Natural Language to G-Code Translation for XY Gantry Robotic Systems

Ian Ding, Wai Teng Sin

Boston University, Boston, MA

Introduction

Controlling robotic systems like XY gantries typically requires users to interact with low-level programming languages such as G-code. With this, controlling these systems generally requires certain levels of training and technical skills not common to the general public.

This project explores an alternative approach: using a neural network language model to translate free-form natural language commands into structured robot control instructions for an XY gantry system. The ultimate goal is to enable users to operate the gantry using everyday English without needing technical knowledge of the underlying G-code format.

The goal of this project is to create a model that can return a sequence of structured tokens that can be systematically parsed and inputted into a robotic system for execution. This sequence of tokens should correspond to the human-level instructions that are passed to the model and be formatted in a manner that is safe and legal for the machine to execute.

Motivation

For many people, modern robotic systems offer a level of precision and accuracy for certain tasks that cannot be obtained by them alone. The issue with this is that they remain inaccessible to the general public, with a large majority of the public simply lacking the skills to take advantage of these systems.

By allowing natural language control, we lower the barrier to entry for using complex robotic systems, making them accessible to a broader audience including students, hobbyists, small business owners, and the disabled audience. We envision applications across education, manufacturing, and most importantly in assistive technologies, where individuals stand to benefit the most from simple access to robotic systems.

Ultimately, the goal is to create a system that is easy to use, and enables users to focus on using robotic systems to their advantage, rather than the technical challenges of operating the machine.

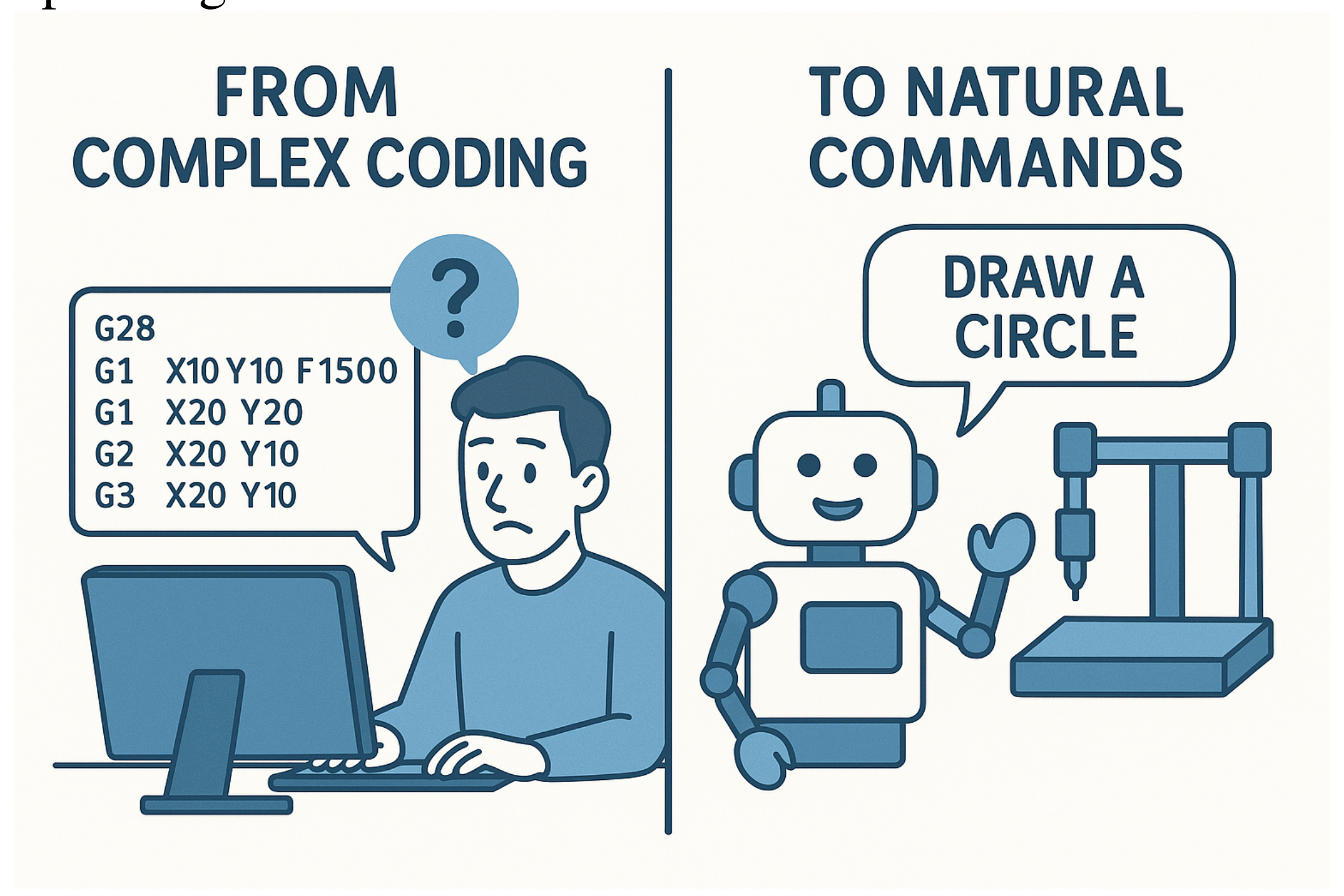

Figure 1: Translating Natural Language to G-Code

Data Collection

We originally sought to generate our own data entries using some LLM capabilities to use a custom token space that was a mix of machine code and human readable instruction sets, but this proved to be infeasible due to LLM capabilities to generate complex instructions and the difficulty in verifying the correctness of the data entries.

The challenge when pivoting away from this was to find a way to collect G-code data and corresponding instructions in a way that creates a data set of entries. We were unable to find prexisting data sets that matched our requirements so we needed to find a way of building our data set from the existing resources online.

To construct this dataset, we first extracted G-code repository and file names from the BigCode Stacks project. We then scraped the corresponding G-code files directly from public GitHub repositories based on these references. For each file, a natural language instruction was generated from the file context and metadata. Then each natural language instruction is paired with the associated G-code file contents.

```
{
  "prompt": "3D print a small mechanical part.
    ",
  "completion": "G28\nM104 S195\nM140 S40\
    nM190 S40\nM109 S195\nG0 F1000 X0 Y0 Z10 \
    nM104 S0 \nM140 S0 \nG91 \nG1 E - 1 F300 \
    nG1 Z + 0.5 E - 5 X - 20 Y - 20 F2700 \
    nG28 X0 Y0 \nM84 \nG90\nG0 F9000 X0 Y0 Z0"
}
```

Listing 1: Example data entry from Gantry Instruction Dataset

Model Details

For this project we implemented three different models that approach the problem of G-code translation in different ways. The Baseline RNN model processes the input instruction one token at a time, maintaining a hidden recurrent state that is updated at each step.

In looking to move further than a RNN model, we sought to approach the problem with a Transformer model with 6 layers and 4 attention heads. The Transformer model encodes the entire input instruction in parallel using self-attention mechanisms, and generates the output sequence with a masked decoder that hides the remainder of the target output.

Our last model is an alteration of a pretrained model, which is tuned to understand and execute our specific task. We used the StarCoder 3B parameter model and trained it using Low Rank Adaption Training (LoRA) to freeze a majority of its parameters to optimize it for our task while keeping training quick and computationally feasible.

Model Architecture

We compare two sequence-to-sequence models for gantry command generation. Both map a natural language instruction $\mathbf{x} = (x_1, \dots, x_T)$ to a sequence of G-code tokens $\mathbf{y} = (y_1, \dots, y_{T'})$. The RNN model uses a recurrent hidden state to process one token at a time, while the Transformer uses global attention over the entire sequence.

Lastly we compare a third pretrained model that is a decoder-only Transformer model that is fine tuned for code generation.

Baseline RNN model:

$$\mathbf{h}_0 = \mathbf{0}$$
$$\mathbf{e}_t = \text{one\_hot}(x_t)$$
$$\mathbf{h}_t = \sigma(W_e \mathbf{e}_t + U_h \mathbf{h}_{t-1} + \mathbf{b}_h)$$
$$\mathbf{z}_t = W_o \mathbf{h}_t + \mathbf{b}_o$$
$$Pr(y_t | \mathbf{x}_{1:t}) = \text{softmax}(\mathbf{z}_t)$$

Transformer model:

$$\mathbf{e}_t = \text{TokenEmb}(x_t) + \text{PosEmb}(t)$$
$$Q = \mathbf{X}W^Q,\ K = \mathbf{X}W^K,\ V = \mathbf{X}W^V$$
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right) V$$
$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$
$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$
$$\text{FFN}(x) = \text{ReLU}(xW_1 + b_1)W_2 + b_2$$
$$\text{LayerNorm}(x + \text{SubLayer}(x))$$
$$\mathbf{H}^{(l)} = \text{EncoderLayer}(\mathbf{H}^{(l-1)})$$
$$\mathbf{Y}^{(l)} = \text{DecoderLayer}(\mathbf{Y}^{(l-1)}, \mathbf{H}_{\text{enc}})$$
$$\mathbf{z}_t = \mathbf{H}_{\text{dec},t} W_o + b_o$$
$$P(y_t | \mathbf{x}, \mathbf{y}_{<t}) = \text{softmax}(\mathbf{z}_t)$$

StarCoder 3B (decoder-only transformer):

$$\mathbf{e}_t = \text{TokenEmb}(x_t) + \text{PosEmb}(t)$$
$$\mathbf{H} = \text{DecoderStack}(\mathbf{e}_1, \dots, \mathbf{e}_T)$$
$$\mathbf{z}_t = W_o \mathbf{h}_t + \mathbf{b}_o$$
$$Pr(x_t | x_{<t}) = \text{softmax}(\mathbf{z}_t)$$

Experiments

In training our transformer model we ran into a significant issue with the memory limitations. The entries in our data set often contain sequences that well exceed hundreds of thousands of characters in length, something unfeasible for us to produce or train. As a result we were forced to truncate sequences. This combined with the challenge of understanding natural language in and of itself we realized that our model would simply not have enough time or complexity to learn the patterns needed to properly predict output tokens. This was also clear in our training as we see the total loss come to a complete plateau.

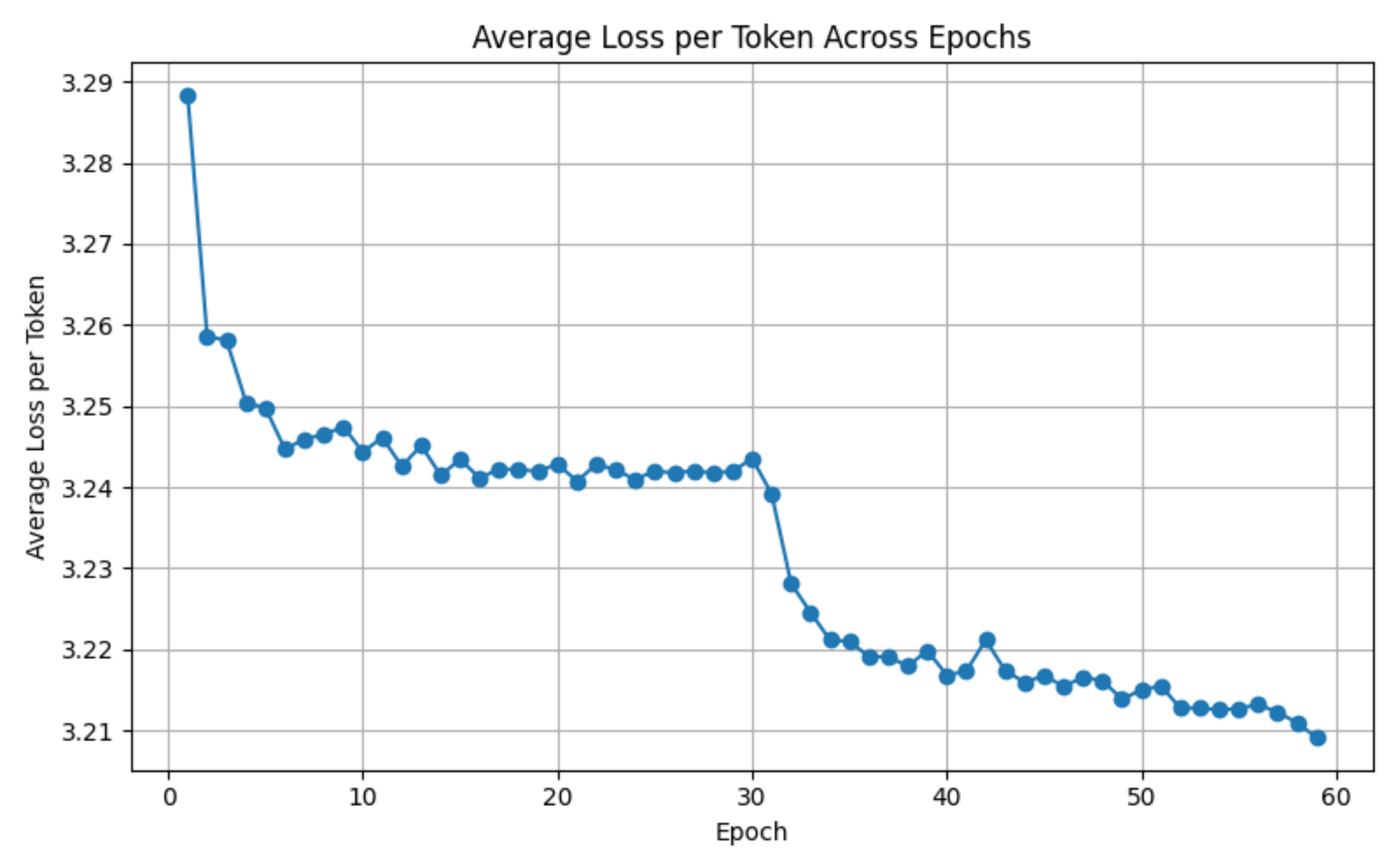

Figure 2: Transformer Training Loss

Later we transferred our focus to a pre-trained model that would allow us to leverage the parameters that have already been carefully tuned. When using our training data to modify this model further we experienced much better results. Below we can see many metrics of this model that highlight its rapid improvement.

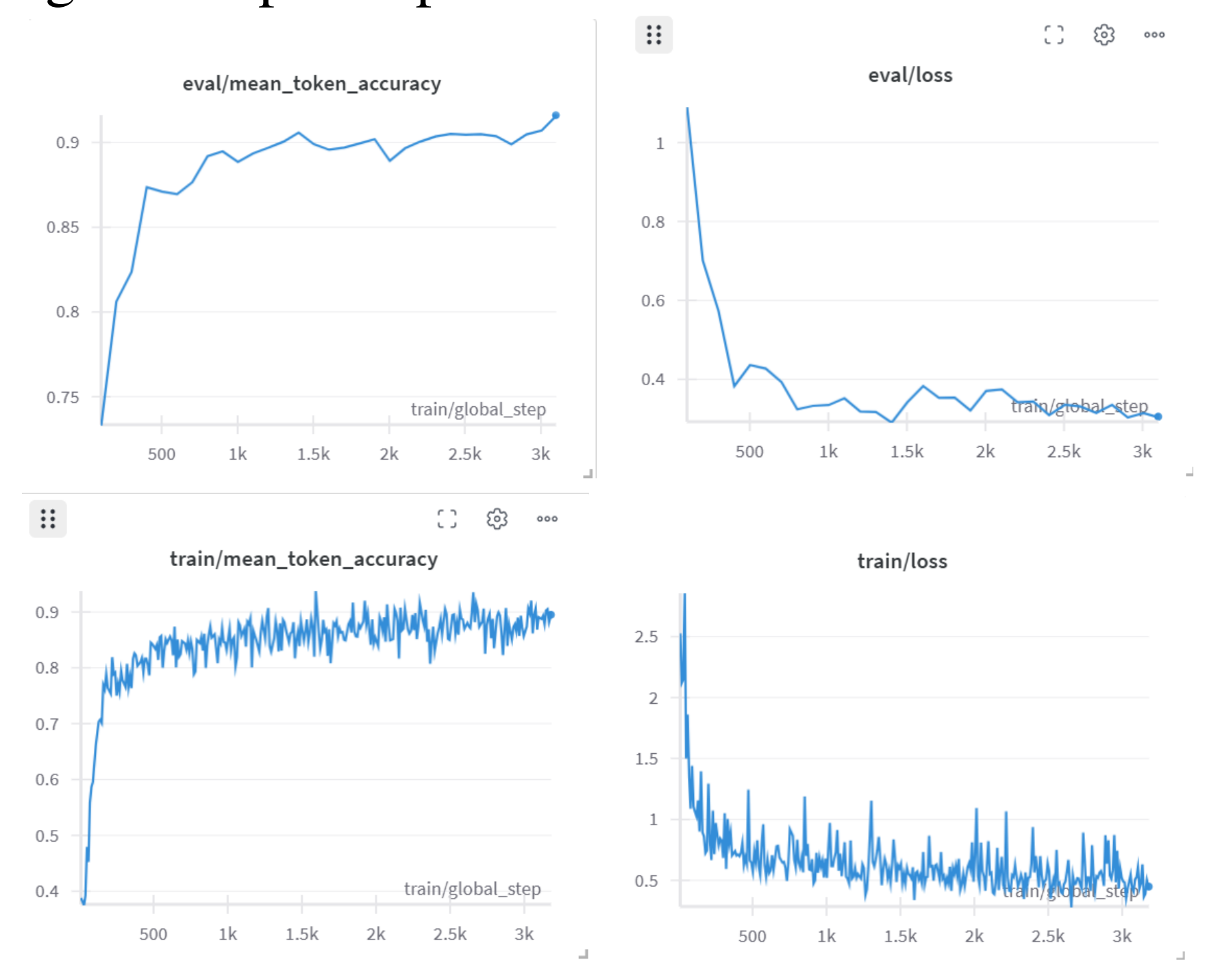

Figure 3: Evolution of both accuracy and loss across 3,000 epochs for the train and test sets.

Conclusion

Through our project, we demonstrated the ability to translate natural human language into deterministic and executable machine code. Although our original models were constrained by the resources/time we have available, which act as a limitation to the complexity of the model we did find that fine tuning an already trained model served as a feasible starting point for expanding upon this project.

This work highlights the promise of a such a tool, that can help us bring intuitive natural language interfaces to the are of robotic systems. In tearing down the technical barrier to robot control we hope to make a meaningful step towards bridging the gap between human-robot interactions to make systems like a Gantry more accessible to the general public.