Supplement: Constrained Generative Adversarial Networks for Interactive Image Generation

Eric Heim
Air Force Research Laboratory
Information Directorate
Rome, NY USA
eric.heim.1@us.af.mil

1. Formal Definition of LSTM Component

Equation (4) is a standard LSTM cell:

$$
\begin{aligned}
LSTM\left(\mathbf{z}, \mathbf{q}_{t-1}^{*}\right) &= \mathbf{o}_t * \tanh\left(\mathbf{h}_t\right), \text{where} \\
\mathbf{o}_t &= \sigma\left(\mathbf{w}_{\mathbf{o}} \cdot \left[\mathbf{q}_{t-1}^{*}, \mathbf{z}\right] + \mathbf{b}_{\mathbf{o}}\right) \\
\mathbf{h}_t &= \mathbf{f}_t * \mathbf{h}_{t-1} + \mathbf{i}_t * \tilde{\mathbf{h}}_t \\
\mathbf{f}_t &= \sigma\left(\mathbf{w}_{\mathbf{f}} \cdot \left[\mathbf{q}_{t-1}^{*}, \mathbf{z}\right] + \mathbf{b}_{\mathbf{f}}\right) \\
\mathbf{i}_t &= \sigma\left(\mathbf{w}_{\mathbf{i}} \cdot \left[\mathbf{q}_{t-1}^{*}, \mathbf{z}\right] + \mathbf{b}_{\mathbf{i}}\right) \\
\tilde{\mathbf{h}}_t &= \tanh\left(\mathbf{w}_{\tilde{\mathbf{h}}} \cdot \left[\mathbf{q}_{t-1}^{*}, \mathbf{z}\right] + \mathbf{b}_{\tilde{\mathbf{h}}}\right)
\end{aligned}
$$

Here, $\mathbf{z}$ is used as what is commonly referred to as "input" to the LSTM, $\mathbf{q}_{t-1}^{*}$ is commonly called the "hidden state" of the previous iteration, and $LSTM$ returns the hidden state of the current iteration.

2. Neural Network Architectures used in Experiments

In this section, we outline the neural network architecture used in all experiments in the main paper, layer by layer. Rows of the network in descending order (top to bottom) indicate layers from input to ouput. The following naming conventions are used throuout. "Conv" indicates a convolutional layer, "FC" indicates a fully connected layer, and "TConv" indicates a transpose convolutional layer. The column labeled "Ker" indicates the kernel size, "Str" indicates stride, and "Act" indicates the activation function used. Columns labeled "In" and "Out" indicate the shape of the input to the layer and shape of the output of the layer.

2.1. MNIST Experiments

Below you will find architecture descriptions for the networks used in the MNIST experiments. Note that after each two convolutional or transpose convolutional layers in all networks, layer normalization is used.

ϕ Network (Encoder)					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	32x32x1	3x3	1	ReLU	32x32x4
Conv	32x32x4	3x3	2	ReLU	16x16x8
Conv	16x16x8	3x3	2	ReLU	8x8x16
Conv	8x8x16	3x3	2	ReLU	4x4x32
Conv	4x4x32	3x3	2	ReLU	2x2x64
FC	2x2x64			None	2

ϕ Network (Decoder)					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
FC	2			None	2x2x64
TConv	2x2x64	3x3	2	ReLU	4x4x32
TConv	4x4x32	3x3	2	ReLU	8x8x16
TConv	8x8x16	3x3	2	ReLU	16x16x8
TConv	16x16x8	3x3	2	ReLU	32x32x4
Conv	32x32x4	3x3	1	tanh	32x32x1

Discriminator Network (WGAN and CONGAN)					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	32x32x1	3x3	1	ReLU	32x32x64
Conv	32x32x64	3x3	2	ReLU	16x16x128
Conv	16x16x128	3x3	2	ReLU	8x8x256
Conv	8x8x256	3x3	2	ReLU	4x4x512
FC	4x4x512			None	1

Read CNN					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	32x32x1	5x5	1	ReLU	32x32x2
Conv	32x32x2	5x5	2	ReLU	16x16x4
Conv	16x16x4	5x5	2	ReLU	8x8x8
Conv	8x8x8	5x5	2	ReLU	4x4x16
Conv	4x4x16	5x5	2	ReLU	2x2x32
FC	2x2x32			tanh	64

CONGAN Write Network/WGAN Generator					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
FC	64			None	4x4x512
TConv	4x4x512	3x3	2	ReLU	8x8x256
Conv	8x8x256	3x3	1	ReLU	8x8x256
TConv	8x8x256	3x3	2	ReLU	16x16x128
Conv	16x16x128	3x3	1	ReLU	16x16x128
TConv	16x16x128	3x3	2	ReLU	32x32x64
Conv	32x32x64	3x3	1	tanh	32x32x1

2.2. CelebA and Zappos50K Experiments

In this section, we first describe all network architectures used in both the CelebA and Zappos50K experiments. Then we outline the ϕ networks used for each. Here, "Norm" indicates layer norm, "ReLU" indicates the application of a rectified linear unit. The "ID" column is used to identify which layers are used in subsequent operations in the residual block. For the residual blocks, the "In" column is either used to indicate the size of the input or the IDs of the layers used as input. The "Add" layers are simply the addition of the two layers identified in the "In" column with the first ID multiplied by 0.3 before the addition. The "RB↑" layer is a residual block up and "RB↓" is a residual block down.

Discriminator Network					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	64x64x3	3x3	1	ReLU	64x64x64
RB↓	64x64x64				32x32x128
RB↓	32x32x128				16x16x256
RB↓	16x16x256				8x8x512
RB↓	8x8x512				4x4x512
FC	4x4x512			None	1

Read CNN					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	64x64x3	3x3	1	ReLU	64x64x8
RB↓	64x64x8				32x32x16
RB↓	32x32x16				16x16x32
RB↓	16x16x32				8x8x32
FC	8x8x32			tanh	1

CONGAN Write Network/WGAN Generator					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
FC	128			ReLU	4x4x512
RB↑	4x4x512				8x8x512
RB↑	8x8x512				16x16x256
RB↑	16x16x256				32x32x128
RB↑	32x32x128				64x64x64
Conv	64x64x64	3x3	1	tanh	64x64x3

Residual Block (Down)						
ID	**Layer**	**In**	**Ker**	**Str**	**Act**	**Out**
1	Conv	$axbxc$	5x5	2	None	$\frac{a}{2}$x$\frac{b}{2}$xd
2	Conv	$axbxc$	5x5	1	None	$axbxc$
3	Norm	(2)				
4	ReLU	(3)				
5	Conv	(4)	5x5	2	None	$\frac{a}{2}$x$\frac{b}{2}$xd
6	Add	(5), (1)				
7	Norm	(6)				
8	ReLU	(7)				

Residual Block (Up)						
ID	**Layer**	**In**	**Ker**	**Str**	**Act**	**Out**
1	TConv	$axbxc$	5x5	2	None	$(2*a)$x $(2*b)$x d
2	Conv	$axbxc$	5x5	1	None	$axbxc$
3	Norm	(2)				
4	ReLU	(3)				
5	TConv	(4)	5x5	2	None	$(2*a)$x $(2*b)$x d
6	Add	(5), (1)				
7	Norm	(6)				
8	ReLU	(7)				

2.3. Celeba ϕ MCNN

The MCNN we developed for the ϕ network in our CelebA experiments takes an image, and puts it through a "base" network. Then the output of the base network is input to twelve"specialized" networks to predict the presence or absence of each of the twelve attributes we used in our experiment. Each of these architectures are outlined below.

ϕ MCNN Network (Base)					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	64x64x3	7x7	2	ReLU	32x32x64
Conv	32x32x64	5x5	2	ReLU	16x16x128
Norm					

ϕ MCNN Network (Specialized)					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	16x16x128	3x3	2	ReLU	8x8x256
Conv	8x8x256	3x3	2	ReLU	4x4x512
Norm					
Conv	4x4x512	3x3	2	ReLU	2x2x1024
FC	2x2x1024			sigm	1

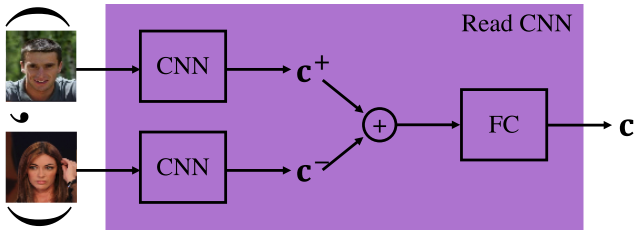

Figure 1: The read network to map a constraint to a vector.

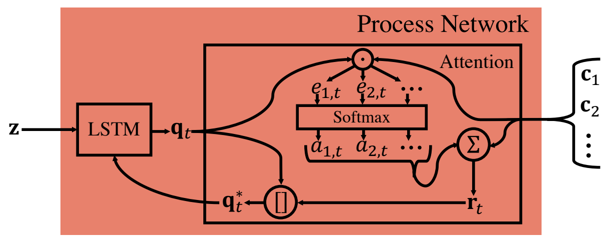

Figure 2: Illustration of the tth iteration of the process network, beginning with the LSTM unit and ending with q_t^*.

2.4. Zappos50K ϕ Triplet Network

A triplet network takes three images and puts them through the same network resulting in an n dimensional embedding for which standard triplet losses can be applied. Below describes the network we used in our Zappos50K experiments. Note that after each two convolutional layers, layer normalization is applied.

ϕ Triplet Network					
Layer	**In**	**Ker**	**Str**	**Act**	**Out**
Conv	64x64x3	5x5	1	ReLU	64x64x8
Conv	64x64x8	5x5	2	ReLU	32x32x8
Conv	32x32x8	5x5	1	ReLU	32x32x16
Conv	32x32x16	5x5	2	ReLU	16x16x16
Conv	16x16x16	5x5	1	ReLU	16x16x32
Conv	16x16x32	5x5	2	ReLU	8x8x32
Conv	8x8x32	5x5	1	ReLU	8x8x64
Conv	8x8x64	5x5	2	ReLU	4x4x64
FC	4x4x64			None	2

3. CelebA ϕ MCNN Training Details and Performance

For training the ϕ MCNN used in the CelebA data experiments, we chose twelve attributes for the network to predict. We used the Adam optimization method with default parameters, a batch size of 32, and trained the model for 100,000 iterations. The test accuracy of the network for the twelve attributes is shown in the table below. We note that these results are slightly worse than those reported in the original paper, but sufficient for the CONGAN generator to learn how to manipulate images. Performance can be increased by employing the "aux" method described in the original MCNN paper, and by designing the architecture to be take advantage of groups of common attributes.

Attribute	Accuracy
Bald	0.9836
Black Hair	0.8870
Blond Hair	0.9414
Brown Hair	0.8242
Eyeglasses	0.9901
Goatee	0.9531
Gray Hair	0.9709
Male	0.9760
Mustache	0.9557
No Beard	0.9360
Pale Skin	0.9601
Wearing Hat	0.9832

4. Zappos50K ϕ Triplet Network Training Details and Performance

We formed the training set for the triplet network by first taking each image in the Zappos50K train set, and placed it into one of the 64 color histogram bins according their highest histogram value. To form each triplet (A, B, C) ("A is more similar to C than C"), we iterated over each bin j, selecting images A and B randomly from j, and image C randomly from another bin. We iterated over each bin 5000 times creating 320,000 triplets for training. We did a similar process for the test set, but with 1000 "passes" over each bin,, making a test set of 64,000 triplets.

We trained the network using default Adam optimization parameters and a batch size of 128. We found that loss leveled out around 25,000 steps and stopped optimization at that point. Upon convergence, the network was able to satisfy 94.504% of the test triplets. Figure 3 shows samples of the Zappos50K data set embedded in two dimensions using the ϕ triplet network.

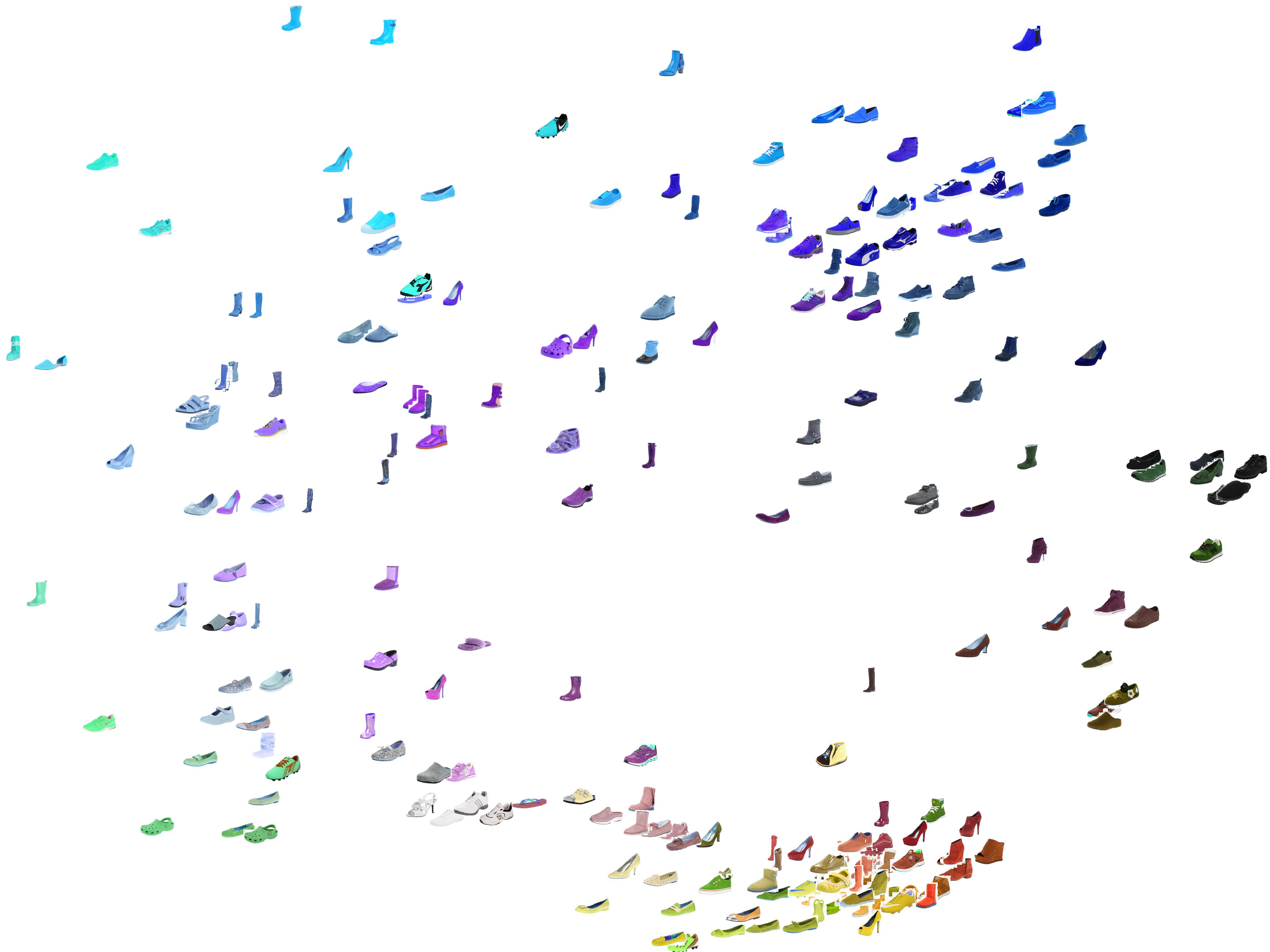

Figure 3: Samples from the Zappos50K data set embedded using the ϕ triplet network.