Seamless Image Cloning with Semantic Segmentation

Yuxiang Gao, Tianshi Liao, Xiaoxiao Liu, Jiteng Mu

***Abstract*—Cropping out your profile in a portrait photo and blending it into a new background or erase something unwanted off your photo are tedious work when they are done manually. In this paper, we propose a two-step solution to this kind of scenario which automatically extracts objects from a source image and seamlessly blend them into the given background. Using a fully convolutional network, we could automatically extract certain objects from the source image. The performance of the neural network with different configurations are tested and evaluated with multiple criteria and several possible improvements were discussed. For the subsequent step, we employ Poisson image editing as the image blending technique. The whole system is designed to be modular and automatic, which makes it expandable and user-friendly.**

***Keywords*—semantic segmentation, fully convolutional networks, Poisson Image Editing.**

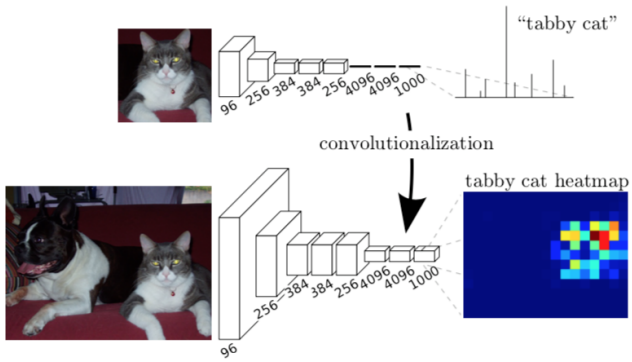


Fig. 1: Relationship between a fully connected network and a fully convolutional network[1].

I. Introduction

The motivation of this project is to implement a functionality to synthesize a new image by seamlessly blending the objects extracted from a source image into the new background image. It consists of two major components, namely, image semantic segmentation for extracting objects from source image and Poisson Image Blending for naturally and seamlessly blending the cropped objects into background without significantly noticeable seams. Such functionality is useful in various scenarios. For example, businesses like souvenir shops in amusement parks will greatly benefit from this product since it greatly simplifies the 'background-changing' task when creating souvenir photos for customers, which is a common marketing strategy but usually done manually. Other scenarios include mobile applications where source objects like faces, people, cars need to be transferred to another background naturally, and so on.

The proposed system can be roughly divided into two parts, the fully convolutional network that automatically generate the mask of the desired object and the Poisson Image Editing to blend object in seamlessly. In the following chapters, we would elaborate on the design of each of these parts.

II. Fully Convolutional Networks

In this paper, we implemented fully convolutional neural network (FCN) with skip architecture for object segmentation which enables us to extract objects from source image.

Fully convolutional networks[1] are designed to make use of the hierarchies of features generated in typical convolutional neural networks, which are widely used in recognition tasks. The output of each layer of convnets is a three-dimensional tensor of size $c \times h \times w$, where c stands for the number of channels and h and w denotes the spatial size.

In each layer the data are down-sampled and reserves only the global information indicating 'what' , and the local information resolving 'where' is discarded in the fully connected layers in recognition networks such as AlexNet[3] and VGG net[4].

By transforming the fully connected layers in these networks into convolutional layers as shown in Fig. 1, we can preserve the spatial information of the recognition results, and output a classification map. This classification map is reduced in size comparing to the original image due to the subsampling process after each layer of the network for keeping the computational requirements acceptable.

To retrieve the information lost in each step of subsampling, [1] proposed a method of up-sampling to combine coarse, higher layer information with the fine, lower layer information. To achieve better performance, the skip architecture was used to combine the final prediction layer output with more lower layers.

Following [1], we use a pre-trained vgg16 network whose fully connected layers are substituted in conv layers as shown in Fig. 2, where finer strides where used and the outputs from last FCN (i.e. FCN32) is combined with earlier pooling layer (i.e. FCN16). In this fashion, FCN 8, which combines the result of FCN16 and pooling result, is also used for segmentation tasks and yield decent result.

The improvement after using the skip architecture is significant, which is stated in [1] as well as proved by our later experiments.

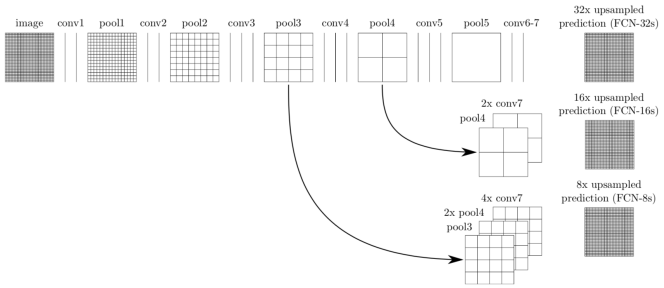


Fig. 2: Architecture of a fully convolutional network[1].

III. Poisson Image Editing

The subsequent task of blending extracted objects by segmentation with new given background is performed by Poisson image editing which smoothing the gradient in intensity between objects extracted and background with Laplacian operator. Poisson image editing, which is first purposed in 2003 by Prez et al.[2] is implemented in our work. In Poisson image editing, gradient in intensity is smoothed out by Laplacian operator, after which seams with barely noticeable effect is achieved, as shown in Fig. 8, where object extracted by segmentation is nicely blend into background image.

Poisson Image editing consists of 2 steps: we first perform convolution to get the gradient of image. After gradient is acquired, we solve the Poisson equation to smooth the transition between extracted objects and background. The Poisson Equation is given as:

$$\Delta\phi = f$$

where Δ is the Laplacian Operator, hence Poisson Equation is:

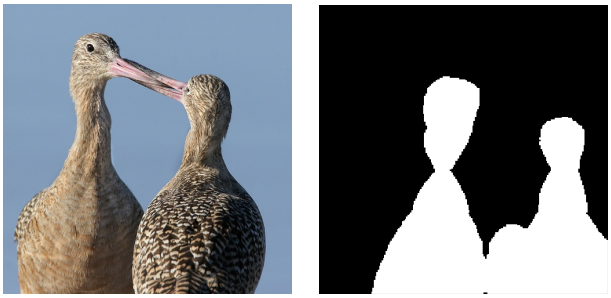
$$\Delta^2\phi = f$$

where Δ^2 is the second degree gradient operator, and f is given as the intensity of the image and we can use Poisson Equation to solve the gradient field of object image (marked as f) with regard to background image (marked as f^*), as shown below:

After the 2 steps of Poisson image editing, we barely notice the seam or any abrupt transition between extracted objects and background image, as shown in Fig. 8.

IV. Experimental Results

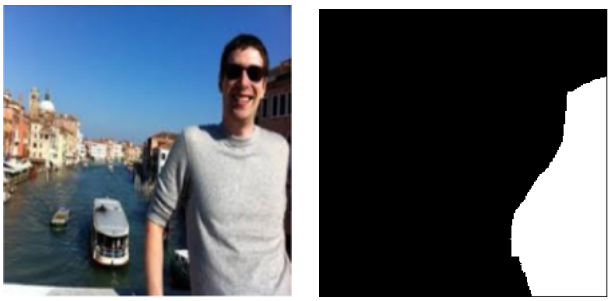
Based on available GPUs and training complexity, we choose PASCAL VOC 2012 as our dataset, which has 21 classes including the black background. The dataset contains 1539 training images and 1635 testing images for segmentation and it provides a train text file and evaluation text file, which are already balanced. We use the pre-trained VGG16 from pytorch and train the network end to end. The pre-trained network can not only make the training much more efficient but also make the training easier. As the pre-trained VGG16 takes images of sizes 224 by 224, so we resize all input images to this size in order to fully take advantage of the pre-trained network. We report the training accuracy with four common methods: Pixel Accuracy, Mean Accuracy, Mean IU and f.w.IU. Here we mainly present three different



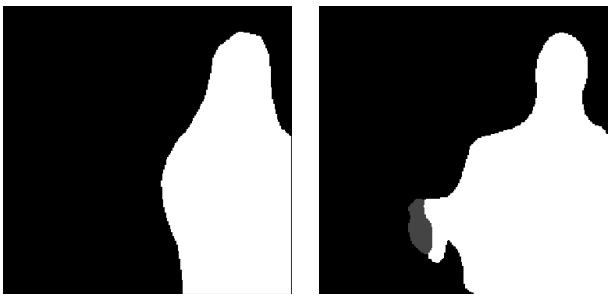
(a) Original image (b) FCN32 segmentation result

(c) FCN16 segmentation result (d) FCN8 segmentation result

Fig. 3: FCN segmentation results compare.

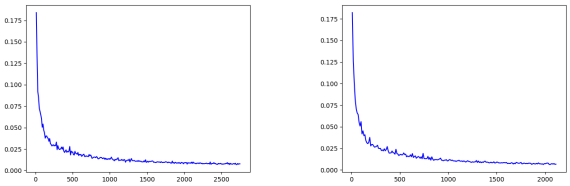


(a) Original image (b) FCN32 segmentation result

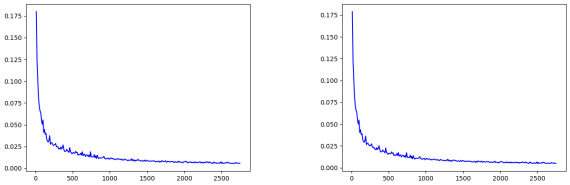


(c) FCN16 segmentation result (d) FCN8 segmentation result

Fig. 4: FCN segmentation results compare.



(a) FCN32 loss (b) FCN16 loss



(c) FCN8 loss (d) FCN4 loss

Fig. 5: FCN training loss compare.

training results: Training in sequential way and training in separate way; the performance of FCN4 and the performance of different spatial sizes have an influence on the performance of the network is shown in Fig. 3.

A. *Our results vs. FCN results*

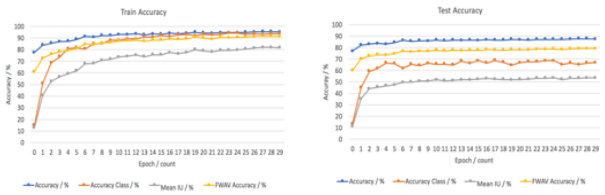
Table. I shows the comparison between our best training results and the results presented by paper[1], which we will just call it paper results for convenience. Our training results are generally worse than the paper results but we are able to show a similar trend for the FCN32, FCN16 and FCN8. The network combined with more low level features is able to capture more details and therefore can achieve a better testing accuracy. A more intuitive feeling of how low the combination of low level features can have an influence on the performance of the network is shown in Fig. 3.

TABLE I: OUR RESULTS VS. PAPER RESULTS[1]

	Network	Pixel Acc.	Mean Acc.	Mean IU	f.w. IU
Our Results	FCN32	87.41	66.90	53.53	79.02
	FCN16	87.87	68.19	54.52	79.74
	FCN8	88.40	70.47	56.16	80.50
Paper Results[1]	FCN32	89.1	73.3	59.4	81.4
	FCN16	90.0	75.7	62.4	83.0
	FCN8	90.3	75.9	62.7	83.2

In the following figures, we just show some of our training results, which can Training loss and training accuracy are correspondingly shown in Fig. 5. and Fig. 6. As is shown, the training loss converge nicely and after 15 epochs, the converge begins to be slow. The accuracy plots we show here are the FCN32 case.

It is shown that the FCN32 can only capture the general profile of two birds but the FCN8 case can capture much more detailed features and hence generate a much refined



(a) Accuracy on train set (b) Accuracy on test set

Fig. 6: FCN32 training accuracy.

classification mask, even for some really thin areas like beaks shown in the figure. A similar trend is also shown in the human case Fig. 4.

B. *Training in sequence vs. Training separately*

Table. II shows the difference between training FCN 32, FCN 16, FCN8 in sequence, which means each network uses previous weights, and training each network separately, which means each network is training based on the pre-trained network VGG16. For the training in sequence condition, FCN32 uses the learning rate 10^{-5} whereas FCN16 and FCN8 use 10^{-7}.

For each network, we trained 30 epochs and the reported accuracy is chosen to be the highest one during the training process. For the separate training case, we train each network purely starting from pre-trained VGG16 and like before, each one is also trained for 30 epochs. In this case, 10^{-5} is used as the learning rate for all three networks.

TABLE II: TRAINING IN SEQUENCE VS. TRAINING SEPARATELY

	Network	Pixel Acc.	Mean Acc.	Mean IU	f.w. IU
Training in Sequence	FCN32	87.41	66.90	53.53	79.02
	FCN16	87.52	68.71	54.44	79.21
	FCN8	87.55	68.78	54.48	79.22
Training separately	FCN32	87.41	66.90	53.53	79.02
	FCN16	87.87	68.19	54.52	79.74
	FCN8	88.40	70.47	56.16	80.50

The result suggests that performance of training in sequence is slightly worse than the training separately case. There could be multiple reasons. The main cause is that the performance of training in sequence depends a lot on the previous training and it is hard to tune training parameters that optimize this whole sequential training process. Therefore, it is easy to overfit the network. In contrast, for the separate training case, as the training is separately, parameters are much easier to tune.

C. *FCN8 vs. FCN4*

It is natural to come up with the idea that if FCN16 is better than FCN32 and FCN8 is better than FCN16, would FCN4 better than FCN8? In the paper[1], it claims that with further combination of low level features, it doesnt help to improve the overall performance a lot. However, with our experiments,

we show that the FCN4 is still slightly better. We trained both of these networks with exactly same parameters and 30 training epochs and the results are shown in the Table. III. So, it is possible that we werent able to find the best parameters that optimize the performance of FCN8 and perhaps it is better to use different parameters. However, from our results, it suggests that it seems that FCN4 converge faster and can better represent the initial image than FCN8.

TABLE III: FCN8 VS. FCN4

Network	AP	Mean IU	Three	Four
FCN4	88.63	71.95	56.77	80.99
FCN8	88.40	70.47	56.16	80.50

D. Performance of different spatial sizes

The spatial size is essential as it contains all classification information of the image. Lets think about an extreme case first, suppose the spatial size of the input before up-sampling is 1 by 1, which is the same as the structure of the original VGG16 network, it reasonable to say that its almost impossible to restore the pixel-wise classification from that. Therefore, in the paper[1], the spatial size before up-sampling is 8 by 8 and the way to achieve this is by padding 100 for the first convolution layer and use a kernel of 7 for the sixth convolution layer. So intuitively, a larger spatial size has more flexibility and may be able to make better dense predictions, whereas a smaller one could make the training harder as it has fewer parameters. So we tried two different padding at the first convolution layer where all other parameters remain the same and results are shown in the Table. IV. It indicates that the performance of a smaller spatial size is obviously worse and the larger size maintains a similar performance as the original case. Without loss of generosity, the testing network we use here is just FCN32. Thus, with our experiments, we show that using the parameters put forward in [1] is a reasonable choice.

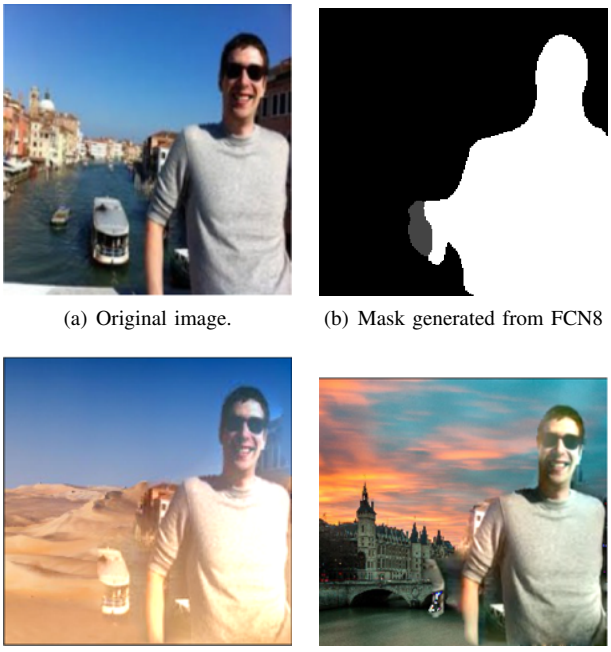
TABLE IV: RESULTS OF DIFFERENT SPATIAL SIZES

Spatial size of last conv layer	Padding of the first conv layer	Pixel Acc.	Mean Acc.	Mean IU	f.w. IU
6×6	80	85.31	63.12	48.68	75.88
8×8	100	87.41	66.90	53.53	79.02
10×10	130	87.28	67.41	53.69	78.92

E. Poisson image blending

Subsequent image editing is required in order to blend extracted objects from origin image into input background image to form a new image as result. Result without implementing said Poisson image blending is shown in Fig. 7, where seams are obvious due to abrupt change in intensity of gradient near the border between objects and background. In order to smooth out such abrupt change in gradient of intensity, Poisson image editing comes into the horizon.

Fig. 7: Result by simply cut-and-paste from the mask.



(a) Original image. (b) Mask generated from FCN8

(c) Result of Poisson Image Editing. (d) Result of Poisson Image Editing.

Fig. 8: Poisson Image Editing results

V. Conclusion

In this paper, we build a seamless image cloning system, which combines FCN and Poisson image editing techniques together and creates blending images. To improve the performance of FCN, besides using different training methods and experimenting with different parameters, we also modify the architecture in different ways, such as the proposal of FCN4. We finally achieve a high segmentation accuracy with respect to the VOC dataset and experiments with images beyond the

dataset are also displayed. Based our experiments, though Poisson image editing is sometimes sensitive to background choices, it still works well in most cases. The overall algorithm proves to be robust and efficient and we believe that it can be further extended for real applications.

REFERENCES

[1] J. Long, E. Shelhamer, T. Darrell. *Fully convolutional networks for semantic segmentation*. Computer Vision and Pattern Recognition, p3431-3440, June 2015.

[2] P. Patrick, M. Gangnet, A. Blake. *Poisson Image Editing*. ACM Transactions on Graphics (TOG). Vol. 22. No. 3. ACM, 2003.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks*. In NIPS, 2012

[4] K. Simonyan and A. Zisserman. *Very deep convolutional networks for large-scale image recognition*. CoRR, abs/1409.1556, 2014