(H. Kim) Texture Synthesis Using CNNs

작성일시	@November 30, 2021 9:08 AM
Keywords	Style Transfer
Reviewer	㉧ 김하연
Paper	
발표일자	@September 17, 2021
발표영상	
최종 편집	@February 7, 2022 5:11 PM
Prof.	☑
PPT	
Q&A Link	
비고 (추가 노트)	https://junklee.tistory.com/69 - 정리가 잘 되어있어서 참고
추가자료	
Reviewed	☑
열	
속성	
속성 1	

```
211130 늦었지만 그래도 시작
211201 큰 틀 다 잡기
211202 디테일 완성
* 나중에 Gram matrix 직접 풀어서 손실함수 구해보기 !!(손으로 해보는게 최고!)
211203 보충(뿌-듯)
```

Texture Synthesis Using CNNs

0. Introduction

> The goal of visual texture synthesis is to infer a generating process from an example texture, which then allows to produce arbitrarily many new samples of that texture.

→ 텍스처 합성의 목적은 주어진 텍스처로 부터 많은 샘플을 만들어 내는 것.(주어진 sample texture로 더 많은 sample texture를 만들어 내는 것이 목표)

→ 이때 평가기준 : human inspection(합성한 많은 샘플들 중 실제 original texture를 못찾으면 성공적). 즉, 합성한 것을 보고 original texture를 못 찾을만큼 합성이 잘 된 것!(아래 사진에서 우측이 actual, 좌측이 generated image인데, 둘 중 어느 것이 original인지 찾기 힘듦 → 성공적)

Approach to find a texture generating process

1. original 텍스쳐로부터 각각의 픽셀 / 전체 패치를 리샘플링 함으로써 생성(implicit)

   → non-parametic resampling(효율적으로 높은 퀄리티의 텍스쳐 생성 가능), not define an actual model for natural textures(단순히 sample을 랜덤하게 뽑는 과정)

2. parametic texture model을 명시적으로 정의(explicit) ← 이 논문은 이를 채택

   → 조금 더 정량적인 방법으로 측정 가능(같은 결과를 생산한 모든 이미지를 같은 이미지로 여김)

* 선례연구(Julesz가 제안한 n개의 인접한 픽셀의 히스토그램, Gabor filter, ***Portilla와 simoncelli가 제안한 linear filter bank/steerable pyramid 방법***)는 모두 natural texture의 모든 범위를 잡아내는데 실패

▼ ***그래서, 이 연구는 왜 필요한가(연구 목적)***

- 새로운 parametic texture model로 natural texture의 모든 범위를 잡아내기 위해서 !

→ 객체 인식으로 훈련된 CNN + feature response에 대한 공간적인 statistics를 결합(이러한 방식으로, 공간적으로 불변하도록 parameterised된 텍스쳐 모델 획득 가능)

- 선례연구 중 linear filter bank와 유사해 보이지만 가장 큰 차이점 : 입력 이미지 자체가 Variable. 기존 방식은 input image를 입력하고 그에 맞는 variable을 찾아 최적화하는 방향으로 움직였으나, 이 논문의 모델은 입력 이미지 자체를 variable로 사용할 수 있다!

→ 그래서, 이 연구에서는 feature response간의 상관관계를 단 하나의 statistic으로 사용한다.

1. CNN(convolutional neural network)

1) CNN : 이름 그대로 'convolution'을 이용한 NN. (재미있어서 가져와본 TMI) 초창기에 CNN을 개발한 분들이 고양이가 보는 것마다 자극 받는 뇌의 위치가 다른 것을 보고 아이디어를 얻어 CNN을 만들었다고 합니다(아! 뇌에 모든 것을 보기 보다 '부분'에 집중을 해야겠구나! 하고 만든 게 CNN) → 그래서 CNN은 filter(일종의 '부분')를 이용하여 설정할 수 있다.

→ **filter**를 사용하여 모든 이미지를 훑을 때, 이동이 필요한데 이때 얼만큼 이동하는가를 **stride**라고 부름! 이렇게 convolution을 활용하면 실제크기 보다 작은 결과값을 도출하게 되는데 이때, 손실을 막아주고자 남은 크기에 0을 채우는 것을 **padding**이라고 부름

Padding에서 0이라는 값을 인위적으로 넣어서 오류를 초래할 것 같지만, padding없이 convolution으로 인한 손실보다는 더 적게 초래하므로 이를 활용 = trade-off

2) 이 논문에서는 VGG-19 사용 / 16 convolutional and 5 pooling layers(fully connected layer X)

→ filter size = 3*3*k(k는 input feature maps의 수), input과 output의 stride와 padding이 동일

→ maximum pooling = 2*2 → down sample

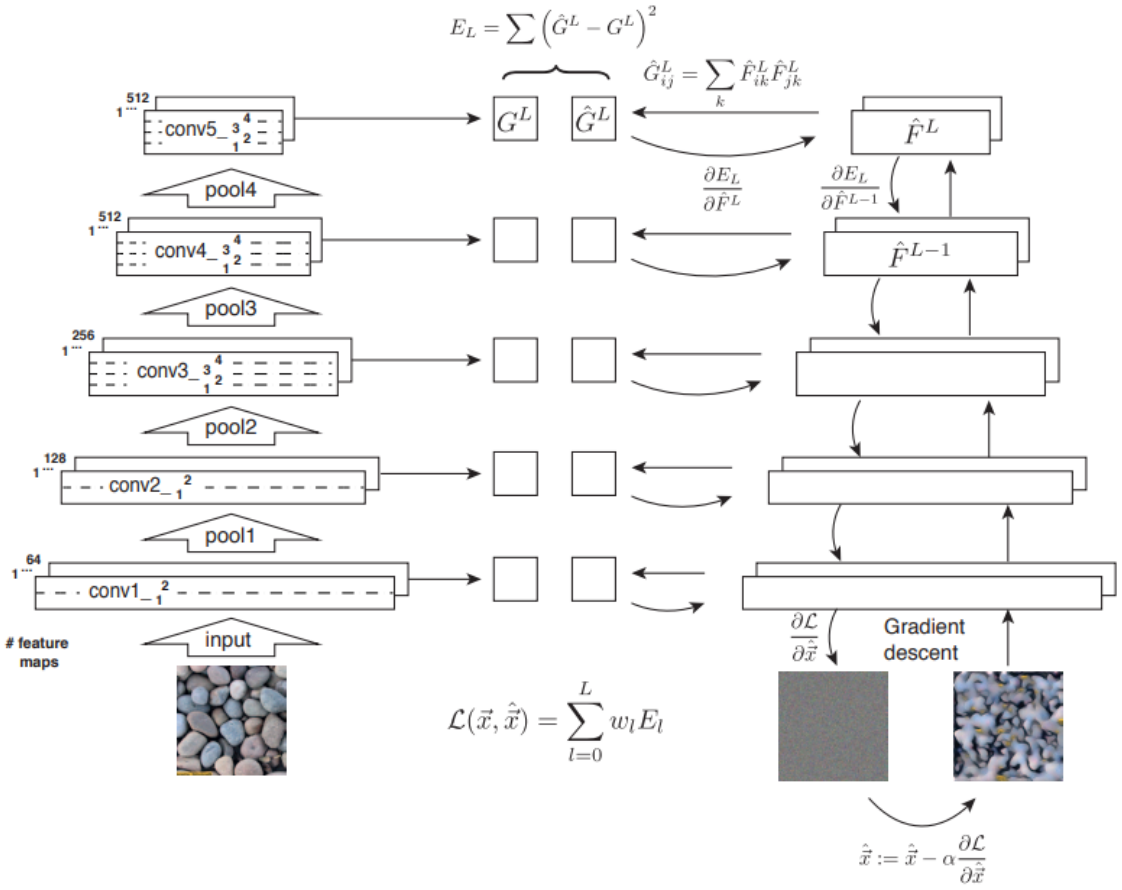
2. Texture model & Texture generation(Method)

Texture model = feature correlations를 계산하는 것

1) Texture model

- Portilla and Simoncelli의 방식을 채택 : 이미지로 부터 다른 사이즈들의 이미지 feature를 추출 → feature에 대한 spatial summary statistics을 계산 → white noise로 초기화 되어진 랜덤 이미지에 gradient descent를 수행함으로써 같은 stationary description을 가지는 이미지를 찾기(explicit한 방법에서 같은 결과를 나타내는 이미지를 같은 이미지로 여긴다고 하였으므로!)
- But, Portilla&Simoncelli와 가장 큰 차이점 : linear filter band & summary statistics → feature space provided by a high performing deep NN & only one spatial summary statistics = 'Feature Correlation'

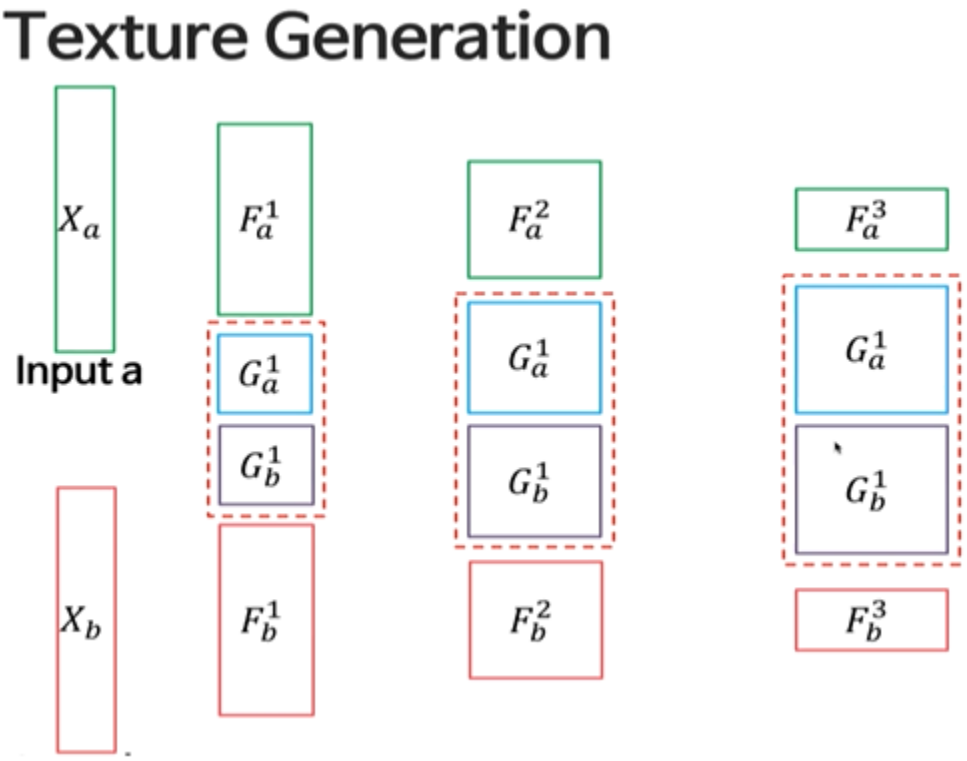
2) Texture generation



목적 : 왼쪽이 input이미지, 그리고 오른쪽이 generated이미지이다. 이때 오른쪽에서 회색 노이즈와 자갈 비슷한 모양의 이미지가 있는데, 우리는 회색 노이즈 이미지에서 왼쪽 input이미지와 유사한 오른쪽 자갈 패턴 이미지로 변환 할 것이다.

방법 : 왼쪽을 input, 오른쪽을 output이라고 했을때, 우리는 각 레이블 당 중간에 있는 정사각형에 Gram matrix를 구해서 입력하고 두 값을 비교하는 E_L 함수(MSE)를 구한다 → 이 함수의 값이 최소가 되었을 때, input과 유사한 output이미지를 생성할 수 있다.

⇒ 즉, 랜덤으로 노이즈 이미지가 주어졌을때, input과 노이즈 이미지에서는 똑같은 레이블(conv)을 사용하는데 이때 Gram matrix를 구하여 두 값의 차이가 최소가 되는 방향으로 계속 학습을 거쳐 유사한 이미지로 만들 수 있다.(가장 중요한 건, 이미지 그 자체가 variable이 된 것이다! → 우리는 따로 어떠한 변수를 설정한 것이 없다!)



▲ 위와 같이 input a가 주어졌을 때, 다른 feature들을 capture한 후 각각의 Gram matrix를 구한다. 이때 a와 b간의 gram matrix를 비교하여 둘의 차이를 최소화 시킴으로써, texture generation을 수행한다.

*** 식으로 보는 Texture Generation**

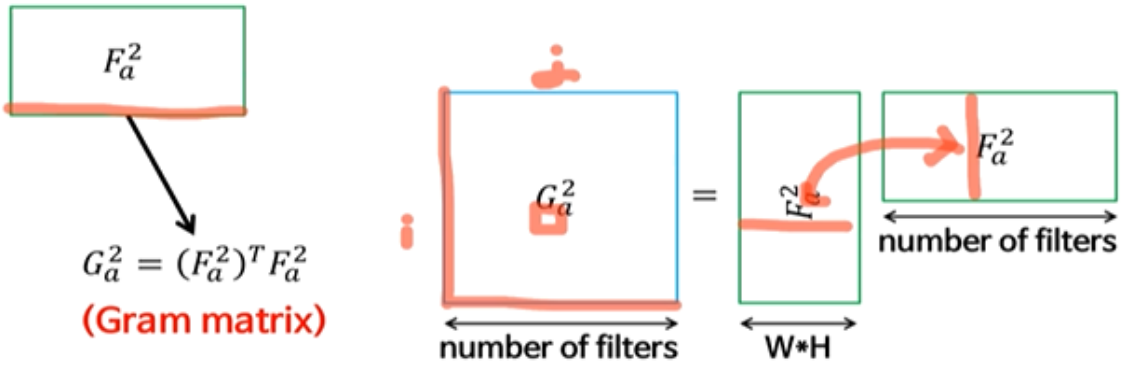
(N_l : filter, M_l : feature map의 size, $F^l \in R^{N_l * M_l}$: feature map, G_{ij}^l : feature map의 i와 j사이의 내적 = Gram matrix)

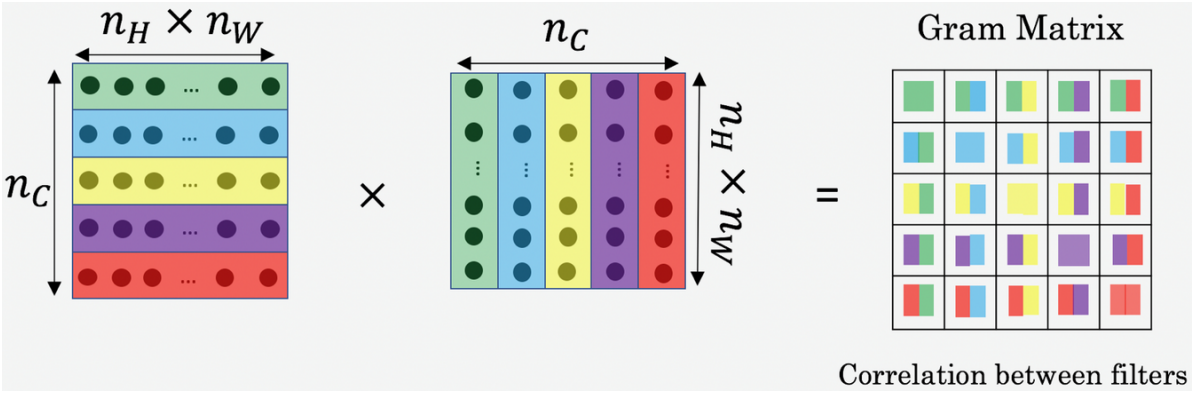
$$G_{i,j}^l = \sum_k F_{ik}^l F_{jk}^l$$

1. 이 논문에서는 **feature의 correlation**만을 유일한 stationary statistic으로 사용하므로, l이라는 레이블의 k위치에 있는 i번째, j번째의 feature map을 내적한다. 이것을 gram matrix라고 부르는데, 각 레이블에 있는 RGB채널의 공분산을 이용하여 style representation을 레이블링으로 하여 학습을 시킬 수 있다.

▼ ***Gram Matrix***

Feature Correlations





$$G = A^T A = A A^T$$

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

2. 원본 이미지를 $\vec{x}$, 생성된 이미지를 $\hat{\vec{x}}$라고 했을 때, 각 레이블에서 그 둘의 Gram matrix의 차이를 이용하여 위 공식처럼 해당 레이블의 total loss를 구할 수 있다.(MSE)

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

3. 이번엔, 해당 label이 아닌 total loss를 구하면 위의 공식과 같이 나온다. 여기서 w_l은 해당 레이블의 가중치이다. 모든 w_l값을 더하면 1이라는 걸 알 수 있다.(가중치는 모두 합하면 1이므

로)

$$\frac{\partial E_l}{\partial \hat{F}_{ij}^l} = \begin{cases} \frac{1}{N_l^2 M_l^2}((\hat{F}^l)^T(G^l - \hat{G}^l))ji & \text{if} \hat{F}^l ij > 0 \\ 0 & \text{if} \hat{F}_{ij}^l < 0 \end{cases}$$

4. 이후에는 다음과 같이 해당 레이블의 손실 함수를 생성된 이미지의 feature map에 대해 미분을 하여서 최솟값으로 근사시킴으로써 이미지를 최적화시킬 수 있다.(L-BFGS채택)

> 💡 **결론 :** 하나의 레이블(l)에서 원본 이미지와 생성된 이미지의 Gram matrix를 구해서 차이를 최소화(MSE이용) = $E_l \rightarrow E_l$을 생성된 이미지의 feature map에 대해 미분 → 손실을 최소화 하는 방식으로 texture를 생성

3. Result & Discussion

▼ 나를 위한 용어 정리

1. caffe-framework : Caffe (Convolutional Architecture for Fast Feature Embedding)는 빠른 속도와 모듈성을 내세운 딥러닝 프레임워크, Matlab에서 CNN을 구현해 놓은 것
2. L-BFGS : LBFGS는 Limited-memory quasi-Newton methods의 한 예시로써, Hessian 행렬을 계산하거나 저장하기 위한 비용이 합리적이지 않을 경우 유용하게 사용된다. 이 방법은 밀도가 높은 n×n의 Hessian 행렬을 저장하는 대신 n차원의 벡터 몇 개만을 유지하여 Hessian 행렬을 추정(approximation)하는 방식

   *** 나중에 hessian행렬에 대해 더 알아보기 !**
