Supplement to: Inferring Graphics Programs from Images

Anonymous Author(s)
Affiliation
Address
email

1 Neural network architecture

1.1 Convolutional network

The convolutional network takes as input 2 256×256 images represented as a $2 \times 256 \times 256 \times$ volume. These are passed through two layers of convolutions separated by ReLU nonlinearities and max pooling:

- Layer 1: 20 8×8 convolutions, 2 16×4 convolutions, 2 4×16 convolutions. Followed by 8×8 pooling with a stride size of 4.
- Layer 2: 10 8×8 convolutions. Followed by 4×4 pooling with a stride size of 4.

Training takes a little bit less than a day on a Nvidia TitanX GPU. The network was trained on 10^5 synthetic examples.

1.2 Autoregressive decoding of drawing commands

Given the image features f, we predict the first token using logistic regression:

$$\mathbb{P}[T_1] \propto W_{T_1} f \tag{1}$$

where W_{T_1} is a learned weight matrix.

Subsequent tokens are predicted as:

$$\mathbb{P}[T_n | T_{1:(n-1)}] \propto \text{MLP}_{T_1,n}(I \otimes \bigotimes_{j<n} \text{oneHot}(T_j)) \tag{2}$$

Thus each token of each drawing primitive has its own learned MLP. For predicting the coordinates of lines we found that using 32 hidden nodes with sigmoid activations worked well; for other tokens the MLP's are just logistic regression (no hidden nodes).

1.3 A learned likelihood surrogate

Our architecture for $L_{\text{learned}}(\text{render}(T_1)|\text{render}(T_2))$ has the same series of convolutions as the network that predicts the next drawing command. We train it to predict two scalars: $|T_1 - T_2|$ and $|T_2 - T_1|$. These predictions are made using linear regression from the image features followed by a ReLU nonlinearity; this nonlinearity makes sense because the predictions can never be negative but could be arbitrarily large positive numbers.

We train this network by sampling random synthetic scenes for T_1, and then perturbing them in small ways to produce T_2. We minimize the squared loss between the network's prediction and the ground truth symmetric differences. T_1 is rendered in a "simulated hand drawing" style which we describe next.

2 Simulating hand drawings

We introduce noise into the rendering process by:

- Rescaling the image intensity by a factor chosen uniformly at random from $[0.5, 1.5]$
- Translating the image by ± 3 pixels chosen uniformly random
- Rendering the LaTeX using the `pencildraw` style, which adds random perturbations to the paths drawn by LaTeXin a way designed to resemble a pencil.
- Randomly perturbing the positions and sizes of primitive LaTeXdrawing commands