**OBJECT REMOVAL USING KERAS**

**ABSTRACT**

Object removal based on deep learning uses a basic helix network over the depraved image, using helix filter responses conditioned on both valid pixels as well as the simulated values in the masked holes. This leads to artifacts such as colour differences and blurriness. We included a mechanism to automatically generate an updated mask for the next layer as part of a forward pass. Our model outperforms other methods for irregular masked. We show qualitative and quantitative comparisons with other method to validate our approach.

**INTRODUCTION**

Object Removal, the task of filling in holes in an image, can be used in many applications. For example, it can be used in image editing to remove unwanted image content, while filling in the resulting space with plausible imagery.

****

Deep neural networks learn semantic priors and meaningful hidden representations in an end-to-end fashion, which have been used for recent image inpainting efforts. These networks employ convolutional filters on images, replacing the removed content with a fixed value. As a result, these approaches suffer from dependence on the initial hole values, which often manifests itself as lack of texture in the hole regions, obvious colour contrasts, or artificial edge responses surrounding the hole.

Another limitation of many recent approaches is the focus on rectangular shaped holes, often assumed to be Centre in the image. We find these limitations may lead to over fitting to the rectangular holes, and ultimately limit the utility of these models in application. To properly handle irregular masks, we propose the use of a Partial Convolutional Layer, comprising a masked and re-normalized convolution operation followed by a mask-update step. Our use of partial convolutions is such that given a binary mask our convolutional results depend only on the non-hole regions at every layer. Our main extension is the automatic mask update step, which removes any masking where the partial convolution was able to operate on an unmasked value. Given sufficient layers of successive updates, even the largest masked holes will eventually shrink away, leaving only valid responses in the feature map. The partial convolutional layer ultimately makes our model agnostic to placeholder hole values.

**RELATED WORK**

Deep learning based methods typically initialize the holes with some constant placeholder values, which are then passed through a convolutional network. Due to the resulting artifacts, post-processing is often used to ameliorate the eﬀects of conditioning on the placeholder values. Non-learning approaches to image inpainting rely on propagating appearance information from neighboring pixels to the target region using some mechanisms like distance ﬁeld. With standard convolutional layers, the raw features of noise or wrong hole initialization values in the encoder stage will propagate to the decoder stage. Our work also does not depend on placeholder values in the hole regions, but we also aim to achieve good results in a single feed forward pass and enable the use of skip links to create detailed predictions.

Our work makes extensive use of a masked or reweighted convolution operation, which allows us to condition output only on valid inputs. Harley made use of this approach with a soft attention mask for semantic segmentation. It has also been used for full-image generation to condition the next pixel only on previously synthesized pixels. For object removal, Ren proposed Shepard convolution layer where the same kernel is applied for both feature and mask convolutions. The mask convolution result acts as both the reweighting denominator and updated mask, which does not guarantee the hole to evolve during updating due to the kernel’s possible negative entries. It cannot handle big holes properly either.

**APPROACH**

Our proposed model uses stacked partial convolution operations and mask updating steps to perform object removal. We ﬁrst deﬁne our convolution and mask update mechanism,then discuss model architecture and loss functions.

1. Network Architecture and Implementation:

**Implementation**: Partial convolution layer is implemented by extending existing standard PyTorch, although it can be improved both in time and space using custom layers. The straightforward implementation is to deﬁne binary masks of size C×H×W, the same size with their associated images/features, and then to implement mask updating is implemented using a ﬁxed convolution layer, with the same kernel size as the partial convolution operation, but with weights identically set to 1 and no bias. The entire network inference on a 512×512 image takes 0.029s on a single NVIDIA V100 GPU, regardless of the hole size.

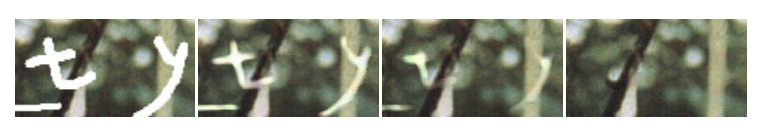
**Network Design**: We design a UNet-like architecture similar to the one used in [11], replacing all convolutional layers with partial convolutional layers and using nearest neighbour up-sampling in the decoding stage. The skip links will concatenate two feature maps and two masks respectively, acting as the feature and mask inputs for the next partial convolution layer. The last partial convolution layer’s input will contain the concatenation of the original input image with hole and original mask, making it possible for the model to copy non-hole pixels. Network details are found in the supplementary ﬁle.

**Partial Convolution as Padding**: We use the partial convolution with appropriate masking at image boundaries in lieu of typical padding. This ensures that the inpainted content at the image border will not be aﬀected by invalid values outside of the image – which can be interpreted as another hole.

**EXPERIMENTS**

**1. Irregular Mask Dataset:**

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 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 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 have 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. Some examples of each category’s masks.



Progressive nature of the algorithm. Several intermediate steps of the reconstruction are shown.

**2. Training Process**:

**Training Data**: We use 3 separate image datasets for training and testing: Image Net dataset, Places2 dataset and CelebA-HQ. We use the original train, test, and Val splits for Image Net 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 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. Image Net and Places2 models train for 10 days, whereas CelebA-HQ trains in 3 days. All ﬁne-tuning is performed in one day.

**3. Comparisons:**

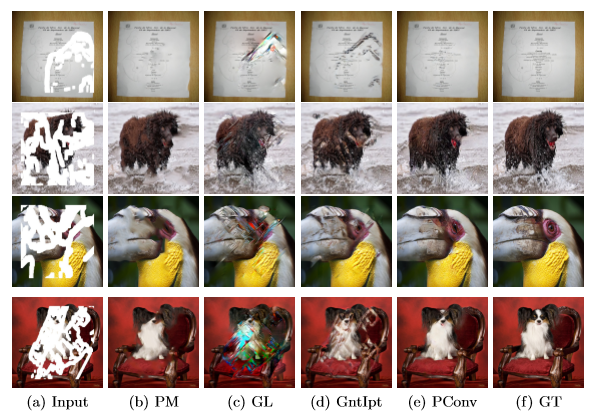
We compare with 4 methods:

– PM: Patch Match, the state-of-the-art non-learning based approach

– GL: Method proposed by Iizuka et al.

– GntIpt: Method proposed by Yu et al.

– Conv: Same network structure as our method but using typical convolutional layers. Loss weights were re-determined via hyperparameter search.

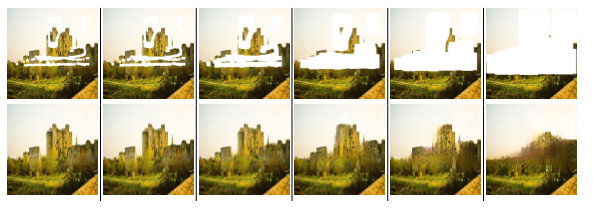


Our method is denoted as **PConv**. A fair comparison with GL and GntIpt would require retraining their models on our data. However, the training of both approaches use local discriminators assuming availability of the local bounding boxes of the holes, which would not make sense for the shape of our masks. As such, we directly use their released pre-trained models1. For Patch Match, we used a third-party implementation2. As we do not know their train-test splits, our own splits will likely diﬀer from theirs. We evaluate on 12,000 images randomly assigning our masks to images without replacement.

**DISCUSION & EXTENION**

**1. Discussion:**

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 such as the bars on the door like most methods, struggles on the largest of holes.



**REFRENCES**

1. Ballester, C., Bertalmio, M., Caselles, V., Sapiro, G., Verdera, J.: Filling-in by joint interpolation of vector ﬁelds and grey levels. IEEE transactions on image processing 10(8), 1200–1211

2. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, and D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Transactions on Graphics-TOG 28(3).

3. Bertalmio, M., Sapiro, G., Caselles, V., Ballester, C.: Image inpainting. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques. pp. 417–424. ACM Press/Addison-Wesley Publishing Co.