

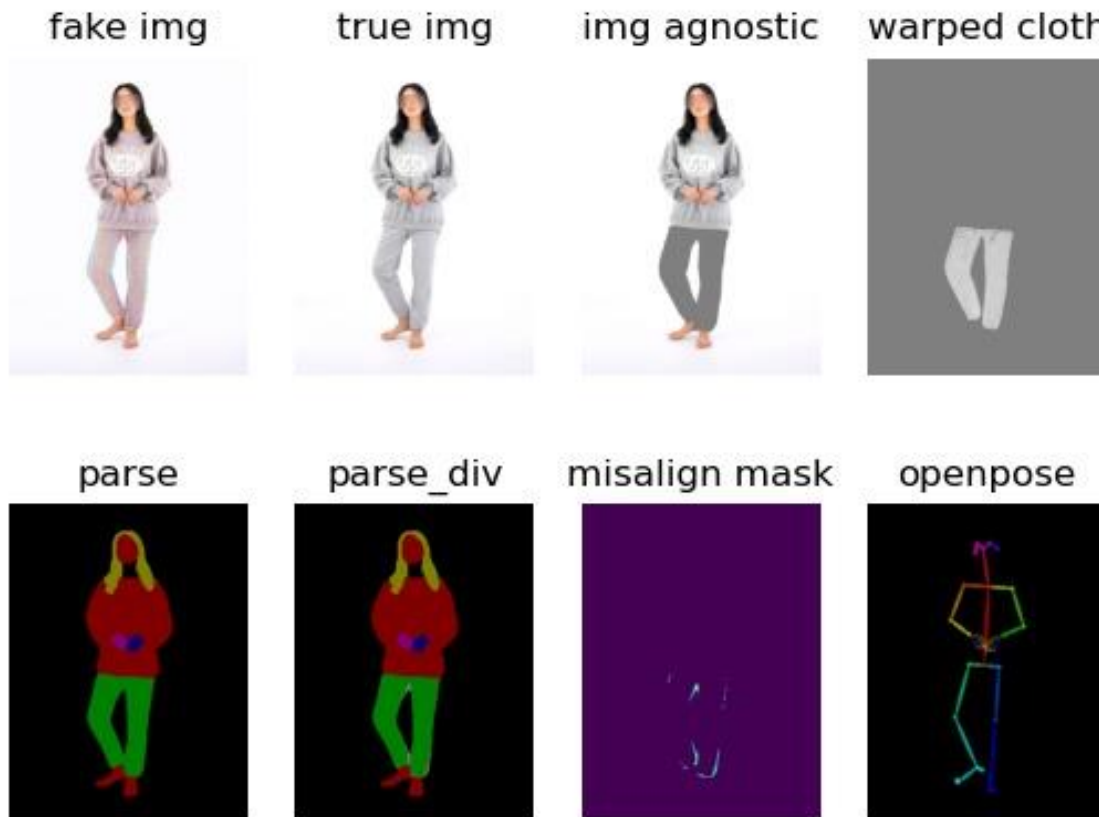
SESAME

Semantic Editing of Scenes

by Adding, Manipulating or Erasing objects

vs ALIAS(image generation)

.GAN_Feat:0.19 D_real:0.49 D_fake:0.51 _timestamp:1655696162.00 _runt



Abstract. Recent advances in image generation gave rise to powerful tools for semantic image editing. However, existing approaches can either operate on a single image or require an abundance of additional information. They are not capable of handling the complete set of editing operations, that is addition, manipulation or removal of semantic concepts. To address these limitations, we propose SESAME, a novel generator-discriminator pair for Semantic Editing of Scenes by Adding, Manipulating or Erasing objects. In our setup, the user provides the semantic labels of the areas to be edited and the generator synthesizes the corresponding pixels. In contrast to previous methods that employ a discriminator that trivially concatenates semantics and image as an input, the SESAME discriminator is composed of two input streams that independently process the image and its semantics, using the latter to manipulate the results of the former. We evaluate our model on a diverse set of datasets and report state-of-the-art performance on two tasks: (a) image manipulation and (b) image generation conditioned on semantic labels.

- 이전 연구에서는 단순히 하나의 이미지만 생성
- Pixel level에서 의미론적 정보를 추가, 조정, 제거하여 이미지를 편집하는 것이 목적
- SESAME discriminator는 image와 semantic를 단순 연결하는 것이 아니라 독립적으로 처리 후 semantic 결과로 image 결과 조정

1 Introduction

Image manipulation has been used in the literature to refer to various tasks. In this paper, we follow the formulation of Bau *et al.* [3], and define the task of semantic image editing as the process of adding, altering, and removing instances of certain classes or semantic concepts in a scene. Examples of such manipulations include but are not limited to: removing a car from a road scene, changing the size of the eyes of a person, adding clouds in the sky, etc. We use the term *semantic concepts* to refer to various class labels that can not be identified as objects, *e.g.* mountains, grass, etc.

Training neural networks for visual editing is not a trivial task. It requires a high level of understanding of the scene, the objects, and their interconnections [45]. Any region of an image added or removed should look realistic and should also fit harmoniously with the rest of the scene. In contrast to image generation, the co-existence of real and fake pixels makes the fake pixels more detectable, as the network cannot take the "easy route" of generating simple textures and shapes or even omit a whole class of objects [4]. Moreover, the lack of natural image datasets, where a scene is captured with and without an object, makes it difficult to train such models in a supervised manner.

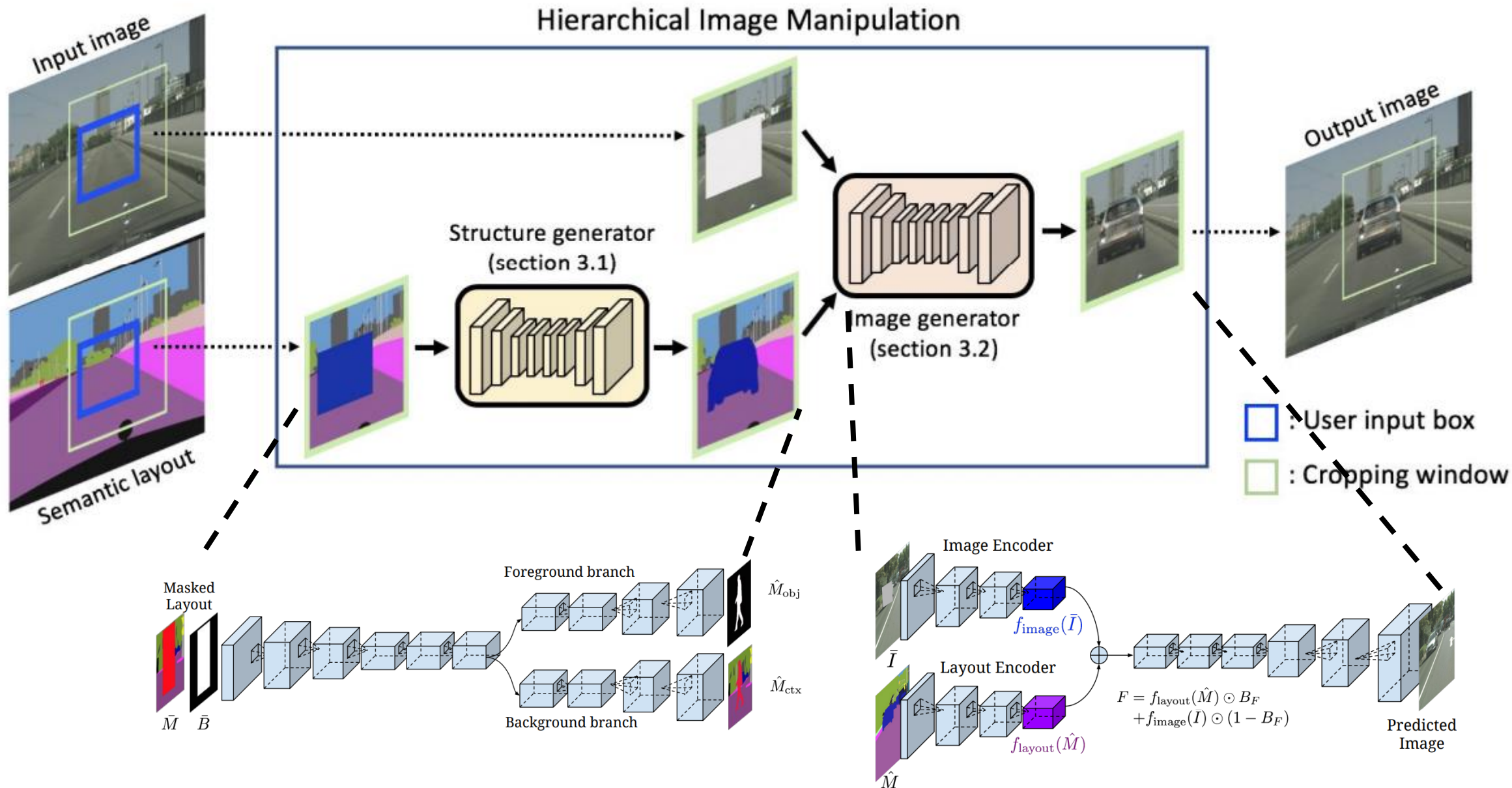
- Image manipulation은 이 이미지에서 특정 class 또는 semantic 인스턴스를 추가, 제거, 변경하는 것을 의미
- image generation과 달리 image manipulation은 real pixel을 유지해야 하면서 대응되는 texture들을 생성해야 하므로 더 어렵다.
- 적당한 데이터셋이 없다

One way to circumvent this problem is by inpainting the regions of an image we seek to edit. Following this scheme, we mask out and remove all the pixels we want to manipulate. Recent works [55,32,37,16] improve upon this approach by incorporating sketch and color inputs to further guide the generation of the missing areas and thus provide higher level control. However, inpainting can only tackle some aspects of semantic editing. To address this limitation, Hong *et al.* [12] manipulate the semantic layout of an image and subsequently, they utilize for inpainting the image. Yet, this approach requires access to the full semantic information of the image, which is costly to acquire.

To this end, we propose SESAME, a novel semantic editing architecture based on adversarial learning, able to manipulate images based on a semantic input. In particular, our method is able to edit images with pixel-level guidance of semantic labels, permitting full control over the output. We propose using the semantics *only* for regions to be edited, which is more cost-efficient and produces better results in certain scenarios. Moreover, we introduce a new approach for semantics-conditioned discrimination, by utilizing two independent streams to process the input image and the corresponding semantics. We use the output of the semantics stream to adjust the output of the image stream. We employ visual results along with quantitative analysis and a human study to validate the performance and flexibility of the proposed approach.

- Edit할 부분을 inpaint => 수정할 부분의 pixel을 mask & 제거
- Inpaint image + manipulate semantic layout
- 수정할 영역의 label만 제공 => 실험적으로 더 좋음

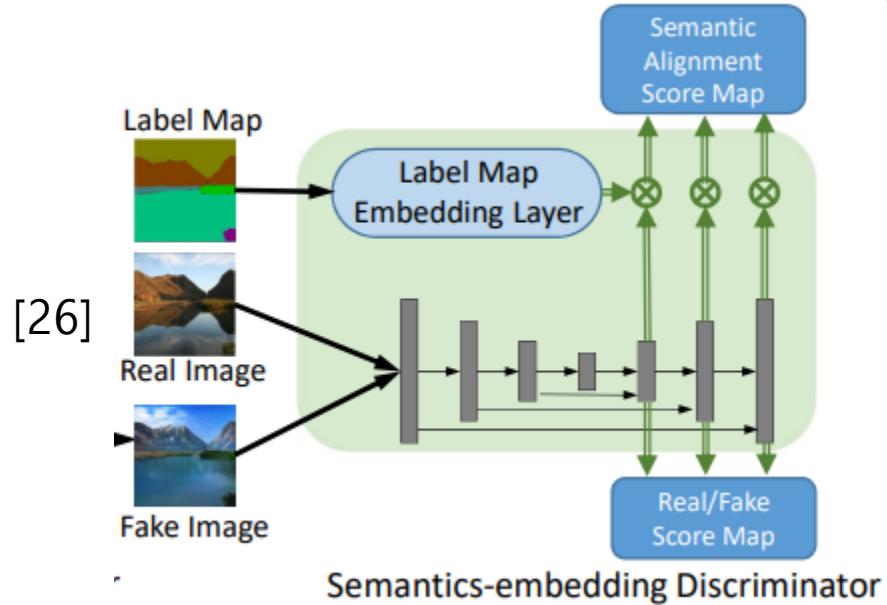
[12]Hong et al.



The core of the majority of the aforementioned works rely on adjusting the generator for the task of image editing. Most recent models use a PatchGAN variant [14] which is able to discriminate on the high frequencies of the image. This is a desired attribute as conventional losses like *Mean Squared Error* and *Mean Absolute Error* can only convey information about the lower frequencies to the generator. PatchGAN can also be used for conditional generation of images on semantic maps, similar to our case study. Previous works targeting a similar problem concatenate the semantic information to the image and use it as an input to the discriminator. However, conventional conditional generation literature suggests that concatenation is not the optimal approach for conditional discrimination [39,33,31].

- 이전 연구에서 true/fake image를 semantic 정보와 concatenate하여 Patch GAN discriminator의 입력으로 사용
- 이는 최선이 아님??(12슬라이드와 연관?)

We believe that a good discriminator should focus on two indispensable and complementary aspects: high-fidelity details such as texture and edges, and semantic alignment with the input semantic map. Existing methods for semantic image synthesis apply a multi-scale PatchGAN discriminator [29, 25], where images concatenated with the semantic label maps are scaled to multiple resolutions and fed into different discriminators with identical structure. But it still struggles to discriminate the fine details, and does not pose strong constraints on the spatial semantic alignment between the generated image and the input label map. [26]



To address this, Liu *et al.* [26] propose a feature pyramid semantics-embedding (FPSE) discriminator using an Encoder-Decoder architecture. Each upsampling layer outputs two per-patch score maps, one trying to measure the *realness* and one to gauge the *semantic matching* with the labels; the later is derived after a patch-wise inner product operation with the down-sampled semantic embeddings. Rather than incorporating a semantics loss, we use semantics to guide the image discrimination. Our model incorporates conditional information by processing it separately from the image input. In a later stage of the network, the two processed streams are merged to produce the final output of the discriminator.

- Patch GAN discriminator는 input label 과 generate image간 spatial semantic alignment를 시키지 못함??
- SPADE block 처럼 semantic 정보를 처리
- 각 upsampling layer(feature 피라미드)에서는 real/fake score와 label과 얼마나 일치하는지

3 SESAME

In this work we describe a deep learning pipeline for semantically editing images, using conditional Generative Adversarial Networks (cGANs). Given an image I_{real} and a semantic guideline of the regions that should be altered by the network, denoted by M_{sem} , we want to produce a realistic output I_{out} . The real pixels values corresponding to M_{sem} are removed from the input image. The generated pixels in their place should be both true to the semantics dictated by the mask and coherent with the rest of the pixels of I_{real} . In order to achieve this, our network is trained end-to-end in an adversarial manner. The generator is an Encoder-Decoder architecture, with dilated convolutions [53] and SPADE [35] layers and the discriminator is a two stream patch discriminator.

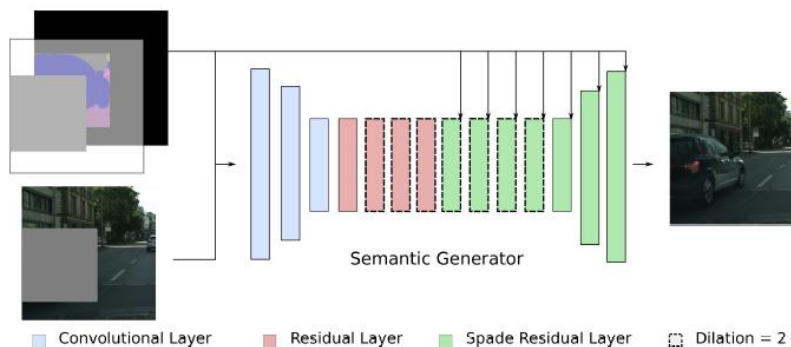


Fig. 2. The SESAME Generator aims to generate the pixels designated by the semantic mask so they are both (1) true to their label and (2) fit naturally to the rest of the picture. It is an encoder-decoder architecture with dilated convolutions to increase the receptive field as well as SPADE layers in the decoder to guide in-class generation

- Edit 영역의 pixel은 semantic information을 따라야 하고 edit 영역이 아닌 부분은 수정 전 이미지와 일치해야 함
- Encoder-Decoder 구조
- Dilated convolution(큰 receptive field), SPADE layer
- Two stream patch discriminator

SESAME Generator. Semantically editing a scene is an Image to Image translation problem. We want to transform an image where we substituted some of the RGB pixels with an one-hot semantics vector. From the generator's output, only the pixels on the masked out regions are retained, while the rest are retrieved from the original image:

$$I_{gen} = G(I_m, M, M_{sem}), \quad (1)$$

$$I_{out} = I_{gen} \cdot M + I_{real} \cdot (1 - M). \quad (2)$$

This architecture has two goals: generated pixels should 1) be coherent with their real neighboring ones as well as 2) be true to the semantic input. To achieve these goals we adapt our generator from the network proposed by Johnson *et al.* [17] to fill the gaps: two downsampling layers, a semantic core made of multiple residual layers and two upsampling ones.

We conceptually divide our architecture into an encoder and a decoder part. The first extracts the contextual information of the pixels we want to synthesize. The second combines the semantic information using Spatially Adaptive De-Normalization [35] blocks to every layer. As the area to be edited can span over a large region, we would like the receptive field of our network to be relatively large. Thus, we use dilated convolutions in the last and first layers of the encoder and the decoder, respectively. A scheme of our SESAME generator can be seen in the Fig. 2, and for further details refer to the supplementary materials.

[17]: Perceptual Losses for Real-Time Style Transfer and Super-Resolution

pix2pixhd generator가 참고했던 논문

- Generator 입력은 RGB color 와 one-hot encoding 된 semantic vector(?)
- Encoder는 contextual 정보 추출, decoder는 SPADE block으로 semantic 정보 결합
- 넓은 receptive field를 위해 dilated conv를 generator 양 끝단에

[B, 3, 1024, 1024]

Conv_img

[B, 64, 1024, 1024]

Up
sample

SPADE_up_1

[B, 128, 512, 512]

Up
sample

SPADE_up_0

[B, 256, 256, 256]

SPADE_4

[B, 256, 256, 256]

SPADE_3

[B, 256, 256, 256]

SPADE_2

[B, 256, 256, 256]

SPADE_1

[B, 256, 256, 256]

SPADE_0

[B, 256, 256, 256]

Res_blk_3

[B, 256, 256, 256]

Res_blk_2

[B, 256, 256, 256]

Res_blk_1

[B, 256, 256, 256]

Res_blk_0

[B, 256, 256, 256]

Down
sample

Conv_down_1

[B, 128, 512, 512]

Down
sample

Conv_down_0

[B, 64, 1024, 1024]

Conv_init

[B, 184, 1024, 1024]

SESAME Discriminator. Layout to image editing can be seen as a sub-task of label to image translation. Inspired by the Pix2Pix [14], more recent approaches [47,35] employ a variation of the PatchGAN discriminator. The Markovian discriminator, as it is also called, was a paradigm shift that made the discriminator focus on the higher frequencies by limiting the attention of the discriminator into local patches, producing a different fake/real prediction value for each of them. The subsequent methods added a multiscale discrimination approach, the Feature Matching-Loss [47] and the use of Spectral Normalization [30] instead of Instance Normalization [35], which stabilized training and further improved the quality of the generated samples. However, the way that the conditional information was provided to the discriminator remained unchanged.

- PatchGAN discriminator
는 local patch에만 집중하
여 detail 향상에 도움을
주었다. + multiscale
- Instance normalization 대
신 spectral normalization
을 사용하면 학습을 안정
화된다고 함

Label to image generation is a sub-task of conditional image generation. In this more general category of methods, the discriminator has evolved from the cGAN's **input concatenation** [29], to **concatenating the class information with a hidden layer** [39], and lastly, **to take the form of the projection discriminator** [31]. In the latter approach, the inner product of the embedding of the conditional information and a feature extracted from the hidden layers of the discriminator are summed with the output of the discriminator to produce the final prediction. Each step of the conditional discriminator evolution improved the results over the naive concatenation at its input [31]. On all aforementioned methods the discriminator produces, nonetheless, a scalar output for the whole image.

- Patch 기반이 아닌 discriminator이므로 scalar 값을 반환
- 우측으로 갈 수록 성능 향상

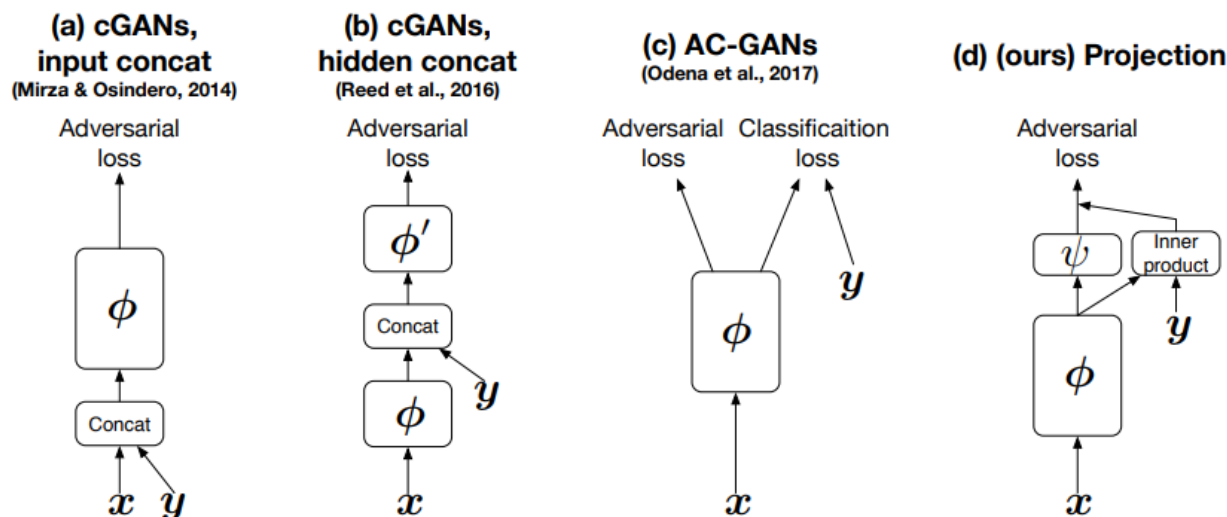



Figure 1: Discriminator models for conditional GANs

[29]

[39]

[31]

We aim to design a discriminator for label to image generation that combines the **aforementioned** attributes. On the one hand, it should preserve the ability of **PatchGAN to discriminate on high-frequencies**. On the other hand, we want to **enforce the semantic information guidance on the discriminator's decision**. If the pixels of the whole image shared semantic class, the projection discriminator would be easily extended to PatchGAN. In contrast, our case is characterized by fine-grained per pixel semantics: each output patch encompasses a variety of classes and different compositions of them. 

- 앞에서 patch 기반이 아닌 discriminator에서 단순히 rgb와 semantic label을 concatenate하지 않는 것이 성능 향상이 있었다는 것을 확인
- 단순히 입력단에서 concatenate하는 Patch GAN discriminator를 수정.

Our proposed SESAME discriminator is comprised by two independent streams that handle the RGB and Semantic Labels inputs. As Fig. 3 depicts, the two streams have identical architectures. Before the information is merged a *Sum Global Pooling* operation is applied to the output of the Semantic Stream. The output of the semantic stream is used to *scale* each output coming from the RGB stream. The resulted feature map is passed as input to a last 3×3 convolutional layer, which produces the final output. The process can be written as follows:

$$D(I, Sem) = Conv_{3 \times 3}(D_{RGB}(I_{out}) \cdot (1 + \sum_{channels} D_{sem}(Sem))), \quad (3)$$

where the D_{RGB} is the output of the RGB stream and D_{sem} of the semantic stream before the *Global Sum Pooling*. We also integrate the changes made to PatchGAN by Pix2PixHD [47] and SPADE [35]. We use a *multiscale discrimination* scheme with squared patches and two different edge-sizes of 70 and 140 pixels, in order to provide also discrimination at a coarser level and *Spectral Normalization*. The input to the semantic stream is the same for both fake and real images discrimination, so we only need to calculate D_{sem} once. Moreover, it makes sense to apply the Feature Matching Loss only to the Feature Maps produced by the RGB stream.

- 독립적인 RGB stream과 Semantic label stream으로 나뉨.
- Semantic feature를 RGB feature에 적용(scale)하기 전에 Sum Global Pooling을 함.
- Fake/true의 semantic label stream(condition) 입력은 동일

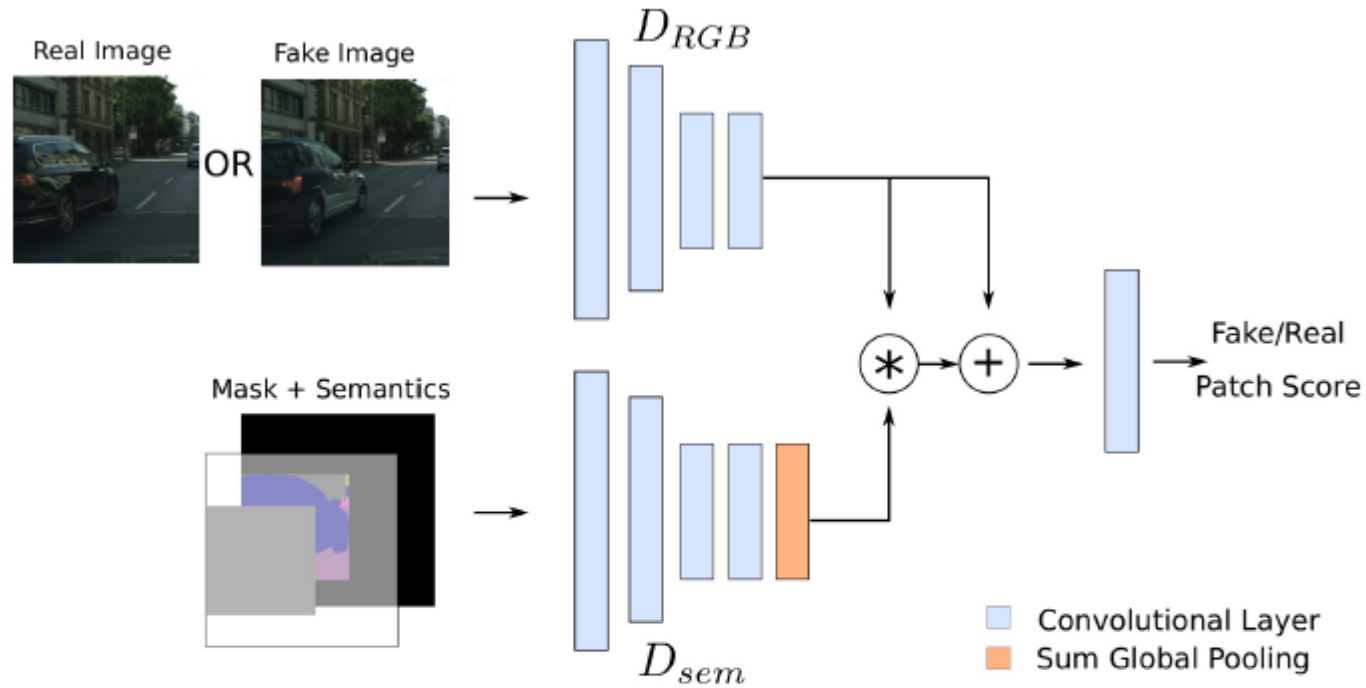


Fig. 3. The SESAME discriminator, in contrast to the commonly used PatchGAN, is handling the RGB Image and its Semantics independently. Before the last convolutional layer the two streams, D_{RGB} and D_{Sem} , are merged. The semantics stream is reduced via a *Sum Global Pooling* operation to a 2D matrix of spatial dimensions equal to the number of output patches. The feature vector of D_{RGB} at each path is scaled by D_{Sem} and a residual is added to product

Edit 하지 않는 부분 즉 fake/true의 공통분모에 해당 하는 patch 기댓값은 receptive field가 모두 edit부분이 아니면 1이어야 하지 않나?

Training Losses. We train the Generator in an adversarial manner using the following losses: Perceptual Loss [17], Feature Matching Loss [41], and Hinge Loss [24,46,30] as the Adversarial Loss. Early experiments with ~~Style Loss [17]~~ ~~did not improve the results.~~ Accordingly:

$$L_G = \lambda_{percept} \cdot L_{perc} + \lambda_{feat} \cdot L_{FM} - E_{z \sim p(z)}[D_k(I_{out}, M, M_{sem})], \quad (4)$$

Where each λ represents the relative importance of each loss component.

For the discriminator at each scale, the Hinge Loss takes the following form:

$$L_{D_k} = E_{z \sim q_{data}(x)}[\min(0, -1 + D_k(I_{real}, M, M_{sem}))] + E_{z \sim p(z)}[\min(1, -1 - D_k(I_{real}, M, M_{sem}))], \quad (5)$$

which is then combined to form the full discrimination loss,

$$L_D = L_{D_1} + L_{D_2}. \quad (6)$$