Context Encoders
: Feature Learning by Inpainting

Introduction

(a) Input context

(b) Human artist

(c) Context Encoder ($L2$ loss)

(d) Context Encoder ($L2$ + Adversarial loss)

사람의 경우, (a)와 같은 그림을 봤을 때 쉽게 빈 공간의 내용을 주위 pixel을 사용해서 상상 가능 (한 번도 정확히 본 적 없어도!)

=> 이를 CNN을 통해 가능하게 하겠다!

Introduction

- Model : Context encoder
  - Encoder : capture the context of image into a latent feature representation
  - Decoder : 위에서 얻은 feature로 produce the missing image content

⇒ Closely related to autoencoder

Autoencoder : memory↓ 위해! (원본 img → bottle neck → 다시 복원)

- Autoencoder : input image -> pass through low-dim 'bottleneck' layer -> reconstruct

-> aim : obtain a compact feature representation of the scene

-> cons : jus compress the image content without learning a semantically meaningful representation

→ 위치 정보 남아 있음

- Denoising autoencoder : 위의 issue 다룸 by corrupting the input image and require the network to undo the image (training data 에 noise 더함 -> 학습결과가 noise 없는 데이터와의 error 최소화 하는 것이 목적)

-> cons : very localized & low-level -> not require much semantic information to undo

(위에 noise 를 얻는 경우 전체적인 image 정보가 필요하지 patch 수준의 정보는 필요 없음)

Noise + input → bottleneck에서 사라짐

- Context encoder : 더 어려운 문제 해결! -> fill in large missing area of the images (심지어 nearby pixel로 부터 hint를 얻을 수도 없게 큰 부분을 채워야 함!)
  - Require much deeper semantic understanding of scene & ability to synthesize high lvl feature over large spatial extents

Introduction

- Context encoder : train in completely unsupervised manner (autoencoder와 동일)
  - Understand the content of image & produce plausible hypothesis for the missing parts

-> fill missing region & maintain coherence with the given context => multi-modal

⇒ 이를 loss function을 결합함으로써 해결 : Reconstruction loss and adversarial loss 모두 minimize!

- Reconstruction loss (L2) : capture the overall structure of the missing region in relation to the context
- Adversarial loss : the effect of picking a particular mode from the distribution

분포에서 전체를 모두 해보고, 그 중 좋았던 걸 선택해서 하는 것이라 생각!

(c) Context Encoder ($L2$ loss)

(d) Context Encoder ($L2$ + Adversarial loss)

L2 loss 만 사용한 경우 : blurry result
L2 loss + Ad loss : much sharper prediction

Introduction

- Evaluating encoder and decoder independently
  - Encoder : image patch 의 context encoding -> encoding result feature 사용 -> dataset 에서 가장 가까운 context 검색 -> original(unseen) patch와 semantically 비슷한 patch 생성
  - Fine-tuning the encoder -> validate the quality of the learned feature representation
  - Decoder : can fill in realistic image content

-> first parametric inpainting algorithm (can give reasonable result for semantic hole-filling)

- Context encoder : non-parametric inpainting method 에서 computing nearest neighbor 위한 better visual feature 제공 가능!

Cf) parametric : unknown의 분포를 알 고 있다고 가정하는 것 (ex. P(X|S1), P(X|S2) 알고 있음 -> P(S1|X) 구함
Nonparametric : unknown의 분포 모름 -> pdf 먼저 estimation

Parametric 하다는 것 : 빈 부분의 data 역시 나머지 부분의 분포와 동일하다고 가정해서!

Related Work

- Unsupervised Learning

: Semantically informative & Generalizable feature from raw images without any labels – open question
-> autoencoder 가 먼저 연구 되기 시작
-> denoising autoencoder : local corruption 에서 image reconstruct (corruption으로 부터 encoding robust하게)
-> context encoder : denoising autoencoder의 일종으로 볼 수 있음 but corruption의 범위가 훨씬 큼

- Weakly-supervised and self-supervised learning
  - Useful source of supervision : use the temporal information contained in video
  - Use Consistency : track patches

: temporal frame 사이의 consistency -> embedding 학습에서 label로 사용

- Supervisory signal로 사용 가능한 매우 많은 spatial context 사용 어디에 빈칸이 생기든 나머지 갖고 사용!
- Visual Memex : nonparametrical 하게 object relation modeling & scene에서 masked object predict 하는데 context 사용 -> unsupervised object discovery 에 관련성 제공하기 위해 context 사용

=> Hand designed feature 사용 & representation learning perform X

Context encoders for image generation
Encoder-Decoder Pipeline

- Simple encoder-decoder pipeline 사용
  - Encoder : missing region있는 img -> 이 img에 대한 latent feature 생성
  - Decoder : 이 feature로 missing img content 생성

-> encoder 와 decoder 를 channel wise-fully connected layer 생성

⇒ Decoder의 각 unit에서 전체 img content에 대해 추론할 수 있음

- Encoder : AlexNet 사용 -> 무작위로 초기화된 weights 를 사용해서 원래의 context 예측!

⇒ Encoder 가 CNN에만 제한되어 있다면 : feature map의 one corner에서 다른 corner로 propagate할 정보 X
(∵ convolutional layer의 모든 feature map은 서로 연결 but 모든 location이 특정한 feature map과 연결 X)

⇒ Information propagation을 이 논문에선 fully-connected or inner product layer로 해결! (모든 activation이 서로 연결되어 있음) Information propagation : encoder → decoder로 latent feature 넘겨줌

- Encoder/decoder latent feature : 6X6X256

⇒ Original input reconstruct 필요 X & bottleneck 줄일 필요 X

⇒ But fully connecting the encoder and decoder : parameter 많음 (100M 넘음)

⇒ Channel-wise fully connected layer 로 해결! (connect the encoder feature to decoder 하기 위해!)

Latent feature만 넣으면 안 중요한/중요한 모두 같은 비중으로 들어감
but weight 곱해서 넣어주면 중요한 것만 강조!

Context encoders for image generation
Encoder-Decoder Pipeline

- Channel-wise fully-connected layer

: group 간에 essentially fully-connected layer (각 feature map의 activation으로 부터 information을 전달하기 위해)

  - Input : nXn size의 m개의 feature map -> output : nXn size의 m개의 feature map

  ⇒ Fully connected layer와 달리, 다른 feature map과 연결 시 parameter 필요 없음

  ⇒ 그저 feature map의 information 만 전달!

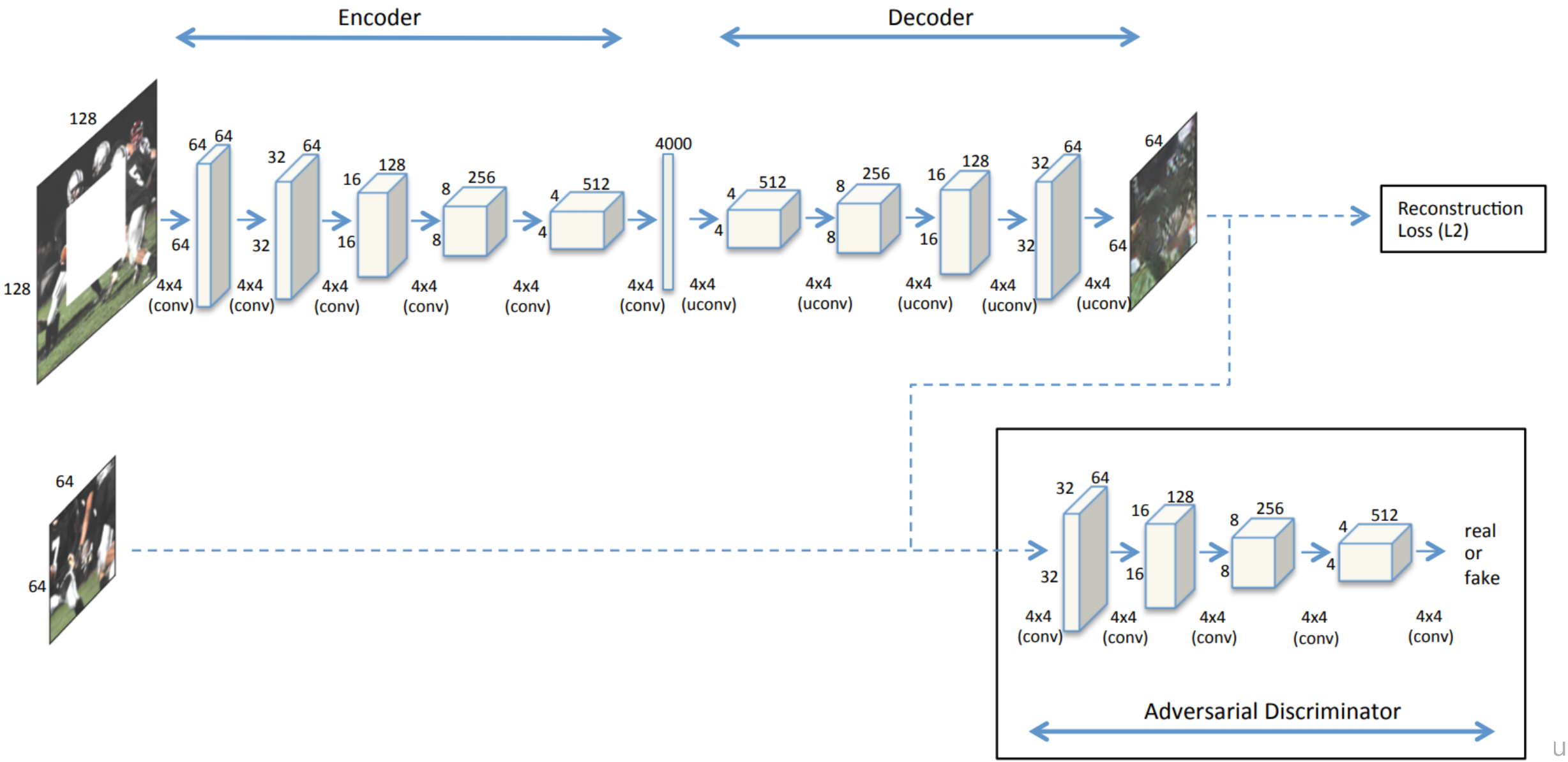
  - # parameter : mn^4 (fully connected layer : m^2n^4)
  - Followed by 1X1 conv to propagate information across channel

- Decoder : generate pixels of the image using the encoder features
  - encoder feature -> decoder feature by channel-wise fully connected layer
  - 5 up-convolutional layer (up conv -> relu)

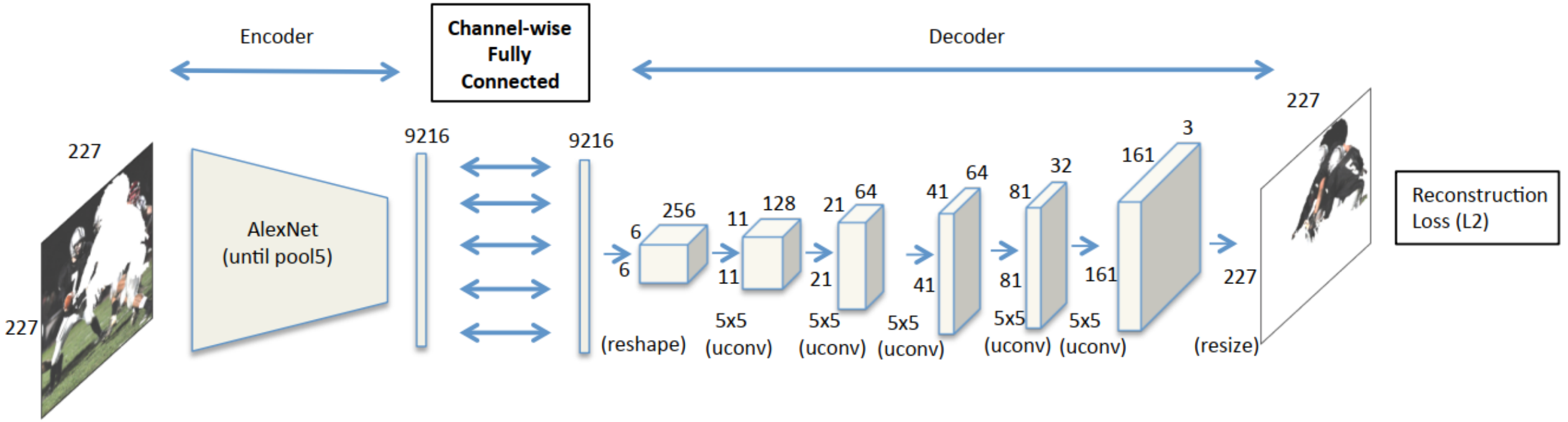
m^2n^4 : 1) n^2 : nxn 연산 -> 1x1xm의 fully connected로 만들어줌
2) m^2 : m개의 fully connected와 그 다음 m개의 fully connected 사이의 연산
3) n^2 : 1x1xm의 fully connected를 다시 nxnxm으로 만들어줌

mn^4 : 1) n^2 : nxn 연산 -> 1x1xm의 fully connected 로 만들어줌
2) m : fully connected 사이에 일대일로 matmul 계산만 있음
3) n^2 : 1x1xm의 fully connected를 다시 nxnxm으로 만들어줌

Context encoders for image generation
Encoder-Decoder Pipeline



Context encoders for image generation
Encoder-Decoder Pipeline



(b) Context encoder trained with reconstruction loss for feature learning by filling in *arbitrary region dropouts* in the input.

Context encoders for image generation
Encoder-Decoder Pipeline

```
def channel_wise_fc_layer(self, input, name): # bottom: (7x7x512)
    _, width, height, n_feat_map = input.get_shape().as_list()
    input_reshape = tf.reshape( input, [-1, width*height, n_feat_map] )
    input_transpose = tf.transpose( input_reshape, [2,0,1] )

    with tf.variable_scope(name):
        W = tf.get_variable(
                "W",
                shape=[n_feat_map,width*height, width*height], # (512,49,49)
                initializer=tf.random_normal_initializer(0., 0.005))
        output = tf.batch_matmul(input_transpose, W)

    output_transpose = tf.transpose(output, [1,2,0])
    output_reshape = tf.reshape( output_transpose, [-1, height, width, n_feat_map] )

    return output_reshape
```

1) reshape → transpose : fully connected 라고 생각

2) batch_matmul : 일대일로 넘겨주게 되는데 이때, weigh만 곱해서 넘겨줌

3) 다시 fully connected → conv 모양으로 만들어줌

Context encoders for image generation
Loss function

- Loss function
  - Missing region content 찾는 것이 training 목표
  - Context 일관성 사용해서 missing image region 하는 방법은 다양함

⇒ Decouple joint loss function 으로 modeling

⇒ Context continuity와 output multiple mode 모두 다룰 수 있음

- Reconstruction loss (L2 loss) : capture the overall structure of missing region & context 관련 coherence 유지 가능 but prediction에서 multiple mode 평균화 하는 경향 있음

- Adversarial loss : make prediction look real & pick a particular mode from the distribution

Context encoders for image generation
Loss function

- Reconstruction loss : masked L2 distance $L_{rec}(x) = \left\| \hat{M} \odot \left(x - F\left((1 - \hat{M}) \odot x \right) \right) \right\|_2$
  - $\odot$: element-wise product operation
  - L1 사용해도 별 차이 없음
- 이 loss만 사용할 경우 : decoder의 predict가 rough한 outline만 생성 → high freq의 detail은 잘 잡지 못함

⇒ L2/L1 loss : blurry solution 가능 (큰 틀 잡는 것만 가능) → 정확한 texture 선호 X

⇒ L2 : 분포의 mean을 예측하는 경향 -> mean pixel-wise error minimize => blurry한 평균 img 결과 나옴

⇒ 이를 adversarial loss 로 완화!

- Adversarial loss : based on GAN

GAN loss : $L_{adv} = \min_G \max_D \mathrm{E}_{x \in \chi}[\log(D(x))] + \mathrm{E}_{z \in Z}[\log\left(1 - D\left(G(z)\right)\right)]$

Maximum when D(x) = 1
진짜 image를 최대한 진짜라고 판별하도록!

Maximum when D(G(x)) = 0
가짜 image를 최대한 가짜라고 판독하도록!

⇒ modeling generator by context encoder

Context encoders for image generation
Loss function

- Modeling generator by context encoder : $G \triangleq F$
- Given context information : the mask $\hat{M} \odot x$

- Conditional GAN 의 경우 : context prediction training 쉽지 않음 (Discriminator가 generated region 과 original context 사이의 discontinuity를 쉽게 구분 가능)

⇒ Conditioning only the generator on context

⇒ Result improve : when the generator was not conditioned on a noise vector

$$\Rightarrow L_{adv} = \max_D E_{x \in X} \left[\log\left(D(x)\right) + \log(1 - D\left(F\left(\left(1 - \hat{M}\right) \odot x\right)\right)) \right]$$

Generator에 대한 loss만 구하고, discriminator의 성능에 대한 loss는 구하지 않음 (너무 쉬워서)

Discriminator가 원래 input을 진짜라고 판독

$(1-M) \odot x$: mask 아닌 부분과 input의 곱

$F((1-M) \odot x)$: 그 부분만 갖고 만들어낸 generated region

-> Discriminator가 이를 가짜라도 판독