Final Report

Introduction (Purpose):

The purpose of this Independent Study Course was to explore new approaches to combine program synthesis and deep learning in order to improve data efficiency and generalizability in both supervised and reinforcement learning settings. I was going to achieve this by surveying modern program synthesis and deep learning techniques that I found the most relevant devoting 20-30hrs a week throughout the Spring semester to this study.

Originally my plan was to: "By the end of the course I will have written a short survey of the field addressing what I've learned and implemented some state-of-the-art techniques in program synthesis and deep learning on problems of my own."

Deliverables (What I've Done):

Short Survey:

Program Synthesis is the task of finding a program that satisfies a given specification, whether that be input/output pairs, a high-level definition, or structural exemplars with a few placeholders that need to be found. The field has been mainly based on formal methods in programming languages and proof checking techniques. The goal is to automate the processes of writing programs.

Even though the problem of synthesizing programs dates back to the 1950's with Alonzo Church, Program Synthesis has recently grown more popular due to a recent breakthrough (in 2013) showing that it is possible to encode program synthesis problems as Boolean logic operations and use algorithms for the Boolean satisfiability problem with SAT solver to find programs automatically.

Since then the merger of Program Synthesis with Machine Learning and Deep Learning techniques seems to be a promising burgeoning under-explored area, with major works like Dreamcoder (K. Ellis et al.) coming to the scene using data-driven deep learning approaches to achieve state-of-the-art performance on program synthesis tasks.

My work this semester hopes to further this trend by using a combination of Program Synthesis and Deep Learning to help with cases that have limited data, a space which Deep Learning alone seems to underperform.

Data Collection:

One of the research questions we were looking into is whether grounding language in image classification systems will lead to a better ability to generalize quickly to new images? And if so, how do you go about doing this? Our approach is to extract keywords from animal description text and then use a generic deep image classifier, to train a model to classify from pixel images to our key wordset (extracted from descriptions). Then from the key wordset try to synthesize a program to handle some of the semantic ambiguity of the words and classify the animal.

To collect the data for visual words used to describe animals I made a website:

https://inversepictionary.github.io/

Which is maintained by me and my Dr. Solar-Lezama (via a github organization: https://github.com/inversepictionary) The code for the website is available here:

https://github.com/inversepictionary/inversepictionary.github.io

Inside the github:

There is data of 1000 Descriptions and 1000 Drawings generated from MTurk in the file *inversepictionary.github.io/inversepictionary-export.json*

There is a file *inversepictionary.github.io/json_format.txt* which contains the explained structure of the data json:

- Description
    - unique id
        - description
        - drawer_list
            - array
        - image_ID
        - score_list
            - array
        - user_ID

- Drawings

- unique id
    - description
    - raw_image
    - score_list
        - array
    - strokes
        - array
    - user_ID
    - writer_link

- Images
  - array

- Judging
  - info_vec
  - score
  - tag_description
  - tag_drawing

Note: the models trained on this data are still underwork and therefore are not in the github

Not inside the github:

- I wrote multiple python scripts to parse through animal descriptions to extract visually meaningful words (Bulls have [horns] -> horns). I manually evaluated which way of parsing was better (the approach that used word embeddings, Stanford OpenIE, or TF/TF-IDF), by running the extraction procedures and seeing how reasonable the output was. It turned out a mixture of word embeddings and Stanford OpenIE worked the best.
- Notes on existing datasets and what information they contained: https://www.kaggle.com/navneetsurana/animaldataset, http://lila.science/datasets/nacti, and https://eol.org/
- Simple Sketch Programs I Implemented using the Sketch Manual (https://people.csail.mit.edu/asolar/manual.pdf)
- Pre Trained Classifiers

Pragmatics Project:

Towards the middle of the Spring Semester I joined a project with Yewen Pu (one of Dr. Solar-Lezama's grad students) on few shot drawing generation and classification through the lens of pragmatics (https://plato.stanford.edu/entries/pragmatics/). The goal of this project was to use pragmatics to leverage a model of the Speaker during classification tasks to more quickly and accurately guess what class is trying to be communicated.

Example: the computer has a set of classes i.e. animals (Cat, Dog, and Pig) and you have a set of ways you can communicate to the computer what animal you want, you can say ("it has cat-eyes", "it is a house pet", or "it is pink"). You say "it is a house pet". A uninformed classifier would guess Cat 50% of the time and Dog the other 50% of the time since they are both house pets, but a pragmatic classifier can reason about the probability of you saying "it is a house pet" if it was actually a Cat and not just saying "it has cat-eyes" (a more informative example).

This has implications to more informed search in Program Synthesis, but for this project I am trying to use this technique for classifying hand-drawn pictures, with basically infinite set of things you can say (i.e. draw).

For this project I created a template interface that will generate and classify sketches to demo our pragmatics model:

https://pragdraw.github.io/

Currently I've set up a pragmatics classifier that will guess information on the probability of the listener using a certain utterance given the knowledge of what class they want to represent. Now I am still trying to get the uninformed classifier working to guess the class of the sketch from a few strokes of the drawing. I am using the quick draw dataset (https://quickdraw.withgoogle.com/data) for training and testing.

Future Work (Still Needs to be Done):

I am continuing to work on both projects remotely, hoping to get some interesting results within the next few months. In the pragmatics project I still need to get the uniformed listener working and for the first project I need to re-extract keywords from peoples' descriptions. Since there is more data, the data is more noisy and therefore might need better ways of cleaning/extracting the key words from the text.

An overview of what was written in this document can be seen on:

https://github.com/wmcclinton/CIS4900-Independent-Study-Spring-2020