Detect or Generate? Towards Visual Understanding of Academic Diagrams

Hashem Elezabi, Ahmed Sharaf

Department of Computer Science, Stanford University

Introduction

- Objective: Given an image of a slide or a LaTeX *tikz* diagram, can we generate the source code used to create it?
- We explore two different approaches to this problem:
  - **Detect:** Detect objects in the image, then map them to source deterministically.

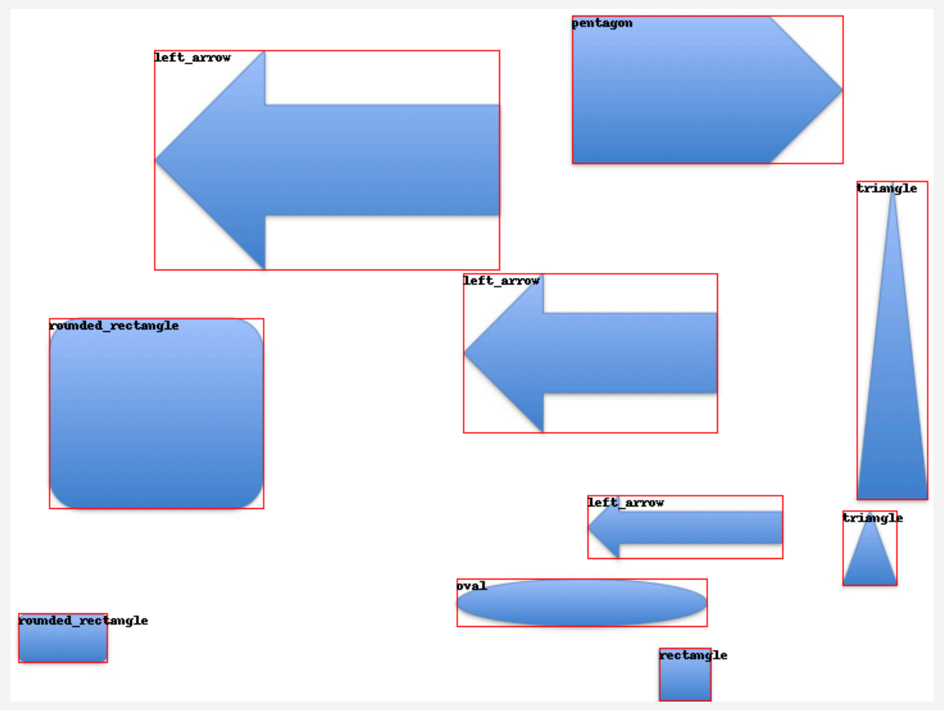
➕ Easier to learn, less prone to brittleness of language models. ➕ More controllable and interpretable. ➖ Can't detect text easily.

  - **Generate:** Train an autoregressive language model to take an input image and generate the corresponding source code end-to-end.

➕ Can detect text better. ➕ Might generalize better to different use cases. ➖ Harder to train, subject to hallucinations.

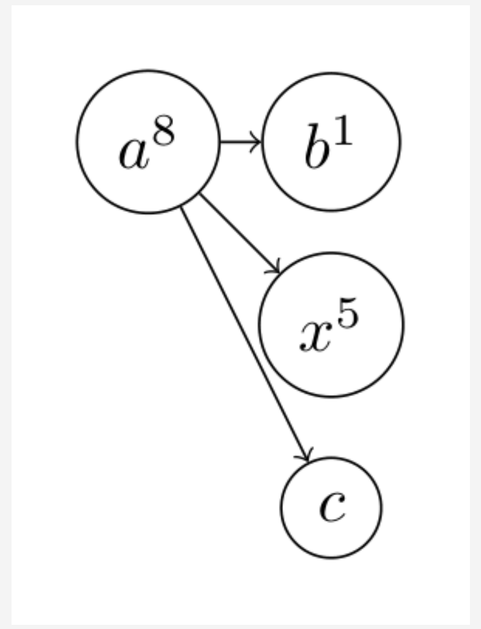
Setup & Dataset

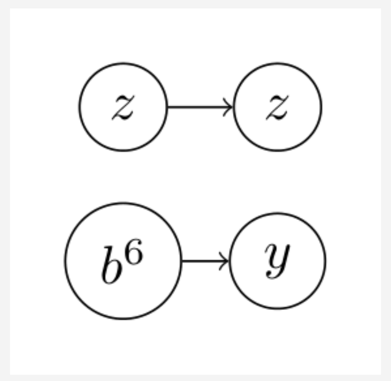
- We use two different datasets for the different approaches.
- **Detect** approach: We generated 10K PowerPoint slides containing randomly located shapes. Each shape can be a 1) pentagon, 2) heart, 3) triangle, 4) oval, 5) down arrow, 6) up arrow, 7) left arrow, 8) right arrow, or 9) rectangle.

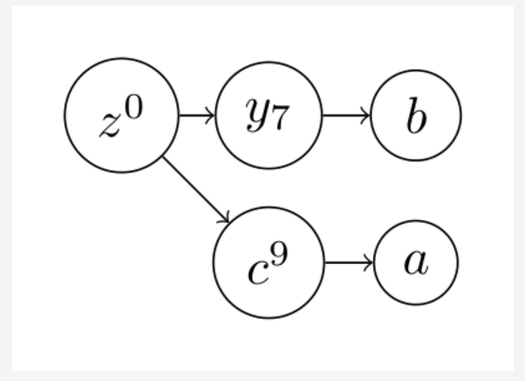


Example of a slide and its ground-truth bounding boxes

- **Generate** approach: We generated 5K tikz graphs restricted to a certain structure, with random number of nodes between 1 and 6 and random node labels of the form <L>{<s><d>}, where <L> is from {a, b, c, x, y, z}, <s> can be "_" (subscript) or "^" (superscript), and <d> is a digit.







Methods

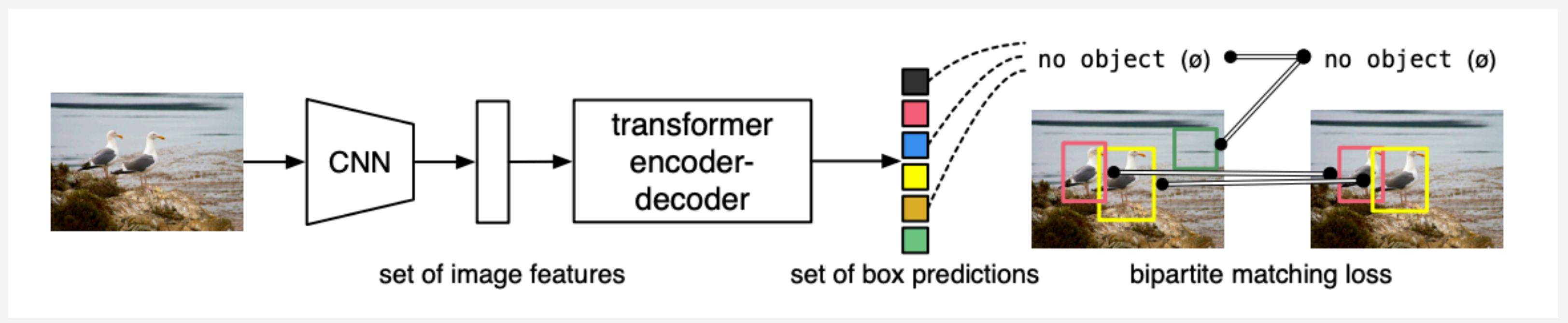
- **Detect** method: We use DETR, a pretrained transformer-based object detection model [1].
  1. DETR feeds the input image into a pretrained ResNet convolutional backbone to obtain a feature map.
  2. This feature map is then projected to match the hidden dimension of an encoder-decoder transformer.
  3. Embeddings representing *object queries* are sent through the decoder.
  4. Two classifier heads are added at the end, one for predicting an object category (or "no object") and one for predicting bounding boxes for each query.
- DETR is trained using a *bipartite matching loss* that compares predicted and ground-truth bounding boxes. It searches for a permutation of predictions with the lowest cost:

$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_i \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$$

- $\mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)})$ is a pairwise matching cost between the ground truth y_i and the prediction with index $\sigma(i)$ that includes the class prediction loss and the bounding box loss:

$$-\hat{p}_{\sigma(i)}(c_i) + \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$$

where c_i is the true category of element i in the ground truth set.
- $\mathcal{L}_{\text{box}}(b_i, \hat{b}_{\sigma(i)})$ is a linear combination of the ℓ_1 loss and the IoU loss.



- **Generate** method: We build on the recent Nougat model for visual document understanding [2], which combines a Swin Transformer as the vision encoder and mBART as the language decoder.
- Instead of using a general pretrained language decoder like BART, GPT, or Llama, we're starting with the easier task of generating the structure of the tikz graph in a specific format:

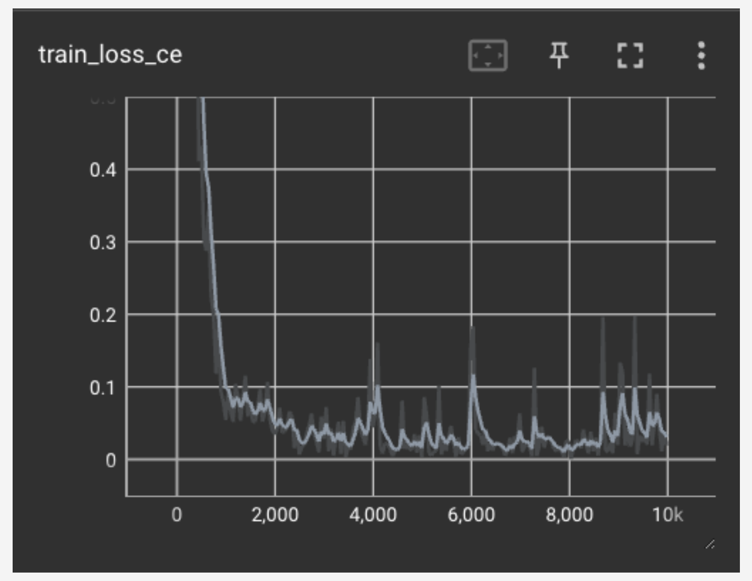
{ y , c_6 } -> { z^5 } -> { c , x^2 , x_1 }

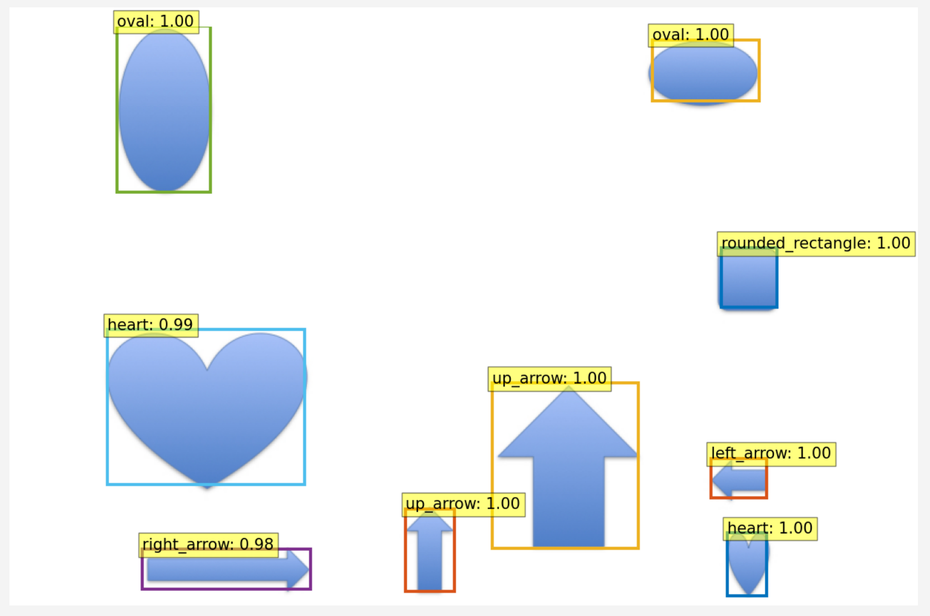
- Since our dataset is restricted, just predicting the above is enough.
- In the above, "{" and "," are tokens and "c_6" and "z^5" are tokens.
- We are designing our own autoregressive decoder architecture to be a simple Transformer model trained from scratch on our dataset.
- This decoder's vocabulary contains only the 130 possible tokens in our dataset: 6 single letters, 6*10*2 = 120 <L><s><d> tokens, and 4 more tokens "{", "}", ",", and "->".

Experiments & Results

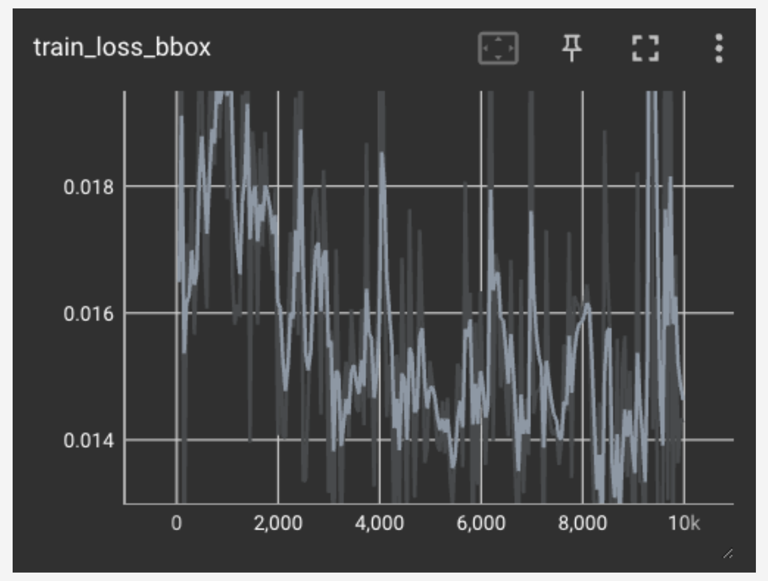
- **Detect** method: We fine-tuned DETR on 8K slide images from our dataset for 5 epochs, then tested it on the 2K remaining images. We achieved a **mean Average Precision value of 75.6%** (up to 89% at IoU=0.5) and a **mean Average Recall of 59.5%** (up to 80.6% if given 10 detections).

	IoU threshold (AP) or maxDets (AR)	Value
AP	0.5:0.95 (average)	75.6%
AP	0.5	89%
Avg Recall	1	59.5%
Avg Recall	10	80.6%





Example prediction



- **Generate** method: In progress.

Next Steps

- Train our DETR model further on images of slides with differently colored shapes and shapes with no fill. This helps it generalize to different styles and, we hope, eventually handwritten sketches!
- Train a language model decoder like Code-Llama to output tikz source code for more complicated graphs.

References

[1] Carion et al. End-to-End Object Detection with Transformers.
[2] Blecher et al. Nougat: Neural Optical Understanding for Academic Documents.