ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 4, April 2019

Image Inpainting using Partial Convolution

(Eswari.B[1], Madhuja.S[2], Sakthi Harini.D[3], Agnes Joshy.S[4])
[1](Dept of IT, UG Scholar, Francis Xavier Engineering College, eshbala06@gmail.com)
[2](Dept of IT, UG Scholar, Francis Xavier Engineering College, madhujasunderaja@gmail.com)
[3](Dept of IT, UG Scholar, Francis Xavier Engineering College, sakthiharinicute@gmail.com)
[4](Dept of IT, Assistant Professor, Francis Xavier Engineering College, sagnesjoshy@gmail.com)

Abstract: Existing deep learning based image inpainting methods use a standard convolutional network over the corrupted image, using convolutional filter responses conditioned on both valid pixels as well as the substitute values in the masked holes (typically the mean value). This often leads to artefacts such as colour discrepancy and blurriness. Postprocessing is usually used to reduce such artifacts but are expensive and may fail. We propose the use of partial convolutions, where the convolution is masked and renormalized to be conditioned on only valid pixels. We further include a mechanism to automatically generate an updated mask for the next layer as part of the forward pass. Our model outperforms other methods for irregular masks. We show qualitative and quantitative comparisons with other methods to validate our approach.

I. INTRODUCTION

Python is an interpreted, high level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.VanRossum led the language community until stepping down as leader in July 2018. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, it also has a comprehensive standard library. Python interpreters are available for many operating systems.

Image Inpainting

This module is concerned with the inpainting of the image. It receives the image from the user where the region to be inpainted is marked in green (R = 0, G = 255, B = 0). As mentioned earlier, we have chosen green color because of its use in the creation of special effects in movies etc.

Let us first describe the terms used in inpainting literature.

a) The image to be inpainted is represented as I.

b) The target region (i.e., the region to be inpainted) is represented as Ω.

c) The source region (i.e., the region from the image which is not to be inpainted and from where the information can be extracted to reconstruct the target region) is represented as Φ.

$\Phi = I - \Omega$

d) The boundary of the target region (i.e., the pixels that separate the target region from the source region) is represented as δΩ.

We have followed the algorithm developed by all with a few modifications. As with all other exemplar based algorithms, this algorithm replaces the target region patch by patch. This patch is generally called the template window, ψ. The size of ψ must be defined for the algorithm. This size is generally kept to be larger than the largest texture element in the source region. We have kept the default patch size of 9 x 9 but we may have to vary it for some images. Once these parameters are assigned the remaining process is completely automatic. The input to the inpainting module is the image with target region marked in green color. The first step is to initialize confidence values. First let us understand what these values represent. In this algorithm, each pixel maintains a confidence value that represents our confidence in selecting that pixel. This confidence value does not change once the pixel has been filled. We initialize the confidence value for all the pixels in the source region (Φ) to be 1 and the confidence values for the pixels in target region (Ω) to be 0. Once we have the target region, we find the boundary of the target region. For this, we first construct a

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 4, April 2019

Boolean matrix where we put 1's corresponding to pixels which are in the target region (Ω) and zero at other places. Let us call this matrix as fillRegion as it denotes the region.We can then find the boundary of the target region by convolving the matrix with a Laplacian filter. We use the followingLaplacian filter for this task. The next step is to compute the priorities for the patches centered on the pixels in δΩ. The result of the inpainting algorithm depends on the order in which the target region is filled. Earlier approaches used the "onion peel" method where the target region is synthesized from outside inward in concentric layers. In however a different method for estimating the filling order is defined which takes into account the structural features of the image. They fill the target region in a best-first-filling order that depends entirely on the priority values assigned to the patches on the fill front (δΩ). They have developed the priority term such that it is biased towards the patches that contain the continuation of edges and are surrounded by high confidence pixels.

Existing system

Deep Learning Method is used. Based on Image Inpainting a standard convolution network is used over the corrupted images,But this leads to color discrepancy and blurriness. To reduce such artifactsPostprocessing is used, but it is expensive and it may fail. Deep learning approaches have focused on rectangular regions. And they fill and detect rectangular region. Located around the center of the image, and often rely on expensive post-processing. It will not detect the irregular holes.

Proposed system

The Partial Convolution method is used. Convolution is masked and renormalized on only valid pixels. We further include a mechanism of partial convolution to automatically generate an updated mask with any form of pixels. By using partial convolution layer, any mask will eventually be all ones, if the input contained any vialed pixels .It detect all the irregular holes. Image inpainting that operates robustly on irregular hole patterns. And produces semantically meaningful predictions that incorporate smoothly with the rest of the image without the need for any additional post-processing or blending operation.

Python Technology

Python is an interpreter, high-level, object oriented programming language. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. Python is a multi-paradigm programming language. They are fully supported, and many of its features support functional programming and (including by metaprogrammingand metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. It is a logical programming and collector for memory management Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc). Python has a simple syntax similar to the English language. Python has syntax that allows developers to write programs with fewer lines than some other programming languages. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick. Python can be treated in a procedural way, an object-orientated way or a functional way. Some Python expressions are similar to languages such as C and Java Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python. They are floor division and integer division. Python also added the ** operator for exponentiation. From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication. In Python, == compares

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 4, April 2019

by value, versus Java, which compares numerics by value [69] and objects by reference. (Value comparisons in Java on objects can be performed with the equals () method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, comparisons may be chained, for example a <= b <= c.Python uses the words and, or, not for its boolean operators rather than the symbolic &&, ||, !used in Java and C.

Extended list comprehensions into a more general expression termed a generator expression. Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression. Conditional expressions in Python are written as x if c else y (different in order of operands from the c ? x : y operator common to many other languages).Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t, thereby effectively "modifying the contents" of t, while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts. Python features sequence unpacking where multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left hand side of the equal sign in an assignment statement.

Architectural diagram

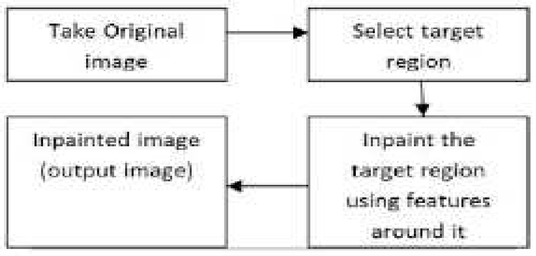

Take Original image
Select target region
Inpaint the target region using features around it
Inpainted image (output image)

Figure 1 Data Flow Diagram

II. IMAGE LOADING

Training Data We use 3 separate image datasets for training and testing. ImageNetdataset ,Places2 dataset and CelebA-HQ . We use the original train, test, and val splits for ImageNet and Places2. For CelebA-HQ, we randomly partition into 27K images for training and 3K images for testing. Training Procedure we initialize the weights using the initialization method described in and use Adam for optimization. We train on a single NVIDIA V100 GPU (16GB) with a batch size of 6.

Initial Training and Fine-Tuning. Holes present a problem for Batch Normalization because the mean and variance will be computed for hole pixels, and so it would make sense to disregard them at masked locations. However, holes are gradually filled with each application and usually completely gone by the decoder stage. In order to use Batch Normalization in the presence of holes, we first turn on Batch Normalization for the initial training using a learning rate of 0.0002. Then, we fine-tune using a learning rate of 0.00005 and freeze the Batch Normalization parameters in the encoder part of the network. We keep Batch Normalization enabled in the decoder. This not only avoids the incorrect mean and variance issues, but also helps us to achieve faster convergence. ImageNetandPlaces2 models train for 10 days, whereas CelebA-HQ trains in 3 days. All fine-tuning is performed in one day.

III. LOAD MASK

Previous works generate holes in their datasets by randomly removing rectangular regions within their image. We consider this insufficient in creating the diverse hole shapes and sizes that we need. As such, we begin by

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 4, April 2019

collecting masks of random streaks and holes of arbitrary shapes. We found the results of occlusion/dis-occlusion mask estimation method between two consecutive frames for videos described in to be a good source of such patterns. We generate 55,116 masks for the training and 24,866 masks for testing. During training, we augment the mask dataset by randomly sampling a mask from 55,116 masks and later perform random dilation, rotation and cropping. All the masks and images for training and testing are with the size of 512×512. We create a test set by starting with the 24,866 raw masks and adding random dilation, rotation and cropping. Many previous methods such as haveImageInpainting for Irregular Holes Using Partial Convolutions.

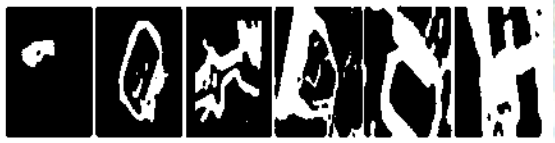

.

Figure 2 Masking

Some test masks for each hole-to-image area ratio category. 1, 3 and 5 are shown using their examples with border constraint; 2, 4 and 6 are shown using their examples without border constraint degraded performance at holes near the image borders. As such, we divide the test set into two: masks with and without holes close to border.

The split that has holes distant from the border ensures a distance of at least 50 pixels from the border. We also further categorize our masks by hole size. Specifically, we generate 6 categories of masks with different hole-to-image area ratios: (0.01, 0.1], (0.1, 0.2], (0.2, 0.3], (0.3, 0.4], (0.4, 0.5], (0.5, 0.6].Each category contains 1000 masks with and without border constraints. In total, we have created 6×2×1000 = 12,000 masks.

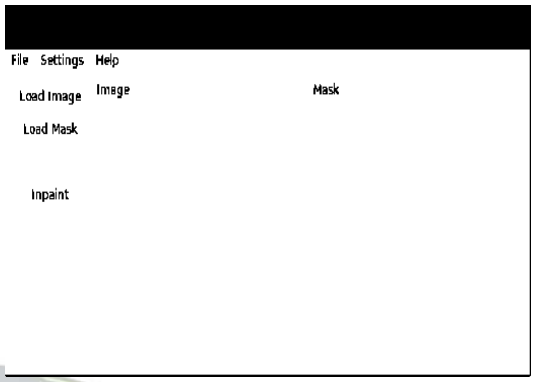
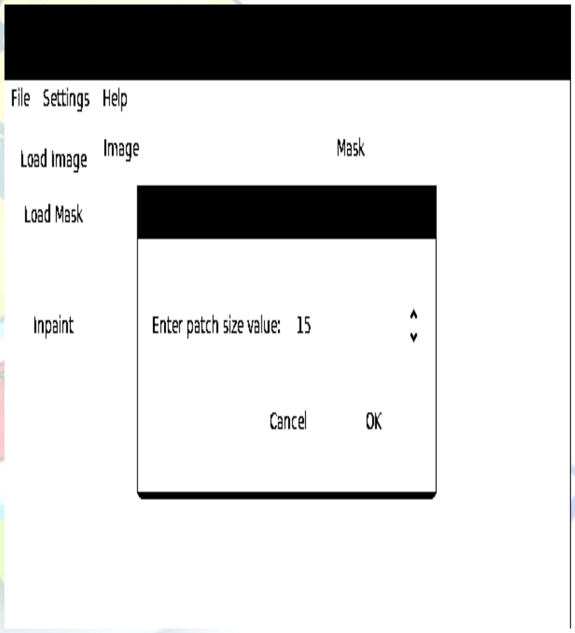


Figure 3 Startup page

This is the page where we load the image by clicking load image button.



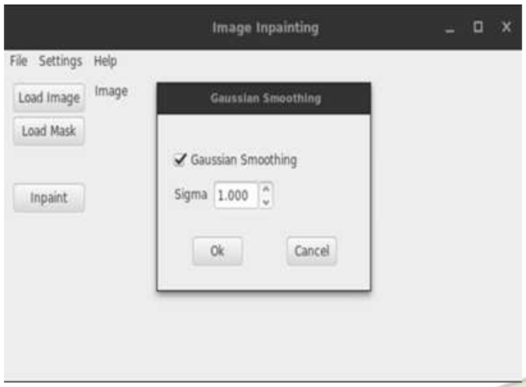
Enter patch size value: 15

Figure 4 Patch Size

In this page the user has to type the patch size of the input image.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 4, April 2019



Gaussian Smoothing: ✓
Sigma: 1.000

Figure 5 Gaussian Smoothing

This is apply for the Gaussian Smoothing which is available in sigma, the use have to type the required size of the sigma.

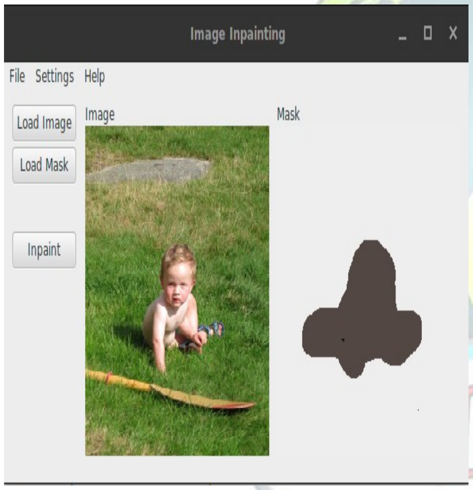


Figure 6 Mask

Now the input image is loaded and the use have to select the portion that the use has to remove and the mask is created.

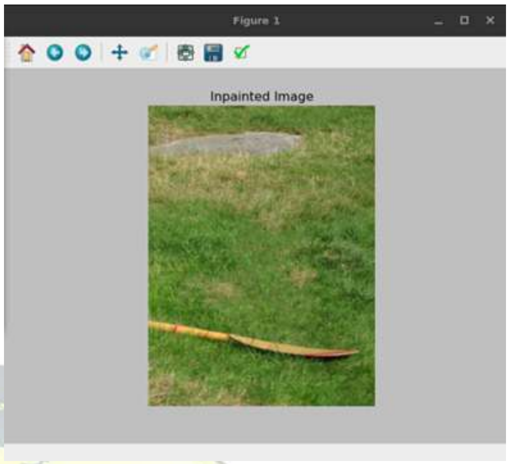


Figure 6 Result

Finally the image is obtained.

IV. Conclusion

We propose the use of a partial convolution layer with an automatic mask updating mechanism and achieve state-of-the-art image inpainting results. Our model can robustly handle holes of any shape, size location, or distance from the image borders. Further, our performance does not deteriorate catastrophically as holes increase in size. However, one limitation of our method is that it fails for some sparsely structured images.

References

[1]. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. CoRR, abs/1703.06211 1(2), 3 (2017)

[2]. Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2017)

[3]. Harley, A.W., Derpanis, K.G., Kokkinos, I.: Segmentation-aware convolutional networks using local attention masks. In: IEEE International Conference on Computer Vision (ICCV). vol. 2, p. 7 (2017)

[4]. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing humanlevel performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision. pp. 1026–1034 (2017)

[5]. Iizuka, S., Simo-Serra, E., Ishikawa, H.: Globally and locally consistent image completion. ACM Transactions on Graphics (TOG) 36(4), 107 (2017)

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 6, Issue 4, April 2019

[6]. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint (2017)

[7]. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. pp. 694–711. Springer (2016)

[8]. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196 (2016)

[9]. Ledig, C., Theis, L., Husza´r, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., et al.: Photo-realistic single image superresolution using a generative adversarial network. arXiv preprint (2016)

[10]. Li, Y., Liu, S., Yang, J., Yang, M.H.: Generative face completion. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). vol. 1, p. 3 (2016)