Neural Style Transfer

Outline

- VGGNet
- Neural Style Transfer
- A pytorch demo

VGGNet

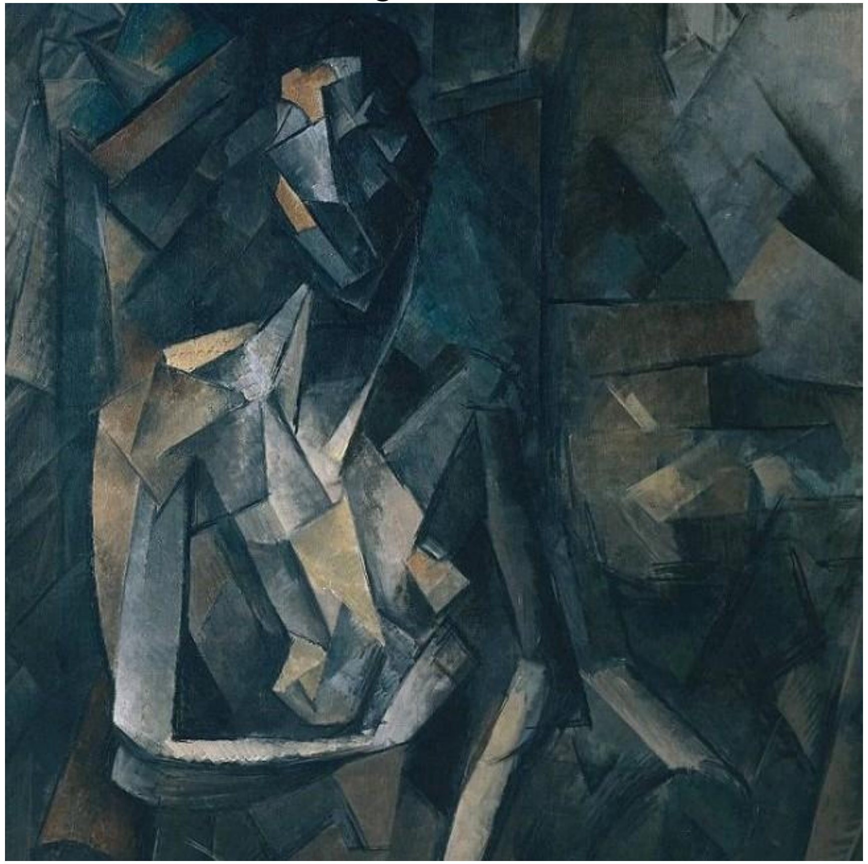
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

5 five groups of conv

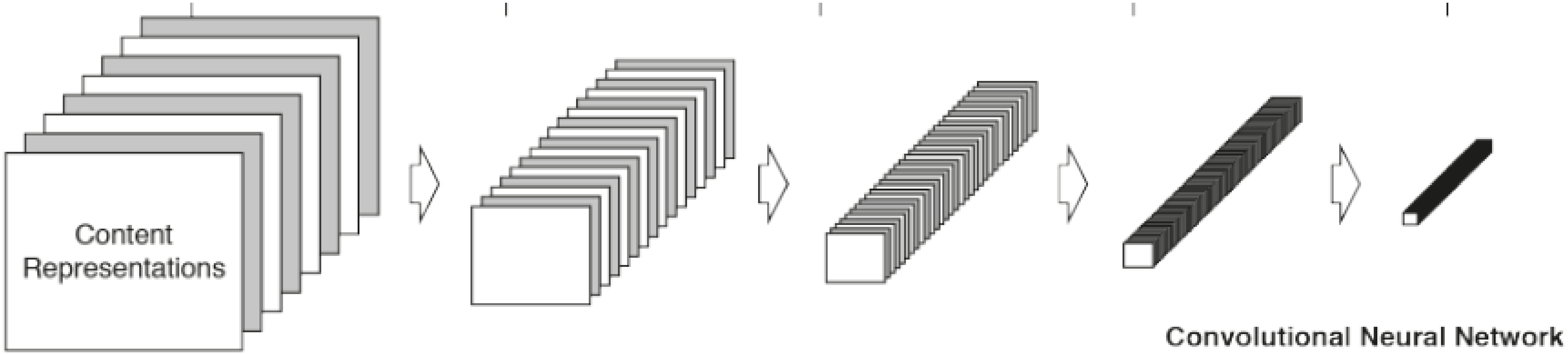
Neural Style transfer

Content

Styl
e



Content


Content
Representations
Convolutional Neural Network


a
b
c
d
e

Notation

N_l: # distinct filters in l layer
M_l: size of feature maps i.e. the height times the width of the feature map
F^l: convolution response where F^l_{ij} is the activation of the i^{th} filter at position j in layer l $\left(F^l \in \mathcal{R}^{N_l \times M_l}\right)$

Content Loss

Let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated and P^l and F^l their respective feature representation in layer l.

We then define the squared-error loss between two feature representations

$$\mathcal{L}_{content}(\vec{p},\vec{x},l) = \frac{1}{2}\sum_{i,j}\left(F_{ij}^{l} - P_{ij}^{l}\right)^2$$

The derivative of this loss with respect to the activations in layer l equals

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^{l}} = \begin{cases} \left(F^{l} - P^{l}\right)_{ij} & \text{if } F_{ij}^{l} > 0 \\ 0 & \text{if } F_{ij}^{l} < 0 \,. \end{cases}$$

The gradient with respect to the image $\vec{x}$ can be computed using standard BP. Thus we can change the initially random image $\vec{x}$ until it generates the same response in a certain layer of the CNN as the original image $\vec{p}$.

Style Loss

To obtain a representation of the *style* of an input image, we use a feature space to capture texture information.

These feature correlations are given by the Gram matrix $G^l \in \mathcal{R}^{N_l \times N_l}$, where G_{ij}^l is the inner product between the vectorized feature map i and j in layer l:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l$$

We minimize the mean-squared distance between the entries of the Gram matrix A^l from the original image and the Gram matrix G^l of the image to be generated

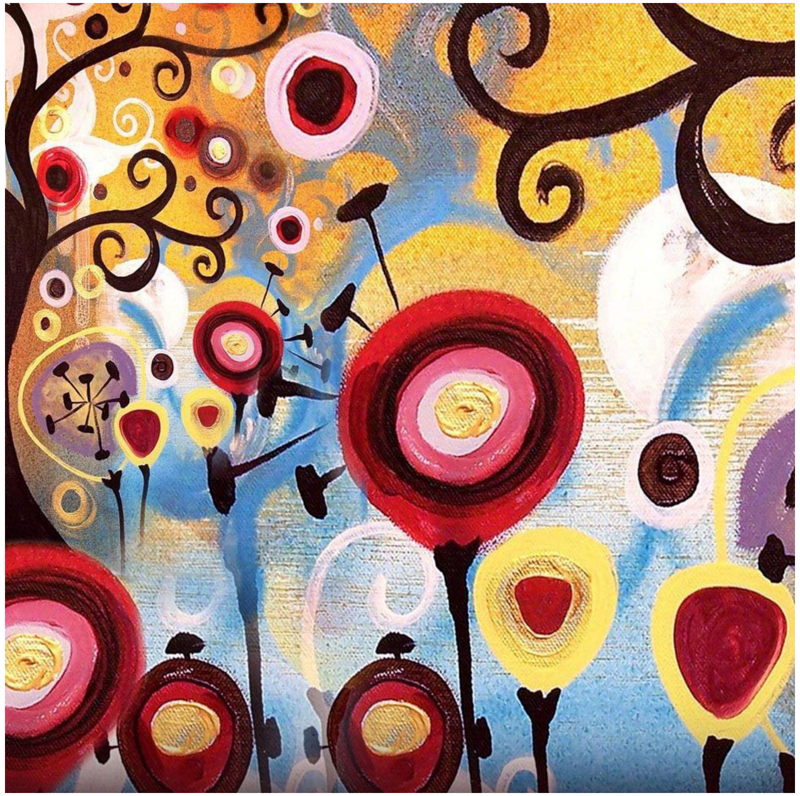
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

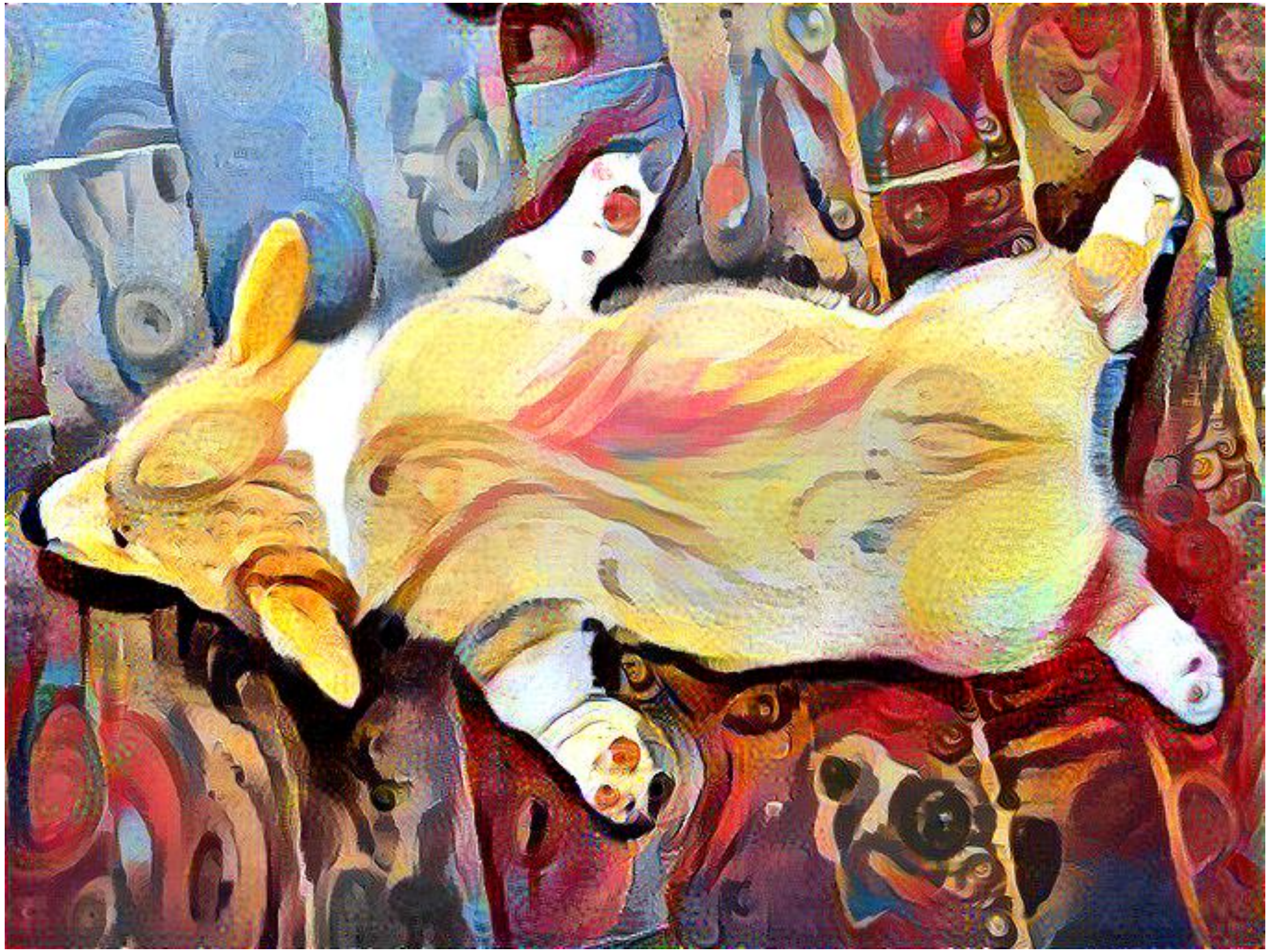
And the total loss is

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Let $\vec{p}$ be the photograph and $\vec{a}$ be the artwork. The loss function we minimize is

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$





Refernces

Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556*(2014).

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." *arXiv preprint arXiv:1508.06576* (2015).

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. “Deep learning.” *Nature* 521.7553 (2015): 436-444.