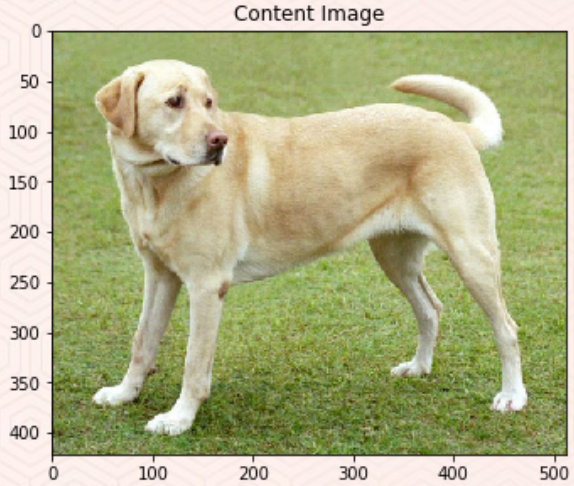
사진을 회화로 만들기 – Neural Style Transfer

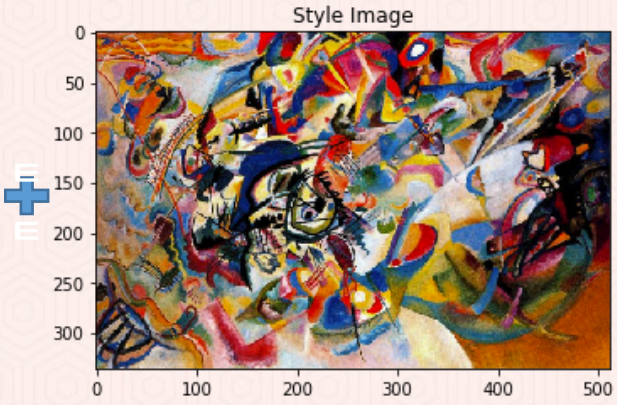
오늘의 학습내용

- 사진을 회화 이미지로 만들기
- Neural Style Transfer 알고리즘
- PyTorch를 이용한 Neural Style Transfer 구현

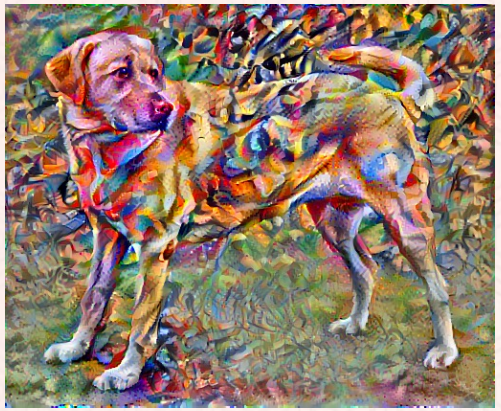
회화 이미지 만들기

❖ 사진에 원하는 회화 이미지를 덧씌워서 사진을 회화스타일로 만들어봅시다.





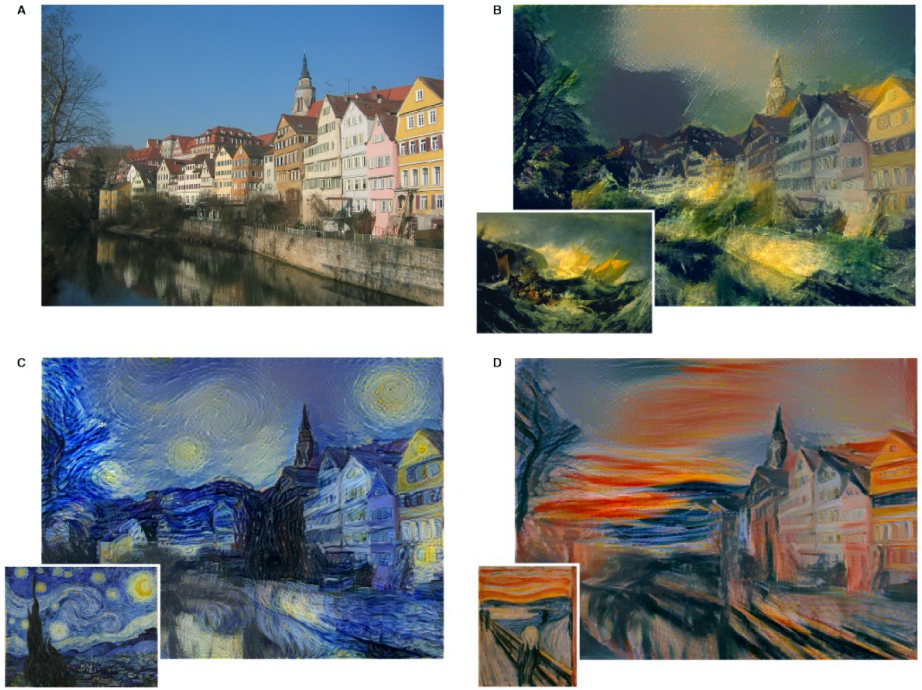
Input



Result

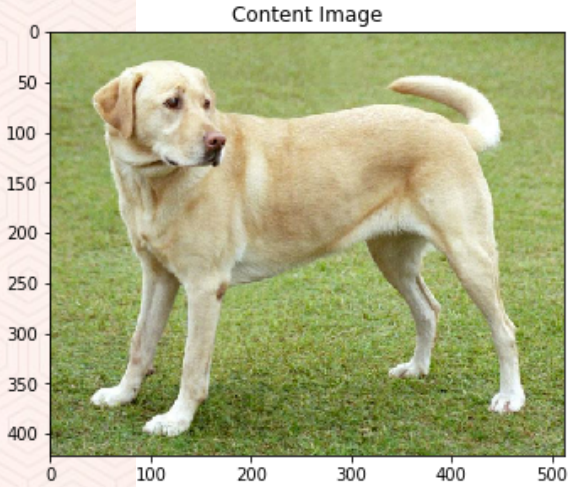
컴퓨터 비전 문제 영역 – Neural Style Transfer

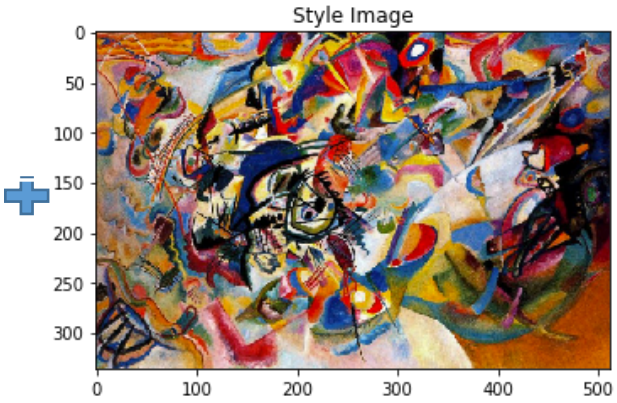
❖ Neural Style Transfer : 콘텐츠 이미지에 스타일 이미지를 덧씌운 합성 이미지를 만드는 문제 영역

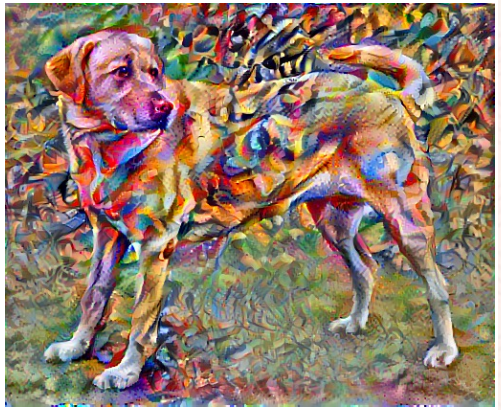


Neural Style Transfer

❖ Content Image + Style Image를 조합해서 Content Image에 원하는 스타일을 덧 씌웁니다.







Input

Result

논문 리뷰 - A neural algorithm of artistic style

- Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).
- https://arxiv.org/pdf/1508.06576.pdf
- 핵심 아이디어 : CNNs에서 style representation과 content representation이 분리가능하다.(separable)

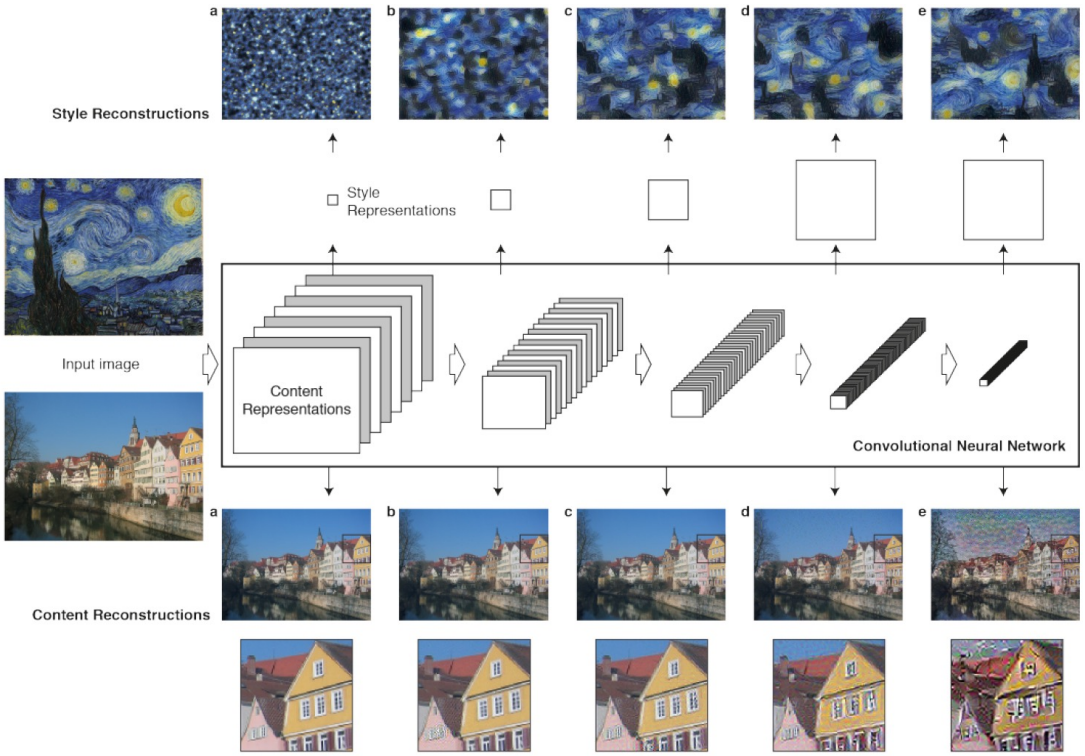
A Neural Algorithm of Artistic Style

Leon A. Gatys,[1,2,3*] Alexander S. Ecker,[1,2,4,5] Matthias Bethge[1,2,4]

[1]Werner Reichardt Centre for Integrative Neuroscience
and Institute of Theoretical Physics, University of Tübingen, Germany
[2]Bernstein Center for Computational Neuroscience, Tübingen, Germany
[3]Graduate School for Neural Information Processing, Tübingen, Germany
[4]Max Planck Institute for Biological Cybernetics, Tübingen, Germany
[5]Department of Neuroscience, Baylor College of Medicine, Houston, TX, USA
*To whom correspondence should be addressed; E-mail: leon.gatys@bethgelab.org

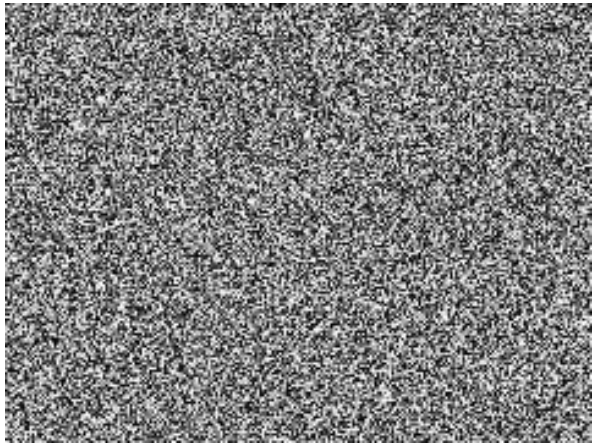
In fine art, especially painting, humans have mastered the skill to create unique visual experiences through composing a complex interplay between the content and style of an image. Thus far the algorithmic basis of this process is unknown and there exists no artificial system with similar capabilities. However, in other key areas of visual perception such as object and face recognition

Overall Architecture



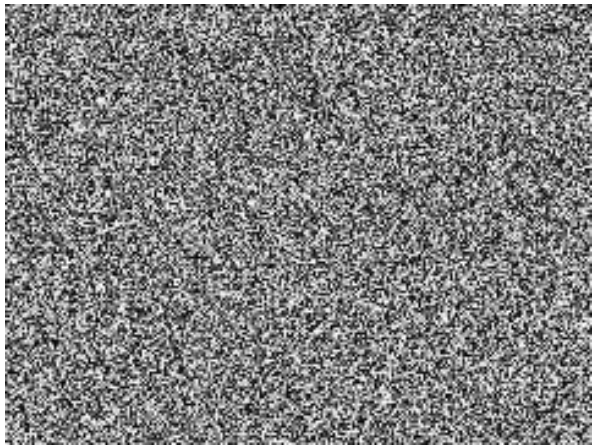
Overall Architecture

- Neural style Transfer 기법은 기존 기법과 달리 최적화 대상이 네트워크 파라미터가 아니라 합성이 이미지입니다.
- 따라서 Networks Weight는 Fix하고 Image를 Gradient Descent로 업데이트합니다.
- 합성 이미지는 처음에는 Noise 이미지로부터 시작합니다.
- 만약 합성 이미지가 Content 이미지에 대해 최적화된다면 아래와 같이 변경될 것입니다.



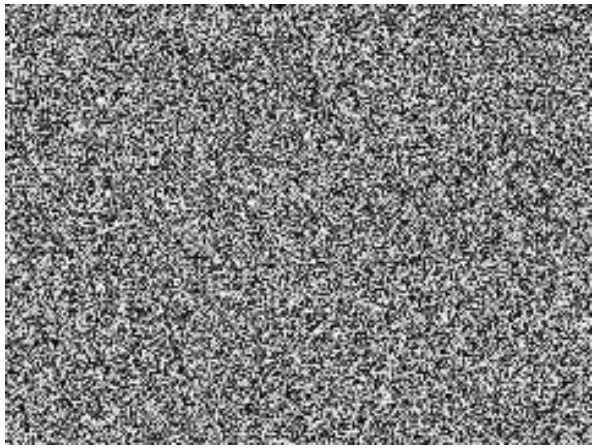
Overall Architecture

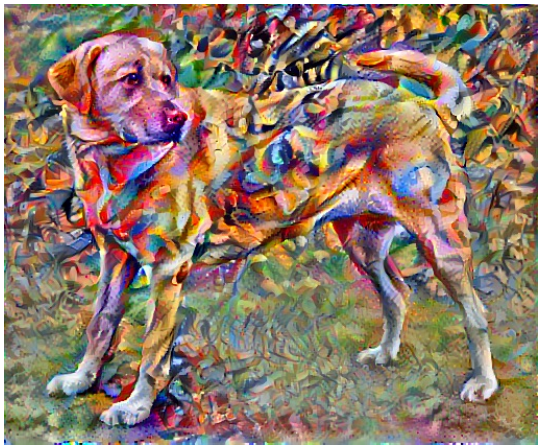
❖ 만약 합성 이미지가 Style 이미지에 대해 최적화된다면 아래와 같이 변경될 것입니다.



Overall Architecture

❖ 만약 합성 이미지가 Content 이미지와 Style 이미지가 적절히 배합된 형태로 최적화된다면 아래와 같이 변경될 것입니다.





Convnet = VGGNet을 사용

- VGGNet E를 사용
- Max Pooling을 Average Pooling으로 바꾼 VGGNet을 사용

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					

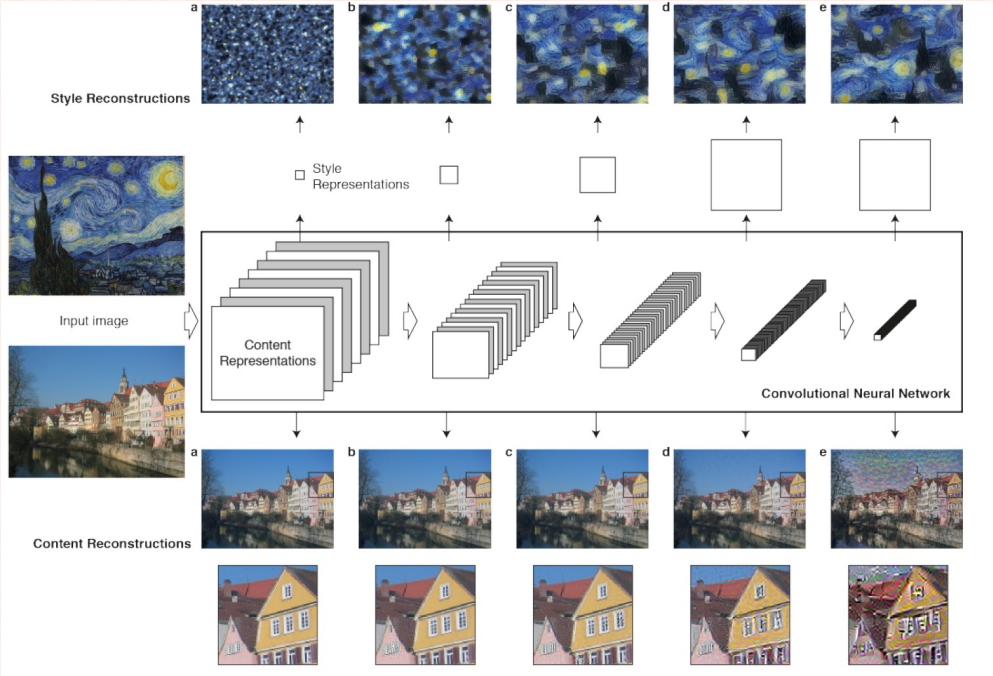
Content Loss

- p : 원본 이미지, x : 생성할 이미지(노이즈로부터 시작)
- P : 원본 이미지의 특징 맵 F : 생성할 이미지의 특징 맵

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

Content Reconstruction

❖ 낮은 레벨의 Layer는 Low-level Feature, 높은 레벨의 Layer는 high-level Feature를 가지고 있는 경향을 확인할 수 있다.



Style Loss

- L : 합성하는 Layer 개수
- w_l : Layer l의 가중치
- E_l : Layer l의 Style Loss

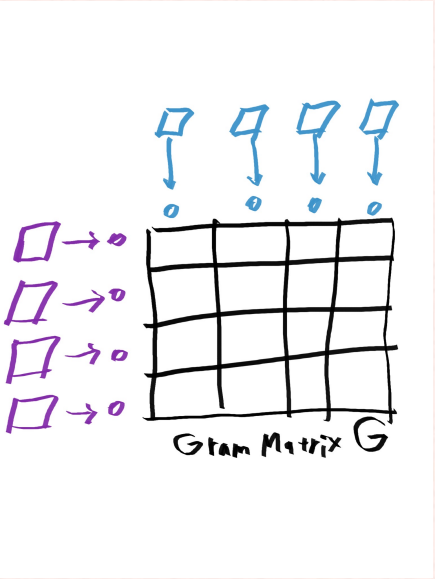
$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

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

Style Loss

- Style = Filter들의 Correlation -> Gram Matrix를 이용해서 계산
- 예를 들어, k개의 WxH 특징 맵들이 있다면 Spatial하게 평균값을 취해서 WxH는 1개의 스칼라 값으로 모으고 k x k 의 Gram Matrix를 계산한다.

$$G_{ij}^{l} = \sum_{k} F_{ik}^{l} F_{kj}^{l}$$



Style Loss

- Gram Matrix를 이용해서 Style Loss를 계산하는 식은 아래와 같다.
- $\mathbf{G_{ij}^l}$ = Style Image의 Gram Matrix, $\mathbf{A_{ij}^l}$ = Generated Image의 Gram Matrix

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

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

Style Loss

- ❖ 전체 Loss를 계산하는 식은 아래와 같다.
- ❖ Alpha, beta값을 조절함으로써 output에 style을 얼마나 줄지 결정할 수 있다.

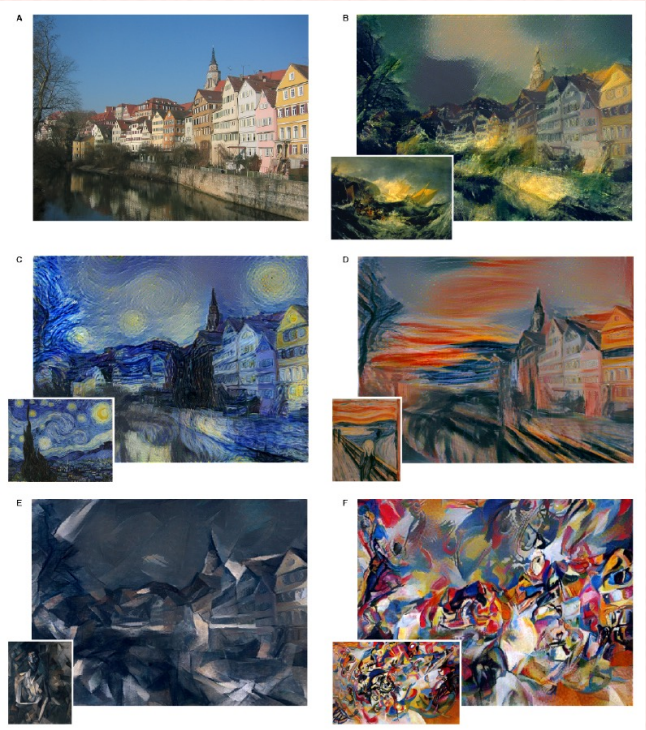
$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Alpha, Beta

❖ Alpha, beta값을 조절함으로써 output에 style을 얼마나 줄지 결정할 수 있다.

Experiment Result

❖ 다양한 회화 이미지를 조합함으로써 다양한 합성 이미지를 만들 수 있다.



파이토치(PyTorch)를 이용한 Neural Style Transfer

- 파이토치(PyTorch)를 이용해서 Neural Style Transfer 알고리즘을 구현해봅시다.
- 6강_PyTorch를_이용한_Neural_Style_Transfer.ipynb
- https://colab.research.google.com/drive/1LybB57frQNCVwfSwOY-RZ0OXrG6giFLS?usp=sharing