Approximation model based on LSTM for predicting the next prime number in an infinite sequence

Petr Pylov[1*], *Roman* Maitak[1], and *Andrey* Protodyakonov[1]

[1]T.F. Gorbachev Kuzbass State Technical University, 28 Vesennya st., 650000 Kemerovo, Russia

Abstract. Prime numbers are a special set of natural numbers that have captured the attention of mathematicians since ancient times. As prime numbers are a fundamental component in many areas of mathematics, they have naturally found wide applications in various fields of knowledge, such as cryptography. The goal of all researchers is to discover the distributional relationships within this infinite set of numbers or, at the very least, to create a mathematical model for predicting the next prime number in a diverging sequence. This article is dedicated to an attempt at solving this problem based on a deep learning model - Long Short-Term Memory (LSTM) neural network.

1 Introduction

Natural numbers that are different from one and have only two divisors - the number itself and one - are commonly referred to as prime numbers. The study of prime numbers dates back to ancient times, with the first mentions of them appearing as early as the Upper Paleolithic era [1].

However, despite the fact that scientists have been studying prime numbers for over five thousand years, it is important to note that a formula relating the numerical sequence and allowing the generation of the next prime number in the series has not yet been obtained.

Given the fact that the sequence of prime numbers is infinite, the task of determining the next prime number in the sequence remains relevant and can be approached through two distinct methods:

1. Deterministic primality tests: These tests aim to precisely determine whether a given number is prime or composite. By establishing the membership of a number in one of the two categories, these tests provide an exact result.
2. Probabilistic primality tests: These tests estimate the probability of a number belonging to the prime number category and therefore cannot guarantee complete reliability.

While the first category of tests allows for a definitive classification of numbers, it requires significant time and computational resources to execute the testing procedure. The second category reduces the time and computational requirements by providing a predictive

* Corresponding author: pylovpa@kuzstu.ru

result at the cost of some reliability. To confirm or refute the prediction, an actual primality test must be performed in either case.

The significant time investments required to compute the next prime number in an infinite sequence necessitate the development of a prediction algorithm that combines the speed of probabilistic primality tests with a level of reliability approaching that of deterministic primality tests. As one variation of such an approximation model, the authors of this article have devised a deep learning model based on the logic of Long Short-Term Memory (LSTM) neural networks.

2 Materials and Methods

The choice of Long Short-Term Memory (LSTM) models is not accidental but justified by a specific characteristic of the prime number sequence. To formalize this characteristic, a mathematical operation of taking the difference between each pair of adjacent elements in the numerical sequence should be performed. In other words, we obtain a sequence of differences between prime numbers that exhibits certain properties explicitly derived from the properties of the prime number sequence:

1. The sequence of differences will be non-decreasing and composed of natural numbers;
2. The sequence of differences is unbounded from above and bounded from below;
3. The series of numbers will be infinite, indicating that the sequence itself is divergent.

Having defined the properties of the data, we can now proceed to examine their peculiarities. Figure 1 illustrates a fragment of the prime number sequence.

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 10
9, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 23
9, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 37
9, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 491, 499, 503, 509, 52
1, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 659, 66
1, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 82
7, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 99
1, 997, 1009, 1013, 1019, 1021, 1031, 1033, 1039, 1049, 1051, 1061, 1063, 1069, 1087, 1091, 1093, 1097, 1103, 1109,
1117, 1123, 1129, 1151, 1153, 1163, 1171, 1181, 1187, 1193, 1201, 1213, 1217, 1223, 1229, 1231, 1237, 1249, 1259, 1
277, 1279, 1283, 1289, 1291, 1297, 1301, 1303, 1307, 1319, 1321, 1327, 1361, 1367, 1373, 1381, 1399, 1409, 1423, 14
27, 1429, 1433, 1439, 1447, 1451, 1453, 1459, 1471, 1481, 1483, 1487, 1489, 1493, 1499, 1511, 1523, 1531, 1543, 154
9, 1553, 1559, 1567, 1571, 1579, 1583, 1597, 1601, 1607, 1609, 1613, 1619, 1621, 1627, 1637, 1657, 1663, 1667, 166
9, 1693, 1697, 1699, 1709, 1721, 1723, 1733, 1741, 1747, 1753, 1759, 1777, 1783, 1787, 1789, 1801, 1811, 1823, 183
1, 1847, 1861, 1867, 1871, 1873, 1877, 1879, 1889, 1901, 1907, 1913, 1931, 1933, 1949, 1951, 1973, 1979, 1987, 199
3, 1997, 1999, 2003, 2011, 2017, 2027, 2029, 2039, 2053, 2063, 2069, 2081, 2083, 2087, 2089, 2099, 2111, 2113, 212
9, 2131, 2137, 2141, 2143, 2153, 2161, 2179, 2203, 2207, 2213, 2221, 2237, 2239, 2243, 2251, 2267, 2269, 2273, 228
1, 2287, 2293, 2297, 2309, 2311, 2333, 2339, 2341, 2347, 2351, 2357, 2371, 2377, 2381, 2383, 2389, 2393, 2399, 241
1, 2417, 2423, 2437, 2441, 2447, 2459, 2467, 2473, 2477, 2503, 2521, 2531, 2539, 2543, 2549, 2551, 2557, 2579, 259
1, 2593, 2609, 2617, 2621, 2633, 2647, 2657, 2659, 2663, 2671, 2677, 2683, 2687, 2689, 2693, 2699, 2707, 2711, 271
3, 2719, 2729, 2731, 2741, 2749, 2753, 2767, 2777, 2789, 2791, 2797, 2801, 2803, 2819, 2833, 2837, 2843, 2851, 285
7, 2861, 2879, 2887, 2897, 2903, 2909, 2917, 2927, 2939, 2953, 2957, 2963, 2969, 2971, 2999, 3001, 3011, 3019, 302
3, 3037, 3041, 3049, 3061, 3067, 3079, 3083, 3089, 3109, 3119, 3121, 3137, 3163, 3167, 3169, 3181, 3187, 3191, 320
3, 3209, 3217, 3221, 3229, 3251, 3253, 3257, 3259, 3271, 3299, 3301, 3307, 3313, 3319, 3323, 3329, 3331, 3343, 334
7, 3359, 3361, 3371, 3373, 3389, 3391, 3407, 3413, 3433, 3449, 3457, 3461, 3463, 3467, 3469, 3491, 3499, 3511, 351
7, 3527, 3529, 3533, 3539, 3541, 3547, 3557, 3559, 3571, 3581, 3583, 3593, 3607, 3613, 3617, 3623, 3631, 3637, 364
3, 3659, 3671, 3673, 3677, 3691, 3697, 3701, 3709, 3719, 3727, 3733, 3739, 3761, 3767, 3769, 3779, 3793, 3797, 380
3, 3821, 3823, 3833, 3847, 3851, 3853, 3863, 3877, 3881, 3889, 3907, 3911, 3917, 3919, 3923, 3929, 3931, 3943, 394
7, 3967, 3989, 4001, 4003, 4007, 4013, 4019, 4021, 4027, 4049, 4051, 4057, 4073, 4079, 4091, 4093, 4099, 4111, 412
7, 4129, 4133, 4139, 4153, 4157, 4159, 4177, 4201, 4211, 4217, 4219, 4229, 4231, 4241, 4243, 4253, 4259, 4261, 427
1, 4273, 4283, 4289, 4297, 4327, 4337, 4339, 4349, 4357, 4363, 4373, 4391, 4397, 4409, 4421, 4423, 4441, 4447, 445
```

Fig. 1. Fragment of the prime number sequence

Figure 2 depicts a fragment of the difference between adjacent elements in the first sequence.

[1, 2, 2, 4, 2, 4, 2, 4, 6, 2, 6, 4, 2, 4, 6, 6, 2, 6, 4, 2, 6, 4, 6, 8, 4, 2, 4, 2, 4, 14, 4, 6, 2, 10, 2, 6, 6,
4, 6, 6, 2, 10, 2, 4, 2, 12, 12, 4, 2, 4, 6, 2, 10, 6, 6, 6, 2, 6, 4, 2, 10, 14, 4, 2, 4, 14, 6, 10, 2, 4, 6, 8, 6,
6, 4, 6, 8, 4, 8, 10, 2, 10, 2, 6, 4, 6, 8, 4, 2, 4, 12, 8, 4, 8, 4, 6, 12, 2, 18, 6, 10, 6, 6, 2, 6, 10, 6, 6, 2,
6, 6, 4, 2, 12, 10, 2, 4, 6, 6, 2, 12, 4, 6, 8, 10, 8, 10, 8, 6, 6, 4, 8, 6, 4, 8, 4, 14, 10, 12, 2, 10, 2, 4, 2, 1
0, 14, 4, 2, 4, 14, 4, 2, 4, 20, 4, 8, 10, 8, 4, 6, 6, 14, 4, 6, 6, 8, 6, 12, 4, 6, 2, 10, 2, 6, 10, 2, 10, 2, 6, 1
8, 4, 2, 4, 6, 6, 8, 6, 6, 22, 2, 10, 8, 10, 6, 6, 8, 12, 4, 6, 6, 2, 6, 12, 10, 18, 2, 4, 6, 2, 6, 4, 2, 4, 12, 2,
6, 34, 6, 6, 8, 18, 10, 14, 4, 2, 4, 6, 8, 4, 2, 6, 12, 10, 2, 4, 2, 4, 6, 12, 12, 8, 12, 6, 4, 6, 8, 4, 8, 4, 14,
4, 6, 2, 4, 6, 2, 6, 10, 20, 6, 4, 2, 24, 4, 2, 10, 12, 2, 10, 8, 6, 6, 6, 18, 6, 4, 2, 12, 10, 12, 8, 16, 14, 6,
4, 2, 4, 2, 10, 12, 6, 6, 18, 2, 16, 2, 22, 6, 8, 6, 4, 2, 4, 8, 6, 10, 2, 10, 14, 10, 6, 12, 2, 4, 2, 10, 12, 2, 1
6, 2, 6, 4, 2, 10, 8, 18, 24, 4, 6, 8, 16, 2, 4, 8, 16, 2, 4, 8, 6, 6, 4, 12, 2, 22, 6, 2, 6, 4, 6, 14, 6, 4, 2, 6,
4, 6, 12, 6, 6, 14, 4, 6, 12, 8, 6, 4, 26, 18, 10, 8, 4, 6, 2, 6, 22, 12, 2, 16, 8, 4, 12, 14, 10, 2, 4, 8, 6, 6,
4, 2, 4, 6, 8, 4, 2, 6, 10, 2, 10, 8, 4, 14, 10, 12, 2, 6, 4, 2, 16, 14, 4, 6, 8, 6, 4, 18, 8, 10, 6, 6, 8, 10, 12,
14, 4, 6, 6, 2, 28, 2, 10, 8, 4, 14, 4, 8, 12, 6, 12, 4, 6, 20, 10, 2, 16, 26, 4, 2, 12, 6, 4, 12, 6, 8, 4, 8, 22,
2, 4, 2, 12, 28, 2, 6, 6, 6, 4, 6, 2, 12, 4, 12, 2, 10, 2, 16, 2, 16, 6, 20, 16, 8, 4, 2, 4, 2, 22, 8, 12, 6, 10,
2, 4, 6, 2, 6, 10, 2, 12, 10, 2, 10, 14, 6, 4, 6, 8, 6, 6, 16, 12, 2, 4, 14, 6, 4, 8, 10, 8, 6, 6, 22, 6, 2, 10, 1
4, 4, 6, 18, 2, 10, 14, 4, 2, 10, 14, 4, 8, 18, 4, 6, 2, 4, 6, 2, 12, 4, 20, 22, 12, 2, 4, 6, 6, 2, 6, 22, 2, 6, 1
6, 6, 12, 2, 6, 12, 16, 2, 4, 6, 14, 4, 2, 18, 24, 10, 6, 2, 10, 2, 10, 2, 10, 6, 2, 10, 2, 10, 6, 8, 30, 10, 2, 1
0, 8, 6, 10, 18, 6, 12, 12, 2, 18, 6, 4, 6, 6, 18, 2, 10, 14, 6, 4, 2, 4, 24, 2, 12, 6, 16, 8, 6, 6, 18, 16, 2, 4,
6, 2, 6, 6, 10, 6, 12, 12, 18, 2, 6, 4, 18, 8, 24, 4, 2, 4, 6, 2, 12, 4, 14, 30, 10, 6, 12, 14, 6, 10, 12, 2, 4, 6,
8, 6, 10, 2, 4, 14, 6, 6, 4, 6, 2, 10, 2, 16, 12, 8, 18, 4, 6, 12, 2, 6, 6, 6, 28, 6, 14, 4, 8, 10, 8, 12, 18, 4,
2, 4, 24, 12, 6, 2, 16, 6, 6, 14, 10, 14, 4, 30, 6, 6, 6, 8, 6, 4, 2, 12, 6, 4, 2, 6, 22, 6, 2, 4, 18, 2, 4, 12, 2,
6, 4, 26, 6, 6, 4, 8, 10, 32, 16, 2, 6, 4, 2, 4, 2, 10, 14, 6, 4, 8, 10, 6, 20, 4, 2, 6, 30, 4, 8, 10, 6, 6, 8, 6,
12, 4, 6, 2, 6, 4, 6, 2, 10, 2, 16, 6, 20, 4, 12, 14, 28, 6, 20, 4, 18, 8, 6, 4, 6, 14, 6, 6, 10, 2, 10, 12, 8, 10,
2, 10, 8, 12, 10, 24, 2, 4, 8, 6, 4, 8, 18, 10, 6, 6, 2, 6, 10, 12, 2, 10, 6, 6, 6, 8, 6, 10, 6, 2, 6, 6, 6, 10, 8,
24, 6, 22, 2, 18, 4, 8, 10, 30, 8, 18, 4, 2, 10, 6, 2, 6, 4, 18, 8, 12, 18, 16, 6, 2, 12, 6, 10, 2, 10, 2, 6, 10, 1
4, 4, 24, 2, 16, 2, 10, 2, 10, 20, 4, 2, 4, 8, 16, 6, 6, 2, 12, 16, 8, 4, 6, 30, 2, 10, 2, 6, 4, 6, 6, 8, 6, 4, 12,
6, 8, 12, 4, 14, 12, 10, 24, 6, 12, 6, 2, 22, 8, 18, 10, 6, 14, 4, 2, 6, 10, 8, 6, 4, 6, 30, 14, 10, 2, 12, 10, 2,
16, 2, 18, 24, 18, 6, 16, 18, 6, 2, 18, 4, 6, 2, 10, 8, 10, 6, 6, 8, 4, 6, 2, 10, 2, 12, 4, 6, 6, 2, 12, 4, 14, 18,
4, 6, 20, 4, 8, 6, 4, 8, 4, 14, 6, 4, 14, 12, 4, 2, 30, 4, 24, 6, 6, 12, 12, 14, 6, 4, 2, 4, 18, 6, 12, 8, 6, 4, 1
2, 2, 12, 30, 16, 2, 6, 22, 14, 6, 10, 12, 6, 2, 4, 8, 10, 6, 6, 24, 14, 6, 4, 8, 12, 18, 10, 2, 10, 2, 4, 6, 20,
6, 4, 14, 4, 2, 4, 14, 6, 12, 24, 10, 6, 8, 10, 2, 30, 4, 6, 2, 12, 4, 14, 6, 34, 12, 8, 6, 10, 2, 4, 20, 10, 8, 1
6, 2, 10, 14, 4, 2, 12, 6, 16, 6, 8, 4, 8, 4, 6, 8, 6, 6, 12, 6, 4, 6, 6, 8, 18, 4, 20, 4, 12, 2, 10, 6, 2, 10, 12,
2, 4, 20, 6, 30, 6, 4, 8, 10, 12, 6, 2, 28, 2, 6, 4, 2, 16, 12, 2, 6, 10, 8, 24, 12, 6, 18, 6, 4, 14, 6, 4, 12, 8,
6, 12, 4, 6, 12, 6, 12, 2, 16, 20, 4, 2, 10, 18, 8, 4, 14, 4, 2, 6, 22, 6, 14, 6, 6, 10, 6, 2, 10, 2, 4, 2, 22, 2,
4, 6, 6, 12, 6, 14, 10, 12, 6, 8, 4, 36, 14, 12, 6, 4, 6, 2, 12, 6, 12, 16, 2, 10, 8, 22, 2, 12, 6, 4, 6, 18, 2, 1
2, 6, 4, 12, 8, 6, 12, 4, 6, 12, 6, 2, 12, 12, 4, 14, 6, 16, 6, 2, 10, 8, 18, 6, 34, 2, 28, 2, 22, 6, 2, 10, 12, 2,
6, 4, 8, 22, 6, 2, 10, 8, 4, 6, 8, 4, 12, 18, 12, 20, 4, 6, 6, 8, 4, 2, 16, 12, 2, 10, 8, 10, 2, 4, 6, 14, 12, 22,
8, 28, 2, 4, 20, 4, 2, 4, 14, 10, 12, 2, 12, 16, 2, 28, 8, 22, 8, 4, 6, 6, 14, 4, 8, 12, 6, 6, 4, 20, 4, 18, 2, 12,
6, 4, 6, 14, 18, 10, 8, 10, 32, 6, 10, 6, 6, 2, 6, 16, 6, 2, 12, 6, 28, 2, 10, 8, 16, 6, 8, 6, 10, 24, 20, 10, 2, 1
0, 2, 12, 4, 6, 20, 4, 2, 12, 18, 10, 2, 10, 2, 4, 20, 16, 26, 4, 8, 6, 4, 12, 6, 8, 12, 12, 6, 4, 8, 22, 2, 16, 1
4, 10, 6, 12, 12, 14, 6, 4, 20, 4, 12, 6, 2, 6, 6, 16, 8, 22, 2, 28, 8, 6, 4, 20, 4, 12, 24, 20, 4, 8, 10, 2, 16,
2, 12, 12, 34, 2, 4, 6, 12, 6, 6, 8, 6, 4, 2, 6, 24, 4, 20, 10, 6, 6, 14, 4, 6, 6, 2, 12, 6, 10, 2, 10, 6, 20, 4, 2
6, 4, 2, 6, 22, 2, 24, 4, 6, 2, 4, 6, 24, 6, 8, 4, 2, 34, 6, 8, 16, 12, 2, 10, 2, 10, 6, 8, 4, 8, 12, 22, 6, 14, 4,

Fig. 2. Fragment of the sequence of pairwise differences between prime numbers

By comparing the figures, it can be observed that the second sequence contains repeated elements. Furthermore, all of these repetitions are even numbers (except for the first difference), and the frequency of occurrence of the difference repetitions can be calculated. The ranking of the differences was performed based on the increasing magnitude of the difference and computed for prime number values limited to the first million values (Figure 3).

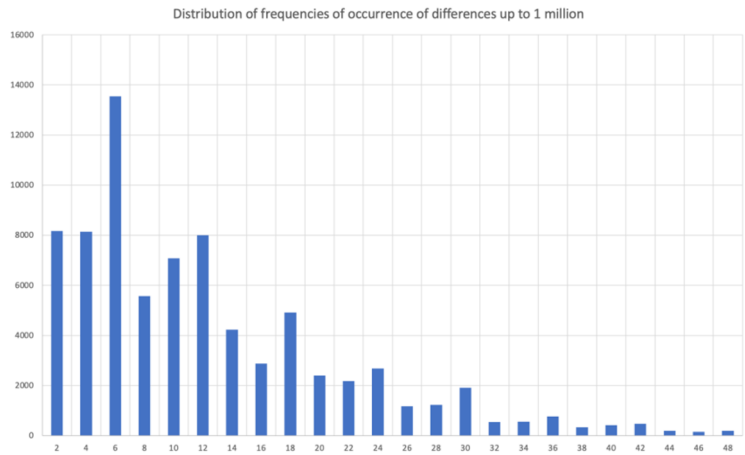


Fig. 3. Distribution of frequencies of repeated differences for prime number values limited to the first million

The comparison of differences reveals an important characteristic in the data: values that are multiples of 6 exhibit a higher frequency of repetitions.

To confirm this observation, let us consider the numerical sequence of differences between prime numbers based on the restriction of prime number series up to ten million values (Figure 4). If there is a change in the distribution trend, it will be reflected in the diagram itself.

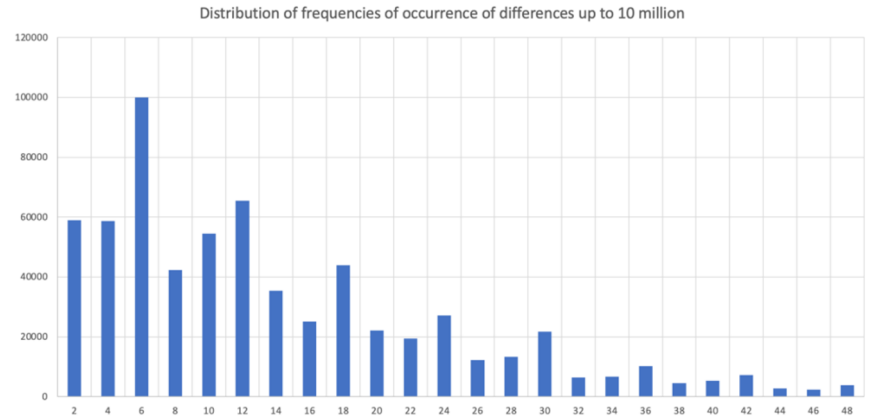


Fig. 4. The distribution of frequencies of repeated differences for prime numbers, limited to values exceeding ten million

As evident from the provided diagram (Figure 4), when restricting the sequence of prime numbers to the first ten million values, a trend of the highest popularity of differences divisible by 6 is maintained.

A similar distribution pattern is observed for differences between prime numbers up to one hundred million values (Figure 5).

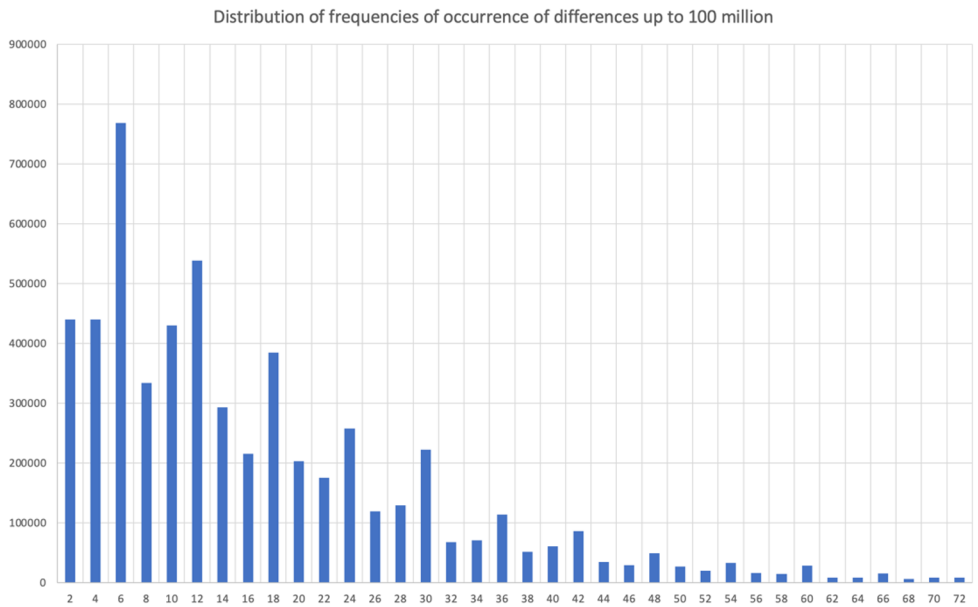


Fig. 5. The distribution of frequencies of repeated differences for prime numbers, limited to values exceeding one hundred million

This statement holds true for every "triplet" of difference values, thus playing a key role in deep learning models with a memory effect, with the long short-term memory (LSTM) neural network being a prominent example [2].

3 Results and discussion

The task of predicting the next prime number in an infinite sequence can be reformulated as the problem of determining the next difference between the positions of adjacent prime numbers. By formulating the task in this way, the main idea is preserved: knowing the current prime number allows us to add the predicted difference to it, thus obtaining the next prime number.

To implement a software solution, it is necessary to use a long short-term memory (LSTM) neural network. This is justified by the fact that the most frequently occurring differences, which are multiples of 6, can be more accurately predicted by a machine learning model if the algorithm learns the current distribution of these differences beforehand.

Furthermore, the primary motivation behind LSTM networks is to remember the context of the information it explores for as long as possible. The network will have to "figure out" what context needs to be remembered on its own [2]. The overall configuration of the long short-term memory network is shown in Figure 6.

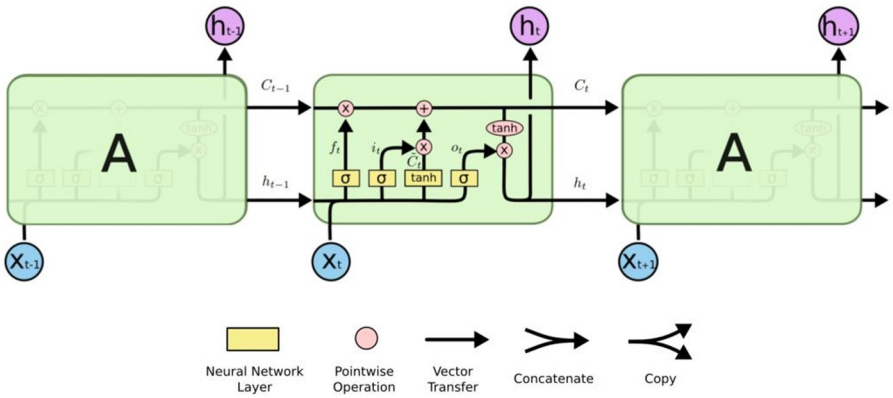
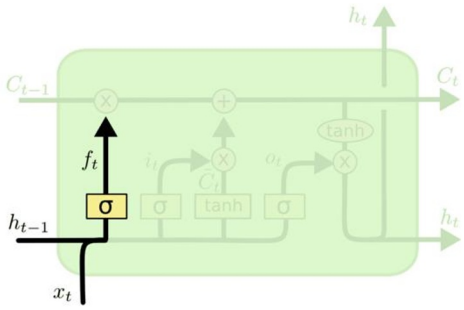


Fig. 6. Generalized LSTM Architecture

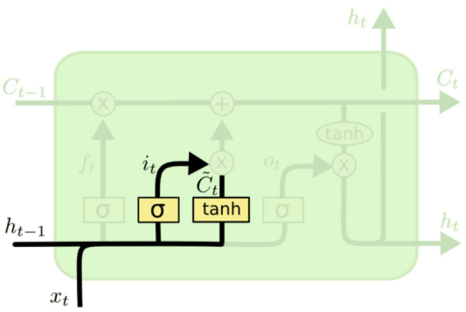
The presented structure (Figure 6) has an important feature necessary for solving the given task: at its core lies the forget gate (Figure 7), which autonomously determines which coordinates of the state vector need to be remembered [3].



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$$
$$\tilde{C}_t = \mathrm{th}\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t \odot \mathrm{th}(C_t)$$

Fig. 7. The forget gate in the LSTM structure

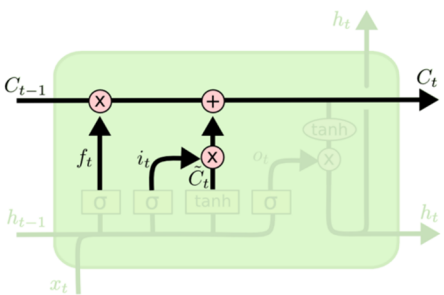
Furthermore, the input gate autonomously determines which weights in the model need to be updated (Figure 8).



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$$
$$\tilde{C}_t = \mathrm{th}\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t \odot \mathrm{th}(C_t)$$

Fig. 8. The input gate in the LSTM architecture

Another important element in the LSTM architecture is the new state, which is formed in the LSTM model as a composition of the old state, modified by the gate, and the vector of values for new weight "candidates" (Figure 9).



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right)$$
$$\tilde{C}_t = \mathrm{th}\left(W_c \cdot [h_{t-1}, x_t] + b_c\right)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t \odot \mathrm{th}(C_t)$$

Fig. 9. The new state gate in the LSTM structure

The final element in the neural network structure is the output state gate (Figure 10). It allows for the formation of the output prediction, which represents the predicted difference between the current prime number and the next unknown prime number, located at the "+1" position relative to the considered number.

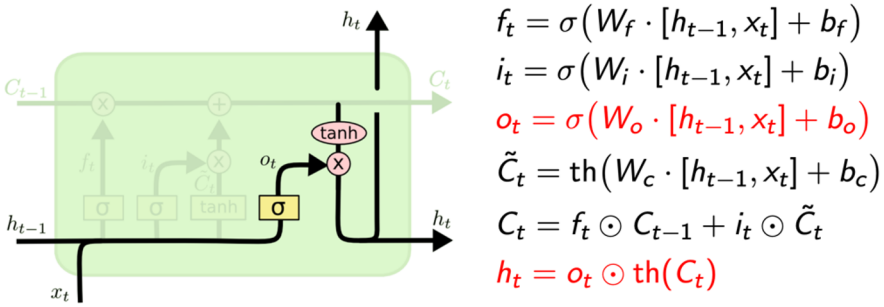


Fig. 10. The output state gate in the LSTM architecture

The software implementation of the LSTM architecture (Figure 11) was based on 100 training epochs.

```
import numpy as np
import tensorflow as tf

# Data preparation
sequence = spis  # Numerical sequence of the difference of prime numbers

# Calculating differences between neighboring elements
diff_sequence = np.diff(sequence)

# Converting data to a format suitable for LSTM training
input_sequence = np.array(diff_sequence[:-1])  # Input sequence (excluding the last element)
output_sequence = np.array(diff_sequence[1:])  # Output sequence (next difference)

input_sequence = np.reshape(input_sequence, (len(input_sequence), 1, 1))
output_sequence = np.reshape(output_sequence, (len(output_sequence), 1))

# Creating an LSTM model
model = tf.keras.models.Sequential([
    tf.keras.layers.LSTM(10, activation='tanh', recurrent_activation='sigmoid', input_shape=(1, 1)),
    tf.keras.layers.Dense(1)
])

# Compiling the model
model.compile(loss='mean_squared_error', optimizer='adam')

# Model Training
model.fit(input_sequence, output_sequence, epochs=100, batch_size=1, verbose=2)

# Predicting the next difference in the sequence
next_diff = model.predict(np.array([[diff_sequence[-1]]]))
print("Next sequence difference:", next_diff[0][0])

# Restoring the next prime number
next_number = int(sequence[-1] + next_diff[0][0])
print("The next prime number in the sequence:", next_number)
```

Fig. 11. Software implementation of the LSTM model for predicting the next difference between prime numbers in an infinite sequence

The software algorithm allowed for accurate prediction of the next difference and determination of the prime number at the "+1" position relative to the considered number. For example, for the number "14081," the next prime number would be "14083," and thus, the difference between these numbers would be "2." The predictive model (Figure 12) yielded completely identical results.

```
Epoch 87/100
1658/1658 - 1s - loss: 38.3891 - 788ms/epoch - 475us/step
Epoch 88/100
1658/1658 - 1s - loss: 38.4063 - 788ms/epoch - 475us/step
Epoch 89/100
1658/1658 - 1s - loss: 38.4135 - 786ms/epoch - 474us/step
Epoch 90/100
1658/1658 - 1s - loss: 38.3782 - 794ms/epoch - 479us/step
Epoch 91/100
1658/1658 - 1s - loss: 38.4018 - 777ms/epoch - 469us/step
Epoch 92/100
1658/1658 - 1s - loss: 38.3935 - 777ms/epoch - 469us/step
Epoch 93/100
1658/1658 - 1s - loss: 38.4021 - 788ms/epoch - 475us/step
Epoch 94/100
1658/1658 - 1s - loss: 38.4066 - 786ms/epoch - 474us/step
Epoch 95/100
1658/1658 - 1s - loss: 38.3957 - 798ms/epoch - 481us/step
Epoch 96/100
1658/1658 - 1s - loss: 38.4013 - 802ms/epoch - 484us/step
Epoch 97/100
1658/1658 - 1s - loss: 38.3925 - 789ms/epoch - 476us/step
Epoch 98/100
1658/1658 - 1s - loss: 38.4060 - 801ms/epoch - 483us/step
Epoch 99/100
1658/1658 - 1s - loss: 38.3915 - 799ms/epoch - 482us/step
Epoch 100/100
1658/1658 - 1s - loss: 38.3889 - 803ms/epoch - 484us/step
1/1 [==============================] - 0s 131ms/step
Next sequence difference: 2
The next prime number in the sequence: 14083
```

Fig. 12. Correspondence of predictive results to the truth in the LSTM model for the next difference between prime numbers

4 Conclusion

From the obtained results of the machine learning model predictions, it can be concluded that determining the next difference between prime numbers is a solvable task from the perspective of approximation algorithms.

In cases where an additional level of reliability is required, one can resort to verifying the predicted prime number using a primality test (factoring the number). This approach can increase the probability of classifying the predicted number as prime to unity (or reduce it to zero in the opposite case) and reduce the computational resources needed for exhaustive search through the entire sequence of natural numbers to find the next prime number.

References

1. G. I. Kolesnikova, Artificial Intelligence: Problems and Prospects (2018)
2. S. Hochreiter, Schmidhuber J. Neural Computetion, **9(8)** (1997)
3. Cristopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs (Last accessed 11.05.2023)