest

Layers of Interest



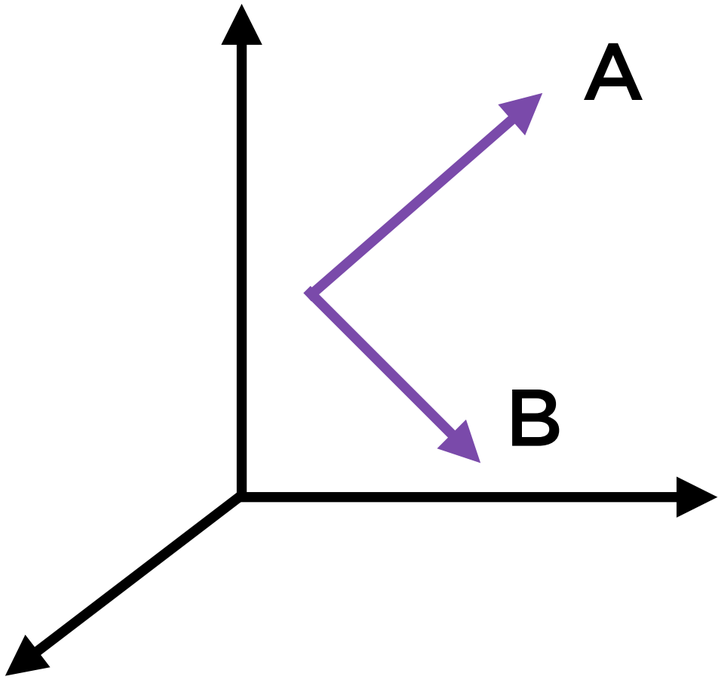
Designate specific layers of CNN as

- Content layers of interest
- Style layers of interest

Cosine Similarity

Cosine similarity is a measure of similarity between two non-zero vectors, widely used in ML algorithms - especially in document modeling applications.

Orthogonal Vectors



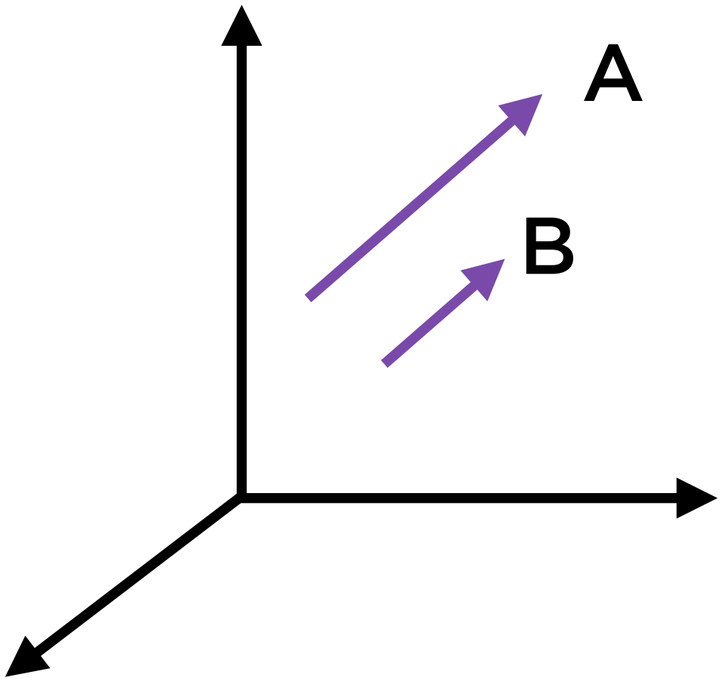
Vectors A and B are at 90 degrees

Orthogonal vectors represent uncorrelated data

A and B are unrelated, independent

Cosine of 90 degrees = 0

Aligned Vectors



Vectors A and B are parallel

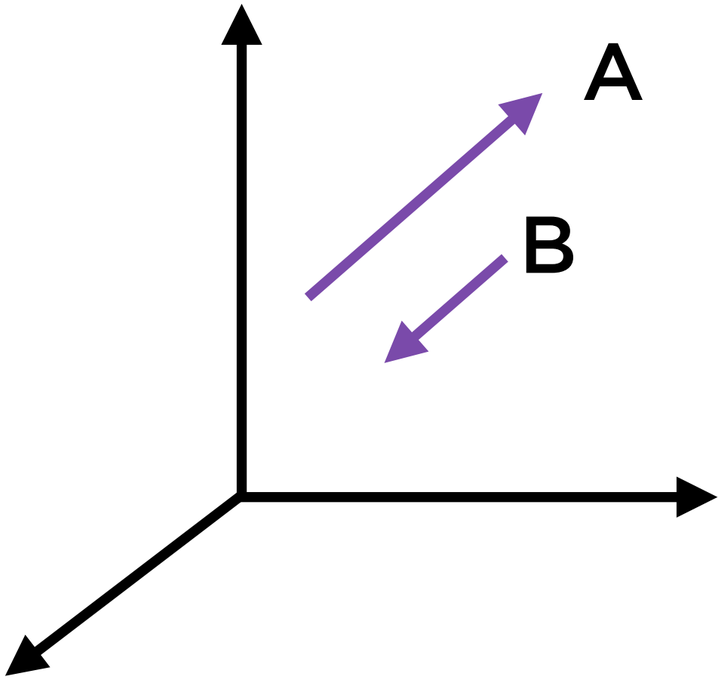
Angle between them is 0 degrees

Perfectly aligned

Correlation of 1 (highest possible)

Cosine of 0 degrees = 1

Opposite Vectors



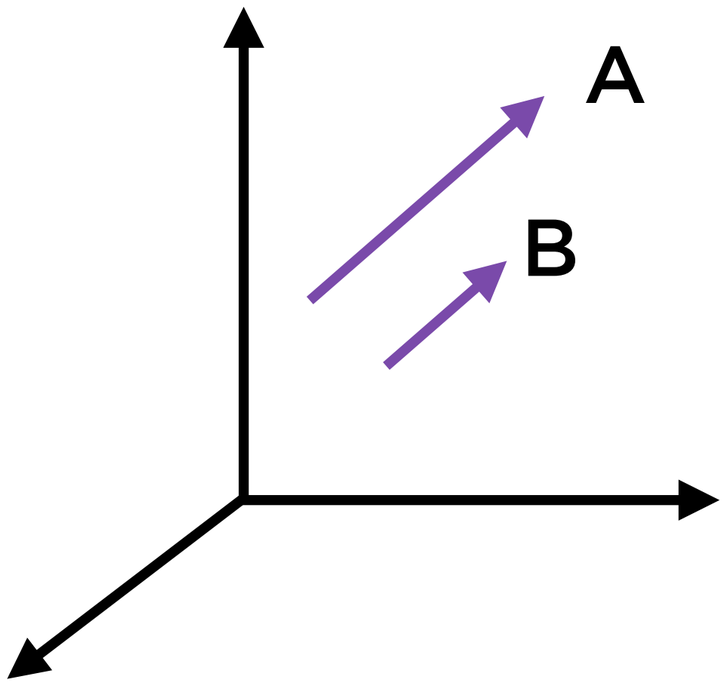
Vectors A and B point in opposite directions

Angle between them is 180 degrees

Perfectly opposed

Cosine of 180 degrees = -1

Cosine Similarity



Quick and intuitive way to express alignment between two vectors

Each vector represents a single point

In three dimensions, a point is represented as

$$(x_i , y_i , z_i)$$

Cosine Similarity

$$\cos(\theta) = \frac{\langle A, B \rangle}{||A||\,||B||}$$

$$||A|| = \sqrt{x_A^2 + y_A^2 + z_A^2}$$

$$||B|| = \sqrt{x_B^2 + y_B^2 + z_B^2}$$

$$\langle A, B \rangle = x_A x_B + y_A y_B + z_A z_B$$

Inner Product

$$\cos(\theta) = \frac{\langle A, B \rangle}{||A||\,||B||}$$

$$||A|| = \text{sqrt}(x_A^2 + y_A^2 + z_A^2)$$

$$||B|| = \text{sqrt}(x_B^2 + y_B^2 + z_B^2)$$

$$\langle A, B \rangle = x_A x_B + y_A y_B + z_A z_B$$

Inner (dot) product is the element-wise product of the two vectors

Vector Magnitudes

$$\cos(\theta) = \frac{\langle A, B \rangle}{||A||\ ||B||}$$

$$||A|| = \text{sqrt}(x_A^2 + y_A^2 + z_A^2)$$

$$||B|| = \text{sqrt}(x_B^2 + y_B^2 + z_B^2)$$

$$\langle A, B \rangle = x_A x_B + y_A y_B + z_A z_B$$

Magnitude of a vector is the inner product of vector with itself

Smaller the dot product, **less similar** the two vectors

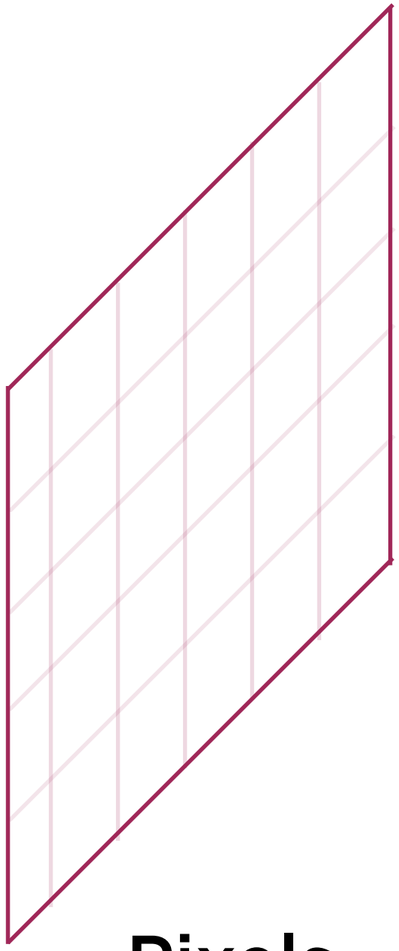
Larger the dot product, **more similar** the two vectors

Gram Matrix

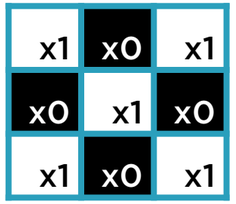
How do we extract **style** information from the style image?

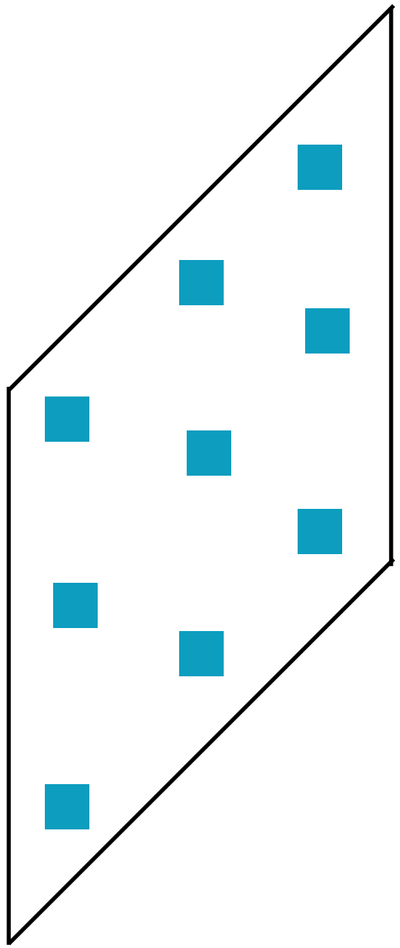
Feature Maps

Image



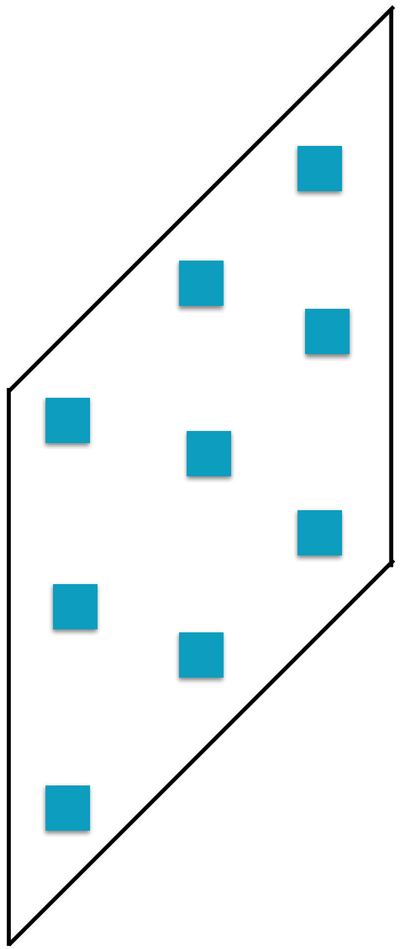
Pixels





Feature Map

Feature Maps



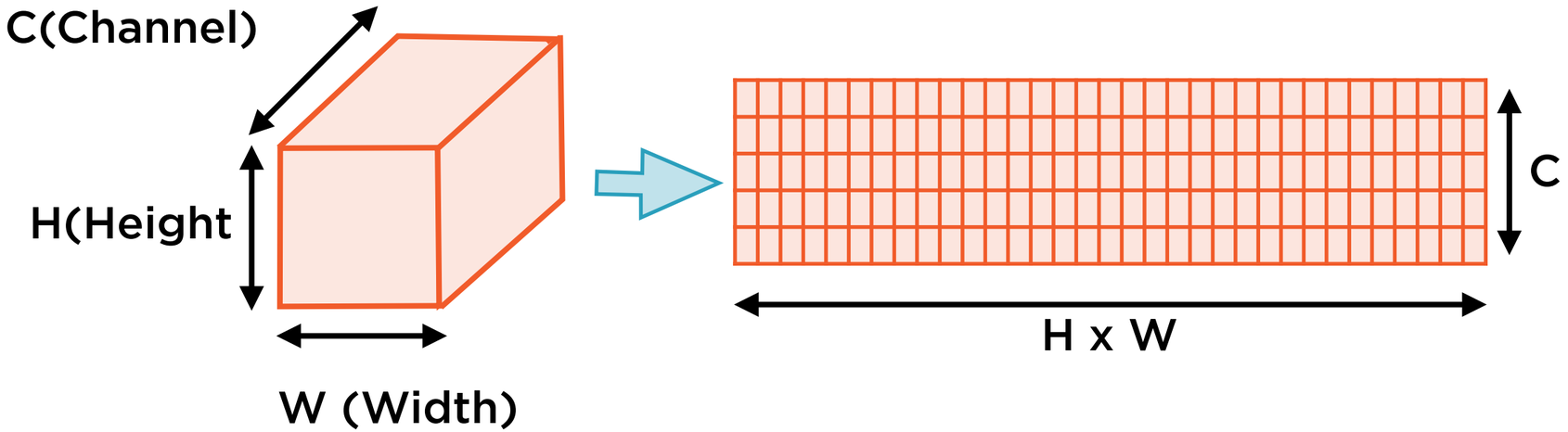
2-dimensional representation of features

Contain spatial, structure information

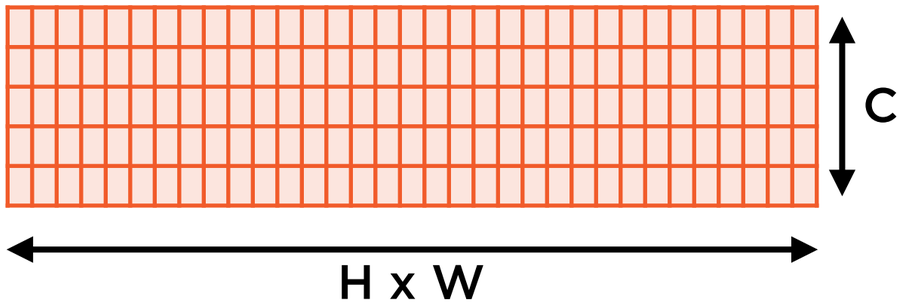
Also contain style information

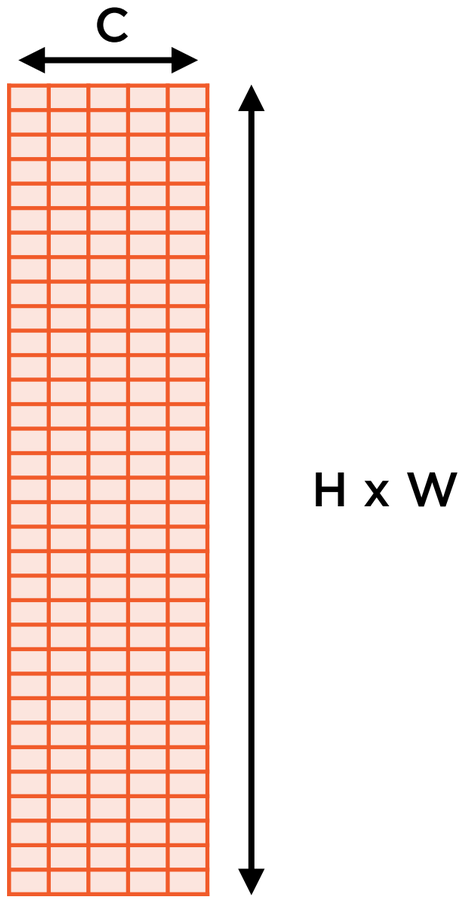
Flattening feature maps will preserve style and lose structure

Flattening Feature Maps



Calculate Dot Product

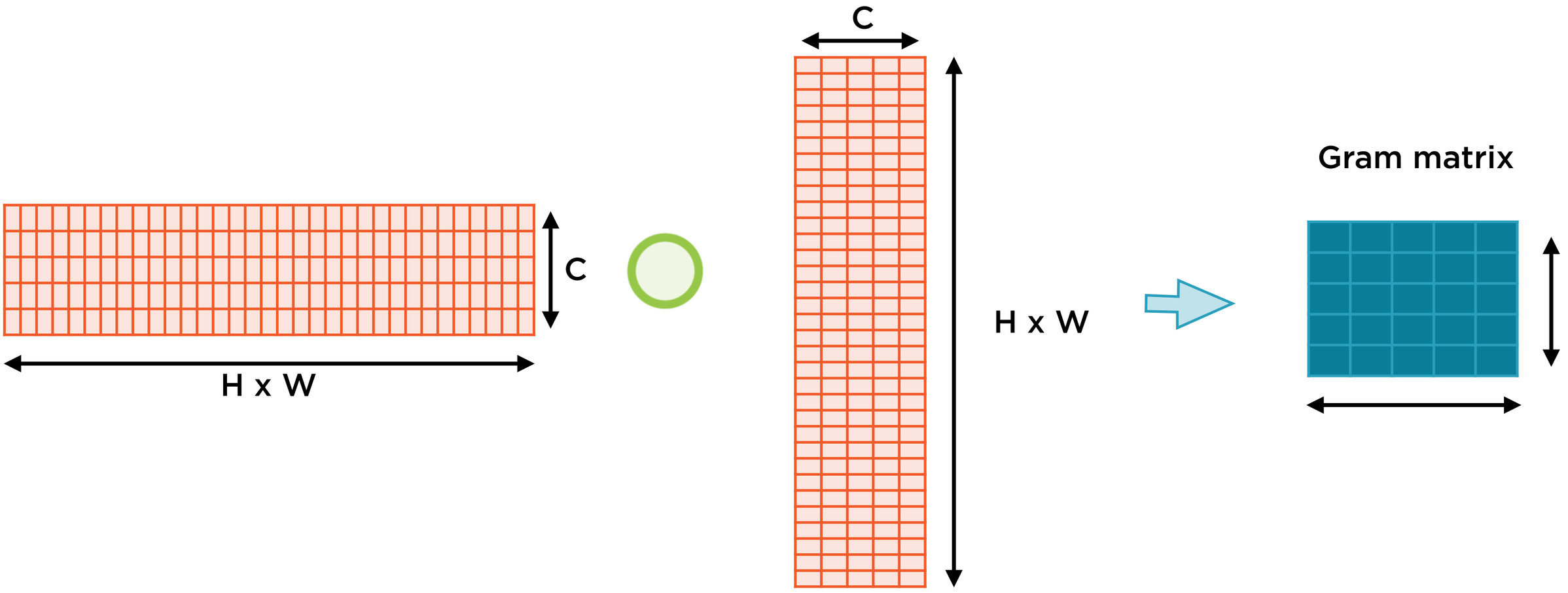




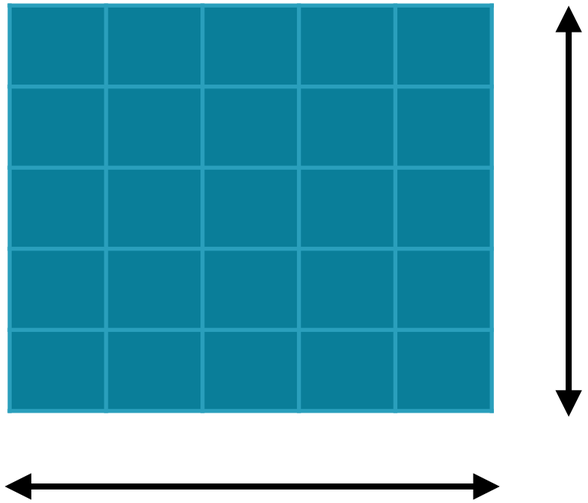
F

F^T

Gram Matrix



Gram Matrix

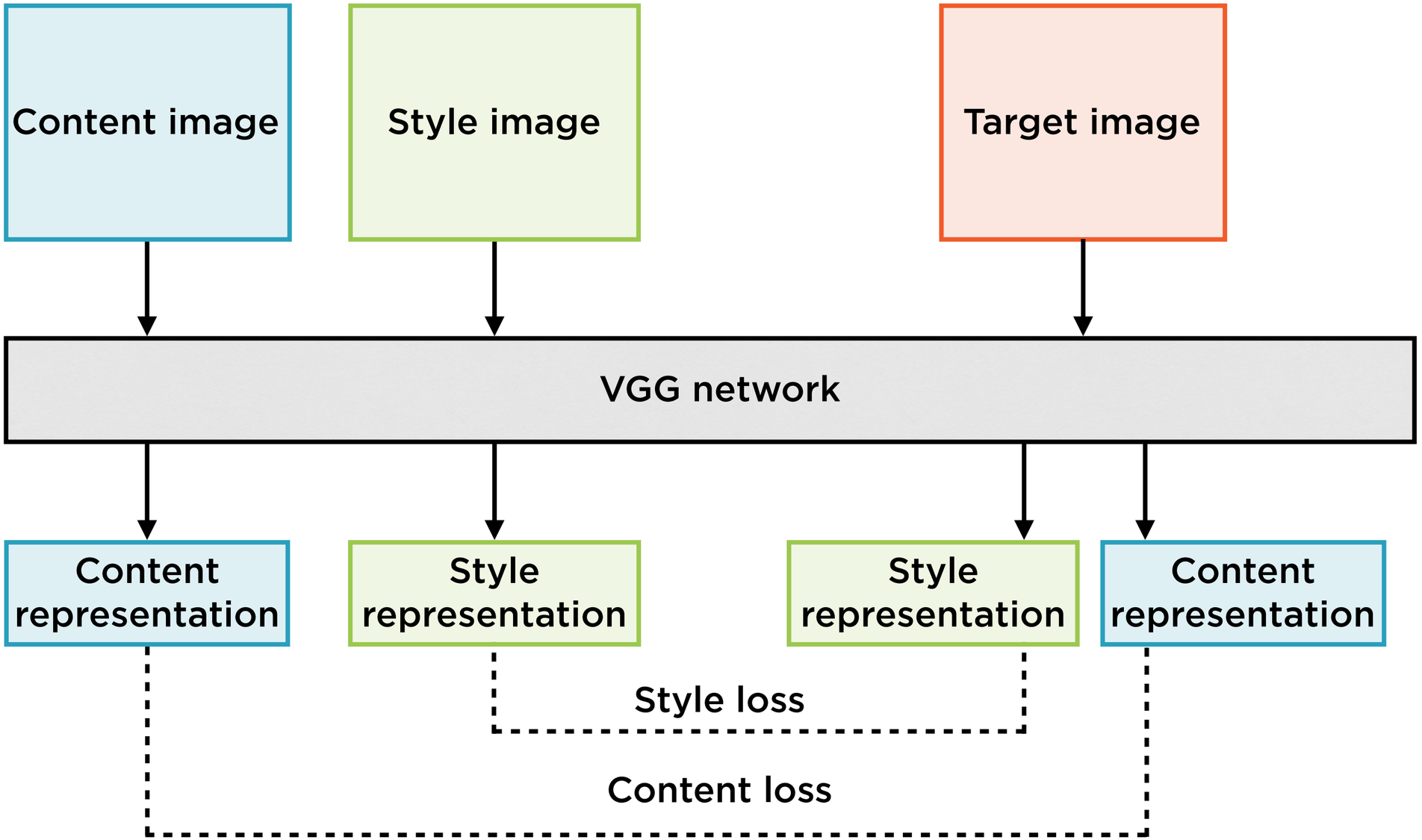


Flattening loses spatial information

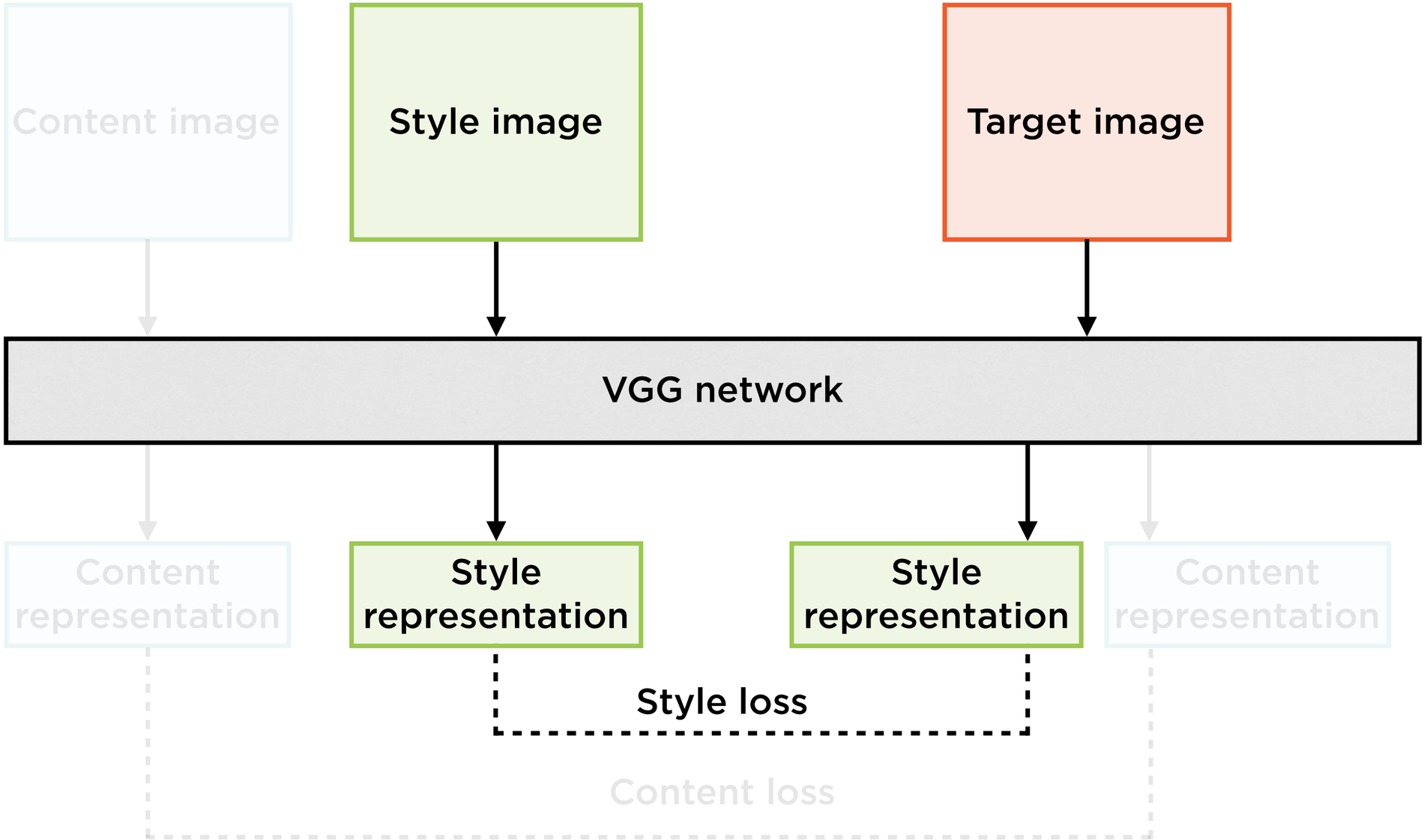
Dot product of the flattened feature maps show how features co-occur

Gives us style information from the image

Style Transfer



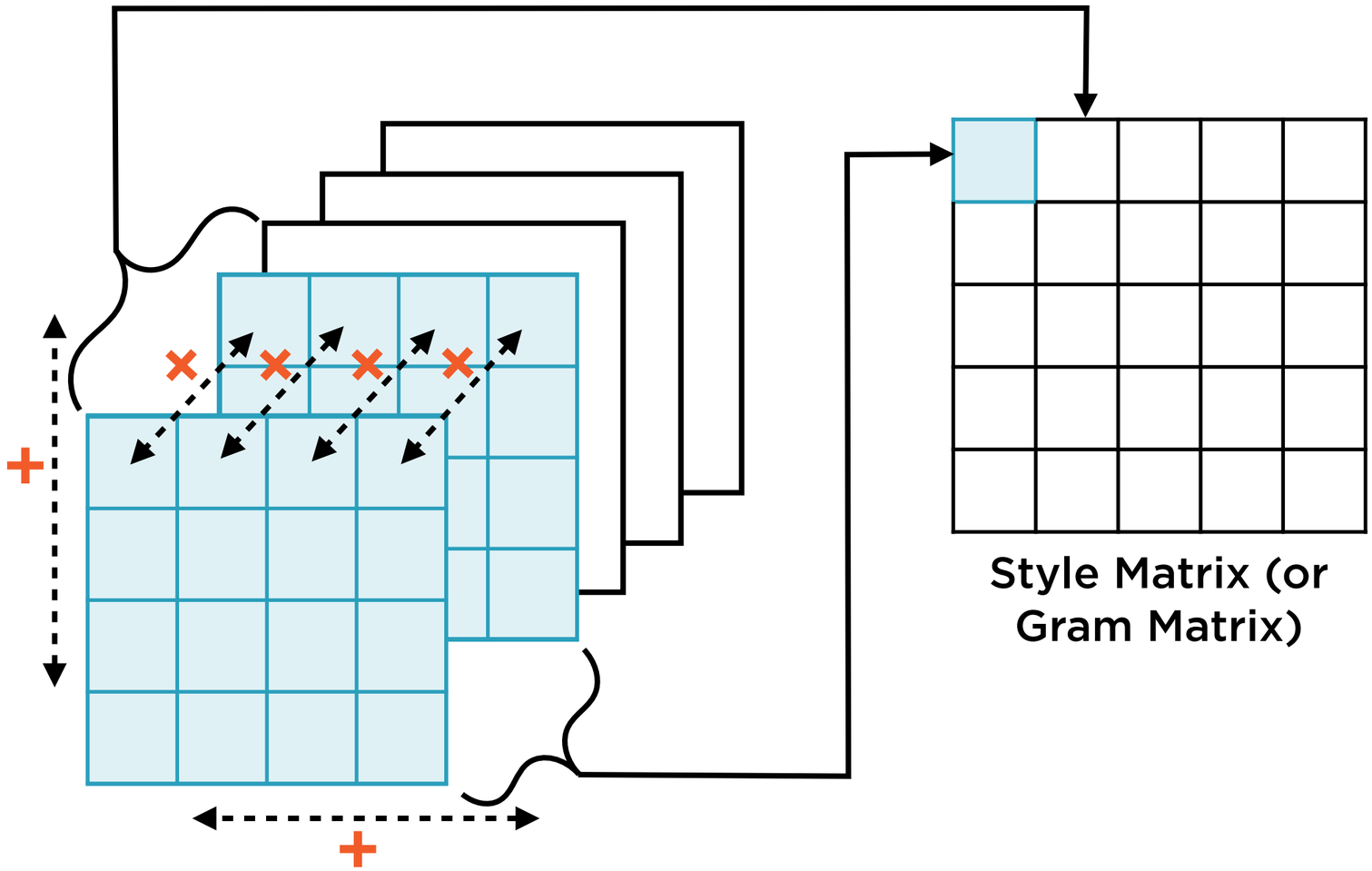
Minimize Style Loss



Difference between the **Gram matrix** of the target image and the style image

Gram matrices from the **style layers of interest**

Computation of Style Matrix (Gram Matrix)



Minimize Style Loss

Style loss: Difference between gram matrices of target and style image

Use gram matrices from style layers of interest

Typically use style information from several layers

MSE loss between style gram matrices and target gram matrices

A Gram Matrix summarizes the dot (inner) products of columns in a data matrix

Gram Matrix

$$G(W) = W * W^T$$

k x k k x z z x k

Each element of the Gram matrix corresponds to the inner (dot) product of corresponding vectors of W

Demo

Set up a deep learning VM

Demo

Use convolutional filters to detect features

Summary

Understand what style transfer is

Convolution layers in style transfer

Content, style, and target images

Content loss and style loss

Feature maps and Gram matrices