Adversarial Scene Editing: Automatic Object Removal from Weak Supervision

Rakshith Shetty **Mario Fritz** **Bernt Schiele**
Max Planck Institute for Informatics
Saarland Informatics Campus
Saarbrücken, Germany
firstname.lastname@mpi-inf.mpg.de

Abstract

While great progress has been made recently in automatic image manipulation, it has been limited to object centric images like faces or structured scene datasets. In this work, we take a step towards general scene-level image editing by developing an automatic interaction-free object removal model. Our model learns to find and remove objects from general scene images using image-level labels and unpaired data in a generative adversarial network (GAN) framework. We achieve this with two key contributions: a two-stage editor architecture consisting of a mask generator and image in-painter that co-operate to remove objects, and a novel GAN based prior for the mask generator that allows us to flexibly incorporate knowledge about object shapes. We experimentally show on two datasets that our method effectively removes a wide variety of objects using weak supervision only.

1 Introduction

Automatic editing of scene-level images to add/remove objects and manipulate attributes of objects like color/shape etc. is a challenging problem with a wide variety of applications. Such an editor can be used for data augmentation [1], test case generation, automatic content filtering and visual privacy filtering [2]. To be scalable, the image manipulation should be free of human interaction and should learn to perform the editing without needing strong supervision. In this work, we investigate such an automatic interaction free image manipulation approach that involves editing an input image to remove target objects, while leaving the rest of the image intact.

The advent of powerful generative models like generative adversarial networks (GAN) has led to significant progress in various image manipulation tasks. Recent works have demonstrated altering facial attributes like hair color, orientation [3], gender [4] and expressions [5] and changing seasons in scenic photographs [6]. An encouraging aspect of these works is that the image manipulation is learnt without ground truth supervision, but with using unpaired data from different attribute classes. While this progress is remarkable, it has been limited to single object centric images like faces or constrained images like street scenes from a single point of view [7]. In this work we move beyond these object-centric images and towards scene-level image editing on general images. We propose an automatic object removal model that takes an input image and a target class and edits the image to remove the target object class. It learns to perform this task with only image-level labels and without ground truth target images, i.e. using only unpaired images containing different object classes.

Our model learns to remove objects primarily by trying to fool object classifiers in a GAN framework. However, simply training a generator to re-synthesize the input image to fool object classifiers leads to degenerate solutions where the generator uses adversarial patterns to fool the classifiers. We address this problem with two key contributions. First we propose a two-stage architecture for our generator, consisting of a mask generator, and an image in-painter which cooperate to achieve

Preprint. Work in progress.

removal. The mask generator learns to fool the object classifier by masking some pixels, while the in-painter learns to make the masked image look realistic. The second part of our solution is a GAN based framework to impose shape priors on the mask generator to encourage it to produce compact and coherent shapes. The flexible framework allows us to incorporate different shape priors, from randomly sampled rectangles to unpaired segmentation masks from a different dataset. Furthermore, we propose a novel locally supervised real/fake classifier to improve the performance of our in-painter for object removal. Our experiments show that our weakly supervised model achieves on par results with a baseline model using a fully supervised Mask-RCNN [8] segmenter in a removal task on the COCO [9] dataset. We also demonstrate the flexibility of our model to go beyond objects, by training it to remove brand logos from images automatically with only image level labels.

2 Related work

Generative adversarial networks. Generative adversarial networks (GAN) [10] are a framework where a generator learns by competing in an adversarial game against a discriminator network. The discriminator learns to distinguish between the real data samples and the "fake" generated samples. The generator is optimized to fool the discriminator into classifying generated samples as real. The generator can be conditioned on additional information to learn conditional generative models [11].

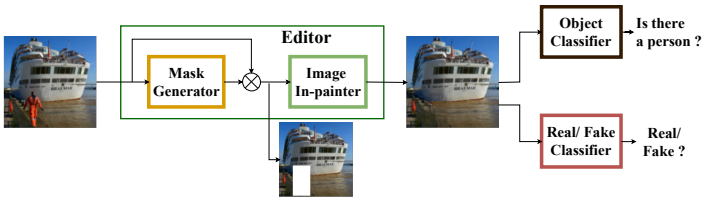
Image manipulation with unpaired data. A conditional GAN based image-to-image translation system was developed in [12] to manipulate images using paired supervision data. Li et al. [6] alleviated the need for paired supervision using cycle constraints and demonstrated translation between two different domains of unpaired images including (horse↔zebras) and (summer↔winter). Similar cyclic reconstruction constraints were extended to multiple domains to achieve facial attributes manipulation without paired data [5]. Nevertheless these image manipulation works have been limited to object centric images like faces [5] or constrained images like street scenes from one point of view [6]. In our work we take a step towards general scene-level manipulation by addressing the problem of object removal from generic scenes. Prior works on scene-level images like the COCO dataset have focused on synthesizing entire images conditioned on text [13–15] and scene-graphs [16]. However generated image quality on scene-level images [16] is still significantly worse than on structured data like faces [17]. In contrast we focus on the manipulation of parts of images rather than full image synthesis and achieve better image quality and control.

Object removal. We propose a two-staged editor with a mask-generator and image in-painter which jointly learn to remove the target object class. Prior works on object removal focus on algorithmic improvements to in-painting while assuming users provide the object mask [18–20]. One could argue that object segmentation masks can be obtained by a stand alone segmenter like Mask-RCNN [8] and just in-paint this masked region to achieve removal. However, this needs expensive mask annotation to supervise the segmentation networks for every category of image entity one wishes to remove for example objects or brand logos.Additionally, as we show in our experiments, even perfect segmentation masks are not sufficient for perfect removal. They tend to trace the object shapes too closely and leave object silhouettes giving away the object class. In contrast, our model learns to perform removal by jointly optimizing the mask generator and the in-painter for the removal task with only weak supervision from image-level labels. This joint optimization allows the two components to cooperate to achieve removal performance on par with a fully supervised segmenter based removal.

3 Learning to remove objects

We propose an end-to-end model which learns to find and remove objects automatically from images without any human interaction. It learns to perform this removal with only access to image-level labels without needing expensive ground-truth location information like bounding boxes or masks. Additionally, we do not have ground-truth target images showing the expected output image with the target object removed since it is infeasible to obtain such data in general.

We overcome the lack of ground-truth location and target image annotations by designing a generative adversarial framework (GAN) to train our model with only unpaired data. Here our editor model learns from weak supervision from three different classifiers. The model learns to locate and remove objects by trying to fool an object classifier. It learns to produce realistic output by trying to fool an adversarial real/fake classifier. Finally, it learns to produce realistic looking object masks by trying to fool a mask shape classifier. Let us examine these components in detail.



(a) Our editor is composed of a mask-generator and an image in-painter

(b) Two-stage generator avoids adversarial patterns

Figure 1: Illustrating (a) the proposed two-staged architecture and (b) the motivation for this approach

3.1 Editor architecture: A two-staged approach

Recent works [4, 5] on image manipulation utilize a generator network which takes the input image and synthesizes the output image to reflect the target attributes. While this approach works well for structured images of single faces, we found in own experiments that it does not scale well for removing objects from general scene images. In general scenes with multiple objects, it is difficult for the generator to remove only the desired object while re-synthesizing the rest of the image exactly. Instead, the generator finds the easier solution to fool the object classifier by producing adversarial patterns. This is also facilitated by the fact that the object classifier in crowded scenes has a much harder task than a classifier determining hair-colors in object centric images and thus is more susceptible to adversarial patterns. Figure 1b illustrates this observation, where a single stage generator from [5] trying to remove the person, fools the classifier using adversarial noise. We can also see that the colors of the entire image have changed even when removing a single local object.

We propose a two-staged generator architecture shown in Figure 1a to address this issue. The first stage is a mask generator, G_M, which learns to locate the target object class, c_t, in the input image x and masks it out by generating a binary mask $m = G_M(x, c_t)$. The second stage is the in-painter, G_I, which takes the generated mask and the masked-out image as input and learns to in-paint to produce a realistic output. Given the inverted mask $\widetilde{m} = 1 - m$, final output image y is computed as

$$y = \widetilde{m} \cdot x + m \cdot G_I\left(\widetilde{m} \cdot x\right) \tag{1}$$

The mask generator is trained to fool the object classifier for the target class whereas the in-painter is trained to only fool the real/fake classifier by minimizing the loss functions shown below.

$$L_{\text{cls}}(G_M) = -\mathbb{E}_x\left[\log(1 - D_{cls}(y, c_t))\right] \tag{2}$$

$$L_{\text{rf}}(G_I) = -\mathbb{E}_x\left[D_{\text{rf}}(y)\right] \tag{3}$$

where $D_{cls}(y, c_t)$ is the object classifier score for class c_t and D_{rf} is the real/fake classifier.

Here D_{rf} is adversarial, i.e. it is constantly updated to classify generated samples y as "fake". The object classifier D_{cls} however is not adversarial, since it leads to the classifier using the context to predict the object class even when the whole object is removed. Instead, to make the D_{cls} robust to partially removed objects, we train it on images randomly masked with rectangles. The multiplicative configuration in (1) makes it easy for G_M to remove the objects by masking them out. Additionally, the in-painter also does not produce adversarial patterns as it is not optimized to fool the object classifier but only to make the output image realistic. The efficacy of this approach is illustrated in the image on the right on Figure 1b, where our two-staged model is able to cleanly remove the person without affecting the rest of the image.

3.2 Mask priors

While the two-stage architecture avoids adversarial patterns and converge to desirable solutions, it is not sufficient. The mask generator can still produce noisy masks or converge to bad solutions like masking most of the image to fool the object classifier. A simple solution is to favor small sized masks. We do this by simply minimizing the exponential function of the mask size, $\exp(\Sigma_{ij} m_{ij})$. But this only penalizes large masks but not noisy or incoherent masks.

To avoid these degenerate solutions, we propose a novel mechanism to regularize the mask generator to produce masks close to a prior distribution. We do this by minimizing the Wasserstein distance between the generated mask distribution and the prior distribution $P(m)$ using Wasserstein

GAN (WGAN) [21] as shown in Figure 2. The WGAN framework allows flexibility while choosing the prior since we only need samples from the prior and not a parametric form for the prior.

The prior can be chosen with varying complexity depending on the amount of information available, including knowledge about shapes of different object classes. For example we can use unpaired segmentation masks from a different dataset as a shape prior to the generator. When this is not available, we can impose the prior that objects are usually continuous coherent shapes by using simple geometric shapes like randomly generated rectangles as the prior distribution.

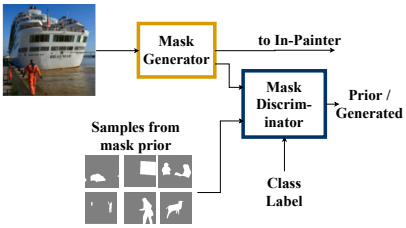


Figure 2: Imposing mask priors with a GAN framework

Given a class specific prior mask distribution, $P(m^p|c_t)$, we setup a discriminator, D_M to assign high scores to samples from this prior distribution and the masks generated by $G_M(x, c_t)$. The mask generator is then additionally optimized to fool the discriminator D_M.

$$L(D_M) = \mathbb{E}_{m^p \sim P(m^p|c_t)} \left[D_M(m^p, c_t) \right] - \mathbb{E}_x \left[D_M(G_M(x, c_t), c_t) \right] \tag{4}$$

$$L_{\text{prior}}(G_M) = -\mathbb{E}_x \left[D_M(G_M(x, c_t), c_t) \right] \tag{5}$$

3.3 Optimizing the in-painting network for removal

The in-painter network G_I is tasked with synthesizing a plausible image patch to fill the region masked-out by G_M, to produce a realistic output image. Similar to prior works on in-painting [22–24], we train G_I with self-supervision by trying to reconstruct random image patches and weak supervision from fooling an adversarial real/fake classifier. The reconstruction loss encourages G_I to keep consistency with the image while the adversarial loss encourages it to produce sharper images.

Reconstruction losses. To obtain self-supervision to the in-painter we mask random rectangular patches m^r from the input and ask G_I to reconstruct these patches. We minimize the L_1 loss and the perceptual loss [25] between the in-painted image and the input as follows:

$$L_{\text{recon}}(G_I) = \left\| G_I\left(\widetilde{m}^r \cdot x\right) - x \right\|_1 + \Sigma_k \left\| \phi_k\left(G_I\left(\widetilde{m}^r \cdot x\right)\right) - \phi_k(x) \right\|_1 \tag{6}$$

Mask buffer. The masks generated by $G_M(x, c_t)$ can be of arbitrary shape and hence the in-painter should be able to fill in arbitrary holes in the image. We find that the in-painter trained only on random rectangular masks performs poorly on masks generated by G_M. However, we cannot simply train the in-painter with reconstruction loss in (6) on masks generated by G_M. Unlike random masks m^r which are unlikely to align exactly with an object, generated masks $G_M(x, c_t)$ overlap the objects we intend to remove. Using reconstruction loss here would encourage the in-painter to regenerate this object. We overcome this by storing generated masks from previous batches in a *mask buffer* and randomly applying them on images from the current batch. These are not objects aligned anymore due to random pairing and we train the in-painter G_I with the reconstruction loss, allowing it to adapt to the changing mask distribution produced by the $G_M(x, c_t)$.

Local real/fake loss. In recent works on in-painting using adversarial loss [22–24], in-painter is trained adversarially against a classifier D_{rf} which learns to predict global "real" and "fake" labels for input x and the generated images y respectively. A drawback with this formulation is that only a small percentage of pixels in the output y is comprised of truly "fake" pixels generated by the in-painter, as seen in Equation (1). This is a hard task for the classifier D_{rf} hard since it has to find the few pixels that contribute to the global "fake" label. We tackle this by providing local pixel-level real/fake labels on the image to D_{rf} instead of a global one. The pixel-level labels are available for free since the inverted mask $\widetilde{m}$ acts as the ground-truth "real" label for D_{rf}. Note that this is different from the patch GAN [12] where the classifier producing patch level real/fake predictions is still supervised with a global image-level real/fake label. We use the least-square GAN loss [26] to train the D_{rf}, since we found the WGAN loss to be unstable with local real/fake prediction. This is because, D_{rf} can minimize the WGAN loss with assigning very high/low scores to one patch, without bothering with the other parts of the image. However, least-squares GAN loss penalizes both very high and very low predictions, thereby giving equal importance to different image regions.

$$L(D_{\text{rf}}) = \frac{1}{\Sigma_{ij}\widetilde{m}_{ij}} \sum_{ij} \widetilde{m}_{ij} \cdot \left(D_{\text{rf}}(y)_{ij} - 1\right)^2 + \frac{1}{\Sigma_{ij} m_{ij}} \sum_{ij} m_{ij} \cdot \left(D_{\text{rf}}(y)_{ij} + 1\right)^2 \tag{7}$$

Penalizing variations. We also incorporate the style-loss (L_{sty}) proposed in [24] to better match the textures in the in-painting output with that of the input image and the total variation loss (L_{tv}) since it helps produce smoother boundaries between the in-painted region and the original image.

The mask generator and the in-painter are optimized in alternate epochs using gradient descent. When the G_M is being optimized, parameters of G_I are held fixed and vice-versa when G_I is optimized. We found that optimizing both the models at every step led to unstable training and many training instances converged to degenerate solutions. Alternate optimization avoids this while still allowing the mask generator and in-painter to co-adapt. The final loss function for G_M and G_I is given as:

$$L_{\text{total}}(G_M) = \lambda_c L_{\text{cls}} + \lambda_p L_{\text{prior}} + \lambda_{sz} \exp(\Sigma_{ij} m_{ij}) \tag{8}$$

$$L_{\text{total}}(G_I) = \lambda_{rf} L_{\text{rf}} + \lambda_r L_{\text{recon}} + \lambda_{tv} L_{\text{tv}} + \lambda_{sty} L_{\text{sty}} \tag{9}$$

4 Experimental setup

Datasets. Keeping with the goal of performing removal on general scene images, we train and test our model mainly on the COCO dataset [9] since it contains significant diversity within object classes and in the contexts in which they appear. We test our proposed GAN framework to impose priors on the mask generator with two different priors namely rotated boxes and unpaired segmentation masks. We use the segmentation masks from Pascal-VOC 2012 dataset [27] (without the images) as the unpaired mask priors. To facilitate this we restrict our experiments on 20 classes shared between the COCO and Pascal datasets. To demonstrate that our editor model can generalize beyond objects and can learn to remove to different image entities, we test our model on the task of removing logos from natural images. We use the Flickr Logos dataset [28], which has a training set of 810 images containing 27 annotated logo classes and a test set of 270 images containing 5 images per class and 135 random images containing no logos. Further details about data pre-processing and network architectures is presented in the supplementary material.

Evaluation metrics. We evaluate our object removal for three aspects: *removal performance* to measure how effective is our model at removing target objects and *image quality assessment* to quantify how much of the original image is edited and finally *human evaluation* to judge removal.

- **Removal performance:** We quantify the removal performance by measuring the performance of an object classifier on the edited images using two metrics. *Removal success rate* measures the percentage of instances where the editor successfully fools the object classifier score below the decision boundary for the target object class.*False removal rate* measures the percentage of cases where the editor removes the wrong objects while trying to remove the target class. This is again measured by monitoring if the object classifier score drops below decision boundary for other classes.
- **Image quality assessment:** To be useful, our editor should remove the target object class while leaving the rest of the image intact.Thus, we quantify the usefulness by measuring similarity between the output and the input image using three metrics namely peak signal-to-noise ratio (pSNR), structural similarity index (ssim) [29] and perceptual loss [30]. The first two are standard metrics used in image in-painting literature, whereas the perceptual loss [30] was recently proposed as a learned metric to compare two images. We use the squeezenet variant of this metric.
- **Human evaluation:** We conduct a study to obtain human judgments of removal performance. We show hundred randomly selected edited images to a human judge and asked if they see the target object class. To keep the number of annotations reasonable, we conduct the human evaluation only on person class (largest class). Each image is shown to three separate judges and removal is considered successful when all three humans agree that they do not see the object class.

Baseline with additional supervision. Since there is no prior work proposing a fully automatic object removal solution, we compare our model against removal using a stand-alone fully supervised segmentation model, Mask-RCNN [8]. We obtain segmentation mask predictions from Mask-RCNN and use our trained in-painter to achieve removal. Please note that this method uses much stronger supervision in terms of object localization and object segmentation than our proposed method.

5 Results

We present qualitative and quantitative evaluations of our editor and comparisons to the Mask-RCNN based removal. Qualitative results show that our editor model works well across diverse scene types and object classes. Quantitative analysis shows that our weakly supervised model performs on par with the fully supervised Mask-RCNN in the removal task, in both automatic and human evaluation.

Figure 3: Qualitative examples of removal of different object classes

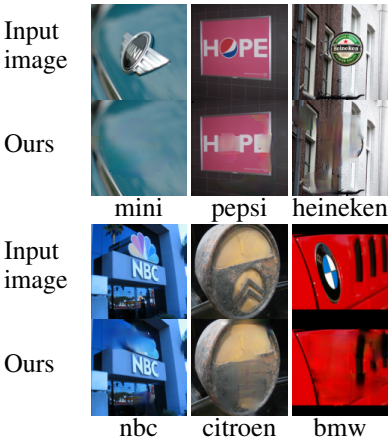


Figure 4: Results of logo removal

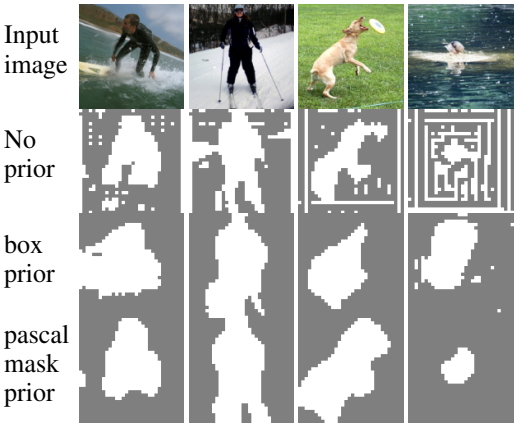


Figure 5: Effect of priors on generated masks

5.1 Qualitative results

Figure 3 shows the results of object removal performed by our model (last row) on the COCO dataset compared to the Mask-RCNN baseline. We see that our model works across diverse scene types, with single objects (columns 1-4) or multiple instances of the same object class (col. 5-6) and even for a fairly large object (last column). Figure 3 also highlight the problems with simply using masks from a segmentation model, Mask-RCNN, for removal. Mask-RCNN is trained to accurately segment the objects and thus the masks it produces very closely trace the object boundary, too closely for removal purposes. We can clearly see the silhouettes of objects in all the edited images on the second row. These results justify our claim that segmentation annotations are not needed to learn to remove objects and might not be the right annotations anyway.

Our model is not tied to notion of objectness and can be easily extended to remove other image entities. The flexible GAN based mask priors allow us to use random rectangular boxes as priors when object shapes are not available. To demonstrate this we apply our model to the task of removing brand logos automatically from images. The model is trained using image level labels and box prior. Qualitative examples in Figure 4 shows that our model works well for this task, despite the fairly small training set (800 images). It is able to find and remove logos in different contexts with only image level labels. The image on the bottom left shows a failure case where the model fails to realize that the text "NBC" belongs to the logo.

Figure 5 shows the masks generated by our model with different mask priors on the COCO dataset. These examples illustrate the importance of the proposed mask priors. The masks generated by the model using no prior (second row) are very noisy since the model has no information about object shapes and is trying to infer everything from the image level classifier. Adding the box prior already makes the masks much cleaner and more accurate. We can note that the generated masks are "boxier" while not strictly rectangles. Finally using unpaired segmentation masks from the pascal dataset as

Table 1: Quantifying the effect of using more accurate mask priors

Prior	Removal Performance			Image quality metrics			Mask accuracy	
	removal success ↑		false ↓ removal	percep. loss ↓	pSNR ↑	ssim ↑	mIou ↑	% masked area ↓
	all	person						
None	**94**	**96**	36	0.13	19.97	0.743	0.15	37.7
boxes	83	88	23	0.11	20.41	0.777	0.18	28.1
pascal (10)	67	59	17	**0.07**	**23.81**	**0.833**	**0.23**	**16.7**
pascal (100)	70	75	**16**	**0.07**	23.02	0.821	0.22	18.1
pascal (all)	73	81	**16**	0.08	22.64	0.803	0.22	20.2

shape priors makes the generated masks more accurate and the model is able to recover the object shapes better. This particularly helps in object with diverse shapes, for example people and dogs.

5.2 Quantitative evaluation of removal performance

To quantify the removal performance we run an object classifier on the edited images and measure its performance. We use a separately trained classifier for this purpose, not the one used in our GAN training, to fairly compare our model and the Mask-RCNN based removal.

Sanity of object classifier performance. The classifier we use to evaluate our model achieves per-class average F1-score of 0.57, overall average F1-score of 0.67 and mAP of 0.58. This is close to the results achieved by recent published work on multi-label classification [31] on the COCO dataset, which achieves class average F1-score of 0.60, overall F1-score of 0.68 and mAP of 0.61. While these numbers are not directly comparable (different image resolution, different number of classes), it shows that our object classifier has good performance and can be relied upon. Furthermore, human evaluation shows similar results as our automatic evaluation.

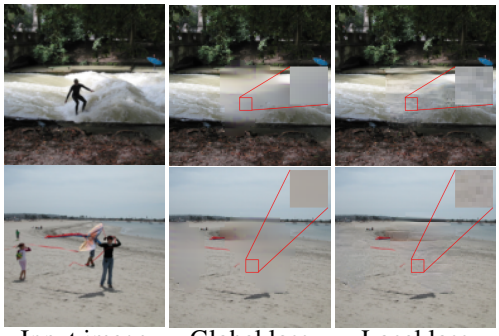
Effect of priors. Table 1 compares the different versions of our model using different priors. The box prior uses randomly generated rectangles of different aspect ratios, area and rotations. The *Pascal (n)* prior uses n randomly chosen unpaired segmentation masks for each class from the Pascal dataset. The table shows metrics measuring the removal performance, image quality and mask accuracy. The arrows ↑ and ↓ indicate if higher or lower is better for the corresponding metric. Comparing removal performance in Table 1 we see that while the model with no prior achieves very high removal rate (94%), but it does so with large masks (37 %) which causes low output image quality. As we add priors, the generated masks become smaller and compact. We also see that mIou of the masks increase with stronger priors (0.22-0.23 for pascal prior), indicating they are more accurate. Smaller and more accurate masks also improve the image quality metrics and false removal rates which drop more than half from 36% to 16%. This is inline with the visual examples in Figure 5, where model without prior produces very noisy masks and quality of the masks improve with priors.

Another interesting observation from Table 1 is that using very few segmentation masks from pascal dataset leads to a drop in removal success rate, especially for the *person* class. This is because the *person* class has very diverse shapes due to varying poses and scales. Using only ten masks in the prior fails to capture this diversity and performs poorly (59%). As we increase the number of mask samples in the prior, removal performance jumps significantly to 81% on the person class. Considering these results, we note that the *pascal all* version offers the best trade-off between removal and image quality due to more accurate masks and we will use this model in comparison to benchmarks.

Benchmarking against GT and Mask-RCNN. Table 2 compares the performance of our model against baselines using ground-truth (GT) masks and Mask-RCNN segmentation masks for removal. These benchmarks use the same in-painter as *our-pascal* model. We see that our model outperforms the fully supervised Mask-RCNN masks and even the GT masks in terms of removal (66%& 68% vs 73%). While surprising, this is explained by the same phenomenon we saw in qualitative results with Mask-RCNN in Figure 3. The GT and Mask-RCNN masks for segmentation are too close to the object boundaries and thus leave object silhouettes behind when used for removal. When we dilate the masks produced by Mask-RCNN before using for removal, the performance improves overall and is on par with our model (slightly better in all classes and a bit worse in the person class). The drawback of weak supervision is that masks are a bit larger which leads to bit higher false removal rate (16% ours compared to 10% Mask-RCNN dilated) and lower image quality metrics. However this is still a remarkable result, given that our model is trained without expensive ground truth segmentation annotation for each image, but instead uses only unpaired masks from a smaller dataset.

Table 2: Comparison to ground truth masks and Mask-RCNN baselines.

Model	Supervision	Removal Performance			Image quality metrics		
		removal success ↑		false ↓ removal	percep. loss ↓	pSNR ↑	ssim ↑
		all	person				
GT masks	-	66	72	5	**0.04**	**27.43**	**0.930**
Mask RCNN	Seg. masks & bound boxes	68	73	6	0.05	25.59	0.900
Mask RCNN (dil. 7x7)		75	77	10	0.07	24.13	0.882
ours-pascal	image labels & unpaired masks	73	**81**	16	0.08	22.64	0.803



Input image Global loss Local loss

Figure 6: Comparing global and local GAN loss. Global loss smooth blurry results, while local one produce sharp, texture-rich images.

Table 3: Evaluating in-painting components

Mask buffer	GAN	TV+ Style	percep. loss ↓	pSNR ↑	ssim ↑
-	G	-	0.13	20.0	0.730
✓	G	-	0.12	**21.9**	**0.772**
✓	L	-	**0.10**	21.5	0.758
✓	L	✓	**0.10**	21.6	0.763

Table 4: Joint training helps improve both mask generation and in-painting

Joint training	Removal success ↑	mIou ↑	percep. loss ↓
-	0.68	0.19	0.10
✓	**0.73**	**0.22**	**0.08**

Human evaluation. We verify our automatic evaluation results using a user study to evaluate removal success as described in Section 4. The human judgements of removal performance follow the same trend seen in automatic evaluation, except that human judges penalize the silhouettes more severely.Our model clearly outperforms the baseline Mask-RCNN model without dilation by achieving 68% removal rate compared to only 30% achieved by Mask-RCNN. With dilated masks, Mask-RCNN performs similar to our model in terms of removal achieving 73% success rate.

5.3 Ablation studies

Joint optimization. We conduct an experiment to test if jointly training the mask generator and the in-painter helps. We pre-train the in-painter using only random boxes and hold it fixed while training the mask generator. The results are shown in Table 4. Not surprisingly, the in-painting quality suffers with higher perceptual loss (0.10 vs 0.08) since it has not adapted to the masks being generated. More interestingly, the mask generator also degrades with a fixed in-painter, as seen by lower mIou (0.19 vs 0.22) and lower removal success rate (0.68 vs 0.73). This result shows that it is important to train both the models jointly to allow them to adapt to each other for best performance.

In-painting components. Table 5 shows the ablation of the in-painter network components. We note that the proposed *mask-buffer*, which uses masks from previous batch to train the in-painter with reconstruction loss, significantly improves the results significantly in all three metrics. Using local loss improves the results in-terms of perceptual loss (0.10 vs 0.12) while being slightly worse in the other two metrics. However on examining the results visually in Figure 6, we see that the version with the global GAN loss produces smooth and blurry in-painting, whereas the version with local GAN loss produces sharper results with richer texture. While these blurry results do better in pixel-wise metrics like pSNR and ssim, they are easily seen by the human eye and are not suitable for removal. Finally addition of total variation and style loss helps slighlty improve the pSNR and ssim metrics.

6 Conclusions

We presented an automatic object removal model which learns to find and remove objects from general scene images. Our model learns to perform this task with only image level labels and unpaired data. Our two-stage editor model with a mask-generator and an in-painter network avoids degenerate solutions by complementing each other. We also developed a GAN based framework to impose different priors to the mask generator, which encourages it to generate clean compact masks to remove objects. Results show that our model achieves similar performance as a fully-supervised segmenter based removal, demonstrating the feasibility of weakly supervised solutions for the general scene-level editing task.

Acknowledgments

This research was supported in part by the German Research Foundation (DFG CRC 1223).

References

[1] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. 4, 2017.

[2] T. Orekondy, M. Fritz, and B. Schiele, "Connecting pixels to privacy and utility: Automatic redaction of private information in images," *arXiv preprint arXiv:1712.01066*, 2017.

[3] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[4] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer *et al.*, "Fader networks: Manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[5] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *arXiv preprint arXiv:1711.09020*, 2017.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[7] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," *arXiv preprint arXiv:1711.11585*, 2017.

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[9] X. Chen, T.-Y. L. Hao Fang, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick, "Microsoft COCO captions: Data collection and evaluation server," *arXiv preprint arxiv:1504.00325*, 2015.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[11] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," in *33rd International Conference on Machine Learning*, 2016.

[14] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[15] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," *arXiv preprint arXiv:1711.10485*, 2017.

[16] J. Johnson, A. Gupta, and L. Fei-Fei, "Image generation from scene graphs," *arXiv preprint arXiv:1804.01622*, 2018.

[17] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[18] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Transactions on image processing*, 2004.

[19] J. Hays and A. A. Efros, "Scene completion using millions of photographs," in *ACM Transactions on Graphics (TOG)*, 2007.

[20] S. S. Mirkamali and P. Nagabhushan, "Object removal by depth-wise image inpainting," *Signal, Image and Video Processing*, 2015.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.

[22] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," *arXiv preprint arXiv:1801.07892*, 2018.

[23] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics (TOG)*, 2017.

[24] G. Liu, F. A. Reda, K. J. Shih, T.-C. Wang, A. Tao, and B. Catanzaro, "Image inpainting for irregular holes using partial convolutions," *arXiv preprint arXiv:1804.07723*, 2018.

[25] L. Gatys, A. Ecker, and M. Bethge, "A neural algorithm of artistic style," *Nature Communications*, 2015.

[26] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[27] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[28] Y. Kalantidis, L. Pueyo, M. Trevisiol, R. van Zwol, and Y. Avrithis, "Scalable triangulation-based logo recognition," in *Proceedings of ACM International Conference on Multimedia Retrieval (ICMR)*, 2011.

[29] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, 2004.

[30] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," *arXiv preprint arXiv:1801.03924*, 2018.

[31] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[32] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[33] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[34] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics-TOG*, 2009.

A Data pre-processing

We pre-process the COCO dataset to filter out images containing large objects. This is done by removing images with single object class covering more than 30% of the image. Removing very large objects requires the model to hallucinate most of the image, and the task becomes very difficult. This is also reasonable from application point of view, since the object larger than 30% of the image are very often the focus of the image, and removing them would not be very useful. After size filtering we are have 39238 training, 2350 validation and 1905 test images.

In both the datasets, the images are preprocessed by resizing the shortest edge to 128 pixels and center-cropping to obtain 128x128 images. Further data augmentation is performed using random horizontal flips and the images are normalized to have zero mean and unit variance.

Similar preprocessing is applied to the masks from the Pascal dataset to obtain 128x128 dimensional masks for the prior. The Pascal dataset has about 215 masks on average for each of the 20 classes. This gives us 4318 masks for our mask-prior, which is much lower compared to 39238 training images in our pre-processed dataset.

B Network architectures

Mask generator.. The mask generator architecture is built on top of a VGG network pre-trained on Image net classification task. Note that this pre-training also uses only image-level labels. We use the features from the convolutional layer before the first fully connected layer from the VGG-19 architecture as our starting point. Additionally, we remove two previous maxpool layers to obtain features of dimensions $512 \times 32 \times 32$. This is concatenated with 20 dimensional one-hot vector representing the target class. On top of this we add the following layers:

$$C^3_{512} - L_{0.1} - R_{512} - C^3_{256} - L_{0.1} - R_{256} - C^3_{128} - L_{0.1} - R_{128} - C^7_{21} - S$$

where C^k_n indicates a convolutional layer with n filters of $k \times k$ size and R_n is a residual block with n filters, $L_{0.1}$ is leaky relu non-linear activation layer and S is the sigmoid activation function.

Each residual block is R_n consists of

$$C^3_n - I_n - L_{0.1} - C^3_n - I_n - L_{0.1}$$

where I_n is a n dimensional instance normalization layer. with slope parameter 0.1. We also concatenate the target class vector after every residual block since it improves the performance.

In-painter.. Our in-painter network architecture is designed borrowing ideas from prior works [5, 23]. We use the same basic architectures as in [5] but incorporate dilated convolutions in the bottleneck layers to improve performance on larger masks as proposed in [23]. The in-painter takes the input image concatenated with the mask and is agnostic to the object class.

The in-painter network is built with a *downsampling* block, *bottle-neck* block and the *upsampling* block. The layers in the *downsampling* block are

$$C^4_{64} - I_{64} - L_{0.1} - D^2_{128} - I_{128} - L_{0.1} - D^2_{256} - I_{256} - L_{0.1} - D^2_{512} - I_{512} - L_{0.1}$$

where D^2_n is a downsampling layer halving the spatial dimensions of the input. It consists of a convolutional layer with filter size 4×4 and stride 2. The *bottle-neck* block is just six back-to-back residual layers, R_{256}. Finally, the *upsampling* block is made of three upsampling blocks followed by an output convolutional layer with tanh non-linearity (T)

$$U^2 - C^3_{256} - I_{256} - L_{0.1} - U^2 - C^3_{128} - I_{128} - L_{0.1} - U^2 - C^3_{64} - I_{64} - L_{0.1} - C^7_3 - T$$

where U_2 is a bilinear up-sampler which doubles the spatial dimensions of the input.

Object Classifier.. Our object classifier is designed on top off the VGG-19 network backbone which is pre-trained on Imagenet. We add the following layers after the last convolutional layer of VGG-19

$$C^3_{512} - B_{512} - L_{0.1} - C^3_{512} - B_{512} - L_{0.1} - G - \text{Lin}_{20} - S$$

where B_n is a n-dimensional batch normalization layer, G is a global pooling layer and Lin_{20} is a linear layer mapping input to 20 class scores.

C Implementation Details

All the components of our model are trained using stochastic gradient descent with Adam optimizer. The adversarial discriminators used in the mask prior and real/fake classification are regularized with the gradient penalty [32]. The model hyper-parameters were chosen using a validation set. The final settings of the hyper-parameters in our loss function were $\lambda_c = 12$, $\lambda_p = 3$, $\lambda_{sz} = 18$, $\lambda_{rf} = 2$, $\lambda_r = 100$,$\lambda_{tv} = 10$, and $\lambda_{sty} = 3000$.

We implement all our models with the PyTorch framework. We will release our code upon publication.

Table 5: Comparing our inpainter to state-of-the art methods. Other results are taken from the paper [24].

Metric	Method	Relative mask size (0-0.1)	[0.1-0.2)	[0.2-0.3)	[0.3-0.4)	[0.4-0.5)	[0.5-0.6)
SSIM	PM [34]	0.947	0.865	0.768	0.675	0.579	0.472
	GL [23]	0.923	0.829	0.721	0.627	0.533	0.440
	PConv [24]	0.945	0.870	0.779	0.689	0.595	0.484
	Ours	**0.972**±0.00	**0.925**±0.00	**0.870**±0.00	**0.813**±0.00	**0.758**±0.00	**0.721**±0.01
pSNR	PM [34]	33.68	27.51	24.35	22.05	20.58	18.22
	GL [23]	29.74	23.83	20.73	18.61	17.38	16.37
	PConv [24]	**34.34**	**28.32**	**25.25**	**22.89**	**21.38**	**19.04**
	Ours	32.68±0.09	26.07±0.03	22.84±0.03	20.67±0.02	19.09±0.04	18.14± 0.17

D Comparing In-painter to state-of-the-art

We compare our in-painter model to state-of-the-art image in-painting models from literature on the Places2 [33] dataset in Table 5. Results from the other papers shown in Table 5 are taken from [24]. Since we do not have access to the masks used to test these models, we test ours using random rotated rectangular masks. We mask the input image with up to four randomly generated rectangular masks and measure the in-painting performance. From the table we see that our model performs better than prior work in-terms of structural similarity index (ssim) and is bit worse than partial convolution based method [24] (PConv) in terms of pSNR.

E Qualitative Results

Figure 7 shows more qualitative examples of objects removal on the COCO dataset. Examples in the first three rows showcase a wide-variety of scenes where our model is able to successfully remove the target object class. It works for objects of different poses and sizes, and with single or multiple instances of the target object class.

Last row of Figure 7 highlights some failure modes we observed in our model. The First four columns in the last row shows failures where the full extent of the object is not removed and some parts of the object is still visible. However, only in case of the *motorcycle* we can clearly identify the removed object class in the edited image. In case of the *boat* image, only one instance of *boat* has been removed by the editor. Similarly in the second last column, although the larger *person* has been removed, the smaller instances of people are visible in the background. Finally, the last column shows a case of *false removal* wherein in an attempt to remove the *horse*, our editor also removes the people riding the horse.

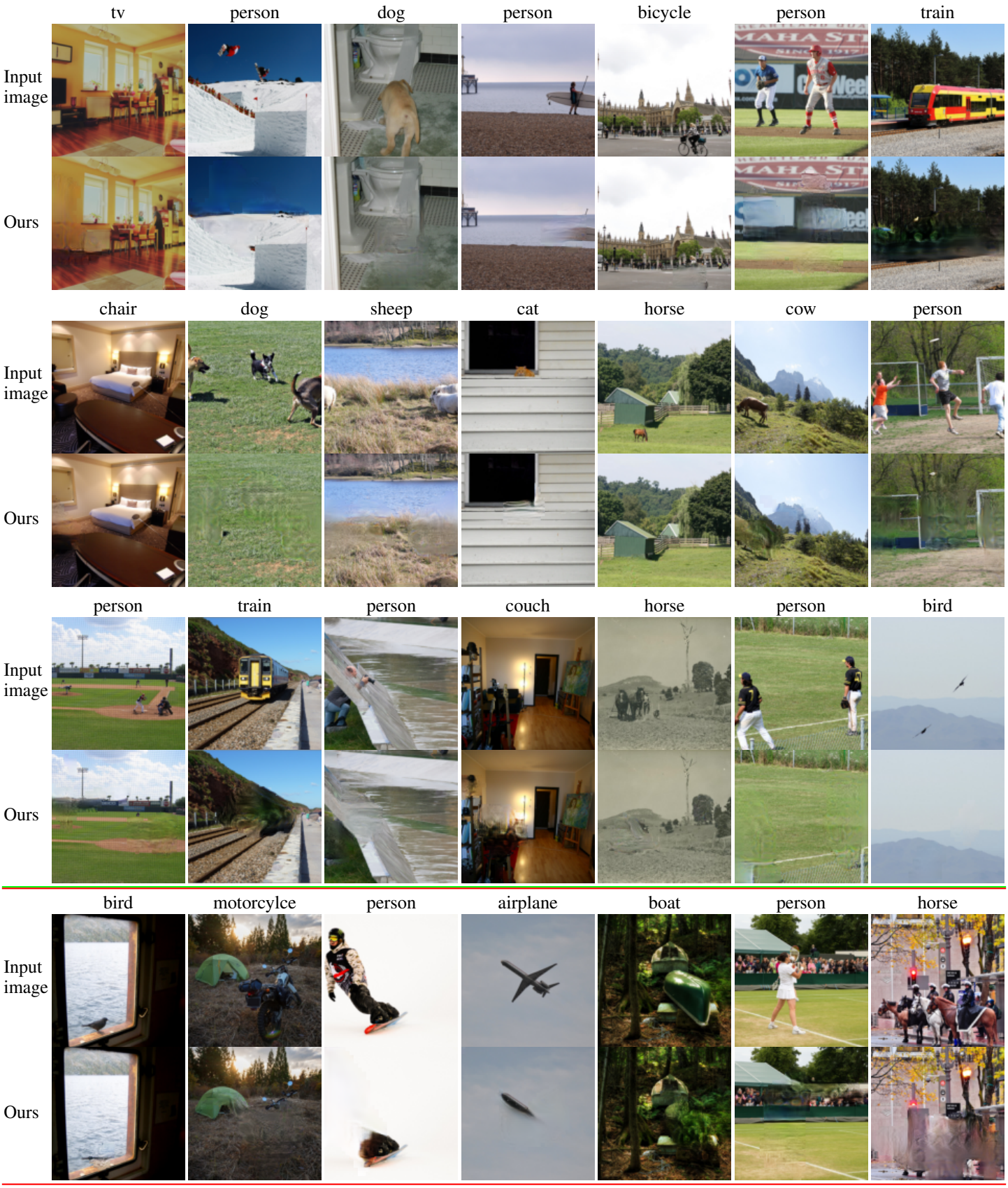


Figure 7: Qualitative examples of removal of different object classes in diverese scenes. First three rows show examples where our model does well, whereas the last row highlights some failure cases.