# Supplementary Material

## 1 Model Architecture

The code to reproduce all experiments and results in this paper (including weights of trained models) is available at `https://github.com/Schlumberger/pixel-constrained-cnn-pytorch`.

### 1.1 MNIST

| Prior Network |
| --- |
| Restricted Gated Conv Block, 32 filters, $5 \times 5$ |
| $14\times$ Gated Conv Block, 32 filters, $5 \times 5$ |
| Conv, 2 filters, $1 \times 1$ |

| Conditioning Network |
| --- |
| $15\times$ Residual Blocks, 32 filters, $5 \times 5$ |
| Conv, 2 filters, $1 \times 1$ |

### 1.2 CelebA

| Prior Network |
| --- |
| Restricted Gated Conv Block, 66 filters, $5 \times 5$ |
| $16\times$ Gated Conv Block, 66 filters, $5 \times 5$ |
| Conv, 1023 filters, $1 \times 1$, ReLU |
| Conv, 96 filters, $1 \times 1$ |

| Conditioning Network |
| --- |
| $17\times$ Residual Blocks, 66 filters, $5 \times 5$ |
| Conv, 96 filters, $1 \times 1$ |

## 2 Model Training

**MNIST**

- Optimizer: Adam
- Learning rate: 4e-4
- $\alpha$: 1
- Epochs: 50

**CelebA**

- Optimizer: Adam
- Learning rate: 4e-4
- $\alpha$: 1
- Epochs: 60

## 3 Data

### 3.1 MNIST

We binarize the MNIST images by setting pixel intensities greater than 0.5 to 1 and others to 0.

### 3.2 CelebA

We quantize the CelebA images from the full 8 bits to 5 bits (i.e. 32 colors). The original (218, 178) images are cropped to (89, 89) and then resized to (32, 32).

## 4 Mask Generation Algorithm

The parameters used for generating masks are `max_num_blobs=4`, `iter_min` = 2, `iter_max` = 7 for both MNIST and CelebA.

## 5 More Samples

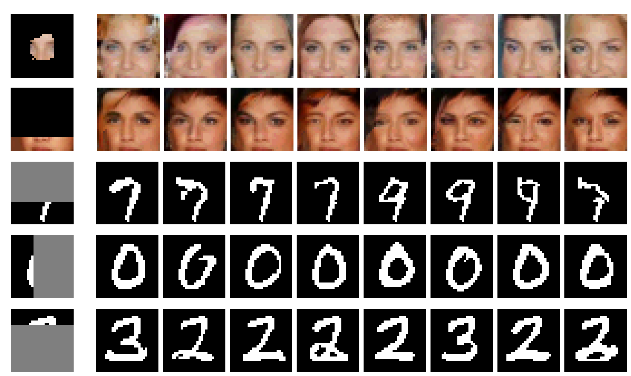

Figure 1: Inpaintings.

## 6 User Study

The user study was designed to verify if likelihood estimates from the model correlate well with human per-

**Algorithm 1** Generate masks of random blobs.

```
Require: Mask height h and width w
Require: max_num_blobs: maximum number of blobs
Require: iter_min: min # of iters to expand blob
Require: iter_max: max # of iters to expand blob
  mask ← zero array of size (h, w)
  num_blobs ~ Unif(1, max_num_blobs)
  for i = 1:num_blobs do
    num_iters ~ Unif(iter_min, iter_max)
    x_0 ~ Unif(1, w)
    y_0 ~ Unif(1, h)
    mask[x_0, y_0] ← 1
    start_positions ← [(x_0, y_0)]
    for j = 1:num_iters do
      next_start_positions ← []
      for pos in start_positions do
        for x, y in neighbors(pos) do
          p ~ Unif(0, 1)
          if p > 0.5 then
            mask[x, y] ← 1
            next_start_positions append (x, y)
          end if
        end for
      end for
      start_positions ← next_start_positions
    end for
  end for
  return mask
```

ception of plausible inpaintings. We include the details of how this test was setup here:

1. We randomly selected 100 images and generated 8 inpaintings for each of them.
2. The samples with highest and lowest likelihood were selected for each of the 100 images.
3. We then calculated the percentage difference between the highest and lowest likelihood sample for each of the 100 pairs and selected the 15 pairs with the largest difference. This was done to ensure the model assigned significantly different likelihoods to each image.
4. The user was then presented with the occluded image and was asked to choose which of the generated images they felt was most plausible. They were also given the option to choose neither as some completions were equally good or equally bad.

Examples of the user interface for the survey are shown in Fig. 2.

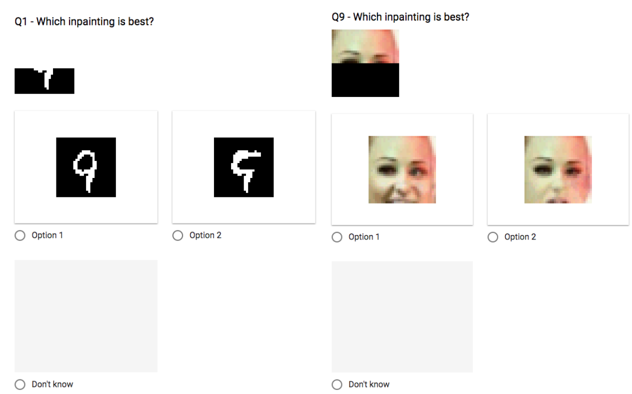


Figure 2: Human survey user interface.