PROGRAM INDUCTION USING NEURAL PROGRAM-INTERPRETER

NOVEL APPLICATION

Souradeepta Biswas
Department of Computer Science
Syracuse University
`sobiswas@syr.edu`

Ritesh Nair
Department of Computer Science
Syracuse University
`rgnair@syr.edu`

Sanman Yadav
Applied Data Science
School of Information Studies
Syracuse University
`smyadav@syr.edu`

ABSTRACT

Our motivation to work on Neural Program-Interpreter (NPI) has been because it can learn programs in very dissimilar environments with different affordances. In the context of sorting, the authors showed that NPI exhibits very strong generalization in comparison to sequence-to-sequence LSTMs. They discuss how a trained Neural Program-Interpreter with a xed core can continue to learn new programs without forgetting already learned programs. We are attempting to use this architecture to teach the NPI how to program Merge Sort, Arithmetic Question Answering and Card Matching. Our estimation is that the Neural network will be able to train on our generated datasets with their execution traces to be able to solve the three tasks with low loss value. We see that the NPI is able to solve addition and card matching problems with high accuracy. But we face problems when it comes to more complex tasks like merge sort. Developing environment for a recursive solution proves to be challenging.

***K*eywords** Neural Program Interpreter(NPI) · Long Short Term Memory(LSTM) · Merge Sort · Card Matching · Arithmetic Answering(Addition)

1 Introduction

Training neural networks to synthesize robust programs from a small number of examples is a challenging task. The space of possible programs is extremely large, and composing a program that performs robustly on the infinite space of possible inputs is difficult—in part because it is impractical to obtain enough training examples to easily disambiguate amongst all possible programs. Nevertheless, we would like the model to quickly learn to represent the right semantics of the underlying program from a small number of training examples, not an exhaustive number of them[1]. For this the authors propose the architecture neural programmer-interpreter(NPI) which is a recurrent and compositional neural network that learns to represent and execute programs. In NPI, the core module is an LSTM-based sequence model that takes as input a learnable program embedding, program arguments passed on by the calling program, and a feature representation of the environment. The output of the core module is a key indicating what program to call next, arguments for the following program and a flag indicating whether the program should terminate. In addition to the recurrent core, the NPI architecture includes a learnable key-value memory of program embeddings. This program-memory is essential for learning and re-using programs in a continual manner.[2]. For our project we implement Neural Programmer-Interpreter (NPI) to solve three problems. To solve Merge Sort, Arithmetic Question Answering and Card Matching. This application falls under novel application, where in we use the existing architecture of NPI and solve the above mentioned three tasks. We chose these tasks since these tasks were solved on other neural architectures and we wanted to see if NPI would be able to solve this. Our code can be found at [11]

2 Methods

This architecture sub-section focuses on the NPI architecture in depth. Plus this also focuses on the architecture for the 3 novel tasks. The experimental tasks subsection focuses on the novel tasks in depth and shed a light on the data-set used for the project.

2.1 Architecture

The NPI core is a long short-term memory (LSTM) network that acts as a router between programs conditioned on the current state observation and previous hidden unit states. At each time step, the core module can select another program to invoke using content-based addressing. It emits the probability of ending the current program with a single binary unit. If this probability is over threshold, control is returned to the caller by popping the caller's LSTM hidden units and program embedding off of a program call stack and resuming execution in this context.

LSTM's basic unit is called a memory cell. Within each memory cell, there is a linear unit with a fixed-weight self-connection. This enforces constant, non exploding, non-vanishing error flow within the memory cell. A multiplicative input gate unit learns to protect the constant error flow within the memory cell from perturbation by irrelevant inputs. Likewise, a multiplicative output gate unit learns to protect other units from perturbation by currently irrelevant memory contents . stored in the memory cell. The gates learn-to open and close access to constant error flow. Why is constant error flow important? For instance, with conventional "back-propagation through time", error signals ”flowing backwards in time" tend to vanish: the temporal evolution of the back-propagated error exponentially depends on the size of the weights.[8]

Our architecture also consists of following:

- Program Termination Network: Feed-Forward Network (fend), takes LSTM Controller hidden state ht and outputs a probability of terminating execution.
- Subroutine Lookup Network: Feed-Forward Network (fprog), takes LSTM Controller hidden state ht and outputs a key embedding kt to look up next subroutine to be called.
- Argument Networks: Feed-Forward Networks (farg), takes LSTM Controller hidden state ht and outputs subroutine arguments a(t + 1).

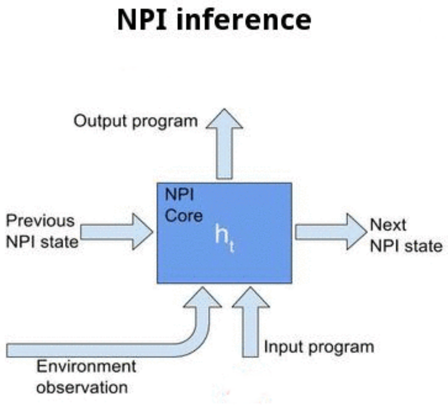


Figure 1: NPI Architecture

The training procedure consisted of Adam optimizer and Stochastic Gradient Descent(SGD). The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. In gradient descent, a batch is the total number of examples you use to calculate the gradient in a single iteration. So far, we've assumed that the batch has been the entire data set. Stochastic gradient descent (SGD) takes this idea to the extreme–it uses only a single example (a batch size of 1) per iteration.

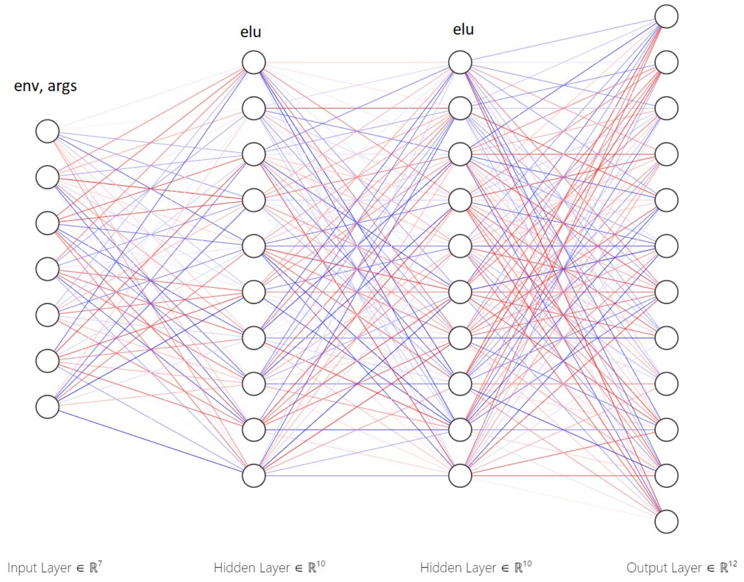


Figure 2: Environment Encoder Network

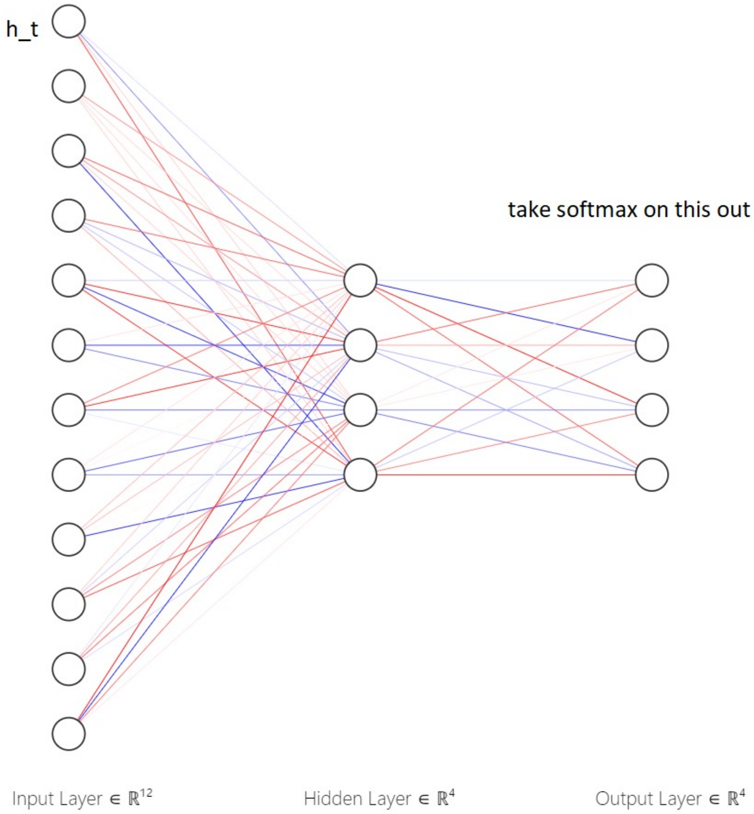


Figure 3: Program Encoder Network

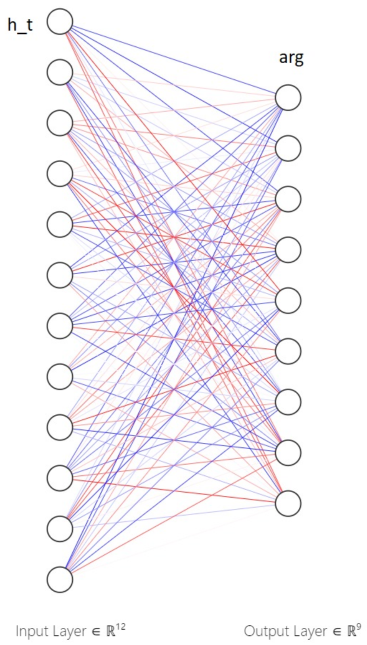


Figure 4: Feed Forward Argument Network

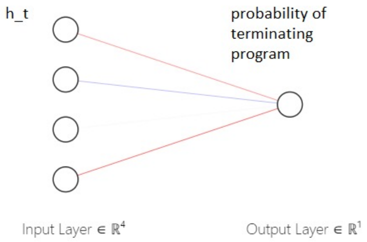


Figure 5: Program Termination Network

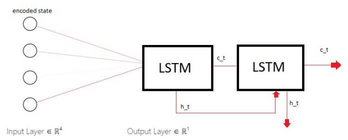
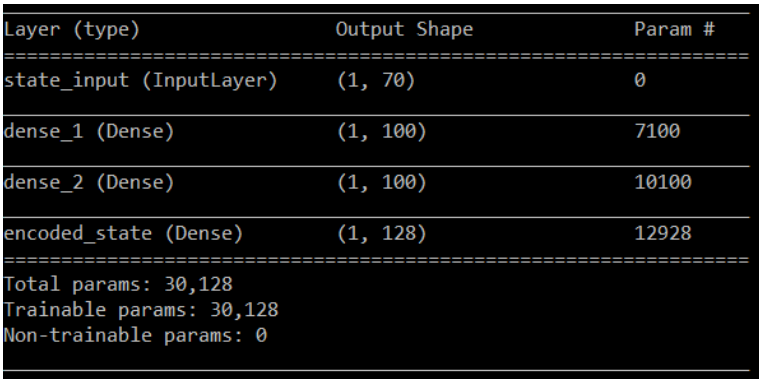
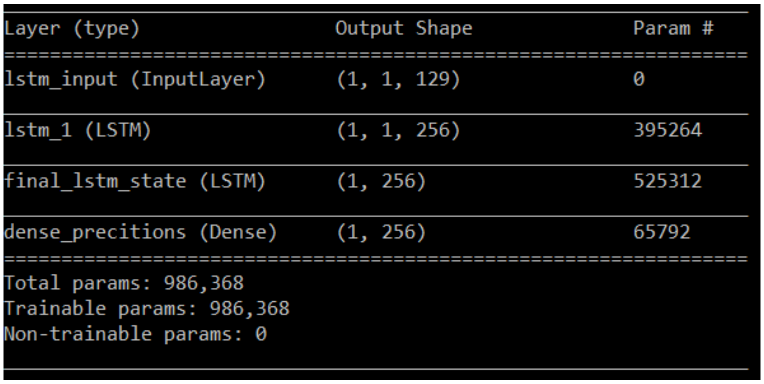


Figure 6: Neural Programmer-Interpreter Core



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
state_input (InputLayer)     (1, 70)                   0
_________________________________________________________________
dense_1 (Dense)              (1, 100)                  7100
_________________________________________________________________
dense_2 (Dense)              (1, 100)                  10100
_________________________________________________________________
encoded_state (Dense)        (1, 128)                  12928
=================================================================
Total params: 30,128
Trainable params: 30,128
Non-trainable params: 0
_________________________________________________________________
```

Figure 7: State Encoder



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_input (InputLayer)      (1, 1, 129)               0
_________________________________________________________________
lstm_1 (LSTM)                (1, 1, 256)               395264
_________________________________________________________________
final_lstm_state (LSTM)      (1, 256)                  525312
_________________________________________________________________
dense_precitions (Dense)     (1, 256)                  65792
=================================================================
Total params: 986,368
Trainable params: 986,368
Non-trainable params: 0
_________________________________________________________________
```

Figure 8: NPI Core

2.2 Experimental Tasks

In this section we will go in depth of the three tasks that have been mention. We will talk about the tasks it self, some example of how the input and output of the task should look like and about the dataset used for the tasks.

Addition - Sanman Yadav In [2] the author discuss the capability of NPI to perform grade-school addition algorithm. However, in the Neural Programmer paper experiments show the Neural Programmer to give results for arithmetic in question-answer format. We would like to train NPI to learn a program which can answer such arithmetic questions like addition. For the scope of the project we are only focusing on the NPI's ability to perform basic addition of two numbers. As the digit size goes on increasing the complexity of addition also increases so here we are only focusing on 2 to 3 digit addition.

Table 1: Addition Example

Input		
Number1	Number2	Output
9	9	18
22	8	30
21	21	42

Merge Sort - Souradeepta Biswas In [2] the authors demonstrate how they train NPI to program sorting using bubble sort. We plan to increase the complexity of this task and train NPI to learn Merge Sort.Merge Sort is a sorting algorithm, which is commonly used in computer science. Merge Sort is a divide and conquer algorithm. It works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly. The solutions to the sub-problems are then combined to give a solution to the original problem. So Merge Sort first divides the array into equal halves and then combines them in a sorted manner. Steps of Merge Sort: 1. If it is only one element in the list it is already sorted, return. 2. Divide the list recursively into two halves until it can no more be divided. 3. Merge the smaller lists into new list in sorted order.

Card Matching - Ritesh Nair In Engineering neural systems for high-level problem solving, the authors demonstrate how their neural framework GALIS solves the Card Matching problem. In similar veins, it would be interesting to see how the NPI can be trained to program a pattern matching algorithm for Card Matching.

3 Results

3.1 Addition - Sanman Yadav

We were able to train the model to correctly give us the addition of two numbers of variable size.

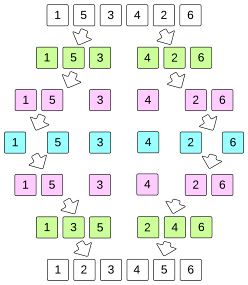


Figure 9: Merge Sort Example

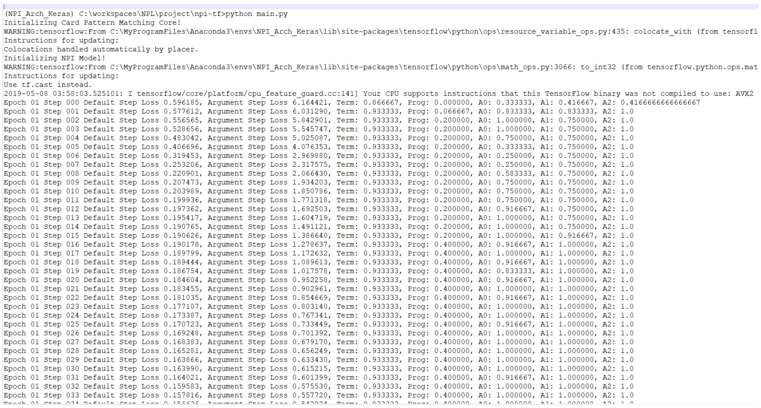
3.1.1 Errors

3.2 Merge Sort - Souradeepta Biswas

Our execution trace was written but development of the environment was challenging.

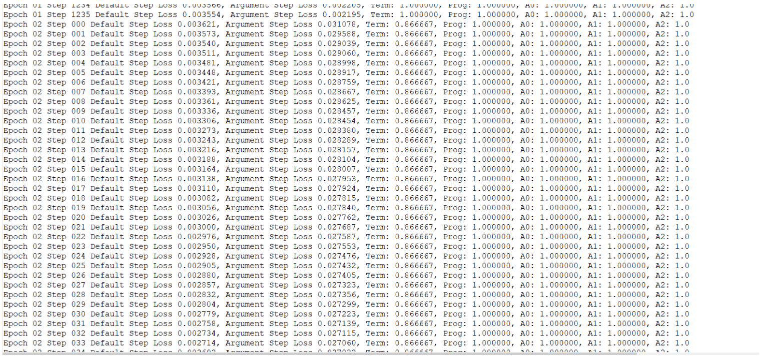
3.3 Card Matching - Ritesh Nair

We were able to train the model to correctly match two suits of cards.



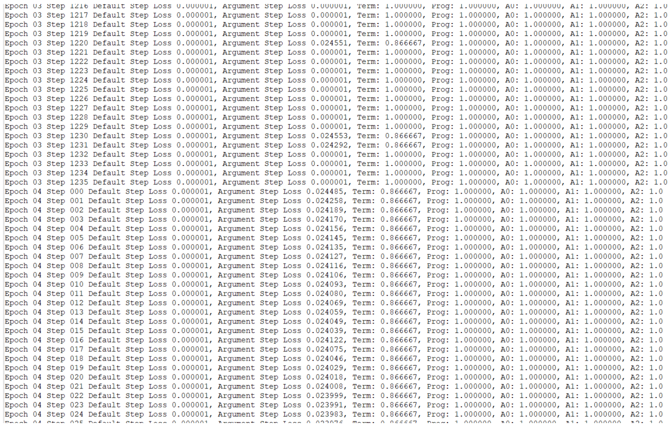
```
(NPI_Arch_Keras) C:\workspaces\NPL\project\npi-tf>python main.py
Initializing Card Pattern Matching Core!
WARNING:tensorflow:From C:\MyProgramFiles\Anaconda3\envs\NPI_Arch_Keras\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tensorfl
Instructions for updating:
Colocations handled automatically by placer.
Initializing NPI Model!
WARNING:tensorflow:From C:\MyProgramFiles\Anaconda3\envs\NPI_Arch_Keras\lib\site-packages\tensorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.ops.mat
Instructions for updating:
Use tf.cast instead.
2019-05-08 03:58:03.525101: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Epoch 01 Step 000 Default Step Loss 0.596185, Argument Step Loss 6.164421, Term: 0.866667, Prog: 0.000000, A0: 0.333333, A1: 0.416667, A2: 0.4166666666666667
Epoch 01 Step 001 Default Step Loss 0.577612, Argument Step Loss 6.031290, Term: 0.933333, Prog: 0.066667, A0: 0.833333, A1: 0.833333, A2: 1.0
Epoch 01 Step 002 Default Step Loss 0.556565, Argument Step Loss 5.842901, Term: 0.933333, Prog: 0.200000, A0: 1.000000, A1: 0.750000, A2: 1.0
Epoch 01 Step 003 Default Step Loss 0.528656, Argument Step Loss 5.545747, Term: 0.933333, Prog: 0.200000, A0: 1.000000, A1: 0.750000, A2: 1.0
Epoch 01 Step 004 Default Step Loss 0.483042, Argument Step Loss 5.025087, Term: 0.933333, Prog: 0.200000, A0: 0.750000, A1: 0.750000, A2: 1.0
Epoch 01 Step 005 Default Step Loss 0.406696, Argument Step Loss 4.076353, Term: 0.933333, Prog: 0.200000, A0: 0.333333, A1: 0.750000, A2: 1.0
Epoch 01 Step 006 Default Step Loss 0.319453, Argument Step Loss 2.969880, Term: 0.933333, Prog: 0.200000, A0: 0.250000, A1: 0.750000, A2: 1.0
Epoch 01 Step 007 Default Step Loss 0.253206, Argument Step Loss 2.317575, Term: 0.933333, Prog: 0.200000, A0: 0.250000, A1: 0.750000, A2: 1.0
Epoch 01 Step 008 Default Step Loss 0.220901, Argument Step Loss 2.066430, Term: 0.933333, Prog: 0.200000, A0: 0.583333, A1: 0.750000, A2: 1.0
Epoch 01 Step 009 Default Step Loss 0.207473, Argument Step Loss 1.934203, Term: 0.933333, Prog: 0.200000, A0: 0.750000, A1: 0.750000, A2: 1.0
Epoch 01 Step 010 Default Step Loss 0.203989, Argument Step Loss 1.850786, Term: 0.933333, Prog: 0.200000, A0: 0.750000, A1: 0.750000, A2: 1.0
Epoch 01 Step 011 Default Step Loss 0.199936, Argument Step Loss 1.771318, Term: 0.933333, Prog: 0.200000, A0: 0.750000, A1: 0.750000, A2: 1.0
Epoch 01 Step 012 Default Step Loss 0.197362, Argument Step Loss 1.692503, Term: 0.933333, Prog: 0.200000, A0: 0.916667, A1: 0.750000, A2: 1.0
Epoch 01 Step 013 Default Step Loss 0.195417, Argument Step Loss 1.604719, Term: 0.933333, Prog: 0.200000, A0: 1.000000, A1: 0.750000, A2: 1.0
Epoch 01 Step 014 Default Step Loss 0.190765, Argument Step Loss 1.491121, Term: 0.933333, Prog: 0.200000, A0: 1.000000, A1: 0.750000, A2: 1.0
Epoch 01 Step 015 Default Step Loss 0.190626, Argument Step Loss 1.386640, Term: 0.933333, Prog: 0.200000, A0: 1.000000, A1: 0.916667, A2: 1.0
Epoch 01 Step 016 Default Step Loss 0.190178, Argument Step Loss 1.278637, Term: 0.933333, Prog: 0.400000, A0: 0.916667, A1: 1.000000, A2: 1.0
Epoch 01 Step 017 Default Step Loss 0.189799, Argument Step Loss 1.172632, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 018 Default Step Loss 0.189444, Argument Step Loss 1.089613, Term: 0.933333, Prog: 0.400000, A0: 0.916667, A1: 1.000000, A2: 1.0
Epoch 01 Step 019 Default Step Loss 0.186754, Argument Step Loss 1.017578, Term: 0.933333, Prog: 0.400000, A0: 0.833333, A1: 1.000000, A2: 1.0
Epoch 01 Step 020 Default Step Loss 0.184604, Argument Step Loss 0.952258, Term: 0.933333, Prog: 0.400000, A0: 0.916667, A1: 1.000000, A2: 1.0
Epoch 01 Step 021 Default Step Loss 0.183455, Argument Step Loss 0.902961, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 022 Default Step Loss 0.181035, Argument Step Loss 0.854669, Term: 0.933333, Prog: 0.400000, A0: 0.916667, A1: 1.000000, A2: 1.0
Epoch 01 Step 023 Default Step Loss 0.177107, Argument Step Loss 0.803140, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 024 Default Step Loss 0.173387, Argument Step Loss 0.767341, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 025 Default Step Loss 0.170723, Argument Step Loss 0.733449, Term: 0.933333, Prog: 0.400000, A0: 0.916667, A1: 1.000000, A2: 1.0
Epoch 01 Step 026 Default Step Loss 0.169248, Argument Step Loss 0.701392, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 027 Default Step Loss 0.168383, Argument Step Loss 0.679170, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 028 Default Step Loss 0.165281, Argument Step Loss 0.656249, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 029 Default Step Loss 0.163866, Argument Step Loss 0.633430, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 030 Default Step Loss 0.163990, Argument Step Loss 0.615215, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 031 Default Step Loss 0.164021, Argument Step Loss 0.601399, Term: 0.933333, Prog: 0.400000, A0: 0.916667, A1: 1.000000, A2: 1.0
Epoch 01 Step 032 Default Step Loss 0.159583, Argument Step Loss 0.575530, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 033 Default Step Loss 0.157816, Argument Step Loss 0.557720, Term: 0.933333, Prog: 0.400000, A0: 1.000000, A1: 1.000000, A2: 1.0
```

Figure 10: Model Running Logs 01



```
Epoch 01 Step 1234 Default Step Loss 0.003566, Argument Step Loss 0.002205, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 01 Step 1235 Default Step Loss 0.003554, Argument Step Loss 0.002195, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 000 Default Step Loss 0.003621, Argument Step Loss 0.031078, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 001 Default Step Loss 0.003573, Argument Step Loss 0.029588, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 002 Default Step Loss 0.003540, Argument Step Loss 0.029039, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 003 Default Step Loss 0.003511, Argument Step Loss 0.029060, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 004 Default Step Loss 0.003481, Argument Step Loss 0.028998, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 005 Default Step Loss 0.003448, Argument Step Loss 0.028917, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 006 Default Step Loss 0.003421, Argument Step Loss 0.028759, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 007 Default Step Loss 0.003393, Argument Step Loss 0.028667, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 008 Default Step Loss 0.003361, Argument Step Loss 0.028625, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 009 Default Step Loss 0.003336, Argument Step Loss 0.028457, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 010 Default Step Loss 0.003306, Argument Step Loss 0.028454, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 011 Default Step Loss 0.003273, Argument Step Loss 0.028380, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 012 Default Step Loss 0.003243, Argument Step Loss 0.028289, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 013 Default Step Loss 0.003216, Argument Step Loss 0.028157, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 014 Default Step Loss 0.003188, Argument Step Loss 0.028104, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 015 Default Step Loss 0.003164, Argument Step Loss 0.028007, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 016 Default Step Loss 0.003138, Argument Step Loss 0.027953, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 017 Default Step Loss 0.003110, Argument Step Loss 0.027924, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 018 Default Step Loss 0.003082, Argument Step Loss 0.027815, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 019 Default Step Loss 0.003056, Argument Step Loss 0.027840, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 020 Default Step Loss 0.003026, Argument Step Loss 0.027762, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 021 Default Step Loss 0.003000, Argument Step Loss 0.027687, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 022 Default Step Loss 0.002976, Argument Step Loss 0.027587, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 023 Default Step Loss 0.002950, Argument Step Loss 0.027553, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 024 Default Step Loss 0.002928, Argument Step Loss 0.027476, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 025 Default Step Loss 0.002905, Argument Step Loss 0.027432, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 026 Default Step Loss 0.002880, Argument Step Loss 0.027405, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 027 Default Step Loss 0.002857, Argument Step Loss 0.027323, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 028 Default Step Loss 0.002832, Argument Step Loss 0.027356, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 029 Default Step Loss 0.002804, Argument Step Loss 0.027299, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 030 Default Step Loss 0.002779, Argument Step Loss 0.027223, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 031 Default Step Loss 0.002758, Argument Step Loss 0.027139, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 032 Default Step Loss 0.002734, Argument Step Loss 0.027115, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 02 Step 033 Default Step Loss 0.002714, Argument Step Loss 0.027060, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
```

Figure 11: Model Running Logs 02



```
Epoch 03 Step 1216 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1217 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1218 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1219 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1220 Default Step Loss 0.000001, Argument Step Loss 0.024551, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1221 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1222 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1223 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1224 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1225 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1226 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1227 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1228 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1229 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1230 Default Step Loss 0.000001, Argument Step Loss 0.024553, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1231 Default Step Loss 0.000001, Argument Step Loss 0.024292, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1232 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1233 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1234 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 03 Step 1235 Default Step Loss 0.000001, Argument Step Loss 0.000001, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 000 Default Step Loss 0.000001, Argument Step Loss 0.024485, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 001 Default Step Loss 0.000001, Argument Step Loss 0.024258, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 002 Default Step Loss 0.000001, Argument Step Loss 0.024189, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 003 Default Step Loss 0.000001, Argument Step Loss 0.024170, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 004 Default Step Loss 0.000001, Argument Step Loss 0.024156, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 005 Default Step Loss 0.000001, Argument Step Loss 0.024145, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 006 Default Step Loss 0.000001, Argument Step Loss 0.024135, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 007 Default Step Loss 0.000001, Argument Step Loss 0.024127, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 008 Default Step Loss 0.000001, Argument Step Loss 0.024116, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 009 Default Step Loss 0.000001, Argument Step Loss 0.024106, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 010 Default Step Loss 0.000001, Argument Step Loss 0.024093, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 011 Default Step Loss 0.000001, Argument Step Loss 0.024080, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 012 Default Step Loss 0.000001, Argument Step Loss 0.024069, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 013 Default Step Loss 0.000001, Argument Step Loss 0.024059, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 014 Default Step Loss 0.000001, Argument Step Loss 0.024049, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 015 Default Step Loss 0.000001, Argument Step Loss 0.024039, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 016 Default Step Loss 0.000001, Argument Step Loss 0.024122, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 017 Default Step Loss 0.000001, Argument Step Loss 0.024075, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 018 Default Step Loss 0.000001, Argument Step Loss 0.024046, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 019 Default Step Loss 0.000001, Argument Step Loss 0.024029, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 020 Default Step Loss 0.000001, Argument Step Loss 0.024018, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 021 Default Step Loss 0.000001, Argument Step Loss 0.024008, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 022 Default Step Loss 0.000001, Argument Step Loss 0.023999, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 023 Default Step Loss 0.000001, Argument Step Loss 0.023991, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 04 Step 024 Default Step Loss 0.000001, Argument Step Loss 0.023983, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
```

Figure 12: Model Running Logs 03

```
Epoch 05 Step 1194 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1195 Default Step Loss 0.000000, Argument Step Loss 0.024298, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1196 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1197 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1198 Default Step Loss 0.000000, Argument Step Loss 0.024234, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1199 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1200 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1201 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1202 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1203 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1204 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1205 Default Step Loss 0.000000, Argument Step Loss 0.024287, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1206 Default Step Loss 0.000000, Argument Step Loss 0.024082, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1207 Default Step Loss 0.000000, Argument Step Loss 0.024019, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1208 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1209 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1210 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1211 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1212 Default Step Loss 0.000000, Argument Step Loss 0.024218, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1213 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1214 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1215 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1216 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1217 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1218 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1219 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1220 Default Step Loss 0.000000, Argument Step Loss 0.024239, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1221 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1222 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1223 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1224 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1225 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1226 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1227 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1228 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1229 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1230 Default Step Loss 0.000000, Argument Step Loss 0.024244, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1231 Default Step Loss 0.000000, Argument Step Loss 0.024045, Term: 0.866667, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1232 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1233 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1234 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Epoch 05 Step 1235 Default Step Loss 0.000000, Argument Step Loss 0.000000, Term: 1.000000, Prog: 1.000000, A0: 1.000000, A1: 1.000000, A2: 1.0
Model generation complete!

(NPI_Arch_Keras) C:\workspaces\NPL\project\npi-tf>
```

Figure 13: Model Running Logs 04

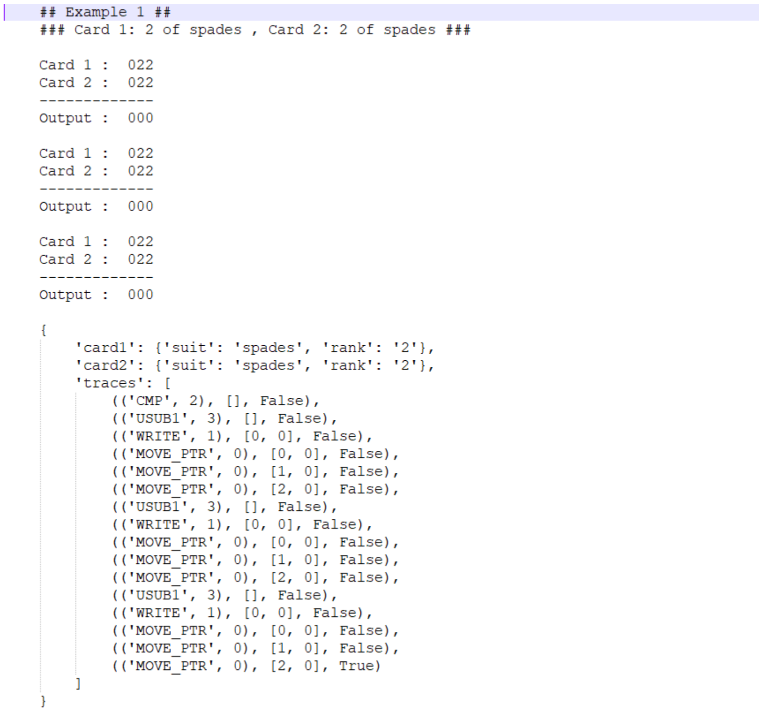
4 Discussion

The main goal of the project was to tweak the NPI architecture to make it learn addition, merge sort and card matching. In this project we take an NPI architecture which had an LSTM as the core. Of the three mentioned tasks, addition and card matching were done successfully. The important results are summerized below.

The most challenging thing we had to deal with was making the NPI architecture understand what we were doing. In the beginning we were working on the simple problem of making the architecture understand how to do addition. But when it came to card matching we had to make different traces for it. Building totally different traces for the card matching problem was a different problem altogether. After making the traces for card matching we thought that the same architecture of the addition. But after trying seeing the training errors we came to an understanding that we again need to tweak the NPI architecture a bit. But the problem wasn't solved there, it wasn't easy to find the proper combination of traces and NPI architecture to get the testing error as low as possible. When we thought we were done with card matching, we had to face the problems of making the merge sort work. Figuring out the merge sort was a different problem altogether, because here the list size was not constant. So for different splits we had to write different splits. Then as the split lists were also of different sizes, this added to the complexity of building the traces. The traces it-self were the hardest part of this project. The addition task was the easiest of the three tasks and its were we had the most success.

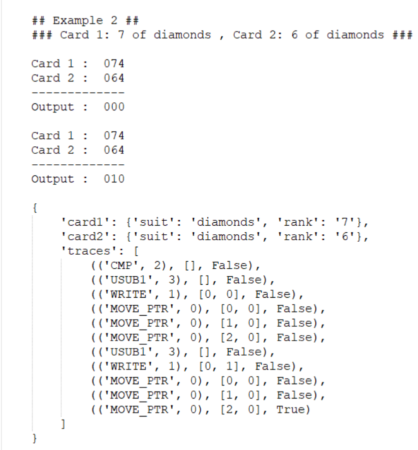
4.1 Model performance

The execution traces executed successfully for all the tasks.



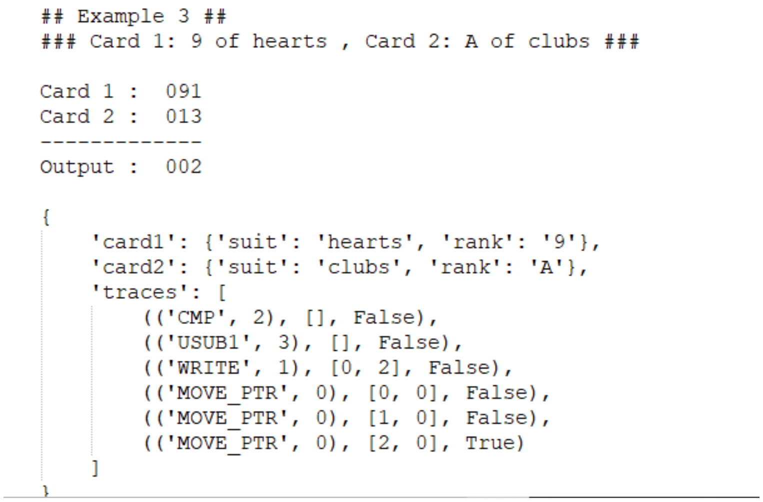
```
## Example 1 ##
### Card 1: 2 of spades , Card 2: 2 of spades ###

Card 1 :  022
Card 2 :  022
-------------
Output :  000

Card 1 :  022
Card 2 :  022
-------------
Output :  000

Card 1 :  022
Card 2 :  022
-------------
Output :  000

{
    'card1': {'suit': 'spades', 'rank': '2'},
    'card2': {'suit': 'spades', 'rank': '2'},
    'traces': [
        (('CMP', 2), [], False),
        (('USUB1', 3), [], False),
        (('WRITE', 1), [0, 0], False),
        (('MOVE_PTR', 0), [0, 0], False),
        (('MOVE_PTR', 0), [1, 0], False),
        (('MOVE_PTR', 0), [2, 0], False),
        (('USUB1', 3), [], False),
        (('WRITE', 1), [0, 0], False),
        (('MOVE_PTR', 0), [0, 0], False),
        (('MOVE_PTR', 0), [1, 0], False),
        (('MOVE_PTR', 0), [2, 0], False),
        (('USUB1', 3), [], False),
        (('WRITE', 1), [0, 0], False),
        (('MOVE_PTR', 0), [0, 0], False),
        (('MOVE_PTR', 0), [1, 0], False),
        (('MOVE_PTR', 0), [2, 0], True)
    ]
}
```

Figure 14: Result 01



```
## Example 2 ##
### Card 1: 7 of diamonds , Card 2: 6 of diamonds ###

Card 1 :  074
Card 2 :  064
-------------
Output :  000

Card 1 :  074
Card 2 :  064
-------------
Output :  010

{
    'card1': {'suit': 'diamonds', 'rank': '7'},
    'card2': {'suit': 'diamonds', 'rank': '6'},
    'traces': [
        (('CMP', 2), [], False),
        (('USUB1', 3), [], False),
        (('WRITE', 1), [0, 0], False),
        (('MOVE_PTR', 0), [0, 0], False),
        (('MOVE_PTR', 0), [1, 0], False),
        (('MOVE_PTR', 0), [2, 0], False),
        (('USUB1', 3), [], False),
        (('WRITE', 1), [0, 1], False),
        (('MOVE_PTR', 0), [0, 0], False),
        (('MOVE_PTR', 0), [1, 0], False),
        (('MOVE_PTR', 0), [2, 0], True)
    ]
}
```

Figure 15: Result 02



```
## Example 3 ##
### Card 1: 9 of hearts , Card 2: A of clubs ###

Card 1 :  091
Card 2 :  013
-------------
Output :  002

{
    'card1': {'suit': 'hearts', 'rank': '9'},
    'card2': {'suit': 'clubs', 'rank': 'A'},
    'traces': [
        (('CMP', 2), [], False),
        (('USUB1', 3), [], False),
        (('WRITE', 1), [0, 2], False),
        (('MOVE_PTR', 0), [0, 0], False),
        (('MOVE_PTR', 0), [1, 0], False),
        (('MOVE_PTR', 0), [2, 0], True)
    ]
}
```

Figure 16: Result 03

4.2 Ablation Studies

In this project we tried to do ablation on addition task. But due to the complexity of the architecture we were getting inconsistent results which did not conform to our understanding. Hence we did not continue with ablation.

4.3 Future work

If this project were to be continued in the future, following are some of the things we would give the highest priority:

- Building proper traces for Merge Sort.
- Tweaking the current NPI architecture for it to learn Merge Sort.
- In the beginning we thought that we will try a one hat fits all approach,ie, for the NPI architecture we wanted to have an generalized NPI to do all the three tasks in one architecture. So if this project was continued in the future, this is something we would like to work on the this aspect particularly.
- The card matching was done on a smaller scale, in future we want to do it on a larger scale so that that card matching is done on a bigger stack of cards.

5 Challenges

The first challenge we faced was with the libraries to use for building the NPI core. We started by building the core using vanilla Tensor flow. However, this initial attempt failed due to lack of knowledge with the library coupled with the complex nature of the architecture. We later tried to build the core using TFLearn but eventually settled with building the core using Keras. We have adapted our code based on [10]

The second challenge was compilation of the traces. We were able to come up with the traces for card pattern matching but as discussed earlier due to recursive nature of the merge sort we could not build its traces.

Finally, for the card pattern matching problem we were able to successfully train the model. However, we were not able to run our predictions on the training data set.

Given time and more familiarity with the helper libraries, we should be able to come up with a working prediction for the models.

References

[1] Jonathon Cai, Richard Shin and Dawn Son Making Neural Programming Architectures generalize via recursion *In ICLR*, 2017.

`https://arxiv.org/pdf/1704.06611.pdf`

[2] Scott Reed and Nando de Freitas Neural programmer-interpreters *In ICLR*, 2016. `https://arxiv.org/pdf/1511.06279.pdf`

`https://arxiv.org/pdf/1511.06279.pdf`

[3] George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.

`https://ieeexplore.ieee.org/document/6981055`

[4] George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.

`https://ieeexplore.ieee.org/document/7008025`

[5] Alex Graves, Greg Wayne, Ivo Danihelka. Neural Turing Machines.

`https://arxiv.org/pdf/1410.5401.pdf`

[6] Arvind Neelakantan, Quoc V. Le, Ilya Sutskever Neural Programmer: Inducing Latent Program with Gradient Descent In *Published as a conference paper at ICLR, 2016*

`https://arxiv.org/pdf/1511.04834.pdf`

[7] Jared Sylvester, James Reggia. Engineering neural systems for high-level problem solving (2016).

`http://www.jsylvest.com/papers/2016-sylvester-reggia-nn.pdf`

[8] Sepp Hochreiter, Jiirgen Schmidhuber. LSTM can solve hard log time lag problems.

`http://papers.nips.cc/paper/1215-lstm-can-solve-hard-long-time-lag-problems.pdf`

[9] Illia Polosukhin. Review of Neural Programmer-Interpreters. Published as a blog post on Medium.(2018)

`https://medium.com/near-ai/review-of-neural-programmer-interpreters-854a14a494fb`

[10] Sidd Karamcheti. Neural Programmer-Interpreters Published on github profile

`https://github.com/siddk/npi`

[11] Authors Program induction using Neural Programmer-Interpreters Published on github profile

`https://github.com/riteshgn/neural-learning-NPI--python`