Guided Image Generation with Conditional Invertible Neural Networks
– Supplementary Material –

Contents
1. Architecture details – MNIST generation . . . . . 1
2. Architecture details – Colorization . . . . . . . . 1
3. Colorization – Ablations . . . . . . . . . . . . . . . 3
4. Colorization – Interpolations . . . . . . . . . . . . . 4
5. Colorization – Additional examples . . . . . . . . . 9
   5.1. General examples . . . . . . . . . . . . . . 9
   5.2. Humans . . . . . . . . . . . . . . . . . . . 21
   5.3. Lacking consistency . . . . . . . . . . . . 25
   5.4. Color ignores semantic content . . . . . . . 26
   5.5. Outright failures . . . . . . . . . . . . . . 29

1 Architecture details – MNIST generation

The network consists of 24 segments with a conditional coupling block and random permutation each:

\[
24 \times \begin{cases} 
\text{Cond. coupling block} \\
\text{Random permutation}
\end{cases}
\]

The Adam optimizer was used, with \(\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-6}\), L2 weight decay \(10^{-5}\). The initial learning rate is \(10^{-4}\), exponentially decaying to \(10^{-6}\). The network requires roughly 2GB VRAM, at a large batch size 256, and can be easily trained on a single GPU. The affine coefficients \(s_j, t_j\) are produced jointly by a single MLP of the following form:

- Fully connected \(392 + 10 \rightarrow 512\)
- ReLU
- Fully connected \(512 \rightarrow 512\)
- ReLU
- Fully connected \(512 \rightarrow 512\)
- ReLU
- Fully connected \(512 \rightarrow 784 (\equiv [s_j, t_j])\)

2 Architecture details – Colorization

The network uses 5 resolution levels, segmenting the latent space into 4 partitions. Each conditional coupling block uses a separate head on top of the feature extraction network, except for the lowest resolution level, which only uses a single head for all coupling blocks. We produce the multiplicative and additive affine coefficients \(s_j, t_j\) jointly though a single network (‘Coefficient functions’ below).

All convolutions, unless otherwise specified, use kernel size 3, stride 1, and padding 1. The Adam optimizer was used, with \(\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-6}\), L2 weight decay \(10^{-5}\). The initial learning rate is \(1.6 \cdot 10^{-4}\), reducing by a factor 0.2 when reaching a plateau. A batch size of 16 fits within 8GB VRAM, and we train with a batch size of \(3 \times 16 = 48\) on 3 GPUs.
2.0.1 Level 1 (2x64x64 Px)

\[ 4 \times \left\{ \begin{array}{l} \text{Cond. coupling block} \\
\text{Random orthog. 1x1 conv.} \\
\text{Reshape} \\
\end{array} \right\} \]

\[ 2 \times \left\{ \begin{array}{l} \text{Random orthog. 1x1 conv.} \\
\text{Coupling block (Uncond.)} \\
\end{array} \right\} \]

**Conditional coefficient functions:**
- Convolution: \(1 + 32 \rightarrow 32\) Ch.
- Leaky ReLU: Slope \(5 \cdot 10^{-2}\)
- Convolution: \(2 \rightarrow 32\) Ch.
- Leaky ReLU: Slope \(5 \cdot 10^{-2}\)
- Convolution: \(32 \rightarrow 2\) Ch. \((= [s_j, t_j])\)

**Unconditional coefficient functions:**
- Convolution: \(4 \rightarrow 64\) Ch., \(1 \times 1\) Kernel
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(64 \rightarrow 64\) Ch., \(1 \times 1\) Kernel
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(64 \rightarrow 8\) Ch., \(1 \times 1\) Kernel

**Conditioning heads:**
- Convolution: \(256 \rightarrow 32\) Ch.
- Batch normalization

2.0.2 Level 2 (8x32x32 Px)

\[ 6 \times \left\{ \begin{array}{l} \text{Cond. coupling block} \\
\text{Random orthog. 1x1 conv.} \\
\text{Reshape} \\
\end{array} \right\} \]

\[ 2 \times \left\{ \begin{array}{l} \text{Random orthog. 1x1 conv.} \\
\text{Coupling block (Uncond.)} \\
\text{Split off 16 channels as } z^{(1)} \end{array} \right\} \]

**Conditional coefficient functions:**
- Convolution: \(8 + 128 \rightarrow 128\) Ch.
- Leaky ReLU: Slope \(5 \cdot 10^{-2}\)
- Convolution: \(128 \rightarrow 128\) Ch.
- Leaky ReLU: Slope \(5 \cdot 10^{-2}\)
- Convolution: \(128 \rightarrow 16\) Ch.

**Unconditional coefficient functions:**
- Convolution: \(32 \rightarrow 256\) Ch., \(1 \times 1\) Kernel
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(256 \rightarrow 256\) Ch., \(1 \times 1\) Kernel
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(256 \rightarrow 64\) Ch., \(1 \times 1\) Kernel

**Conditioning heads:**
- Convolution: \(256 \rightarrow 128\) Ch., Stride 2
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(128 \rightarrow 128\) Ch., Stride 2
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(128 \rightarrow 128\) Ch.
- Batch normalization

2.0.3 Level 3 (16x16x16 Px)

\[ 6 \times \left\{ \begin{array}{l} \text{Cond. coupling block} \\
\text{Random orthog. 1x1 conv.} \\
\text{Reshape} \\
\end{array} \right\} \]

\[ 2 \times \left\{ \begin{array}{l} \text{Random orthog. 1x1 conv.} \\
\text{Coupling block (Uncond.)} \\
\text{Split off 32 channels as } z^{(2)} \end{array} \right\} \]

**Conditional coefficient functions:**
- Convolution: \(8 + 128 \rightarrow 128\) Ch.
- Leaky ReLU: Slope \(5 \cdot 10^{-2}\)
- Convolution: \(128 \rightarrow 128\) Ch.
- Leaky ReLU: Slope \(5 \cdot 10^{-2}\)
- Convolution: \(128 \rightarrow 16\) Ch.

**Unconditional coefficient functions:**
- Convolution: \(32 \rightarrow 256\) Ch., \(1 \times 1\) Kernel
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(256 \rightarrow 256\) Ch., \(1 \times 1\) Kernel
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(256 \rightarrow 64\) Ch., \(1 \times 1\) Kernel

**Conditioning heads:**
- Convolution: \(256 \rightarrow 128\) Ch., Stride 2
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(128 \rightarrow 128\) Ch., Stride 2
- Leaky ReLU: Slope \(10^{-2}\)
- Convolution: \(128 \rightarrow 128\) Ch.
- Batch normalization
2.0.4 Level 4 (32x8x8 Px)

\[
6 \times \left\{ \begin{array}{l}
\text{Cond. coupling block} \\
\text{Random orthog. 1x1 conv.} \\
\text{Reshape}
\end{array} \right. \\
2 \times \left\{ \begin{array}{l}
\text{Random orthog. 1x1 conv.} \\
\text{Coupling block (Uncond.)}
\end{array} \right. \\
\text{Split off 96 channels as } z^{(4)}
\]

**Conditional coefficient functions:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>16 + 256 → 256 Ch.</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 5 \cdot 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 256 Ch.</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 5 \cdot 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 32 Ch.</td>
</tr>
</tbody>
</table>

**Unconditional coefficient functions:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>64 → 256 Ch., 1 \times 1 Kernel</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 256 Ch., 1 \times 1 Kernel</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 128 Ch., 1 \times 1 Kernel</td>
</tr>
</tbody>
</table>

**Conditioning heads:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>256 → 128 Ch., Stride 2</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>128 → 128 Ch., Stride 2</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>128 → 256 Ch., Stride 2</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 256 Ch.</td>
</tr>
<tr>
<td>Batch normalization</td>
<td></td>
</tr>
</tbody>
</table>

2.0.5 Level 5 (Fully conn. 512)

\[
8 \times \left\{ \begin{array}{l}
\text{Cond. coupling block} \\
\text{Random permutation}
\end{array} \right. \\
\text{Result is } z^{(5)}
\]

**Conditional coefficient functions:**

| Fully connected | 256 + 512 → 512 |
| ReLU | |
| Fully connected | 512 → 512 |
| ReLU | |
| Fully connected | 512 → 512 |
| ReLU | |
| Fully connected | 512 → 512 |

**Conditioning head (shared for all coupling blocks):**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>256 → 128 Ch., Stride 2</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>128 → 256 Ch., Stride 2</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 256 Ch., Stride 2</td>
</tr>
<tr>
<td>Leaky ReLU</td>
<td>Slope 10^{-2}</td>
</tr>
<tr>
<td>Convolution</td>
<td>256 → 512 Ch., Stride 2</td>
</tr>
<tr>
<td>Global average pool</td>
<td></td>
</tr>
<tr>
<td>Batch normalization</td>
<td></td>
</tr>
</tbody>
</table>

3 Colorization – Ablations

In the following, we compare the following ablated versions of the cINN architecture:

1. No conditioning network at all, the grayscale image is used directly as conditioning input (**No c.n.**).

2. The same VGG-like conditioning network as in the main paper, but it is only pretrained and then left fixed during the training of the cINN (**Fixed c.n.**).

3. Both cINN and conditioning network are trained jointly, as in the main paper.

In general, we observe that it is critical that the conditioning network and the cINN are trained jointly. With a pretrained but fixed conditioning network, the generated images lack global coherence and exhibit exaggerated variance and artifacts.

The quantitative scores for the ablated versions are as follows, computed in the same way as Table 1 in the main paper.

<table>
<thead>
<tr>
<th></th>
<th>No c.n.</th>
<th>Fixed c.n.</th>
<th>Jointly trained</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE best of 8</td>
<td>4.06±0.04</td>
<td>3.37±0.04</td>
<td>3.53±0.04</td>
</tr>
<tr>
<td>Variance</td>
<td>41.1±0.3</td>
<td>58.0±0.4</td>
<td>35.2±0.3</td>
</tr>
<tr>
<td>FID</td>
<td>29.5±0.5</td>
<td>32.4±0.5</td>
<td>25.1±0.3</td>
</tr>
<tr>
<td>VGG top 5</td>
<td>82.9±0.5</td>
<td>80.8±0.6</td>
<td>85.0±0.5</td>
</tr>
</tbody>
</table>
No conditioning network

Fixed conditioning network

Jointly trained conditioning network

No conditioning network

Fixed conditioning network

Jointly trained conditioning network

No conditioning network

Fixed conditioning network

Jointly trained conditioning network

No conditioning network

Fixed conditioning network

Jointly trained conditioning network
4 Colorization – Interpolations

In the following, we show 2-dimensional interpolations in latent space. Two random latent vectors $z^{(1)}, z^{(2)}$ are linearly combined:

$$z^* = a_1 z^{(1)} + a_2 z^{(2)}$$

with varying $a_1, a_2 \in [-0.9, 0.9]$ across each axis of a grid. The center image has $z^* = 0$. Note that the images in the corners have a larger magnitude than trained for, $\|z^*\|_2 \approx 1.3 \mathbb{E}[\|z\|_2]$, leading to some oversaturation artifacts, as in Fig. 12 of the main paper.
5  Colorization – Additional examples

On the following pages, we provide some additional colorized images, as well as comparisons to alternative methods. All images are taken from the ImageNet 2012 validation set, and all methods were trained on ImageNet 2012. As we do not observe any significant diversity for the cGAN, we only provide a single sample.

5.1  General examples
<table>
<thead>
<tr>
<th>Ground truth</th>
<th>CNN</th>
<th>cGAN</th>
<th>cINN (ours)</th>
<th>cVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grayscale</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ground truth</th>
<th>CNN</th>
<th>cGAN</th>
<th>cINN (ours)</th>
<th>cVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grayscale</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ground truth</th>
<th>CNN</th>
<th>cGAN</th>
<th>cINN (ours)</th>
<th>cVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grayscale</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

13
Ground truth | CNN | cGAN | cINN (ours) | cVAE
---|---|---|---|---
Grayscale | | | | |

Ground truth | CNN | cGAN | cINN (ours) | cVAE
---|---|---|---|---
Grayscale | | | | |

Ground truth | CNN | cGAN | cINN (ours) | cVAE
---|---|---|---|---
Grayscale | | | | |

15
<table>
<thead>
<tr>
<th>Ground truth</th>
<th>CNN</th>
<th>cGAN</th>
<th>cINN (ours)</th>
<th>cVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grayscale</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

16
<table>
<thead>
<tr>
<th>Ground truth</th>
<th>CNN</th>
<th>cGAN</th>
<th>cINN (ours)</th>
<th>cVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grayscale</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

17
5.2 Humans

We find that the cINN often has difficulties generating convincing skin colors, as shown below. Clothing is colored in diverse ways, but not always with the correct connectivity and consistency.
5.3 Lacking consistency

The following failure cases exhibit a lack in consistency, in occluded objects, multi-part objects, or reflections.
### 5.4 Color ignores semantic content

In the following examples, the semantic content of the image was not recognized, and the generated colors are clearly incorrect.
Ground truth | CNN | cGAN | cINN (ours) | cVAE
--- | --- | --- | --- | ---
Grayscale | Grayscale | Grayscale | Grayscale | Grayscale
Ground truth | CNN | cGAN | cINN (ours) | cVAE
--- | --- | --- | --- | ---
Grayscale | Grayscale | Grayscale | Grayscale | Grayscale
Ground truth | CNN | cGAN | cINN (ours) | cVAE
--- | --- | --- | --- | ---
Grayscale | Grayscale | Grayscale | Grayscale | Grayscale
5.5 Outright failures

For the following images, the cINN fails completely, and generates colors with seemingly little or no connection to the grayscale image.
<table>
<thead>
<tr>
<th>Ground truth</th>
<th>CNN</th>
<th>cGAN</th>
<th>cINN (ours)</th>
<th>cVAE</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Ground truth" /></td>
<td><img src="image2" alt="CNN" /></td>
<td><img src="image3" alt="cGAN" /></td>
<td><img src="image4" alt="cINN (ours)" /></td>
<td><img src="image5" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image6" alt="Ground truth" /></td>
<td><img src="image7" alt="CNN" /></td>
<td><img src="image8" alt="cGAN" /></td>
<td><img src="image9" alt="cINN (ours)" /></td>
<td><img src="image10" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image11" alt="Ground truth" /></td>
<td><img src="image12" alt="CNN" /></td>
<td><img src="image13" alt="cGAN" /></td>
<td><img src="image14" alt="cINN (ours)" /></td>
<td><img src="image15" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image16" alt="Ground truth" /></td>
<td><img src="image17" alt="CNN" /></td>
<td><img src="image18" alt="cGAN" /></td>
<td><img src="image19" alt="cINN (ours)" /></td>
<td><img src="image20" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image21" alt="Ground truth" /></td>
<td><img src="image22" alt="CNN" /></td>
<td><img src="image23" alt="cGAN" /></td>
<td><img src="image24" alt="cINN (ours)" /></td>
<td><img src="image25" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image26" alt="Ground truth" /></td>
<td><img src="image27" alt="CNN" /></td>
<td><img src="image28" alt="cGAN" /></td>
<td><img src="image29" alt="cINN (ours)" /></td>
<td><img src="image30" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image31" alt="Ground truth" /></td>
<td><img src="image32" alt="CNN" /></td>
<td><img src="image33" alt="cGAN" /></td>
<td><img src="image34" alt="cINN (ours)" /></td>
<td><img src="image35" alt="cVAE" /></td>
</tr>
<tr>
<td><img src="image36" alt="Ground truth" /></td>
<td><img src="image37" alt="CNN" /></td>
<td><img src="image38" alt="cGAN" /></td>
<td><img src="image39" alt="cINN (ours)" /></td>
<td><img src="image40" alt="cVAE" /></td>
</tr>
</tbody>
</table>