

Generative Image Inpainting for Person Pose Generation



Vismay Patel and Anubha Pandey

1 Image Inpainting Generator with Skip-Connections

Similar to previous work [1, 2], we also use an encoder-decoder based CNN model with a combination of regular and dilated convolutions to encode the partial image, the decoder uses skip connections from the encoder and combination of deconvolution and convolutions to generate a full image. We use a combination of Reconstruction loss, Perceptual loss, and Adversarial loss to train our model using Adam Optimization Algorithm.

The input to the network is a fixed size image and the mask image concatenated at the color dimension. Both the image and the mask are resized to the size of 128×128 . We use data available in the 'maskdata.json' file to generate binary mask images. The masks contain ones in places of holes and zeros everywhere else (Fig. 1).

1.1 Network Architecture

Our network has the following main modules:

1. **Generator Module:** It is an encoder-decoder based CNN model. In the encoder part initially, we use a series of convolution layers. Each convolution layer is followed by a batch normalization and a ReLU layer. We use dilated convolutions in the later stages of the encoder. Dilated convolutions are also followed by a batch normalization and a ReLU layer. The decoder uses a series of deconvolution followed by convolution layers. We also add skip connections

V. Patel · A. Pandey (✉)
Indian Institute of Technology Madras, Chennai, India

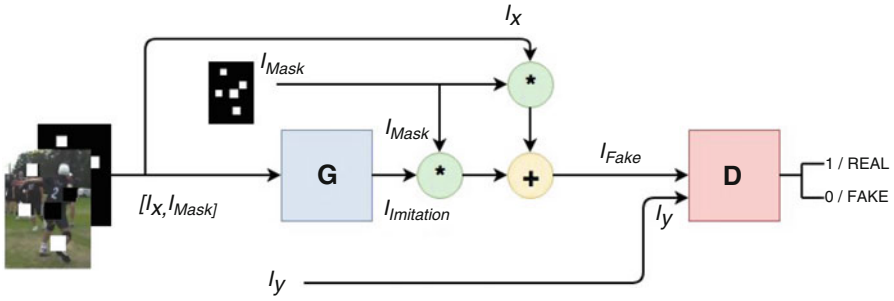


Fig. 1 Full pipeline of the Image Inpainting network

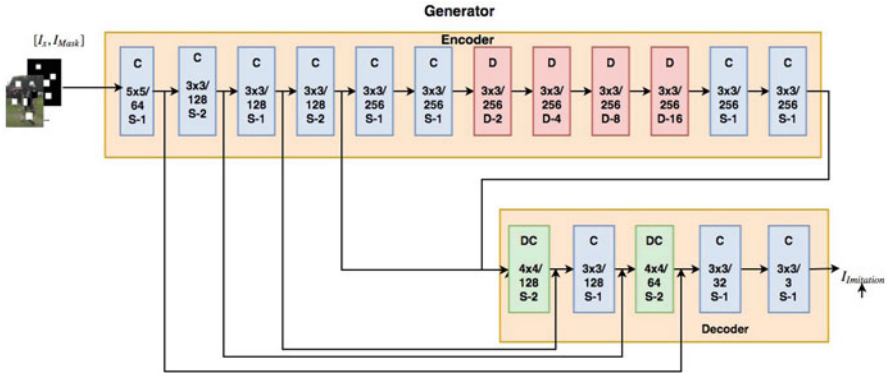


Fig. 2 Architecture of the generator module of the inpainting network. Each building block is described in Fig. 4

from the encoder module to the decoder module. The decoder generates an image where the holes of the input will be filled with some representative patches. The input to the Generator module is $128 * 128 * 4$ sized tensor which is a concatenation of the input image and the mask. The generator architecture is shown in Fig. 2. Our network is inspired by the work of [2].

2. **Discriminator Module:** This module is used while applying GAN [3] loss to the generator. It is responsible to distinguish between Real ground truth images and the fake images generated by the generator. Our discriminator module consists of five convolution layers and a fully connected layer as can be seen from Fig. 3. The discriminator receives two types of inputs. The real images that come from the ground truth data of the training set and the fake images are generated by the generator module. The discriminator outputs the probability of the image being real. Thus the output value should be close to 0 for fake images and it should be close to 1 for real ground truth images.

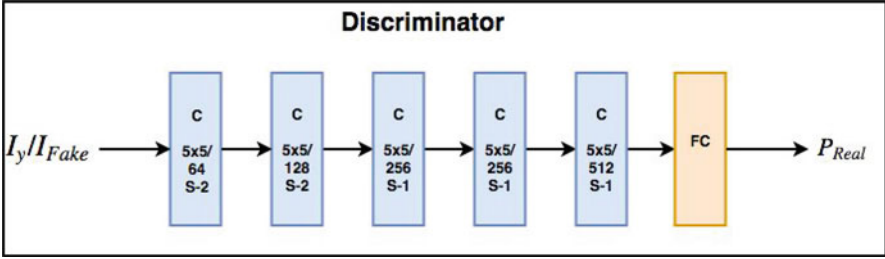
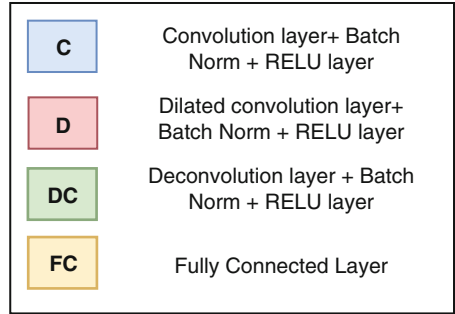


Fig. 3 Architecture of the discriminator module of the inpainting network. Each building block is described in Fig. 4

Fig. 4 Building blocks of the network. Each block in the diagram also contains the filter size/number of output channels and stride (S) or dilation (D) of the respective layer



1.2 Training

We train our network using Adam Optimizer with learning rate 0.001 and batch size 12. For first five epochs we train only the generator module of the network by minimizing reconstruction loss and perceptual loss. For the next 15 epochs, we train the entire GAN network on the adversarial and perceptual loss.

Following loss functions have been used to train the network:

1. **Reconstruction Loss:** To generate images similar to the ground truth image, we try to reduce the absolute error between the generated images and the ground truth images.

$$L_r = \frac{1}{K} \sum_{i=1}^K |I_x^i - I_{imitation}^i| \tag{1}$$

where, K is the batch size and alpha = 0.000001(hyper parameter) and $I_{imitation}^i$ is the output of the decoder.

2. **Adversarial Loss:** In order to maximize the probabilities for real images and minimize the probabilities for fake images, the total discriminator loss L_d is a combination of two partial losses.

$$\begin{aligned}
L_{real} &= -\log(D(I_{groundtruth})) , \quad L_{fake} = -\log(1 - \log(D(G(I_{input})))) \\
L_d &= L_{real} + \beta * L_{fake} \\
L_g &= -L_d
\end{aligned}
\tag{2}$$

where, D is the discriminator function and G is the generator function and β is trade-off parameter between L_{real} and L_{fake} . The best value of β is $1 * 10^{-2}$ which is decided by cross validation.

3. **Perceptual Loss [4]:** The knowledge of contextual information is very crucial for filling the correct missing pixels in any image, it helps to produce a perceptually similar image from the original image. We use the pretrained vgg network as a function to find the semantic difference between ground truth images and the generated images.

$L_p = \frac{1}{K} \sum_{i=1}^K (\phi(I_y) - \phi(I_{imitation}))^2$ where, ϕ represents features taken from various layers of a VGG16 network.

The GAN loss allows us to generate realistic looking images. The skip-connections allows us to pass fine details to the coarse layers in order to generate details in the images.

2 Experiments and Results

We evaluate our model on the dataset provided in the ECCV’18 Satellite Workshop Chalearn LAP Inpainting Competition Track 1-Inpainting of still images of humans. To evaluate the quality of the image reconstruction, metrics MSE, PSNR, DSSIM, WNJD as mentioned on the [competition’s website](#) are used.

The proposed model is implemented using tensorflow-gpu 1.6.0 framework on the top of python 3.6.4 and the platform used is Ubuntu 14.04. The training of the model takes 3 h for one epoch on a GeForce GTX TITAN X graphics card. There are 26,151,776 parameters to be learned in the network. We trained the network on the above-mentioned dataset for 2 days up to 15 epochs on 34,915 images. Testing requires 2.5 h for 6160 images.

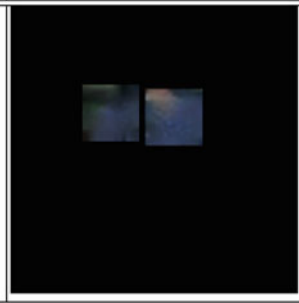
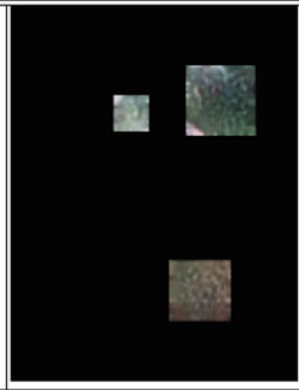
With our proposed solution we secure second position in the competition. Performance comparison of our proposed model with that of other teams are shown in Table 1. The qualitative results of our proposed solution is shown in Table 2.

Table 1 Comparison of performance with other teams

| Team name | Rank | MSE | PSNR | DSSIM | WNJD |
|-----------|------|--------|---------|--------|--------|
| UNLU | 1 | 0.0158 | 21.8712 | 0.2088 | 0.1489 |
| anubhap93 | 2 | 0.0158 | 21.5118 | 0.2048 | 0.1495 |

We are team anubhap93

Table 2 Qualitative results

| Input Image | Ground Truth | Output |
|--|--|---|
|  |  |  |
|  |  |  |
|  |  |  |

3 Conclusions

In this work, we have used a generative CNN for the Image Inpainting task. We have trained our model to generate new patches. Our network can generate patches that do not appear anywhere else in the image. Our network can inpaint images with randomly placed multiple masks of variable sizes. In future work, we aim to improve the resolution of the inpainted patches using multi-stage CNNs. Moreover, techniques to handle the multiple modalities of images and using loss functions related to pose estimation may help us improve the results.

References

1. Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. *arXiv preprint*, 2018.
2. Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa. Globally and locally consistent image completion. *ACM Transactions on Graphics (TOG)*, 36(4):107, 2017.
3. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
4. Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.