Unsupervised Program Synthesis for Images By Sampling Without Replacement Supplementary Material

Chenghui Zhou[1] **Chun-Liang Li**[1] **Barnabás Póczos**[1]

[1]Machine Learning Department, Carnegie Mellon University , Pittsburgh, Pennsylvania, USA
{chenghuz,chunlial,bapoczos}@cs.cmu.edu

A GRAMMAR TREE LSTM GUIDE

This section provides a guide for the tree-LSTM illustration in Figure 1. This guide follows the arrows in the illustration (Figure 1) from left to right:

- A grammar stack with a start token S and an end token \$ as well as an empty image stack is initialized.
- In the first iteration, the token S is popped out. Following Rule (1), all other options will be masked except E, the only possible output. E token is added to the stack.
- In the second iteration, or any iteration where the token E is popped, the input for all examples and all softmax outputs are masked except the entries representing EET and P according to Rule (2). If EET is sampled, T, E and E tokens will be added to the stack separately in that exact order to expand the program further. If P is sampled, it will be added to the stack and the program cannot expand further.
- If T is popped out of the stack, the output space for that iteration will be limited to all the operations (Rule (3)). Similarly, if P is popped out, the output space is limited to all the geometric shapes (Rule (4)).
- When a shape token is sampled, it will not be added to the grammar stack as they do not contribute to the program structure. Instead, the image of the shape will be pushed onto the corresponding image stack.
- When an operation token is sampled, it also will not be added to the grammar stack. Instead, we pop out the top two images to apply the operation on them and push the final image onto the image stack again.
- When the stack has popped out all the added tokens, the end token \$ will be popped out in the last iteration. We then finish the sampling as standard RNN language models.

In practice, we implement the masking mechanism by adding a vector to the output before passing into logsoftmax layer to get the probability. The vector contains 0 for valid output and large negative numbers for invalid ones. This makes sure that invalid options will have almost zero probability of being sampled. The input of the RNN cell includes encoded target image and intermediate images from the image stack, embedded pop-out token from grammar stack and the hidden state from the RNN's last iteration. The exact algorithm is in Algorithm 3.

B SAMPLING WITHOUT REPLACEMENT

This section describes how we achieve sampling without replacement with the help of stochastic beam search [Kool et al., 2019b].

At each step of generation, the algorithm chooses the top-$k+1$ beams to expand based on the $\vec{\mathbf{G}}_{\phi_{i,j}}$ score at time step j in order to find the top-k stochastic sequences at the end. The use for the additional beam will be explained later. Let A be all possible actions at time step j, $\vec{\phi}_{i,j} \in \mathbb{R}^A$ is the log probability of each outcomes of sequence i at time j plus the log probability of the previous $j-1$ actions.

$$\begin{aligned}\vec{\phi}_{i,j} =& \big[\log P(a_1), \log P(a_2), \ldots, \log P(a_A)\big] + \\ & \log P(a_{t_1}, a_{t_2}, \cdots, a_{t_{j-1}}) \cdot \vec{\mathbb{1}} \\ =& \big[\log P(a_1), \log P(a_2), \ldots, \log P(a_A)\big] + \phi_{i,j-1} \cdot \vec{\mathbb{1}}\end{aligned} \tag{1}$$

For each beam, we sample a Gumbel random variable $G_{\phi_{i,j,a}} = \text{Gumbel}(\phi_{i,j,a})$ for each of the element a of the vector $\vec{\phi}_{i,j}$. Then we need to adjust the Gumbel random variable by conditioning on its parent's adjusted stochastic score $G_{s^i_{j-1}}$ being the largest (Equation 2) in relation to all the descendant elements in $\vec{\mathbf{G}}_{\phi_{i,j}}$, the resulting value $\vec{\tilde{\mathbf{G}}}_{\phi_{i,j}} \in \mathbb{R}^A$ is the adjusted stochastic score for each of the potential expansions.

$$\vec{\tilde{\mathbf{G}}}_{\phi_{i,j}} = -\log(\exp(-\vec{\mathbb{1}} \cdot G_{s^i_{j-1}}) - \exp(-\vec{\mathbb{1}} \cdot Z_{i,j}) + \exp(-\vec{\mathbf{G}}_{\phi_{i,j}})) \quad (2)$$

Here $Z_{i,j}$ is the largest value in the vector $\vec{\mathbf{G}}_{\phi_{i,j}} = [G_{\phi_{i,j,a_1}}, G_{\phi_{i,j,a_2}}, \cdots, G_{\phi_{i,j,a_A}}]$ and $G_{s^i_{j-1}}$ is the adjusted stochastic score of i-th beam at step $j-1$ from the last iteration. Conditioning on the parent stochastic score being the largest in this top-down sampling scheme makes sure that each leaf's final stochastic score G_{s^i} is independent, equivalent to sampling the sequences bottom up without replacement [Kool et al., 2019b].

Once we have aggregated all the adjusted stochastic scores in $\vec{\tilde{\mathbf{G}}}_{\phi_{i,j}}$ from all previous $k+1$ beams, we select the top-$k+1$ scored beams from $(k+1) * A$ scores for expansions. The selected $k+1$ adjusted stochastic scores become the $G_{s^i_j}$ $\forall i$ in the new iteration. Note that the reason that we maintain one more beam than we intended to expand because we need the $k+1$ largest stochastic score to be the threshold during estimation of the entropy and REINFORCE objective. This is explained next. Please refer to Algorithm 2 for details in the branching process.

The sampling without replacement algorithm requires importance weighting of the objective functions to ensure unbiasness. The weighting term is $\frac{p_\theta(s^i)}{q_{\theta,\kappa}(s^i)}$. $p_\theta(s^i)$ represents the probability of the sequence s^i (s^i is the i-th completed sequence and $p_\theta(s^i) = \exp\phi_i$) and S represents the set of all sampled sequences s^i for $i = 1, 2, \cdots, k$. $q_{\theta,\kappa}(s^i_j) = P(G_{s^i_j} > \kappa) = 1 - \exp(-\exp(\phi_{i,j} - \kappa))$, where κ is the $(k+1)$-th largest $G_{s^i_j}$ score for all i and $\phi_{i,j} = \log P(a_{t_1}, a_{t_2}, \cdots, a_{t_j})$ is the log likelihood of the first j actions of i-th sequence, can be calculated based on the CDF of the Gumbel distribution. It acts as a threshold for branching selection. We can use the log probability of the sequence here to calculate the CDF because the adjust stochastic scores $G_{s^i_j}$'s are equivalent to the Gumbel scores of sequences sampled without replacement from bottom up. During implementation, we need to keep an extra beam, thus $k+1$ beams in total, to accurately estimate κ in order to ensure the unbiasness of the estimator.

To reduce variance of our objective function, we introduce additional normalization terms as well as a baseline. However, the objective function is biased with these terms. The normalization terms are $W(S) = \sum_{s^i \in S} \frac{p_\theta(s^i)}{q_{\theta,\kappa}(s^i)}$ and $W^i(S) = W(S) - \frac{p_\theta(s^i)}{q_{\theta,\kappa}(s^i)} + p_\theta(s^i)$.

Incorporating a baseline into the REINFORCE objective is a standard practice. A baseline term is defined as $B(S) = \sum_{s^i \in S} \frac{p_\theta(s^i)}{q_{\theta,\kappa}(s^i)} f(s^i)$ and $f(s^i)$ should be the reward of the complete i-th program s^i in this case.

To put everything together, the exact objective is as follows [Kool et al., 2019a]:

$$\nabla_\theta \mathbb{E}_{s \sim p_\theta(s)}[f(s)] \approx \sum_{s^i \in S} \frac{1}{W^i(S)} \cdot \frac{\nabla_\theta p_\theta(s^i)}{q_{\theta,\kappa}(s^i_n)} (f(s^i) - \frac{B(S)}{W(S)}) \quad (3)$$

Entropy estimation uses a similar scaling scheme as the REINFORCE objective:

$$\hat{\mathcal{H}}_D(X_1, X_2, X_3, \cdots, X_n) \approx \sum_{j=1}^{n} \frac{1}{W_j(S)} \sum_{s^i \in S} \frac{p_\theta(s^i_j)}{q_{\theta,\kappa}(s^i_j)} \mathcal{H}(X_j | X_1 = x^i_1, \cdots, X_{j-1} = x^i_{j-1}) \quad (4)$$

where $W_j(S) = \sum_{s^i \in S} \frac{p_\theta(s^i_j)}{q_{\theta,\kappa}(s^i_j)}$ and s^i_j denotes the first j elements of the sequence s^i. The estimator is unbiased excluding the $\frac{1}{W_j(S)}$ term.

C PROOF OF STEPWISE ENTROPY ESTIMATION'S UNBIASNESS

Entropy of a sequence can be decomposed into the sum of the conditional entropy at each step conditioned on the previous values. This is also called the chain rule for entropy calculation. Let $X_1, X_2, \cdots, X_n$ be drawn from $P(X_1, X_2, \cdots, X_n)$ [Cover and Thomas, 2012]:

$$\mathcal{H}(X_1, X_2, \cdots, X_n) = \sum_{j=1}^{n} \mathcal{H}(X_j | X_1, \cdots, X_{j-1}) \quad (5)$$

If we sum up the empirical entropy at each step after the softmax output, we can obtain an unbiased estimator of the entropy. Let S be the set of sequences that we sampled and each sampled sequence s^i consists of $X_1, X_2, \cdots, X_n$:

$$\begin{aligned}
&\mathbb{E}_{X_1,\ldots,X_{j-1}}(\hat{\mathcal{H}}_D) \\
&= \mathbb{E}_{X_1,\ldots,X_{j-1}}(\frac{1}{|S|} \sum_{i \in |S|} \sum_{j=1}^{n} \mathcal{H}(X_j | X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&= \frac{1}{|S|} \cdot |S| \sum_{j=1}^{n} \mathcal{H}(X_j | X_1, \cdots, \quad X_{j-1}) \\
&= \mathcal{H}(X_1, X_2, \cdots, X_n)
\end{aligned}$$

In order to incorporate the stepwise estimation of the entropy into the beam search, we use the similar reweighting scheme as the REINFORCE objective. The difference is that the REINFORCE objective is reweighted after obtaining the full sequence because we only receive the reward at the end and here we reweight the entropy at each step. We denote each time step by j and each sequence by i, the set of sequences selected at time step j is S_j and the complete set of all possible sequences of length j is T_j and $S_j \in T_j$. We are taking the expectation of the estimator over the $G_{\phi_{i,j}}$ scores. As we discussed before, at each step, each potential beam receives a stochastic score $G_{\phi_{i,j}}$. The beams associated with the top-$k+1$ stochastic scores are chosen to be expanded further and κ is the $k+1$-th largest $G_{\phi_{i,j}}$. κ can also be seen as a threshold in the branching selection process and $q_{\theta,\kappa}(s^i_j) = P(G_{s^i_j} > \kappa) = 1 - \exp(-\exp(\phi_{i,j} - \kappa))$. For details on the numerical stable implementation of $q_{\theta,\kappa}(s^i_j)$, please refer to [Kool et al., 2019b].

$$\begin{aligned}
&\mathbb{E}_{G_\phi}\Big(\sum_{j=1}^{n}\sum_{s^i_j \in S_j} \frac{p_\theta(s^i_j)}{q_{\theta,\kappa}(s^i_j)} \mathcal{H}(X_j|X_1 = x^i_1, X_2 = x^i_2, \cdots, \\
&\quad X_{j-1} = x^i_{j-1})\Big) \\
&= \sum_{j=1}^{n} \mathbb{E}_{G_\phi}\Big(\sum_{i\in|T_j|} \frac{p_\theta(s^i_j)}{q_{\theta,\kappa}(s^i_j)} \mathcal{H}(X_j|X_1 = x^i_1, X_2 = x^i_2, \\
&\quad \cdots, X_{j-1} = x^i_{j-1})) \mathbb{1}_{\{x^i_1,\cdots,x^i_j\} \in S_j}\Big) \\
&= \sum_{j=1}^{n}\sum_{i\in|T_j|} p_\theta(s^i_j) \mathcal{H}(X_j|X_1 = x^i_1, X_2 = x^i_2, \cdots, \\
&\quad X_{j-1} = x^i_{j-1}) \mathbb{E}_{G_\phi}\Big(\frac{\mathbb{1}_{\{s^i_j = x^i_1,\cdots,x^i_j\}\in S_j}}{q_{\theta,\kappa}(s^i_j)}\Big) \\
&= \sum_{j=1}^{n} \mathcal{H}(X_j|X_1, X_2, \cdots, X_{j-1}) \cdot 1 \\
&= \mathcal{H}(X_1, X_2, \cdots, X_n)
\end{aligned}$$

For the proof of $\mathbb{E}_{G_\phi}(\frac{\mathbb{1}_{\{s^i_j \in S_j}}{q_{\theta,\kappa}(s^i_j)}) = 1$, please refer to [Kool et al., 2019b], appendix D.

D PROOF OF LOWER VARIANCE OF THE STEPWISE ENTROPY ESTIMATOR

We will continue using the notations from above. We want to compare the variance of the two entropy estimator $\hat{\mathcal{H}}$ and the stepwise entropy estimator $\hat{\mathcal{H}}_D$ and show that the second estimator has lower variance.

Proof. We abuse $\mathbb{E}_{X_j}$ to be $\mathbb{E}_{X_j|X_1,\ldots,X_{j-1}}$ and Var_{X_j} to be $\mathrm{Var}_{X_j|X_1,\ldots,X_{j-1}}$ to simplify the notations.

$$\begin{aligned}
&\mathrm{Var}_{X_1,X_2,\cdots,X_n}\Big(\frac{1}{|S|}\sum_{i\in|S|}\sum_{j=1}^{n} \mathcal{H}(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})\Big) \\
&= \frac{1}{|S|^2}\sum_{i\in|S|}\sum_{j=1}^{n} \mathrm{Var}_{X_1,X_2,\cdots,X_n}(\mathcal{H}(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&= \frac{1}{|S|^2}\sum_{i\in|S|}\sum_{j=1}^{n} (\mathbb{E}(\mathcal{H}^2(X_j|X_1 = x^i_1, \ldots, X_{j-1} = x^i_{j-1})) \\
&\quad - \mathbb{E}^2(\mathcal{H}(X_j|X_1 = x^i_1, \ldots, X_{j-1} = x^i_{j-1}))) \\
&= \frac{1}{|S|^2}\sum_{i\in|S|}\sum_{j=1}^{n} (\mathbb{E}_{X_1,\cdots,X_{j-1}}\mathbb{E}^2_{X_j}(\log P(X_j|X_1 = x^i_1, \\
&\quad \ldots, X_{j-1} = x^i_{j-1})) \\
&\quad - \mathbb{E}^2_{X_1,\cdots,X_{j-1}}\mathbb{E}_{X_j}(\log P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1}))) \\
&= \frac{1}{|S|^2}\sum_{s^i\in S}\sum_{j=1}^{n} (\mathbb{E}_{X_1,\cdots,X_{j-1}}(\mathbb{E}_{X_j}(\log^2 P(X_j|X_1 = x^i_1, \\
&\quad \ldots, X_{j-1} = x^i_{j-1})) \\
&\quad - \mathrm{Var}_{X_j}(\log P(X_j|X_1 = x^i_1, \ldots, X_{j-1} = x^i_{j-1}))) \\
&\quad - \mathbb{E}^2_{X_1,\cdots,X_{j-1}}\mathbb{E}_{X_j}(\log P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1}))) \\
&= \frac{1}{|S|^2}\sum_{i\in|S|}\sum_{j=1}^{n} (\mathbb{E}_{X_1,\cdots,X_j}(\log^2 P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&\quad - \mathbb{E}^2_{X_1,\cdots,X_j}(\log P(X_j|X_1 = x^i_1, \ldots, X_{j-1} = x^i_{j-1}))) \\
&\quad - \mathbb{E}_{X_1,\cdots,X_{j-1}}\mathrm{Var}_{X_j}(\log P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&= \frac{1}{|S|^2}\sum_{i\in|S|}\sum_{j=1}^{n} (\mathrm{Var}_{X_1,\cdots,X_j}(\log P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&\quad - \mathbb{E}_{X_1,\cdots,X_{j-1}}\mathrm{Var}_{X_j}(\log P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&\le \frac{1}{|S|^2}\sum_{i\in|S|}\sum_{j=1}^{n} \mathrm{Var}_{X_1,\cdots,X_j}(\log P(X_j|X_1 = x^i_1, \ldots, \\
&\quad X_{j-1} = x^i_{j-1})) \\
&= \mathrm{Var}_{X_1,X_2,\cdots,X_n}\Big(\frac{1}{|S|}\sum_{i\in|S|} \log P(s^i)\Big)
\end{aligned}$$

□

The fifth equation holds from the fact that $\mathbb{E}^2_X \mathbb{E}_{Y|X}[f(X,Y)] = \mathbb{E}^2_{X,Y}[f(X,Y)]$. The result still stands after applying reweighting for the beam search.

E SHAPE ENCODING DEMONSTRATION

In Figure 1, we show the code name on top of the image that it represents. c, s, and t represent circle, square and triangle respectively. The first two numbers represent the position of

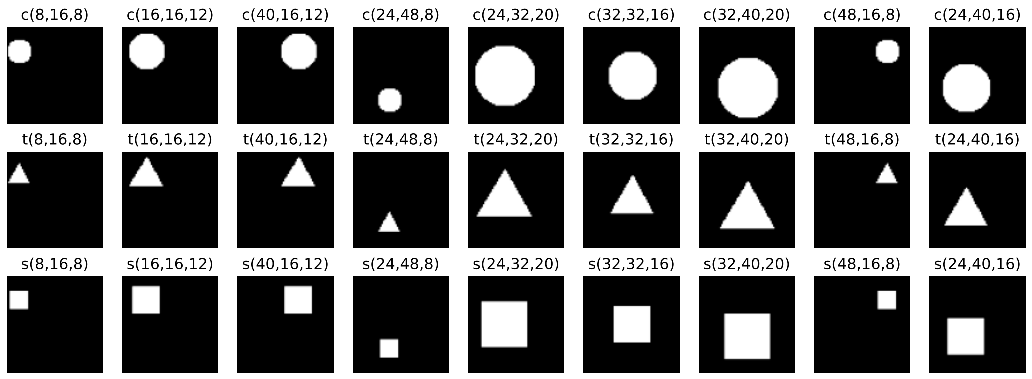
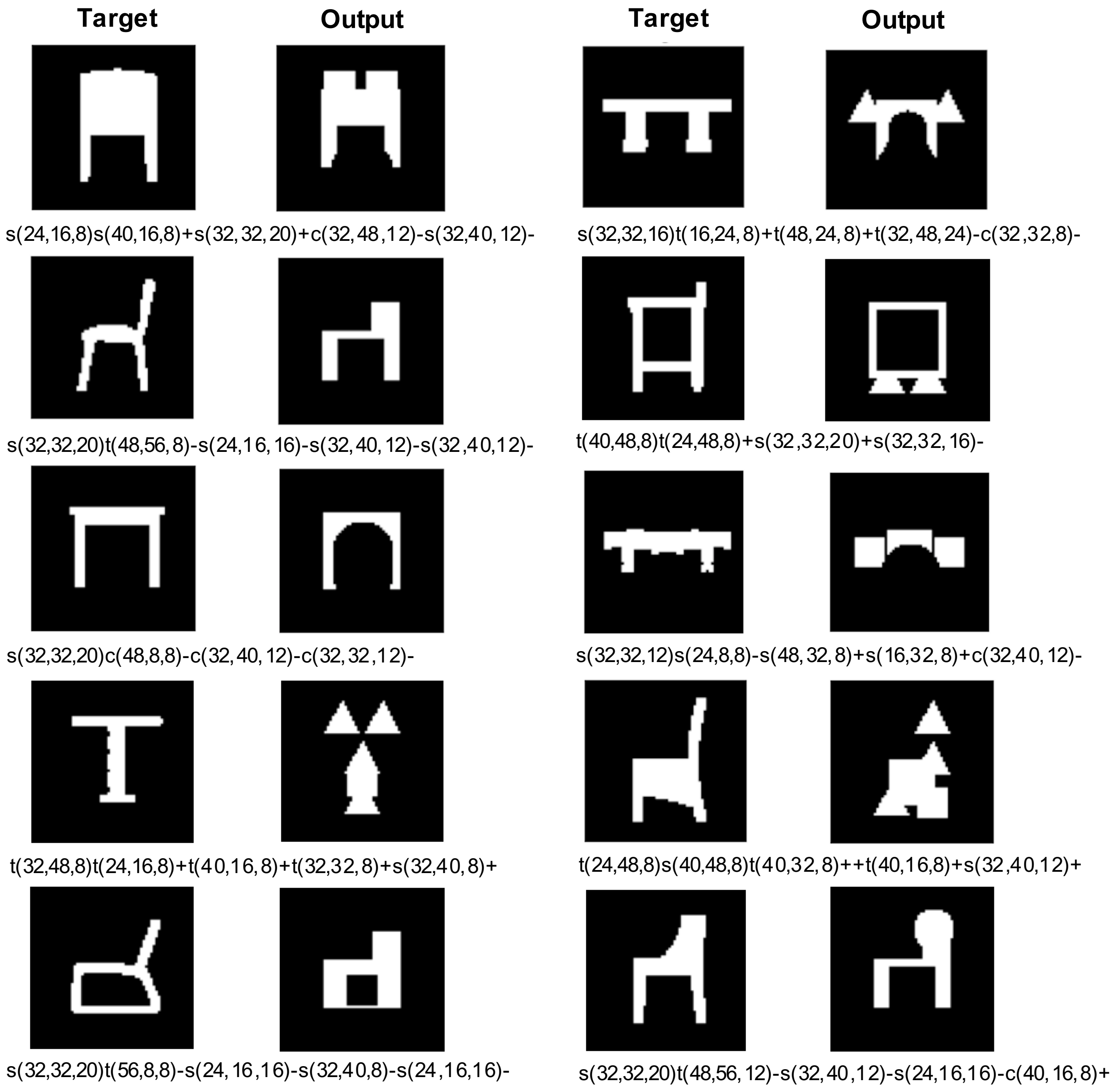


Figure 1: Each shape encoding is on top of the image it represents.



s(24,16,8)s(40,16,8)+s(32,32,20)+c(32,48,12)-s(32,40,12)-

s(32,32,16)t(16,24,8)+t(48,24,8)+t(32,48,24)-c(32,32,8)-

s(32,32,20)t(48,56,8)-s(24,16,16)-s(32,40,12)-s(32,40,12)-

t(40,48,8)t(24,48,8)+s(32,32,20)+s(32,32,16)-

s(32,32,20)c(48,8,8)-c(32,40,12)-c(32,32,12)-

s(32,32,12)s(24,8,8)-s(48,32,8)+s(16,32,8)+c(32,40,12)-

t(32,48,8)t(24,16,8)+t(40,16,8)+t(32,32,8)+s(32,40,8)+

t(24,48,8)s(40,48,8)t(40,32,8)++t(40,16,8)+s(32,40,12)+

s(32,32,20)t(56,8,8)-s(24,16,16)-s(32,40,8)-s(24,16,16)-

s(32,32,20)t(48,56,12)-s(32,40,12)-s(24,16,16)-c(40,16,8)+

Figure 2: Additional test output with corresponding programs. The odd-numbered columns contain the target images and the images to their right are example outputs.

the shape in the image and the last number represents the size.

F ADDITIONAL TEST OUTPUT EXAMPLES FOR THE 2D CAD DATASET

In this section, we include additional enlarged test outputs (Figure 2). We add the corresponding output program below each target/output pair. We observe that the algorithm approximates thin lines with triangles in some cases. Our hypothesis for the cause is the Chamfer Distance reward function being a greedy algorithm and it finds the matching distance based on the nearest features.