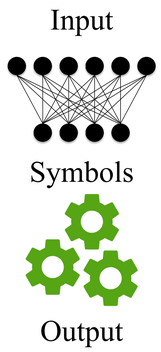
From Perception to Programs: Regularize, Overparameterize, and Amortize

Hao Tang, Kevin Ellis, Cornell University

Neurosymbolic Program Synthesis

We *seek steps toward* AI systems that learn to symbolically process perceptual input, e.g.,

- Pixels → Curves/Parts → 3D shape
- lidar → Objects/Proximities→ Action



Neurosymbolic Program Synthesis

- neural components: perception → symbols
- synthesized programs: symbols → Outputs

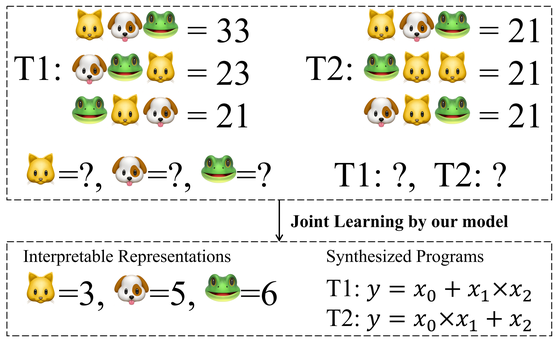
Challenges

- *Large* combinatory search space;
- containing both
  - *continuous* neural networks weights;
  - *discrete* programs;
- *Ill-posed problem* for symbol/concept discovery.
  - (e.g., how to prevent neural networks from directly learning the input-output mapping and setting the program to identity?).

Small-scale Task Example

A small-scale that involves both perception and programmatic reasoning. Introducing multi-tasking to alleviate the ill-posed problem for symbol discovery.

Solving inductive reasoning tasks grounded in perception. inferring a simple formula with noisy perceptual inputs such as images.



Solving this problem involves jointly learning to

- perceive and parse image input into symbolic form
- reason inductively to recover programs that explain each arithmetic task.

Problem Setup

Jointly learn neural symbol grounder and symbolic programs through weak task supervisions.

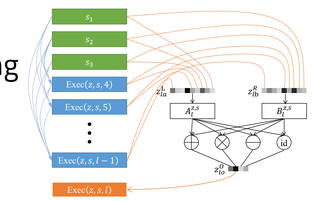
$$g^*(\cdot), z^* = \underset{\substack{g(\cdot),z \\ z's\ entry \in \{0,1\}}}{arg\,min} \left\| y - [\![z]\!](g(x)) \right\|,$$

where x is the perception input, y is the output,

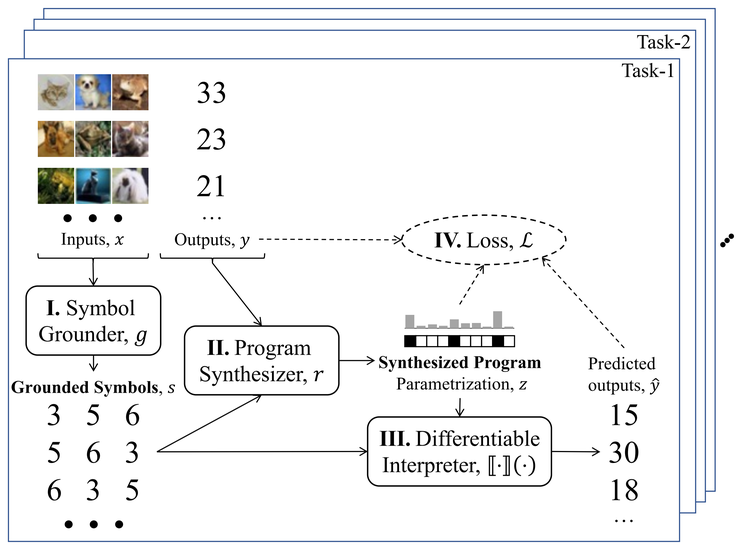
- $g(\cdot)$ is a 'symbol grounder' neural net that generates intermediate symbolic output $s = g(x)$ given the perception input x,
- z is a program to synthesize that transform symbols to predicted output $\hat{y} = [\![z]\!](s)$.

Methodology for ameliorate optimization difficulties

- Relax the space of z's entries from {0,1} to [0,1] for joint learning
- **Overparameterization** by straight-line coding
- Probabilistic VAE Framing for **Amortized Inference** & **Regularize** & **Sampling**:



$$\begin{aligned} p(y|x) &= \textstyle\sum_z p(y|x,z)p(z) \\ &\geq \mathbb{E}_q\left[\log \frac{p(y|x,z)p(z)}{q(z|y,x)}\right], \text{ ELBO} \\ &= \mathbb{E}_q[\log p(y|x,z)] + \mathbb{E}_q[\log p(z)] + \mathbb{H}[q] \\ &\geq \mathbb{E}_q[\log p(y|x,z)] + \mathbb{E}_q[\log p(z)], \text{ (Reconstruction + Regularizer)} \end{aligned}$$



Experimental Results

Metrics for different perspective of models' performances

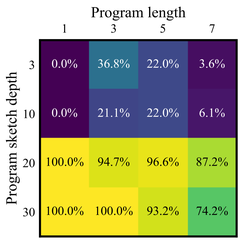
- Program synthesize: program success rate
- Symbol grounding: symbolic loss
- Out-of-distribution generalizability: generalization loss (Train on datasets with digits < 5, while test on digits ≥ 5)

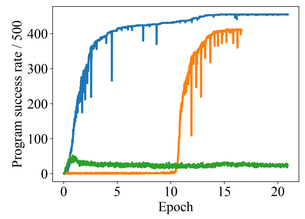
Datasets use CIFAR-10 to build perception inputs. Programs are drawn uniformly from formulas such as (s2+s3)*s1-s2.

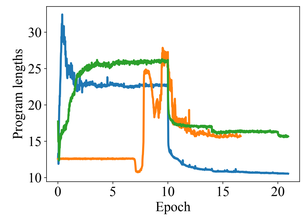
	program success rate	symbolic loss	test loss	generalization loss
ROAP (our model)	**459** / 500	**0.00086**	**0.11**	**0.07**
w/o program, i.e., CNN+MLP	0 / 500	0.95	**0.11**	1.03
w/o amortized inference	136 / 500	0.059	0.20	0.19
w/o gumbel-softmax	8 / 500	0.52	0.33	4.40
w/ Syntax-Tree	345 / 500	0.04	0.16	0.22
w/ depth=10	58 / 500	0.90	0.14	2.12
w/ depth=3	56 / 500	1.10	0.16	2.08

Experimental Results show that

- Joint learning of symbols & programs is feasible.
- All techniques are helpful, including multi-tasking with amortized inference, overparameterization, gumbel-softmax, and the program length regularizer.
- Programmatic prior enables much better *out-of-distribution generalizability* of the models.







- Effectiveness of overparameterization: Necessary for good performances; Also see the improvements w.r.t. the relative overparameterization degree.
- Effectiveness of program length prior: Improve interpretability but not hurt performances when successfully learned (blue line); can largely boost the performances when not successfully learned before (orange line) but not always (green line).