
Inferring Graphics Programs from Images

Anonymous Author(s)

Affiliation

Address

email

Abstract

1

2 1 Introducing visual programs

3 **Comment for reader: this paragraph is a little grandiose and goes beyond what we’ve actually**
4 **done. But these are the kinds of things that motivate the work, and I think that in some form**
5 **these ideas should be in the introduction or the conclusion.** How could an agent go from noisy,
6 high-dimensional perceptual input to a symbolic, abstract object, like a computer program? Here we
7 consider this problem within the context of *graphics program synthesis*. We develop an approach
8 for converting natural images, such as hand drawings, into executable source code for drawing the
9 original image. The source code captures not just the objects in the image, but also describes their
10 geometric relationships.

11 High dimensional perceptual input is ill matched to the abstract semantics of a programming language.
12 But programs with constructs like recursion or iteration produce a simpler *execution trace* of primitive
13 actions. Our hypothesis is that the execution trace of the program is better aligned with the perceptual
14 input, and that the trace can act as a kind of bridge between perception and programs. We test this
15 hypothesis by developing a model that learns to map from an image to the execution trace of the
16 graphics program that drew it. With the execution trace in hand, we can bring to bear techniques from
17 the program synthesis community to recover the latent graphics program.

18 In this work we consider programs that draw diagrams, similar to those found in papers.

19 We develop a hybrid architecture for inferring graphics programs. Our approach uses a deep network
20 infer an execution trace from an image; this recovers primitive drawing operations like lines, circles,
21 or arrows. For added robustness we use the deep network as a proposal distribution for a stochastic
22 search over execution traces. Finally we use techniques in the program synthesis community to
23 recover the program from its trace.

24 Each of these three components – the deep network, the stochastic search, the program synthesizer
25 – confers its own advantages. From the deep network we get a very fast system that can recover
26 plausible execution traces in about a minute. From the stochastic search we get added robustness;
27 essentially the stochastic search can correct mistakes made by the deep network’s proposals. From
28 the program synthesizer we get abstraction: our system recovers coordinate transformations, for
29 loops, and subroutines, which are useful for downstream tasks.

30 2 Neural architecture for inferring image parses

31 We developed a deep network architecture for efficiently inferring a execution trace, T , from an
32 image, I . Our model constructs the trace one drawing command at a time. When predicting the next
33 drawing command, the network takes as input the target image I as well as the rendered output of
34 previous drawing commands. Intuitively, the network looks at the image it wants to explain, as well

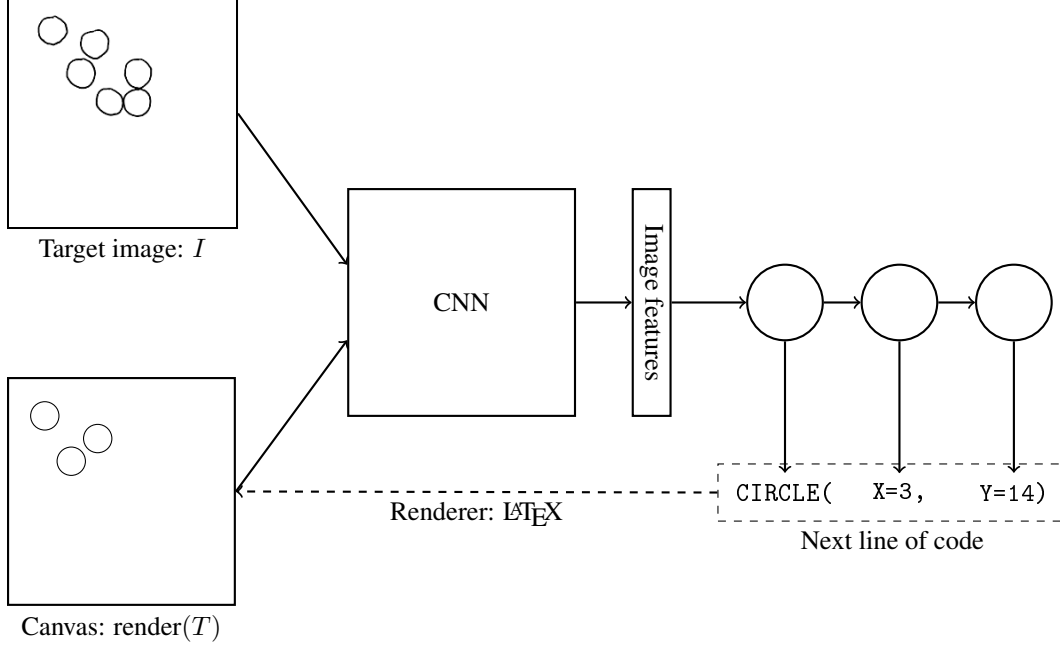


Figure 1: Our neural architecture for inferring the execution trace of a graphics program from its output.

as what it has already drawn. It then decides either to stop drawing or proposes another drawing command to add to the execution trace; if it decides to continue drawing, the predicted primitive is rendered to its “canvas” and the process repeats.

Figure 1 illustrates this architecture. We first pass the target image and a rendering of the trace so far to a convolutional network. Given the features extracted by the convolutional network, a multilayer perceptron then predicts a distribution over the next drawing command to add to the trace. We predict the drawing command token-by-token, and condition each token both on the image features and on the previously generated tokens. For example, the network first decides to emit the `CIRCLE` token conditioned on the image features, then it emits the x coordinate of the circle conditioned on the image features and the `CIRCLE` token, and finally it predicts the y coordinate of the circle conditioned on the image features, the `CIRCLE` token, and the x coordinate.

The distribution over the next drawing command factorizes:

$$\mathbb{P}_\theta[t_1 t_2 \cdots t_K | I, T] = \prod_{k=1}^K \mathbb{P}_\theta[t_k | f_\theta(I, \text{render}(T)), \{t_j\}_{j=1}^{k-1}] \quad (1)$$

where $t_1 t_2 \cdots t_K$ are the tokens in the drawing command, I is the target image, T is an execution trace, θ are the parameters of the neural network, and $f_\theta(\cdot, \cdot)$ is the image feature extractor (convolutional network). The distribution over execution traces factorizes as:

$$\mathbb{P}_\theta[T | I] = \prod_{n=1}^{|T|} \mathbb{P}_\theta[T_n | I, T_{1:(n-1)}] \times \mathbb{P}_\theta[\text{STOP} | I, T] \quad (2)$$

where $|T|$ is the length of execution trace T , and the `STOP` token is emitted by the network to signal that the execution trace explains the image.

We train the network by sampling execution traces T and target images I for randomly generated scenes, and maximizing (2) wrt θ by gradient ascent. Despite the architecture being recurrent, training is fully supervised. In a sense, this model is like an autoregressive variant of AIR.

CIRCLE(x, y)	Circle at (x, y)
RECTANGLE(x_1, y_1, x_2, y_2)	Rectangle with corners at (x_1, y_1) & (x_2, y_2)
LINE(x_1, y_1, x_2, y_2 , arrow $\in \{0, 1\}$, dashed $\in \{0, 1\}$)	Line from (x_1, y_1) to (x_2, y_2) , optionally with an arrow and/or dashed
STOP	Finishes execution trace inference

Table 1: The deep network in (1) predicts drawing commands, shown above.

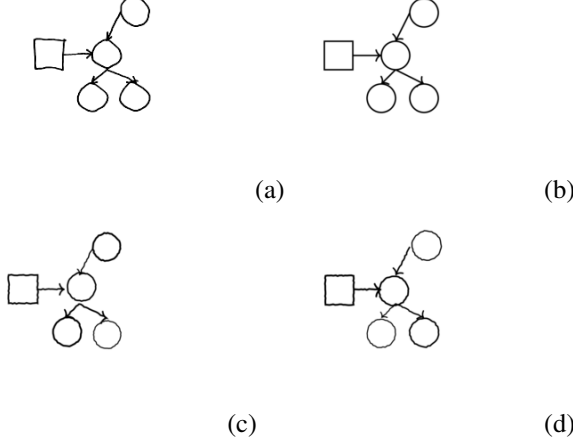


Figure 2: (a): a hand drawing. (b): Rendering of the parse our model infers for (a). We can generalize to hand drawings like these because we train the model on images corrupted by a noise process designed to resemble the kind of noise introduced by hand drawings - see (c) & (d) for noisy renderings of (b).

55 3 Generalizing to hand drawings

56 4 Synthesizing graphics programs from execution traces

57 5 Neural networks for guiding SMC

58 Let $L(\cdot|\cdot) : \text{image}^2 \rightarrow \mathcal{R}$ be our likelihood function: it takes two images, an observed target image
59 and a hypothesized program output, and gives the likelihood of the observed image conditioned on
60 the program output. We want to sample from:

$$\mathbb{P}[p|x] \propto L(x|\text{render}(p))\mathbb{P}[p] \quad (3)$$

61 where $\mathbb{P}[p]$ is the prior probability of program p , and x is the observed image.

Program	→	Command; ... ; Command
Command	→	CIRCLE(Expression, Expression)
Command	→	RECTANGLE(Expression, Expression, Expression, Expression)
Command	→	LINE(Expression, Expression, Expression, Expression, Boolean, Boolean)
Command	→	FOR($0 \leq \text{Var} < \text{Expression}$) { Program }
Command	→	REFLECT(Axis) { Program }
Expression	→	Z * Var + Z
Var	→	A free (unused) variable
Z	→	an integer
Axis	→	X = Z
Axis	→	Y = Z

Table 2: Grammar over graphics programs. We allow loops (FOR), vertical/horizontal reflections (REFLECT), and affine transformations ($Z * \text{Var} + Z$).

Algorithm 1 Neurally guided SMC

Input: Neural network NN, beam size N , maximum length L , target image x

Output: Samples of the program trace

Set $B_0 = \{\text{empty program}\}$

for $1 \leq l \leq L$ **do**

for $1 \leq n \leq N$ **do**

$p_n \sim \text{Uniform}(B_{l-1})$

$p'_n \sim \text{NN}(\text{render}(p), x)$

 Define $r_n = p'_n \cdot p_n$

 Set $\tilde{w}(r_n) = \frac{L(x|r_n)}{L(x|p_n)} \times \frac{\mathbb{P}[p'_n]}{\mathbb{P}[p'_n = \text{NN}(\text{render}(p), x)]}$

end for

 Define $w(p) = \frac{\tilde{w}(p)}{\sum_{p'} \tilde{w}(p')}$

 Set B_l to be N samples from r_n distributed according to $w(\cdot)$

end for

return $\{p : p \in B_{l \leq L}, p \text{ is finished}\}$

62 Let p be a program with L lines, which we will write as $p = (p_1, p_2, \dots, p_L)$. Assume the prior
63 factors into:

$$\mathbb{P}[p] \propto \prod_{l \leq L} \mathbb{P}[p_l] \quad (4)$$

64 Define the distribution $q_L(\cdot)$, which happens to be proportional to the above posterior:

$$q_L(p_1, p_2, \dots, p_{L-1}, p_L) \propto q_{L-1}(p_1, p_2, \dots, p_{L-1}) \times \frac{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}, p_L))}{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}))} \times \mathbb{P}[p_L] \quad (5)$$

65 Now suppose we have some samples from $q_{L-1}(\cdot)$, and that we then sample a p_L from a distribu-
66 tion proportional to $\frac{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}, p_L))}{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}))} \times \mathbb{P}[p_L]$. The resulting programs p are distributed
67 according to q_L , and so are also distributed according to $\mathbb{P}[p|x]$.

68 How do we sample p_L from a distribution proportional to $\frac{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}, p_L))}{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}))} \times \mathbb{P}[p_L]$? We
69 have a neural network that takes as input the target image x and the program so far, and produces
70 a distribution over next lines of code (p_L). We write $\text{NN}(p_L|p_1, \dots, p_{L-1}; x)$ for the distribution
71 output by the neural network. So we can sample from NN and then weight the samples by:

$$w(p_L) = \frac{\mathbb{P}[p_L]}{\text{NN}(p_L|p_1, \dots, p_{L-1}; x)} \times \frac{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}, p_L))}{L(x|\text{render}(p_1, p_2, \dots, p_{L-1}))} \quad (6)$$

72 Then we can resample from these now weighted samples to get a new population of particles (here
73 programs are particles), where each program now has L lines instead of $L - 1$.

74 This procedure can be seen as a particle filter, where each successive latent variable is another line of
75 code, and the emission probabilities are successive ratios of likelihoods under $L(\cdot|\cdot)$.

76 **Comments for Dan.** Right now I'm not actually sampling from the neural network - instead, I
77 enumerate the top few hundred lines of code suggested by the network, and then weight them by their
78 likelihoods. So actually the form of NN is:

$$\text{NN}(p_L|p_1, \dots, p_{L-1}; x) \propto \begin{cases} 1, & \text{if } p_L \in \text{top hundred neural network proposals} \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

79 Do you think this is a problem? The neural network puts almost all of its mass on a few guesses. In
80 order to get the correct line of code I sometimes need to get something like the 50th top guess, so I
81 don't want to literally just sample from the distribution suggested by the neural network.

6 Submission of papers to NIPS 2017

83 NIPS requires electronic submissions. The electronic submission site is

84 <https://cmt.research.microsoft.com/NIPS2017/>

85 Please read carefully the instructions below and follow them faithfully.

86 6.1 Style

87 Papers to be submitted to NIPS 2017 must be prepared according to the instructions presented here.
88 Papers may only be up to eight pages long, including figures. This does not include acknowledgments
89 and cited references which are allowed on subsequent pages. Papers that exceed these limits will not
90 be reviewed, or in any other way considered for presentation at the conference.

91 The margins in 2017 are the same as since 2007, which allow for $\sim 15\%$ more words in the paper
92 compared to earlier years.

93 Authors are required to use the NIPS \LaTeX style files obtainable at the NIPS website as indicated
94 below. Please make sure you use the current files and not previous versions. Tweaking the style files
95 may be grounds for rejection.

96 6.2 Retrieval of style files

97 The style files for NIPS and other conference information are available on the World Wide Web at

98 <http://www.nips.cc/>

99 The file `nips_2017.pdf` contains these instructions and illustrates the various formatting require-
100 ments your NIPS paper must satisfy.

101 The only supported style file for NIPS 2017 is `nips_2017.sty`, rewritten for $\LaTeX 2\epsilon$. **Previous**
102 **style files for $\LaTeX 2.09$, Microsoft Word, and RTF are no longer supported!**

103 The new \LaTeX style file contains two optional arguments: `final`, which creates a camera-ready copy,
104 and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

105 At submission time, please omit the `final` option. This will anonymize your submission and add
106 line numbers to aid review. Please do *not* refer to these line numbers in your paper as they will be
107 removed during generation of camera-ready copies.

108 The file `nips_2017.tex` may be used as a “shell” for writing your paper. All you have to do is
109 replace the author, title, abstract, and text of the paper with your own.

110 The formatting instructions contained in these style files are summarized in Sections 7, 8, and 9
111 below.

112 7 General formatting instructions

113 The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long.
114 The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points.
115 Times New Roman is the preferred typeface throughout, and will be selected for you by default.
116 Paragraphs are separated by $\frac{1}{2}$ line space (5.5 points), with no indentation.

117 The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal
118 rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow $\frac{1}{4}$ inch
119 space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the
120 page.

121 For the final version, authors’ names are set in boldface, and each name is centered above the
122 corresponding address. The lead author’s name is to be listed first (left-most), and the co-authors’
123 names (if different address) are set to follow. If there is only one co-author, list both author and
124 co-author side by side.

125 Please pay special attention to the instructions in Section 9 regarding figures, tables, acknowledgments,
126 and references.

127 **8 Headings: first level**

128 All headings should be lower case (except for first word and proper nouns), flush left, and bold.

129 First-level headings should be in 12-point type.

130 **8.1 Headings: second level**

131 Second-level headings should be in 10-point type.

132 **8.1.1 Headings: third level**

133 Third-level headings should be in 10-point type.

134 **Paragraphs** There is also a `\paragraph` command available, which sets the heading in bold, flush
135 left, and inline with the text, with the heading followed by 1 em of space.

136 **9 Citations, figures, tables, references**

137 These instructions apply to everyone.

138 **9.1 Citations within the text**

139 The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as
140 long as you maintain internal consistency. As to the format of the references themselves, any style is
141 acceptable as long as it is used consistently.

142 The documentation for `natbib` may be found at

143 `http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

144 Of note is the command `\citet`, which produces citations appropriate for use in inline text. For
145 example,

146 `\citet{hasselmo}` investigated\dots

147 produces

148 Hasselmo, et al. (1995) investigated...

149 If you wish to load the `natbib` package with options, you may add the following before loading the
150 `nips_2017` package:

151 `\PassOptionsToPackage{options}{natbib}`

152 If `natbib` clashes with another package you load, you can add the optional argument `nonatbib`
153 when loading the style file:

154 `\usepackage[nonatbib]{nips_2017}`

155 As submission is double blind, refer to your own published work in the third person. That is, use “In
156 the previous work of Jones et al. [4],” not “In our previous work [4].” If you cite your other papers
157 that are not widely available (e.g., a journal paper under review), use anonymous author names in the
158 citation, e.g., an author of the form “A. Anonymous.”

159 **9.2 Footnotes**

160 Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number¹
161 in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote
162 with a horizontal rule of 2 inches (12 picas).

¹Sample of the first footnote.

Table 3: Sample table title

Part		
Name	Description	Size (μm)
Dendrite	Input terminal	~ 100
Axon	Output terminal	~ 10
Soma	Cell body	up to 10^6

163 Note that footnotes are properly typeset *after* punctuation marks.²

164 9.3 Figures

165 All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction.
 166 The figure number and caption always appear after the figure. Place one line space before the figure
 167 caption and one line space after the figure. The figure caption should be lower case (except for first
 168 word and proper nouns); figures are numbered consecutively.

169 You may use color figures. However, it is best for the figure captions and the paper body to be legible
 if the paper is printed in either black/white or in color.

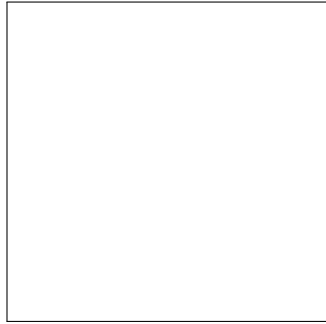


Figure 3: Sample figure caption.

170

171 9.4 Tables

172 All tables must be centered, neat, clean and legible. The table number and title always appear before
 173 the table. See Table 3.

174 Place one line space before the table title, one line space after the table title, and one line space after
 175 the table. The table title must be lower case (except for first word and proper nouns); tables are
 176 numbered consecutively.

177 Note that publication-quality tables *do not contain vertical rules*. We strongly suggest the use of the
 178 booktabs package, which allows for typesetting high-quality, professional tables:

179 <https://www.ctan.org/pkg/booktabs>

180 This package was used to typeset Table 3.

181 10 Final instructions

182 Do not change any aspects of the formatting parameters in the style files. In particular, do not modify
 183 the width or length of the rectangle the text should fit into, and do not change font sizes (except
 184 perhaps in the **References** section; see below). Please note that pages should be numbered.

²As in this example.

11 Preparing PDF files

Please prepare submission files with paper size “US Letter,” and not, for example, “A4.”

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF file uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see <http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf>
- `xfig` “patterned” shapes are implemented with bitmap fonts. Use “solid” shapes instead.
- The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

```
\usepackage{amsfonts}
```

followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for \mathbb{R} , \mathbb{N} or \mathbb{C} . You can also use the following workaround for reals, natural and complex:

```
\newcommand{\RR}{\mathbb{R}} %real numbers
\newcommand{\Nat}{\mathbb{N}} %natural numbers
\newcommand{\CC}{\mathbb{C}} %complex numbers
```

Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

11.1 Margins in L^AT_EX

Most of the margin problems come from figures positioned by hand using `\special` or other commands. We suggest using the command `\includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the `graphics` bundle documentation (<http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf>)

A number of width problems arise when L^AT_EX cannot properly hyphenate a line. Please give L^AT_EX hyphenation hints using the `\-` command when necessary.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can go over 8 pages as long as the subsequent ones contain *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

- 228 [2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the*
229 *GEneral NEural Simulation System*. New York: TELOS/Springer-Verlag.
- 230 [3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent
231 synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.