OCONet: Image Extrapolation by Object Completion
Supplemental Material

Richard Strong Bowen Huiwen Chang Charles Herrmann
Piotr Teterwak Ce Liu Ramin Zabih

1 Architecture and Training Details

The overall pipeline is discussed in section 3 and figure 2 of the main paper. Here we give additional details about architecture, loss functions, and training setup.

1.1 Single-object completion

The object-completion model is an encoder-decoder with gating and skip connections.

The encoder is a 3×3, 32−filter convolution followed by five residual layers as shown in figure 1a. 2×2 average-pooling is used for downsampling. The layers have 32, 64, 128, 256, and 256 channels, respectively. The final $4 \times 4 \times 256$ image is flattened to a length-4096 vector and then bottlenecked to dimension β with a fully-connected layer. In our experiments, $\beta = 256$. To condition on class, we concatenate another size-β vector by feeding a 1-hot class vector into a single fully-connected layer (equivalently, we learn $n_c\beta$ parameters).

The decoder begins with another fully-connected layer, which undoes the bottleneck from dimension 2β back to 4096, and the resulting length-4096 vector is reshaped back to $4 \times 4 \times 256$. The decoder then passes through 6 residual layers as shown in figure 1b. The number of channels for the six layers of the decoder are (256,256,256,128,64,32). A 1×1 convolution is used to match dimension on the residual step. The decoding layer uses a gated convolution [21]: after the skip input is concatenated on to produce an intermediate x, the gated output is $x\sigma(W(x))$ where σ is the sigmoid activation and W is a 1×1 conv with the same number of channels as x.

The resulting $256 \times 256 \times 32$ texture undergoes a final convolutional layer consisting of: 3×3 conv with 32 filters, relu, 3×3 conv with 4 filters, corresponding to the 3 color channels plus predicted signed-distance field, respectively.

The completion network is trained with 3 loss terms. First is a mask loss: an L1 loss on the SDF output.

$$\mathcal{L}_m = \frac{1}{N_{\text{pix}}} ||M_{\text{pred}} - SDF(M_{GT})||_1,$$

where N_{pix} is the number of pixels, and M_{GT} is the ground truth mask, and SDF is the signed distance function (scaled by $1/256$ so that it is roughly on $[-1, 1]$).

We apply an L2 reconstruction loss:

$$\mathcal{L}_{\text{pixel}} = \frac{(\sum M_{\text{GT}}(x,y)(I_{\text{pred}}(x,y) - I_{\text{GT}}(x,y))^2}{\sum M_{\text{GT}}(x,y)}$$

Finally, we use a mask-modulated LPIPS loss [36]. Using a pretrained VGG16 [37] as a baseline network ϕ, we use $N = 5$ layers:

$$\mathcal{L}_{\text{perceptual}} = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{\sum_{x,y} (M^i_{\text{GT}}(x,y)) \left| \hat{\phi}^i(I_{\text{pred}})(x,y) - \hat{\phi}^i(I_{\text{gt}})(x,y) \right|_2^2}{\sum_{x,y} M^i_{\text{GT}}(x,y)} \right)$$

where $\hat{\phi}^i$ is the activations of the ith layer of the network, normalized to have Euclidean norm 1 along the channel dimension, and M_{GT} is the ground-truth mask downsampled to have the same spatial dimensions as ϕ^i.

The overall loss for the object predictor is

$$\mathcal{L}_{\text{object}} = \lambda_{\text{perceptual}} \mathcal{L}_{\text{perceptual}} + \lambda_{\text{pixel}} \mathcal{L}_{\text{pixel}} + \lambda_{\text{mask}} \mathcal{L}_{\text{mask}}.$$

In our experiments, $\lambda_{\text{perceptual}} = \lambda_{\text{pixel}} = 1$ and $\lambda_{\text{mask}} = 0.1$

1.2 Adversarial fine-tuning and discriminator architecture

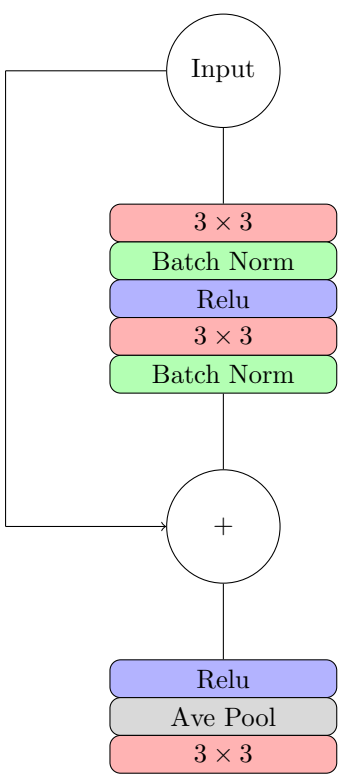
We apply an adversarial loss as a fine-tuning step to improve image quality. We use a patchgan [38] with spectral normalization [39]. The discriminator is a 7×7 convolution with 32 filters, followed by five convolutional layers. These five layers are 3×3 convolutions with leaky relu activation with $\alpha = 0.2$. All but the first have stride 2. There is a final 1×1 convolution with 1 filter, which produces the patch score. There is no normalization applied.

The generator loss comprises GAN loss, feature matching [40] loss and reconstruction loss:

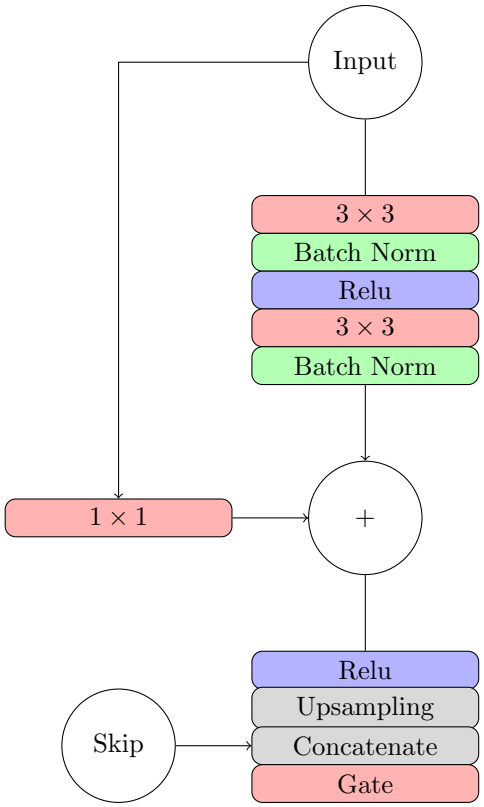
$$\mathcal{L}_{\text{gen}} = \lambda_{\text{adv}} \frac{1}{N_{\text{pix}}} \left(\sum_{(x,y)} -D(x,y) \right) + \mathcal{L}_{\text{object}} + \lambda_{\text{fm}} \mathcal{L}_{\text{fm}}$$

where $\mathcal{L}_{\text{object}}$ is the reconstruction loss described above and $\mathcal{L}_{\text{fm}}$ is a feature-matching loss:

$$\mathcal{L}_{\text{fm}} = \sum_i \frac{1}{N_i} \sum_{x,y} \left(\hat{\phi}_i(I_{\text{real}}) - \hat{\phi}_i(I_{\text{gen}}) \right)^2$$



(a) Object-completion encoder layer. 2×2 average pooling is used for downsampling.



(b) Object-completion decoder layer. Bilinear upsampling is used.

Figure 1: Encoder and decoder layers for object completion

Corruption	FID score
6 pixel erosion	10.35
3 pixel erosion	9.41
None	9.04
3 pixel dilation	8.90
6 pixel dilation	9.14

Table 1: Effect on FID score of misaligning the mask by morphological dilation or erosion. The effect is small, showing the model can handle some misalignment.

with $\hat{\phi}_i$ being features in the ith layer of the discriminator, normalized along the channel dimension. In our experiments $\lambda_{\text{fm}} = 1$ and $\lambda_{\text{adv}} = 3/4$

The discriminator loss is

$$\mathcal{L}_{\text{disc}} = \begin{cases} \frac{1}{N_{\text{pix}}} \sum_{(x,y)} \max(1 - D(x,y), 0) & \text{(real example)} \\ \frac{1}{N_{\text{pix}}} \sum_{(x,y)} \max(1 + D(x,y), 0) & \text{(generated example)} \end{cases}$$

where N_{pix} is the number of pixels at the last layer of the discriminator and D is the discriminator output.

1.3 Implementation and training details

Hyperparameters We train the uncropping networks for 800k train steps with a learning rate of $1e-4$ with gradient clipping (norm 1). The generative fine-tuning uses a learning rate of $1e-5$ for both generator and discriminator. We alternate generator and discriminator training, each taking 3 steps. In total, the fine-tuning runs for 800k cycles (2.4M total steps; 1.2M each discriminator and generator). The background uncroppers are trained using the same hyperparameter settings as in [1] for 2.5M steps. All models are trained using Adam with $\beta_1 = 0.5, \beta 2 = 0.999, \epsilon = 1e^{-8}$.

2 Ground truth and inferred masks

We train on the ground-truth masks, but can use any off-the-shelf object segmenter at inference time. We found that modern instance segmentation typically finds a good mask, even on the cropped input. Occasional mistakes by the segmenter can cause some errors, as shown in figure 2.

3 Impact of mask alignment

To compute the importance of the mask edge aligning with the object edge, we ran our model on the whole dataset – all classes – using the ground-truth masks in OpenImages. This validation set has approximately 177k images. Before

Figure 2: Comparing ground-truth to inferred instance segmentation. The inferred (by an off-the-shelf instance segmentation network) masks are typically very close to the ground-truth masks. The completed objects are also very similar whether the ground-truth or inferred masks are used. The bottom-right example shows an example artifact from an incorrect segmentation

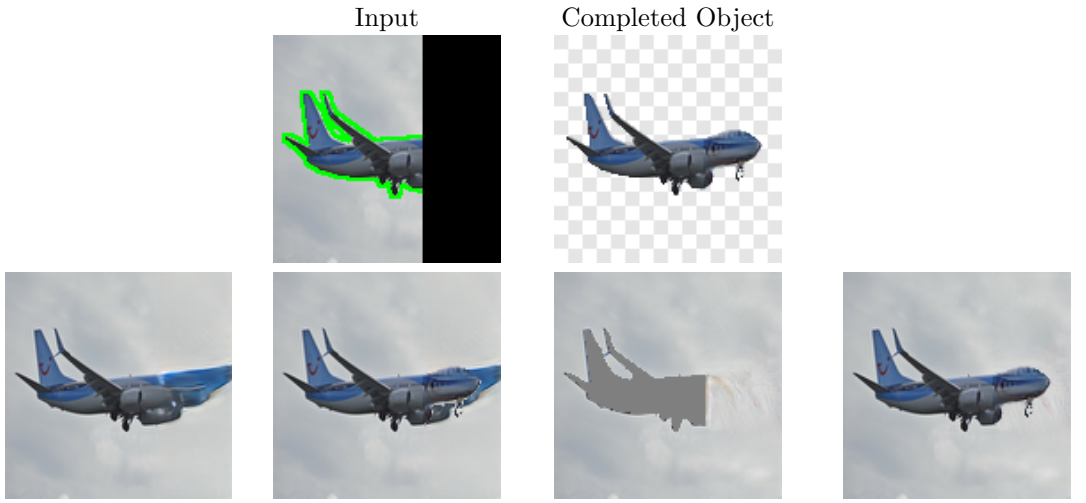


Figure 3: Example demonstrating foreground masking. When using a raw Boundless as a background extrapolator, foreground-object pixels sometimes smear beyond the extent of the completed object's pixels; this is avoided by masking the foreground object.

inference, we corrupt the mask by applying a morphological dilation or erosion of 3 or 6 pixels, then compute the resulting FID score. The results are shown in Table 1; the model is fairly robust to this kind of corruption. Interestingly, the model is more robust to dilation than erosion.

4 Foreground object masking in the background model

In section 3 of the main paper we describe a small change to training a Boundless model: we mask out the object of interest before passing the given region to the generator. Here we show the failure mode this avoids. In figure 3, an example image extrapolation is shown. A Boundless model trained in the standard way smears out the aircraft beyond the extent of the completed object. This in turn causes haloing of texture from the foreground object in the composite. This problem is avoided by not letting the background model see the foreground-object texture in the first place.

5 Dataset statistics

After the filtering described in section 3.2 of the original, the number of examples is about 1.7M. Train and validation split follows OpenImages [18]; the number

of validation examples is about 60k. In validation experiments, each example is repeated three times at different "uncrop ratios" – the fraction of the horizontal extent of the instance that is present in the problem input. Additionally, we must ensure that the method is robust to realistic input masks, such as those that could be generated by an automatic method. So, we generated a subset of our validation set whose masks are produced by an off-the-shelf segmenter. For a number of classes (those in table 1 of the main paper), we begin with the examples where we have a ground-truth annotation of that class. We run the segmenter and keep only the largest mask (if there is one) it produces among those of the class of interest, as long as confidence score is at least 0.5, and the mask touches the right hand side of the image (within 4 pixels). This gives us a subset of the dataset with automatically-generated masks from a number of classes; the sizes of those classes can be seen in table 1 in the main paper.

6 Comparison details

We compare against three main techniques for image extrapolation: Boundless [1], Wide-Context Image Extrapolation[3], and Self-Supervised Scene Deocclusion(SSSD) [2]. There are a few challenges in producing a fair comparison.

The Wide-Context architecture assumes a fixed uncrop ratio, which is not compatible with our problem setup. Therefore we chose a fixed uncrop shape – half the image. In comparisons with Wide Context, their network is always given half of the ground-truth as input, irrespective of the displayed "input" column. A second challenge with Wide Context is that, although we were able to train their network successfully on Celeb-A-HQ, we did not find that it converged on our dataset after 40000 training iteration with their reconstruction loss; therefore, the results appear blurry, even though trained with another 40000 iterations of the finetune stage with the adversarial loss. For this reason, we did not include them in our numerical comparisons.

SSSD solves the deocclusion problem, which is similar but not identical to the extrapolation problem considered in this paper. We used their model pretrained on COCOA datasets and process the images so that the objects are cropped by an adaptive square and resized to 256x256 as inputs. We also tried the model trained on the same dataset for complete the background. But the model was trained to inpaint the occluded regions instead of extrapolating the background, so unsurprisingly the network does not perform well here. Hence we do our best to produce a fairer comparison by using a Boundless model to provide the background, compositing SSSD's object on top. We found that, often, SSSD's output masks do not extend very far into the crop region or not at all; for this reason, they often look similar to Boundless output.