



M. Willis Monroe<sup>1</sup>

willis.monroe@unb.ca

Logan Born<sup>2</sup>

loborn@sfu.ca

Kathryn Kelley<sup>3</sup>

kathrynerin.kelley@unibo.it

Anoop Sarkar<sup>2</sup>

anoop@cs.sfu.ca

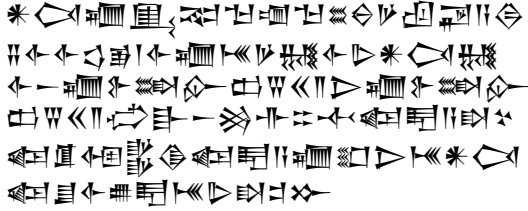
<sup>1</sup>University of New Brunswick  
Department of Historical Studies

<sup>2</sup>Simon Fraser University  
School of Computing Science

<sup>3</sup>Università di Bologna  
Dipartimento di Filologia Classica e Italianistica

## Abstract

This paper announces the discovery of the use of neural nets almost 4,000 years before their use in the modern era. Newly discovered tablets preserve a perceptron used for calculating the numbers on Plimpton 322, the most important object in the history of mathematics. The native programming language used by the ancient Babylonian “cuneogrammers” uses sexagesimal numbering leading to some “weirdness”.



## 1 Introduction




The history of math is long, but the history of programming is longer. Cuneiform, arguably the first writing in the world, is known for its sheep receipts and beer ration lists as well as complaints about substandard copper (Oppenheim, 1954) and women complaining at each other (Matuszak, 2020). This article adds neural network programming to that vaunted list of human achievements. A set of newly discovered cuneiform tablets preserve the mechanism for performing simple neural network calculations. These methods, it seems, were used to calculate the lengths of triangles; a well known example of this exercise is preserved on the tablet known as Plimpton 322<sup>1</sup>. It is remarkable that no his-

<sup>1</sup>An important and real description of this interesting object is found in Robson (2002).

torian of math or cuneiform scholar has ever consider this possibility. This paper covers the background, a description of the cuneogramming language, and includes a facsimile copy of the most important tablet as an appendix.

Unfortunately, the hardware required to execute these programs (i.e. a living Babylonian mathematician) has not been adequately preserved, but we have managed to write a Python library which emulates it.<sup>2</sup> The assumption is that these calculations were done by hand in their copious free time between inventing the wheel and the concept of zero. While the actual output of these tablets is relatively simple by modern standards, the implications of this discovery are profound. Future work will explore how these techniques could have been used in the realm of astronomical calculation and elucidate the full extent of Babylonian computational prowess.








## 2 Description of the Language

Programs in  (EME.ŠID.A “language of counting”) follow a tabular structure with three main sections: (i) a header, denoted by  (DUB “tablet”), (ii) a sequence of instructions, and (iii) a colophon detailing the tablet’s authorship. Each instruction spans four columns, which we have taken to calling the *arguments*, *opcode*, *destination*, and *line number*. These columns are usually tab-separated, though in a few documents they are TAB-separated (using the cuneiform sign TAB ). Instructions are grouped into blocks by means of horizontal lines.

**Arguments** An instruction’s arguments may be numbers, register addresses, or a combination

<sup>2</sup>[github.com/MrLogarithm/emeszida](https://github.com/MrLogarithm/emeszida)

of the two. Numbers are encoded following standard Babylonian conventions, with  $\text{!}$  denoting the radix point which separates the integer part from a following fraction. There is an explicit representation for zero ( $\text{!}$ ), making these tablets some of the earliest unambiguous examples of the mathematical concept of zero.

If an instruction has multiple arguments, these must be delimited by a wide space (distinct from the short space used to separate groups of digits within a number), or by one of the phrases *a-na*  “to” or *i-na*  “from”. By convention, the choice of delimiter depends on the instruction’s opcode (see below), with multiplication preferring space delimiters, addition preferring , and subtraction preferring . There is no mechanism to enforce these conventions, but we recommend following them because    is hard enough to read at the best of times.

**Opcodes** Each instruction has a single opcode belonging to the following vocabulary:

- 卜 ME, “to be”
- 𠄎 𠄎 𠄎 *ta-mar*, “you will see”
- 𠄎 𠄎 NIGIN.NA, “start again”
- 𠄎 𠄎 𠄎 𠄎 TUKUM.BI DIRIG, “if it exceeds”
- 𠄎 𠄎 𠄎 𠄎 TUKUM.BI SIG, “if it it is weak”






The language does not appear to have any kind of binary division operator. Rather, a unary  $\nabla$  operator was used to find the reciprocal of the denominator, which was then multiplied by the numerator using the binary  $\bowtie$  operator.





 includes three types of control-flow instructions.  functions like GOTO, and jumps the program counter to the specified line number.  functions like the x86 jz instruction, and jumps to the specified line number if and only if its argument is zero.  is similar, and jumps if the argument is greater than zero.



Figure 1: BM 34580, courtesy of the Trustees of the British Museum, CC BY-NC-SA 4.0

need not even be distinct. For example, most lines of the perceptron tablet are labeled with the number 𐍪 (zero); only lines that are the destination of some control-flow instruction receive non-zero identifiers.

## 2.1 Fractional Indexing

Both line numbers and register addresses in 𐍪𐍪𐍪𐍪 can have fractional parts. The original scribes seem to have exploited this fact to establish non-overlapping “namespaces” for the different parts of their code. For example, in the perceptron tablet, all of the model parameters are stored in addresses with integer part 1; the program inputs all have integer part 2; the matrix multiplication subroutine uses addresses with integer part 3; and so on. The fractional parts of register addresses also appear to follow some standard conventions, with the  $X;0$  register typically storing a subroutine’s return address, while  $X;1$  onward were used for its arguments.

The perceptron tablet also uses fractional register addresses to perform a kind of multi-dimensional array indexing. As an example, the first layer of the perceptron has a  $50 \times 2$  weight matrix, and this is stored in registers 1;0,0,0 through 1;0,49,1. The integer part of these addresses denotes the “data” portion of memory; the first digit after the radix point identifies this as the 0th model parameter; and the second and third digits can be treated as a pair of indices ranging from 0–49 and 0–1 respectively. To ac-

cess a specific element in this matrix, the scribes use repeated division by 60 to implement a kind of “bit-shift” instruction, in order to shift integer indices into the correct positions after the radix point. By adding the bit-shifted element indices (e.g. 0;0,4,7 for the element in row 5, column 8) to a pointer to the top corner of the matrix (1;0,0,0) they obtain the address of the desired element (1;0,4,7).

Notably, this practice limits the size of their model parameters to at most  $60 \times 60$ , as for larger values the addresses would carry over to higher digits and thus begin to overwrite one another. This limitation may explain why AI never made waves in Babylonian society, as their models were all too small to be truly revolutionary.

## 3 Description of the Texts

The cuneogramming corpus contains numerous fragments implementing recognizable procedures such as bit-shifting, populating an array, computing dot products, and so on. However, only a single text is known to have been preserved in its entirety.<sup>3</sup> Spanning close to 1700 lines, this impressive text is divided into five sections implementing what modern readers will immediately recognize to be a multi-layer perceptron. The first section straightforwardly defines a matrix-

<sup>3</sup>All of the known texts are reproduced in [github.com/MrLogarithm/emeszida/tree/main/programs](https://github.com/MrLogarithm/emeszida/tree/main/programs), and the long text is reproduced in facsimile in the appendix of this work.

multiplication subroutine. This is called by a subroutine defined in section two, which applies each layer of the perceptron to a given input, and applies a ReLU-style activation between each pair of layers. Section three implements the tablet’s “main” method, which loads an input to memory, calls the perceptron subroutine, and prints the resulting output. Section four loads the model parameters, which appear to comprise weight matrices of sizes  $50 \times 2$ ,  $25 \times 50$ , and  $1 \times 25$ , plus bias vectors of sizes 50, 25, and 1 respectively. The final section lists pairs of inputs, whose values (incredibly!) correspond to the second and third columns of Plimpton 322. Interestingly, this section very closely resembles the tables of parameters found in later astronomical calculations (see Figure 1).

When the program is executed, it produces a single numeric output for each input pair. These outputs correspond remarkably closely to the values in the first column of Plimpton 322, as demonstrated in Figure 2. The correspondence is not perfect, however, and the values which should match lines 1, 5, and 6 of Plimpton 322 are significantly larger than expected. This implies that, although the code on this tablet is clearly *related* to Plimpton 322, it could not have been used to directly populate the table in that text. Perhaps the outputs from this model were refined in some later step to produce the more exact ratios in the Plimpton text, or perhaps the Babylonians were disillusioned by the imprecision of their machine learning models and simply abandoned them for tried-and-true manual methods. Given how miniscule the cuneogramming corpus is relative to the larger body of Babylonian administrative writing, we lean towards the latter explanation.

## 4 Implications and Future Work

This completely rewrites the history of modern computing and artificial intelligence. In addition to focusing on important figures like Ada Lovelace and Grace Hopper we should be looking at pioneers thousands of years earlier like Enheduana the world’s first known author (Helle, 2023) and now the world’s first known programmer.

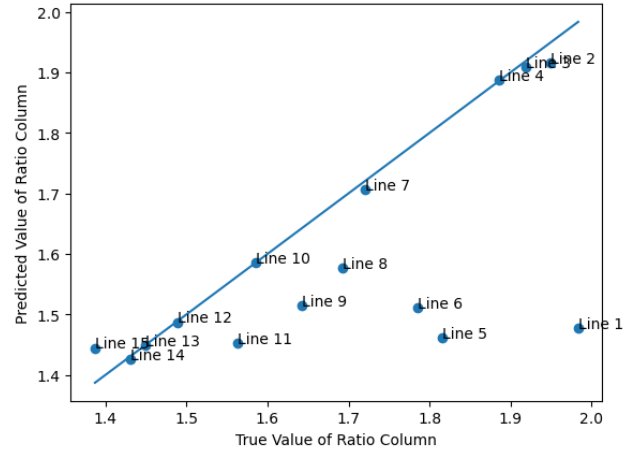


Figure 2: Outputs from the perceptron tablet vs. true value of each line from column 1 of Plimpton 322. If a point falls upon the line, the model output for that line exactly equals the value on Plimpton 322.

Other ancient corpora which have resisted decipherment, and which boast a similar numeric component, may represent additional examples of ancient programming traditions (Kelley et al., 2022).

## Acknowledgments

The search for these tablets and our effort to understand them was inspired by Ramsey Nasser’s **قلب** programming language (<https://github.com/nasser/--->).

## References

- Sophus Helle. 2023. *Enheduana: the complete poems of the world’s first author*. Yale University Press, New Haven, CT.
- Kathryn Kelley, Logan Born, M. Willis Monroe, and Anoop Sarkar. 2022. [Image-aware language modeling for proto-elamite](#). *Lingue e linguaggio*, (2):261–294.
- Jana Matuszak. 2020. “*Und du, du bist eine Frau?!: Editio princeps und Analyse des sumerischen Streitgesprächs ‘Zwei Frauen B’*”. De Gruyter.
- A. L. Oppenheim. 1954. [The seafaring merchants of Ur](#). *Journal of the American Oriental Society*, 74(1):6–17.
- Eleanor Robson. 2002. [Words and pictures: New light on Plimpton 322](#). *The American Mathematical Monthly*, 109(2):105–120.

## **A   Appendix**

The following pages reproduce the perceptron tablet in its entirety.



[illegible][illegible]

一 二 三 四 五 六 七 八 九 十 十一 十二 十三 十四 十五 十六 十七 十八 十九 二十 二十一 二十二 二十三 二十四 二十五 二十六 二十七 二十八 二十九 三十 三十一 三十二 三十三 三十四 三十五 三十六 三十七 三十八 三十九 四十 四十一 四十二 四十三 四十四 四十五 四十六 四十七 四十八 四十九 五十 五十一 五十二 五十三 五十四 五十五 五十六 五十七 五十八 五十九 六十 六十一 六十二 六十三 六十四 六十五 六十六 六十七 六十八 六十九 七十 七十一 七十二 七十三 七十四 七十五 七十六 七十七 七十八 七十九 八十 八十一 八十二 八十三 八十四 八十五 八十六 八十七 八十八 八十九 九十 九十一 九十二 九十三 九十四 九十五 九十六 九十七 九十八 九十九 一百

[illegible][illegible][illegible]

[illegible][illegible]



[illegible][illegible][illegible]

[illegible][illegible]

[illegible][illegible][illegible]

[illegible]

一  
 二  
 三  
 四  
 五  
 六  
 七  
 八  
 九  
 十  
 十一  
 十二  
 十三  
 十四  
 十五  
 十六  
 十七  
 十八  
 十九  
 二十  
 二十一  
 二十二  
 二十三  
 二十四  
 二十五  
 二十六  
 二十七  
 二十八  
 二十九  
 三十  
 三十一  
 三十二  
 三十三  
 三十四  
 三十五  
 三十六  
 三十七  
 三十八  
 三十九  
 四十  
 四十一  
 四十二  
 四十三  
 四十四  
 四十五  
 四十六  
 四十七  
 四十八  
 四十九  
 五十  
 五十一  
 五十二  
 五十三  
 五十四  
 五十五  
 五十六  
 五十七  
 五十八  
 五十九  
 六十  
 六十一  
 六十二  
 六十三  
 六十四  
 六十五  
 六十六  
 六十七  
 六十八  
 六十九  
 七十  
 七十一  
 七十二  
 七十三  
 七十四  
 七十五  
 七十六  
 七十七  
 七十八  
 七十九  
 八十  
 八十一  
 八十二  
 八十三  
 八十四  
 八十五  
 八十六  
 八十七  
 八十八  
 八十九  
 九十  
 九十一  
 九十二  
 九十三  
 九十四  
 九十五  
 九十六  
 九十七  
 九十八  
 九十九  
 一百

[illegible]

[illegible][illegible][illegible]

[illegible][illegible][illegible]

*[The page contains dense, illegible vertical Chinese text arranged in columns.]*

[illegible][illegible]

[illegible][illegible][illegible]



[illegible][illegible][illegible]

*[The page contains dense, illegible vertical columns of Chinese characters.]*

[illegible][illegible]

[illegible][illegible][illegible]

[illegible][illegible]

A vertical strip of 60 small icons arranged in two columns of 30 each. The left column contains various symbols including musical notes, geometric shapes, and abstract patterns. The right column contains similar symbols, some with additional decorative elements like dots or lines.

*[The page contains dense, illegible vertical Chinese text arranged in columns.]*

[illegible][illegible]

[illegible]

一  
二  
三  
四  
五  
六  
七  
八  
九  
十  
十一  
十二  
十三  
十四  
十五  
十六  
十七  
十八  
十九  
二十  
二十一  
二十二  
二十三  
二十四  
二十五  
二十六  
二十七  
二十八  
二十九  
三十  
三十一  
三十二  
三十三  
三十四  
三十五  
三十六  
三十七  
三十八  
三十九  
四十  
四十一  
四十二  
四十三  
四十四  
四十五  
四十六  
四十七  
四十八  
四十九  
五十  
五十一  
五十二  
五十三  
五十四  
五十五  
五十六  
五十七  
五十八  
五十九  
六十  
六十一  
六十二  
六十三  
六十四  
六十五  
六十六  
六十七  
六十八  
六十九  
七十  
七十一  
七十二  
七十三  
七十四  
七十五  
七十六  
七十七  
七十八  
七十九  
八十  
八十一  
八十二  
八十三  
八十四  
八十五  
八十六  
八十七  
八十八  
八十九  
九十  
九十一  
九十二  
九十三  
九十四  
九十五  
九十六  
九十七  
九十八  
九十九  
一百

[illegible]

[illegible][illegible][illegible]

