IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 January 2024, accepted 26 March 2024, date of publication 10 April 2024, date of current version 18 April 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3387306*

RESEARCH ARTICLE

# On the Generative Power of ReLU Network for Generating Similar Strings

**MAMOONA GHAFOOR AND TATSUYA AKUTSU, (Senior Member, IEEE)**
Bioinformatics Center, Institute for Chemical Research, Kyoto University, Uji 611-0011, Japan

Corresponding authors: Mamoona Ghafoor (mamoona.ghafoor@kuicr.kyoto-u.ac.jp) and Tatsuya Akutsu (takutsu@kuicr.kyoto-u.ac.jp)

The work of Tatsuya Akutsu was supported in part by Japan Society for the Promotion of Science (JSPS), Japan, under Grant 22H00532 and Grant 22K19830.

**ABSTRACT** Recently, generative networks are widely used in different applied fields including computational biology for data augmentation, DNA sequence generation, and drug discovery. The core idea of these networks is to generate new data instances that resemble a given set of data. However it is unclear how many nodes and layers are required to generate the desirable data. In this context, we study the problem of generating strings with a given Hamming distance and edit distance which are commonly used for sequence comparison, error detection, and correction in computational biology to comprehend genetic variations, mutations, and evolutionary changes. More precisely, for a given string $e$ of length $n$ over a symbol set $\Sigma$, $m = |\Sigma|$, we proved that all strings over $\Sigma$ with hamming distance and edit distance at most $d$ from $e$ can be generated by a generative network with rectified linear unit function as an activation function. The depth of these networks is constant and are of size $\mathcal{O}(nd)$ and $\mathcal{O}(\max(md, nd))$.

**INDEX TERMS** ReLU neural network, hamming distance, edit distance.

## I. INTRODUCTION

In the recent years, generative adversarial networks have been extensively studies due to their data representation power. These generative networks are basically neural networks which are used to capture the statistical regularities, underlying patterns, and dependencies of a given dataset, and generate new data that have similar characteristics as the original data [1], [2], [3], [4]. These networks have gained widespread applicability in diverse domains, spanning natural language processing, data augmentation, DNA sequence synthesis, and drug discovery [5], [6], [7], [8], [9], [10], [11], [12].

Choosing the right network model, or function family, is crucial in machine learning. If the function family is too extensive, it can lead to issues like high computational costs and overfitting. On the other hand, if the function family is too limited and doesn't encompass the target functions, it may not yield the desired prediction results [13]. However, selecting an appropriate network is a challenging task, and it is still unclear which network should be used for a given problem setting. Extensive research have been carried out to study the representational capabilities of networks to get an effective insight for choosing an appropriate function family.

According to the universal approximation theorem, a depth-2 neural network (a network with two layers) can approximate any given Borel measurable function [14]. However, there are many instances where the number of nodes becomes excessively large, necessitating exploration of more effective network architectures. In this context, relationships between function families and the size of architectures have been studies to find the best architecture with smaller number of nodes. Furthermore, it has been noticed that the representation capability of networks increases exponentially with the depth [15], [16]. Moreover, every function family cannot be efficiently expressed on any arbitrary architecture. For example, Telgarsky [17] discovered a function family that can be represented by deep neural networks and shallow neural networks with nodes of linear and exponential order, respectively. Szymanski and McCane [18] showed that the periodic functions can be efficiently expressed by deep networks. To express a periodic

The associate editor coordinating the review of this manuscript and approving it for publication was Nuno M. Garcia.

function, Chatziafratis et al. [19] gave lower bounds for the width as a function of depth. Hanin and Rolnick [20] demonstrated that the number of regions does not grow rapidly in a network with piecewise linear activation function. Bengio et al. [21] and Biau et al. [22] proved that decision trees and random forests can be efficiently expressed as neural networks by using sigmoidal functions, Heaviside functions, and hyperbolic tangent functions as activation functions. Later on, Kumano and Akutsu [13], extended the result for the neural networks with rectified linear unit (ReLU) and other related activation functions.

Sequence comparison, error detection, and correction in DNA, RNA, and protein sequences are crucial tasks in computational biology to understand genetic variations, mutations, and evolutionary changes. Hamming distance and edit distance are the two most frequently employed distance metrics for handling these tasks when the sequences have a fixed and variable length, respectively, [5], [6], [12], [23], [24], [25].

Motivated by the application of the generative networks and distance metrics, we study the problem of generating all strings that are similar to a given string using neural networks. More precisely, for a given string $e$ of length $n$ over a symbol set $\Sigma$, we theoretically prove the existence of generative networks with ReLU as an activation function that can generate all strings over $\Sigma$ with Hamming distance and edit distance at most $d$ from $e$. For this purpose we express the functions $\delta$ and $[a \geq \theta]$ as ReLU functions, where $\delta(a, b) = 1$ if $a = b$, and $\delta(a, b) = 0$, otherwise; $[a \geq \theta] = 1$ if $a \geq \theta$, and $[a \geq \theta] = 0$ otherwise; $\mathrm{ReLU}(a) = \max(0, a)$, for any real numbers $a, b, \theta$. Furthermore, the implemented networks can be freely accessed from https://github.com/MGANN-KU/ReLU_Networks.

The rest of the paper is organized as follows: Conversion of $\delta$ and $[a \geq \theta]$ functions into ReLU is discussed in Section II. Furthermore, construction of ReLU networks to generate binary and integer strings with Hamming distance at most $d$ is discussed in Section II. Construction of ReLU networks to generate integer strings with edit distance $d$ due to deletion and insertion operations is discussed in Section III. The generation of all strings with edit distance at most $d$ due to simultaneous application of substitution, deletion and insertion operations by ReLU network is discussed in Section IV. Concluding remarks are given in Section V. Proofs of some theorems and explanation of program codes with sample instances are discussed in Appendix.

## II. $H_d$-GENERATIVE RELU

In this section, we discuss the construction of ReLU network that can generate all binary and non-binary strings over a given symbol set with Hamming distance at most $d$ from a given string. Before going to the details, we first prove in Props. 1 and 2 that the functions $\delta$ and $[a \geq \theta]$ can be expressed as ReLU function, resp., by using the Heaviside function $H$ defined as $H(a) = 1$ if $a \geq 0$, and $H(a) = 0$ otherwise.

*Proposition 1: For integers a and b, the function $\delta(a, b)$ can be realized exactly by ReLU function.*

*Proof:* By the definition of functions $\delta$ and $H$, for any two integers $a, b$, it holds that

$$\delta(a, b) = H(a - b) + H(b - a) - 1. \tag{1}$$

By [13, Theorem 1], the Heaviside function $H$ can be expressed by ReLUfunction as

$$H(a - b) = \mathrm{ReLU}((a - b)/\epsilon + 1) - \mathrm{ReLU}((a - b)/\epsilon), \tag{2}$$

where $\epsilon$ is a sufficiently small positive real number, and $a - b \neq 0$. By using Eq. (1) in Eq. (2), we get

$$\begin{aligned}\delta(a, b) = {} & \mathrm{ReLU}((a - b)/\epsilon + 1) - \mathrm{ReLU}((a - b)/\epsilon) \\ & + \mathrm{ReLU}((b - a)/\epsilon + 1) - \mathrm{ReLU}((b - a)/\epsilon) - 1.\end{aligned} \tag{3}$$

This implies that $\delta(a, b)$ can be expressed with four nodes of ReLU as demonstrated in Figure 1(b). □

By Eqs. (2) and (3) the functions $H$ and $\delta$ can be computed by using two and four nodes with ReLU as an activation function, respectively. Figures 1 (i) and (ii) illustrate examples of neural networks that can compute $\delta$ using $H$ and ReLU as activation functions, respectively.

*Proposition 2: For integers a and $\theta$, the function $[a \geq \theta]$ can be realized exactly by ReLU function.*

*Proof:* For a sufficiently small positive real number $\epsilon$ it holds that

$$[a \geq \theta] = \mathrm{ReLU}((a - \theta)/\epsilon + 1) - \mathrm{ReLU}((a - \theta)/\epsilon). \tag{4}$$

Hence $[a \geq \theta]$ can be expressed with two nodes of ReLU as demonstrated in Figure 2. □

Eq. (4) implies that the function $[a \geq \theta]$ can be computed with two nodes with ReLU as an activation function. Figures 2 (a) and (b) illustrate example neural networks with $[a \geq \theta]$ and ReLU as an activation function, respectively. The output of these networks is the same.

Given a binary string $e = (e_1, e_2, \ldots, e_n)$ of length $n$ and a non-negative integer $d$, which implicitly means the distance. We define *binary $H_d$-generative ReLU*, as a ReLU neural network with $d$ input nodes $x = (x_1, x_2, \ldots, x_d)$ and $n$ output nodes $y = y_1, y_2, \ldots, y_n$ such that all binary strings $(y_1, y_2, \ldots, y_n)$ with hamming distance at most $d$ with $e$ can be obtained by appropriately choosing integer input values for $x = (x_1, x_2, \ldots, x_d)$, where $x_j \leq n$ for all $j$.

We discuss the existence of binary $H_d$-generative ReLU networks in the following theorem.

*Theorem 1: For any given binary string e of length n and a non-negative integer d, there exists a binary $H_d$-generative ReLU with size $\mathcal{O}(dn)$ and constant depth.*

*Proof:* We first establish that for each string $y$ with hamming distance at most $d$ with the string $e$, there exists a string $x$ such that:

$$u_i^j = \delta(x_j, i) \text{ for } i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, d\}, \tag{5}$$

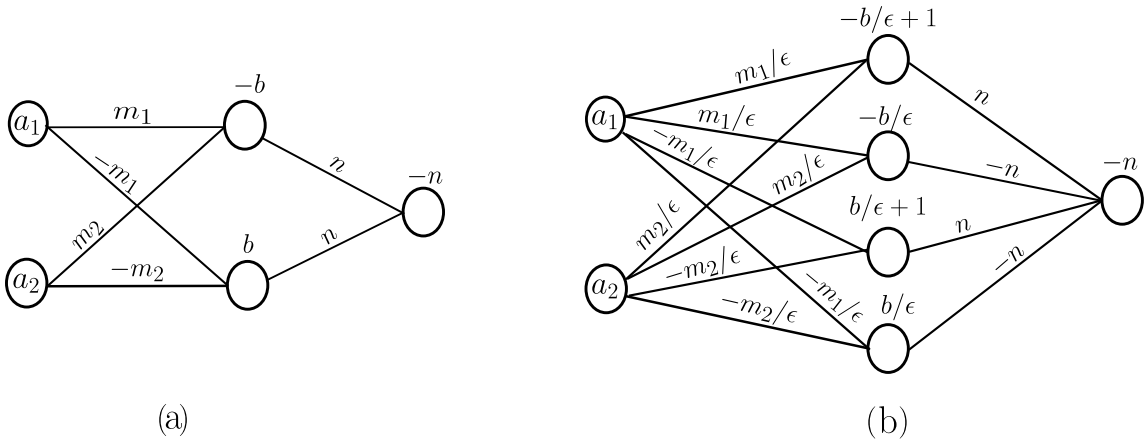


**FIGURE 1.** Computing $\delta$ by $H$ and ReLU: (a) and (b) Example neural networks with one hidden layer with two and four nodes, and $H$ and $\delta$ as an activation, resp., to compute $\delta$. These networks have two input nodes with values $a_1$ and $a_2$ and one output node. The values on the edges and nodes of hidden and output layers are the weights and bias. By setting $a = a_1m_1 + a_2m_2$, the outputs of the neural network in Figures (a) and (b) are $nH(a-b)+nH(b-a)-n = n\delta(a,b)$ and $n\mathrm{ReLU}((a-b)/\epsilon+1) - n\mathrm{ReLU}((a-b)/\epsilon) + n\mathrm{ReLU}((b-a)/\epsilon+1) - n\mathrm{ReLU}((b-a)/\epsilon) - n = n\delta(a,b)$, respectively.

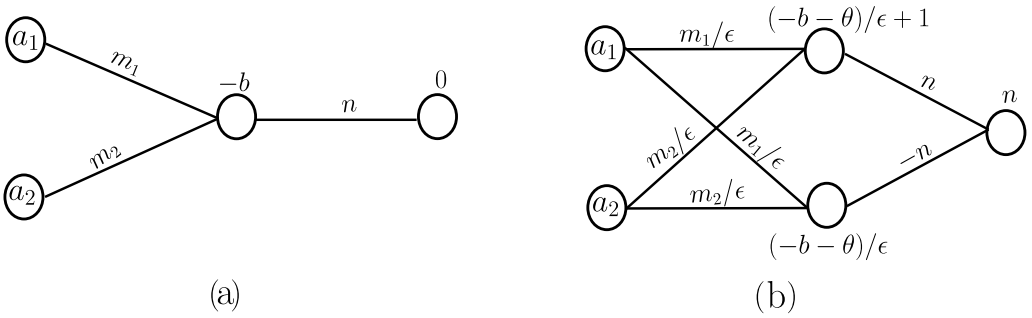


**FIGURE 2.** Computing $[a \geq \theta]$ using ReLU. (a) and (b) Example neural networks with $[a \geq \theta]$ and ReLU as an activation function, resp., with two input nodes with values $a_1$ and $a_2$, one hidden layer. The values on the edges and nodes of hidden and output layers are the weight and bias. By setting $a = a_1m_1 + a_2m_2$, the output of these two networks is the same.

$$v_i = \left[\sum_{j=1}^{d} u_i^j \geq 1\right] \text{ for } i \in \{1, 2, \ldots, n\}, \tag{6}$$

$$y_i = 1 - \delta(v_i, e_i) \text{ for } i \in \{1, 2, \ldots, n\}. \tag{7}$$

Suppose that $e = (e_1, e_2, \ldots, e_n)$ and $y = (y_1, y_2, \ldots, y_n)$ such that the hamming distance between $e$ and $y$ is $d' \leq d$. Then $e$ and $y$ differ at exactly $d'$ positions $k_1, k_2, \ldots, k_{d'}$, i.e., $e_i \neq y_i$, for all $i = k_j$ where $k_j \leq n$ and $1 \leq j \leq d'$. Consider a string $x = (x_1, x_2, \ldots, x_d)$ such that $x_{j'} \in \{0, k_1, k_2, \ldots, k_{d'}\}$, and for each $j$, at least one $x_{j'}$ takes the value $k_j$. This implies that $\sum_{j=1}^{d} u_i^j x_i \geq 1$ if and only if $i \in \{k_1, k_2, \ldots, k_{d'}\}$ from which it follows that $v_i = 1$ if and only if $i \in \{k_1, k_2, \ldots, k_{d'}\}$. Thus we have $1 - \delta(v_i, e_i) = 0$ (resp., 1) if $e_i = 1$ (resp., 0) for $i \in \{k_1, k_2, \ldots, k_{d'}\}$, and $1 - \delta(v_i, e_i) = 0$ (resp., 1) if $e_i = 0$ (resp., 1) for $i \notin \{k_1, k_2, \ldots, k_{d'}\}$, which implies that $y$ can be obtained by $x$.

We construct a neural network by using Eqs. (3) and (4) with five layers, and use ReLUas an activation function to express the computation of Eqs. (5), (6), and (7). An illustration of the network is given in Figure 3. We denote a node and its value with the same symbol. The first and the last layers are the input and output layers with $d$ and $n$ nodes storing the entries of the strings $x$ and $y$, respectively. The second layer (first hidden layer) corresponds to Eqs. (5). There are $dn$ values of $\delta(x_j, i)$, and by Eq. (3) $\delta(x_j, i)$ can be computed using four nodes with ReLUas an activation function. Therefore there are $4dn$ nodes in the second layer. Let $\alpha_{ij}^1$, $\alpha_{ij}^2$, $\beta_{ij}^1$, and $\beta_{ij}^2$ denote such nodes, $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, d\}$, with bias $-i/\epsilon + 1$, $-i/\epsilon$, $i/\epsilon + 1$, $i/\epsilon$, resp., where $\epsilon$ is a sufficiently small positive real number. The edges $(x_j, \alpha_{ij}^1)$, $(x_j, \alpha_{ij}^2)$, $(x_j, \beta_{ij}^1)$, $(x_j, \beta_{ij}^2)$ have the only non-zero weights $1/\epsilon$, $1/\epsilon$, $-1/\epsilon$, $-1/\epsilon$, respectively. This implies that

$$\alpha_{ij}^1 = \mathrm{ReLU}((x_j - i)/\epsilon + 1), \alpha_{ij}^2 = \mathrm{ReLU}((x_j - i)/\epsilon),$$
$$\beta_{ij}^1 = \mathrm{ReLU}((i - x_j)/\epsilon + 1), \beta_{ij}^2 = \mathrm{ReLU}((i - x_j)/\epsilon).$$

The third layer performs the computation of Eq. (6). There are $n$ values of $v_i$, and by Eq. (4) for each $v_i$, two nodes are required to express $[a \geq 1]$ as ReLU. Therefore there are $2n$ nodes in the third layer. Let $\eta_i^1$, and $\eta_i^2$, $i \in \{1, 2, \ldots, n\}$,

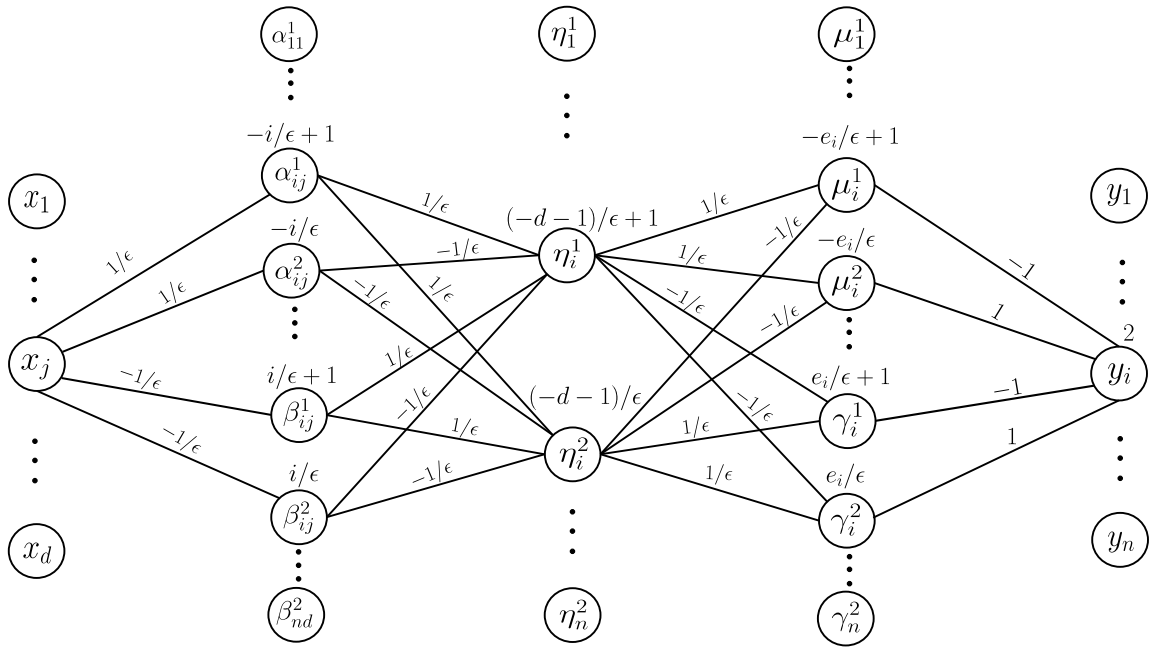


**FIGURE 3.** An illustration of a binary $H_d$-generative neural network with three hidden layers. The bias and non-zero weights of a few nodes and edges are shown.

are such nodes with bias $-(d+1)/\epsilon+1$ and $-(d+1)/\epsilon$, respectively. The edges $(\alpha^1_{ij}, \eta^h_i)$, $(\alpha^2_{ij}, \eta^h_i)$, $(\beta^1_{ij}, \eta^h_i)$, $(\beta^2_{ij}, \eta^h_i)$, $h = 1, 2$ have non-zero weights $1/\epsilon, -1/\epsilon, 1/\epsilon, -1/\epsilon$ respectively.

Thus with ReLU as an activation function and Eq. (1), we have

$$\begin{aligned}\eta^1_i &= \mathrm{ReLU}\left(\sum_{j=1}^{d}(\alpha^1_{ij}-\alpha^2_{ij}+\beta^1_{ij}-\beta^2_{ij})/\epsilon-(d+1)/\epsilon+1\right)\\ &= \mathrm{ReLU}\left(\sum_{j=1}^{d}(H(x_j-i)+H(i-x_j))/\epsilon-(d+1)/\epsilon+1\right),\end{aligned} \tag{8}$$

$$\begin{aligned}\eta^2_i &= \mathrm{ReLU}\left(\sum_{j=1}^{d}(\alpha^1_{ij}-\alpha^2_{ij}+\beta^1_{ij}-\beta^2_{ij})/\epsilon-(d+1)/\epsilon\right)\\ &= \mathrm{ReLU}\left(\sum_{j=1}^{d}(H(x_j-i)+H(i-x_j))/\epsilon-(d+1)/\epsilon\right).\end{aligned} \tag{9}$$

By setting $a=\sum_{j=1}^{d}(H(x_j-i)+H(i-x_j))-d$ in Eqs. (8), and (9), and $\theta=1$, we have

$$\eta^1_i = \mathrm{ReLU}((a-\theta)/\epsilon+1), \qquad \eta^2_i = \mathrm{ReLU}((a-\theta)/\epsilon)).$$

This and Eq. (2) implies that

$$\eta^1_i-\eta^2_i = [a\geq 1] \tag{10}$$

$$= \left[\sum_{j=1}^{d}(H(x_j-i)+H(i-x_j))-d\geq 1\right] \tag{11}$$

$$= \left[\sum_{j=1}^{d}\delta(x_j,i)\geq 1\right]. \tag{12}$$

The fourth layer is used to compute Eq. (7). For each $\delta$, we need four nodes, and therefore the fourth layer has $4n$ nodes. Let $\mu^1_i$, $\mu^2_i$, $\gamma^1_i$ and $\gamma^2_i$ denote such nodes with bias $-e_i/\epsilon+1$, $-e_i/\epsilon$, $e_i/\epsilon+1$ and $e_i/\epsilon$, respectively. The edges $(\eta^1_i, \mu^1_i)$, $(\eta^2_i, \mu^1_i)$ $(\eta^1_i, \mu^2_i)$, $(\eta^2_i, \mu^2_i)$ (resp., $(\eta^1_i, \gamma^1_i)$, $(\eta^2_i, \gamma^1_i)$ $(\eta^1_i, \gamma^2_i)$, $(\eta^2_i, \gamma^2_i)$) have the non-zero weights $1/\epsilon, -1/\epsilon, 1/\epsilon$, $-1/\epsilon$ (resp., $-1/\epsilon, 1/\epsilon, -1/\epsilon, 1/\epsilon$) respectively. By Eq. (10) we have

$$\begin{aligned}\mu^1_i &= \mathrm{ReLU}\left((\eta^1_i-\eta^2_i-e_i)/\epsilon+1\right),\\ \mu^2_i &= \mathrm{ReLU}\left((\eta^1_i-\eta^2_i-e_i)/\epsilon\right),\\ \gamma^1_i &= \mathrm{ReLU}\left((-\eta^1_i+\eta^2_i+e_i)/\epsilon+1\right),\\ \gamma^2_i &= \mathrm{ReLU}\left((-\eta^1_i+\eta^2_i+e_i)/\epsilon\right).\end{aligned}$$

The fifth layer is the output layer with $n$ nodes $y_i$ with bias 2. The non-zero weights of the edges $(\mu^1_i, y_i)$, $(\gamma^1_i, y_i)$, $(\mu^2_i, y_i)$, $(\gamma^2_i, y_i)$ are -1, -1, 1, 1, respectively. By using Eqs. (1) and (4), we have

$$\begin{aligned}y_i &= -\mu^1_i+\mu^2_i-\gamma^1_i+\gamma^2_i+2\\ &= -H(\eta^1_i-\eta^2_i-e_i)-H(e_i-(\eta^1_i-\eta^2_i))+2\\ &= 1-\delta(\eta^1_i-\eta^2_i,e_i)\end{aligned}$$

$$= 1 - \delta\left(\left[\sum_{j=1}^{d} \delta(x_j, i) \geq 1\right], e_i\right)$$
$$= 1 - \delta([p \geq 1], e_i) = 1 - \delta(v_i, e_i).$$

This implies that for each input string $x$ the constructed network can output a string $y$ that has hamming distance at most $d$ with $e$. Furthermore, the size of the network is $d + 4dn + 2n + 4n + n = \mathcal{O}(dn)$, and has a constant depth, which completes the proof. □

*Example 1: Suppose that $e = (1, 1, 0)$, $d = 2$, and $y = (0, 1, 1)$, where the hamming distance between e and y is 2. The strings e and y differ at positions $i = 1, 3$. By Theorem 1, set $x_1 = 1$ and $x_2 = 3$, i.e., $x = (1, 3)$. We demonstrate that by using $x = 1, 3$ as an input to the neural network with ReLU as an activation function constructed in Theorem 1, we get the same output y. The architecture of the neural network is shown in Figure 4, where the edges with zero weights are omitted. The values of each node in the second, third, fourth, and fifth layers are listed in Tables 1, 2, 3, and 4, respectively.*

We extend the definition of binary $H_d$-generative ReLU for integer strings as follows.

Given a string $e = (e_1, e_2, \ldots, e_n)$ of length $n$ over alphabet $\Sigma = \{1, 2, \ldots, m\}$, and a non-negative integer $d$. We define *$H_d$-generative ReLU*, as a ReLU neural network with $2d$ input nodes $x = (x_1, x_2, \ldots, x_{2d})$ and $n$ output nodes $y = y_1, y_2, \ldots, y_n$ such that all strings $y = (y_1, y_2, \ldots, y_n)$ over $\Sigma$ with hamming distance at most $d$ with $e$ can be obtained by appropriately choosing input values for $x = (x_1, x_2, \ldots, x_{2d})$, where $x_j \leq n$, and $x_{j+d} \in \Sigma$ for $1 \leq j \leq d$.

We discuss the existence of $H_d$-generative ReLU in Theorem 2.

*Theorem 2: For a given string e of size n over a symbol set $\Sigma = \{1, 2, \ldots, m\}$, and a non-negative integer d, there exits an $H_d$-generative ReLU network with size $\mathcal{O}(nd)$ and constant depth.*

A proof of Theorem 2 and its computation are demonstrated in Example 3 in Appendix.

## III. $ED_d$ AND $EI_d$-GENERATIVE RELU

Given a string $e = (e_1, e_2, \ldots, e_n)$ of length $n$ over alphabet $\Sigma = \{1, 2, \ldots, m\}$, and a non-negative integer $d$, we define *$ED_d$-generative ReLU*, to be a ReLU neural network with $d$ input nodes $x = (x_1, x_2, \ldots, x_d)$, and $n - d$ output nodes $y = y_1, y_2, \ldots, y_{n-d}$ such that all strings $y = (y_1, y_2, \ldots, y_{n-d})$ over $\Sigma$ with edit distance exactly $d$ from $e$ due to deletion can be obtained by appropriately choosing the input string $x = (x_1, x_2, \ldots, x_d)$, where $x_j \leq n$, $1 \leq j \leq d$.

The following theorem discuss the existence of $ED_d$-generative ReLU.

*Theorem 3: For a string e of size n over $\Sigma$, and a non-negative integer d, there exists an $ED_d$-generative ReLUnetwork with size $\mathcal{O}(dn)$ and constant depth.*

*Proof:* Suppose $e = (e_1, e_2, \ldots, e_n)$ and $y = (y_1, y_2, \ldots, y_{n-d})$ are two strings over $\Sigma$ such that $y$ is obtained from $e$ by deleting exactly $d$ entries at the positions $k_1, k_2, \ldots, k_d$ of $e$. Consider an integer string $x = (x_1, x_2, \ldots, x_d)$ such that $x_j \in \{k_1, k_2, \ldots, k_d\}$. Then the following system of equations implies that we can obtain $y$ from $x$.

$$p_i = \begin{cases} 0 & \text{if } x_j = i, \\ 1 & \text{otherwise,} \end{cases} \quad \text{for } i \in \{1, 2, \ldots, n\}, \text{ for some } j \in \{1, 2, \ldots, d\}, \tag{13}$$

$$q_i = \max(B \sum_{k=1}^{i} p_k - C\delta(p_i, 0), 0) \text{ for } i \in \{1, 2, \ldots, n\}, \quad B \gg n, C \gg B, \tag{14}$$

$$r_i^j = \left[iB \leq q_{i+j-1} \leq iB + 1\right] \text{ for } i \in \{1, 2, \ldots, n-d\}, \; j \in \{1, 2, \ldots, d+1\}, \tag{15}$$

$$t_i^j = \max(e_{i+j-1} - C(1 - r_i^j), 0) \text{ for } i \in \{1, 2, \ldots, n-d\}, \; j \in \{1, 2, \ldots, d+1\}, \tag{16}$$

$$y_i = \sum_{j=1}^{d+1} t_i^j \text{ for } i \in \{1, 2, \ldots, n-d\}. \tag{17}$$

Eq. (13) encodes the indices $i$ as a binary vector to determine if $e_i$ should be deleted or not from $e$. Eq. (14) is used to assign the weights to each index in an increasing order such that the weight $q_i$ is zero if $p_i$ is zero, i.e., the entry $e_i$ should be deleted from $e$. Eqs. (15) and (16) are used to determine the positions in $y$ of those entries of $e$ that should not be deleted. Since exactly $d$ entries can be deleted from $e$ therefore each $i$ can be shifted by at most value $d$. Hence, a given position $i$ in $y$ can have the value $e_{i+j-1}$, for some $j$, depending on the number of zeros before $q_i$, i.e., the number of entries to be deleted before $e_i$ as demonstrated in Example 2.

Construct an eight-layer neural network with ReLU as an activation function by expressing Eqs. (13)-(17) as ReLU as follows. An illustration of such a network is given in Figure 5 where the first layer is the input layer with input $x$, the last layer is the output layer that outputs $y$. The Input layer has $d$ nodes which are denoted by $x_j$, $1 \leq j \leq d$. We can express Eq. (13) as $\delta$, $[a \geq \theta]$ as follows.

$$u_i^j = 1 - \delta(x_j, i), \tag{18}$$

$$z_i = \left[\sum_{j=1}^{d} u_i^j \geq d\right]. \tag{19}$$

By Eq. (19), we have $z_i = p_i$. By using Prop. 1 the second layer performs the computation of Eq. (18) with $4nd$ nodes $\alpha_{ij}^1, \alpha_{ij}^2, \beta_{ij}^1, \beta_{ij}^2$, $1 \leq i \leq n, 1 \leq j \leq d$. The values of these nodes are:

$$\alpha_{ij}^1 = \text{ReLU}((x_j - i)/\epsilon + 1), \alpha_{ij}^2 = \text{ReLU}((x_j - i)/\epsilon),$$
$$\beta_{ij}^1 = \text{ReLU}((i - x_j)/\epsilon + 1), \beta_{ij}^2 = \text{ReLU}((i - x_j)/\epsilon).$$

By using Prop. 2, the third layer performs the computation of Eq. (19) with $2n$ nodes $\eta_i^1$ and $\eta_i^2$, $1 \leq i \leq n$, with

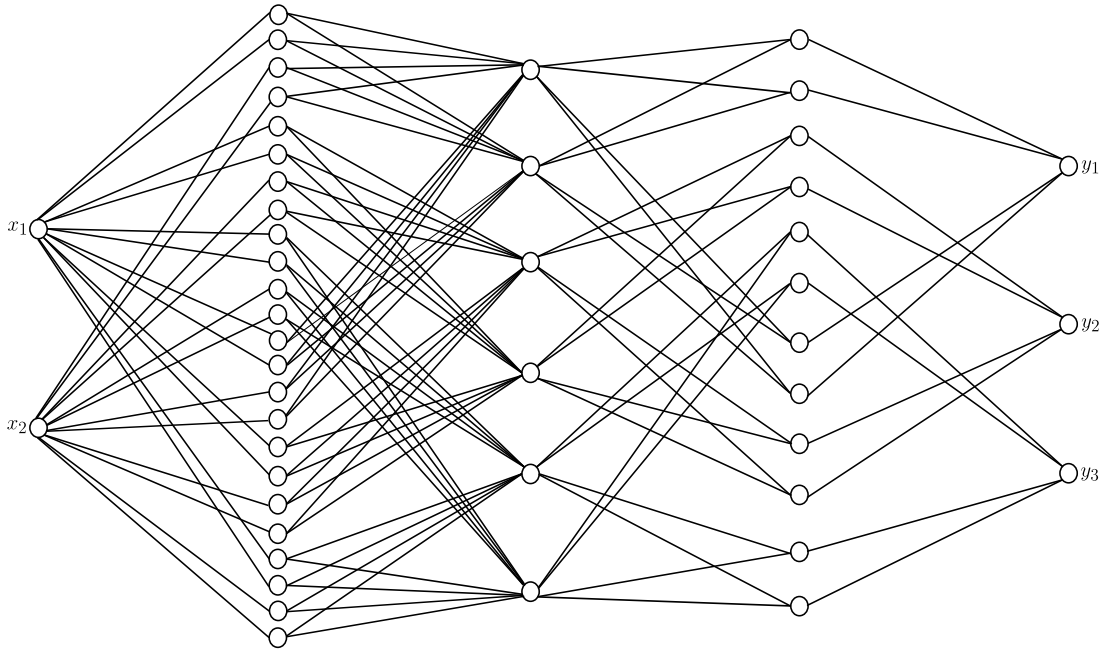


**FIGURE 4.** An example $H_d$-generative ReLU obtained by using Theorem 1. The edges with zero weights are omitted.

values:

$$\eta_i^1 = \mathrm{ReLU}\left(\sum_{j=1}^{d}(-\alpha_{ij}^1 - \beta_{ij}^1 + \alpha_{ij}^2 + \beta_{ij}^2)/\epsilon + d/\epsilon + 1\right),$$

$$\eta_i^2 = \mathrm{ReLU}\left(\sum_{j=1}^{d}(-\alpha_{ij}^1 - \beta_{ij}^1 + \alpha_{ij}^2 + \beta_{ij}^2)/\epsilon + d/\epsilon\right).$$

Fourth and fifth layers are used to perform the computation of Eq. (14). The fourth layer calculates the values of $\mathrm{ReLU}(\sum_{k=1}^{i} z_k)$ and $\delta(z_i, 0)$ with $5n$ nodes $\gamma_i$, $\mu_i^1$, $\mu_i^2$, $\lambda_i^1$, $\lambda_i^2$ by using Prop. 1. The values of these nodes are:

$$\gamma_i = \mathrm{ReLU}(\sum_{k=1}^{i}(\eta_k^1 - \eta_k^2)).$$
$$\mu_i^1 = \mathrm{ReLU}((\eta_i^1 - \eta_i^2)/\epsilon + 1),$$
$$\mu_i^2 = \mathrm{ReLU}((\eta_i^1 - \eta_i^2)/\epsilon),$$
$$\lambda_i^1 = \mathrm{ReLU}((-\eta_i^1 + \eta_i^2)/\epsilon + 1),$$
$$\lambda_i^2 = \mathrm{ReLU}((-\eta_i^1 + \eta_i^2)/\epsilon).$$

Finally, the fifth layer with $n$ nodes $\tau_i$ computes Eq. (14) as follows:

$$\tau_i = \mathrm{ReLU}\left(B\gamma_i - C(\mu_i^1 - \mu_i^2 + \lambda_i^1 - \lambda_i^2) + C\right).$$

The sixth layer is used to compute $\left[q_{i+j-1} \geq iB\right]$ and $\left[-q_{i+j-1} \geq -(iB+1)\right]$ in Eq. (15) by using Prop. 2 with $4(n-d)(d+1)$ nodes $\psi_{ij}^1$, $\psi_{ij}^2$, $\rho_{ij}^1$ and $\rho_{ij}^2$ such that

$$\psi_{ij}^1 = \mathrm{ReLU}(\tau_{i+j-1}/\epsilon + (-iB/\epsilon + 1)),$$
$$\psi_{ij}^2 = \mathrm{ReLU}(\tau_{i+j-1}/\epsilon + (-iB/\epsilon)),$$
$$\rho_{ij}^1 = \mathrm{ReLU}(-\tau_{i+j-1}/\epsilon + (iB+1)/\epsilon + 1),$$
$$\rho_{ij}^2 = \mathrm{ReLU}(-\tau_{i+j-1}/\epsilon + (iB+1)/\epsilon).$$

Thus, Eq. (16) is computed in the seventh layer with $(n-d)(d+1)$ nodes $\zeta_{ij}$ such that

$$\zeta_{ij} = \mathrm{ReLU}\left(e_{i+j-1} - C(-\psi_{ij}^1 + \psi_{ij}^2 - \rho_{ij}^1 + \rho_{ij}^2 + 2)\right).$$

Finally, the eighth layer is used to perform the computation of Eq. (17) with $n-d$ nodes $y_i$ with value

$$y_i = \sum_{j=1}^{d+1} \zeta_{ij}.$$

Hence the constructed neural network is an $ED_d$-generative ReLU with size $\mathcal{O}(dn)$ and constant depth. □

*Example 2: Suppose that the symbol set is* $\Sigma = \{1, 2, 3, 4, 5\}$, $e = (3, 2, 4, 1)$, $d = 2$ *and* $y = (2, 1)$. *The strings obtained by using* $x = (1, 3)$ *and Eqs. (13)- (15) are* $p = [0, 1, 0, 1]$, $q = [0, B, 0, 2B]$, $r = [[0, 1, 0], [0, 0, 1]]$, $t = [[0, 2, 0], [0, 0, 1]]$, *and* $y = (2, 1)$. *We show that the* $ED_d$*-generative ReLU constructed by using Theorem 3 outputs the required string y. The values of each node of the layers are listed in Tables 12- 19.*

Next, we define $EI_d$-generative ReLU for strings as follows.

For a string $e = (e_1, e_2, \ldots, e_n)$ of length $n$ over alphabet $\Sigma = \{1, 2, \ldots, m\}$, and a non-negative integer $d$, we define *$EI_d$-generative ReLU* to be a ReLU neural network with $2d$ input nodes $x = (x_1, x_2, \ldots, x_{2d})$, and $n+d$ output nodes $y = y_1, y_2, \ldots, y_{n+d}$ such that all strings $y = (y_1, y_2, \ldots, y_{n+d})$

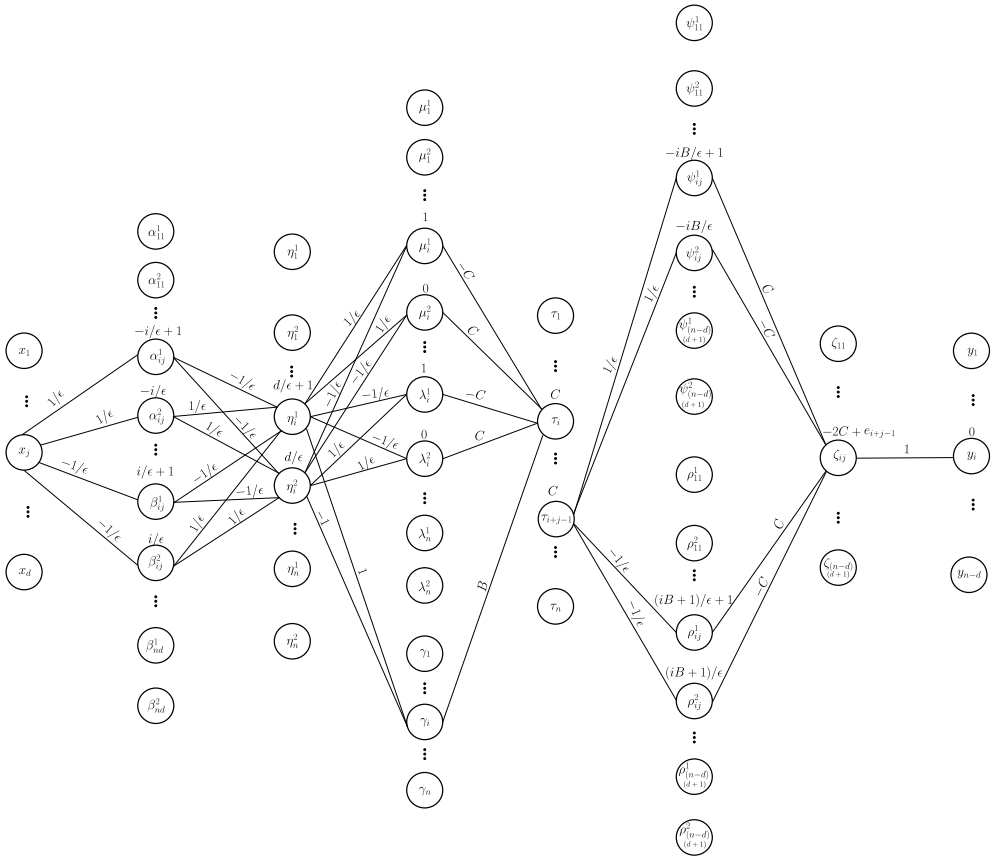

**FIGURE 5.** An illustration of $ED_d$-generative ReLU with the six hidden layers, where a few nodes and edges with their bias and non-zero weights are shown, respectively.

over $\Sigma$ with edit distance exactly $d$ from $e$ due to insertion can be obtained by appropriately choosing the input string $x = (x_1, x_2, \ldots, x_{2d})$, where $x_j \leq n$, and $x_{j+d} \in \Sigma$ for $1 \leq j \leq d$.

The existence of of such $EI_d$-generative ReLU is discussed in Theorem 4.

*Theorem 4: For a string $e$ of size $n$ over $\Sigma = \{1, 2, \ldots, m\}$, and a non-negative integer $d$, there exists an $EI_d$-generative ReLUnetwork with size $\mathcal{O}(nd)$ and constant depth.*

A proof of Theorem 4 and its computation are demonstrated in Example 4 in Appendix.

## IV. $E_d$-GENERATIVE RELU

For a string $e = (e_1, e_2, \ldots, e_n)$ of length $n$ over alphabet $\Sigma = \{1, 2, \ldots, m\}$, and a non-negative integer $d$, we define *$E_d$-generative ReLU* to be a ReLU neural network such that all strings $y$ over $\Sigma$ with edit distance at most $d$ from $e$ due to deletion, substitution and insertion and can be obtained by appropriately choosing input string $x = (x_1, \ldots, x_j, \ldots, x_{5d})$, such that $x_j \in [0, 1)$ and $x_j$ is of the form $i \cdot \Delta$, where $i$ is an integer, and $\Delta$ is a small constant.

*Theorem 5: For a string $e$ of size $n$ over $\Sigma$, and a non-negative integer $d$, there exists an $E_d$-generative ReLU-network with size $\mathcal{O}(\max(dn, dm))$ and constant depth.*

A proof of Theorem 5 and its computation are demonstrated in Example 5 in Appendix.

## V. CONCLUSION

We proved that the functions $\delta$ and $[a \geq \theta]$ can be expressed as a ReLU network with four and two hidden nodes, respectively. By using this results, we discussed the existence of ReLU generative networks that can generate strings similar to a given string with respect to Hamming distance and edit distance. We first proved that all the binary and non-binary strings of length $n$ with Hamming distance at most $d$ from a given string can be generated with a ReLU network with constant depth and size $\mathcal{O}(nd)$. This result is then extended for the case of edit distance when either deletion or insertion operations are allowed, and proved the existence of a ReLU network with constant depth and size $\mathcal{O}(nd)$ to generate all strings over a given symbol set with edit distance exactly $d$ from a given string. Finally, based on these results, we proved the generalized case where substitution, deletion and insertion operations can be applied simultaneously. More concretely, for a given string of size $n$ over a symbol set of size $m$, there exists a ReLU generative

**TABLE 1.** Values of the nodes of the second layer.

| $\alpha^1_{ij} = \text{ReLU}((x_j - i)/\epsilon + 1)$ | $\alpha^2_{ij} = \text{ReLU}((x_j - i)/\epsilon)$ | $\beta^1_{ij} = \text{ReLU}((i - x_j)/\epsilon + 1)$ | $\beta^2_{ij} = \text{ReLU}((i - x_j)/\epsilon)$ |
|---|---|---|---|
| $\alpha^1_{11} = 1$ | $\alpha^2_{11} = 0$ | $\beta^1_{11} = 1$ | $\beta^2_{11} = 0$ |
| $\alpha^1_{12} = 1 + 2/\epsilon$ | $\alpha^2_{12} = 2/\epsilon$ | $\beta^1_{12} = 0$ | $\beta^2_{12} = 0$ |
| $\alpha^1_{21} = 0$ | $\alpha^2_{21} = 0$ | $\beta^1_{21} = 1 + 1/\epsilon$ | $\beta^2_{21} = 1/\epsilon$ |
| $\alpha^1_{22} = 1 + 1/\epsilon$ | $\alpha^2_{22} = 1/\epsilon$ | $\beta^1_{22} = 0$ | $\beta^2_{22} = 0$ |
| $\alpha^1_{31} = 0$ | $\alpha^2_{31} = 0$ | $\beta^1_{31} = 1 + 2/\epsilon$ | $\beta^2_{31} = 2/\epsilon$ |
| $\alpha^1_{32} = 1$ | $\alpha^2_{32} = 0$ | $\beta^1_{32} = 1$ | $\beta^2_{32} = 0$ |

**TABLE 2.** Values of the nodes of the third layer.

| $\eta^1_i = \text{ReLU}(\sum_{j=1}^{d}(\alpha^1_{ij} - \alpha^2_{ij} + \beta^1_{ij} - \beta^2_{ij})/\epsilon - (d+1)/\epsilon + 1)$ | $\eta^2_i = \text{ReLU}(\sum_{j=1}^{d}(\alpha^1_{ij} - \alpha^2_{ij} + \beta^1_{ij} - \beta^2_{ij})/\epsilon - (d+1)/\epsilon)$ |
|---|---|
| $\eta^1_1 = 1$ | $\eta^2_1 = 0$ |
| $\eta^1_2 = 0$ | $\eta^2_2 = 0$ |
| $\eta^1_3 = 1$ | $\eta^2_3 = 0$ |

**TABLE 3.** Values of the nodes of the fourth layer.

| $\mu^1_i = \text{ReLU}((\eta^1_i - \eta^2_i - e_i)/\epsilon + 1)$ | $\mu^2_i = \text{ReLU}((\eta^1_i - \eta^2_i - e_i)/\epsilon)$ | $\gamma^1_i = \text{ReLU}((-\eta^1_i + \eta^2_i + e_i)/\epsilon + 1)$ | $\gamma^2_i = \text{ReLU}((-\eta^1_i + \eta^2_i + e_i)/\epsilon)$ |
|---|---|---|---|
| $\mu^1_1 = 1$ | $\mu^2_1 = 0$ | $\gamma^1_1 = 1$ | $\gamma^2_1 = 0$ |
| $\mu^1_2 = 0$ | $\mu^2_2 = 0$ | $\gamma^1_2 = 1 + 1/\epsilon$ | $\gamma^2_2 = 1/\epsilon$ |
| $\mu^1_3 = 1 + 1/\epsilon$ | $\mu^2_3 = 1/\epsilon$ | $\gamma^1_3 = 0$ | $\gamma^2_3 = 0$ |

**TABLE 4.** Values of the nodes of the fifth layer.

| $y_i = -\mu^1_i + \mu^2_i - \gamma^1_i + \gamma^2_i + 2$ |
|---|
| $y_1 = 0$ |
| $y_2 = 1$ |
| $y_3 = 1$ |

network that can generate all the strings with edit distance at most $d$ due to substitution, deletion and insertion operations with constant depth and size $\mathcal{O}(\max(md, nd))$.

The complexity of the proposed networks is at least $\mathcal{O}(nd)$ which can be computationally expensive with the increase in $n$ and $d$. Therefore it is a natural research problem to improve the complexity of the proposed networks from $\mathcal{O}(nd)$ to $o(nd)$. Furthermore, an interesting future direction can be to extend these results for other distance metrics such as tree distance. From a practical viewpoint, it is important future work to improve and apply the proposed method to generation of real string data such as DNA sequences and protein sequences. An implementation of the constructed generative network is available at https://github.com/MGANN-KU/ReLU_Networks.

## APPENDIX
## PROOFS AND EXAMPLES

*Proof of Theorem 2:* Let $e = (e_1, e_2, \ldots, e_n)$ be a string over $\Sigma$. We can partition the strings $y$ over $\Sigma$ that have hamming distance with $e$ at most $d$ with respect to their exact hamming distance and the positions they differ from $e$. Let $y = (y_1, y_2, \ldots, y_n)$ be a string with hamming distance $d' \leq d$ and differs from $e$ at the positions $k_1, k_2, \ldots, k_{d'}$. Then it holds that $y_{k_j} \in \Sigma \backslash \{e_{k_j}\}$. Construct $x = (x_1, x_2, \ldots, x_{2d})$ such that $x_1, \ldots, x_d$ is a sequence over $\{0, k_1, k_2, \ldots, k_{d'}\}$, where each $k_\ell$ appears at least once in $x_1, \ldots, x_d$, and $x_{d+j} = y_{k_\ell}$ (resp., $a \in \Sigma$) if $x_j = k_\ell$ (resp., 0), for $1 \leq j \leq d$. It is easy to verify that Eqs. (20)-(23) hold for $e$, $y$, and $x$, i.e., we can get $y$ from the constructed $x$.

$$p_j = \max(x_j - C \cdot \sum_{k=1}^{j-1} \delta(x_j, x_k), 0) \text{ for } j \in \{1, 2, \ldots, d\}, \text{ where } C \text{ is a constant with } C \gg \max(m, n), \quad (20)$$

$$q_i = \max(e_i - C \sum_{j=1}^{d} \delta(p_j, i), 0) \text{ for } i \in \{1, 2, \ldots, n\}, \quad (21)$$

$$r_i = \sum_{j=1}^{d} \left(\max(x_{j+d} - C(1 - \delta(p_j, i)), 0)\right) \text{ for } i \in \{1, 2, \ldots, n\}, \quad (22)$$

$$y_i = q_i + r_i \text{ for } i \in \{1, 2, \ldots, n\}. \quad (23)$$

The variable $p_j = 1$ if and only if the value of $x_j$ is not repeated before. Basically, $p_j$ is used to ignore the repetition in the sequence $x_1, \ldots, x_d$. The variable $q_i$ stores the values of $e$ that will not be substituted. More precisely, $q_i = e_i$ if and only if $x_j \neq i$ for all $j$ (as demonstrated in Example 3). The variable $r_i$ stores the values that should be substituted, i.e., $r_i = x_{d+j}$ if and only if $x_j = i$ for some $j$. Finally, the required string $y$ is obtained by adding $q_i$ and $r_i$, since exactly one of these can be non-zero.

Construct a seven-layer neural network with ReLU as an activation function by expressing Eqs. (20)- (23) as ReLU. An illustration of such a network is given in Figure 6. The first layer is the input layer with input $x$ having $2d$ nodes which are denoted by $x_j$, $1 \leq j \leq 2d$. The last layer is the output layer which outputs $y$ using Eq. (23) and $n$ nodes $y_i$,

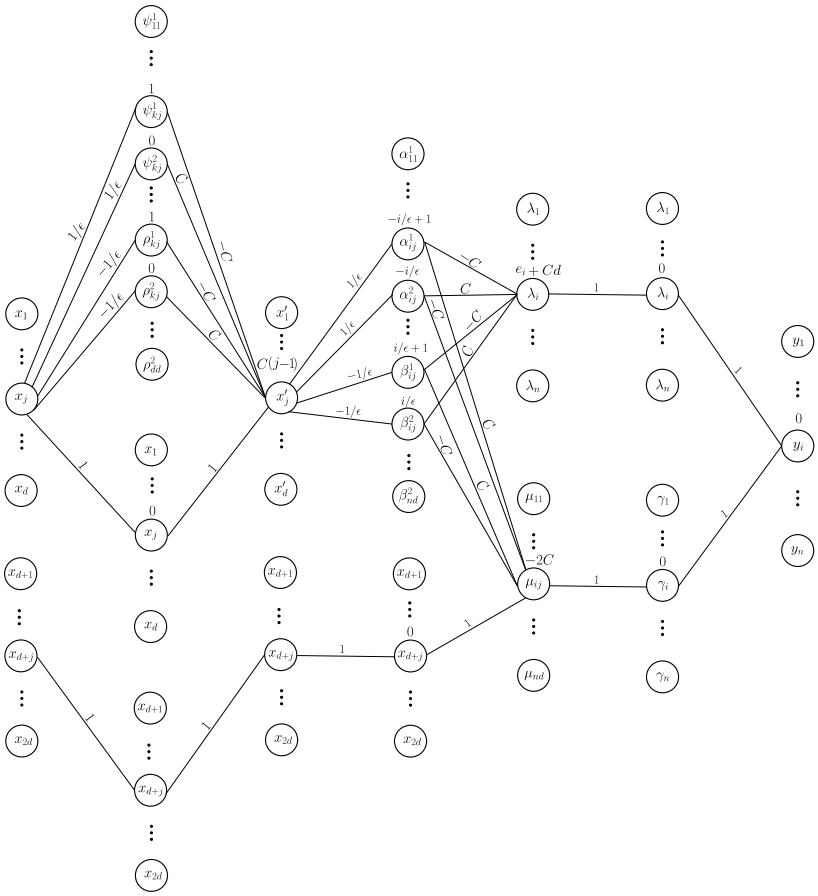

**FIGURE 6.** An illustration of the $H_d$-generative ReLU with five hidden layers, where a few nodes with their bias and edges with non-zero weights are shown.

**TABLE 5.** Values of the nodes $\psi^1_{kj}, \psi^2_{kj}, \rho^1_{kj}, \rho^2_{kj}$ of the second layer for $1 \leq j, k \leq d$.

| $\psi^1_{kj} = \text{ReLU}((x_j - x_k)/\epsilon + 1)$ | $\psi^2_{kj} = \text{ReLU}((x_j - x_k)/\epsilon)$ | $\rho^1_{kj} = \text{ReLU}((x_k - x_j)/\epsilon + 1)$ | $\rho^2_{kj} = \text{ReLU}((x_k - x_j)/\epsilon)$ |
|---|---|---|---|
| $\psi^1_{11} = 1$ | $\psi^2_{11} = 0$ | $\rho^1_{11} = 1$ | $\rho^2_{11} = 0$ |
| $\psi^1_{12} = 2/\epsilon + 1$ | $\psi^2_{12} = 2/\epsilon$ | $\rho^1_{12} = 0$ | $\rho^2_{12} = 0$ |
| $\psi^1_{13} = 1$ | $\psi^2_{13} = 0$ | $\rho^1_{13} = 1$ | $\rho^2_{13} = 0$ |
| $\psi^1_{21} = 0$ | $\psi^2_{21} = 0$ | $\rho^1_{21} = 2/\epsilon + 1$ | $\rho^2_{21} = 2/\epsilon$ |
| $\psi^1_{22} = 1$ | $\psi^2_{22} = 0$ | $\rho^1_{22} = 1$ | $\rho^2_{22} = 0$ |
| $\psi^1_{23} = 0$ | $\psi^2_{23} = 0$ | $\rho^1_{23} = 2/\epsilon + 1$ | $\rho^2_{23} = 2/\epsilon$ |
| $\psi^1_{31} = 1$ | $\psi^2_{31} = 0$ | $\rho^1_{31} = 1$ | $\rho^2_{31} = 0$ |
| $\psi^1_{32} = 2/\epsilon + 1$ | $\psi^2_{32} = 2/\epsilon$ | $\rho^1_{32} = 0$ | $\rho^2_{32} = 0$ |
| $\psi^1_{33} = 1$ | $\psi^2_{33} = 0$ | $\rho^1_{33} = 1$ | $\rho^2_{33} = 0$ |

**TABLE 6.** Values of the nodes $x'_j$ of the third layer.

| $x'_j = \text{ReLU}(x_j - C \cdot \sum_{k=1}^{j-1}(\psi^1_{kj} - \psi^2_{kj} + \rho^1_{kj} - \rho^2_{kj} - 1))$ |
|---|
| $x'_1 = 2$ |
| $x'_2 = 4$ |
| $x'_3 = 0$ |

$1 \leq i \leq n$. The five hidden layers perform the computation of Eqs. (21) and (22). The second layer keeps a copy of $2d$ nodes $x_j$ and performs the computation of $\delta(x_j, x_k)$ with $4d^2$ nodes $\psi^1_{kj}, \psi^2_{kj}, \rho^1_{kj}, \rho^2_{kj}$, $1 \leq j, k \leq d$ based on Prop. 1. The values of these nodes are:

$$\psi^1_{kj} = \text{ReLU}((x_j - x_k)/\epsilon + 1), \psi^2_{kj} = \text{ReLU}((x_j - x_k)/\epsilon),$$
$$\rho^1_{kj} = \text{ReLU}((x_k - x_j)/\epsilon + 1), \rho^2_{kj} = \text{ReLU}((x_k - x_j)/\epsilon).$$

The third layer keeps a copy of $d$ nodes $x_{j+d}$ and performs the computation of Eq. (20) with $d$ nodes $x'_j$ fo $1 \leq j \leq d$ which is given as:

$$x'_j = \text{ReLU}\left(x_j - C \cdot \sum_{k=1}^{j-1}(\psi^1_{kj} - \psi^2_{kj} + \rho^1_{kj} - \rho^2_{kj} - 1)\right).$$

**TABLE 7.** Values of the nodes $\alpha^1_{ij}, \alpha^2_{ij}, \beta^1_{ij}, \beta^2_{ij}$ of the second layer for $1 \le i \le n, 1 \le j \le d$.

| $\alpha^1_{ij} = \mathrm{ReLU}((x'_j - i)/\epsilon + 1)$ | $\alpha^2_{ij} = \mathrm{ReLU}((x'_j - i)/\epsilon)$ | $\beta^1_{ij} = \mathrm{ReLU}((i - x'_j)/\epsilon + 1)$ | $\beta^2_{ij} = \mathrm{ReLU}((i - x'_j)/\epsilon)$ |
|---|---|---|---|
| $\alpha^1_{11} = 1/\epsilon + 1$ | $\alpha^2_{11} = 1/\epsilon$ | $\beta^1_{11} = 0$ | $\beta^2_{11} = 0$ |
| $\alpha^1_{12} = 3/\epsilon + 1$ | $\alpha^2_{12} = 3/\epsilon$ | $\beta^1_{12} = 0$ | $\beta^2_{12} = 0$ |
| $\alpha^1_{13} = 0$ | $\alpha^2_{13} = 0$ | $\beta^1_{13} = 1/\epsilon + 1$ | $\beta^2_{13} = 1/\epsilon$ |
| $\alpha^1_{21} = 1$ | $\alpha^2_{21} = 0$ | $\beta^1_{21} = 1$ | $\beta^2_{21} = 0$ |
| $\alpha^1_{22} = 2/\epsilon + 1$ | $\alpha^2_{22} = 2/\epsilon$ | $\beta^1_{22} = 0$ | $\beta^2_{22} = 0$ |
| $\alpha^1_{23} = 0$ | $\alpha^2_{23} = 0$ | $\beta^1_{23} = 2/\epsilon + 1$ | $\beta^2_{23} = 2/\epsilon$ |
| $\alpha^1_{31} = 0$ | $\alpha^2_{31} = 0$ | $\beta^1_{31} = 1/\epsilon + 1$ | $\beta^2_{31} = 1/\epsilon$ |
| $\alpha^1_{32} = 1/\epsilon + 1$ | $\alpha^2_{32} = 1/\epsilon$ | $\beta^1_{32} = 0$ | $\beta^2_{32} = 0$ |
| $\alpha^1_{33} = 0$ | $\alpha^2_{33} = 0$ | $\beta^1_{33} = 3/\epsilon + 1$ | $\beta^2_{33} = 3/\epsilon$ |
| $\alpha^1_{41} = 0$ | $\alpha^2_{41} = 0$ | $\beta^1_{41} = 2/\epsilon + 1$ | $\beta^2_{41} = 2/\epsilon$ |
| $\alpha^1_{42} = 1$ | $\alpha^2_{42} = 0$ | $\beta^1_{42} = 1$ | $\beta^2_{42} = 0$ |
| $\alpha^1_{43} = 0$ | $\alpha^2_{43} = 0$ | $\beta^1_{43} = 4/\epsilon + 1$ | $\beta^2_{43} = 4/\epsilon$ |

**TABLE 8.** Values of the nodes $\lambda_i$ of the third layer.

| $\lambda_i = \mathrm{ReLU}(\sum_{j=1}^{d}(-C\alpha^1_{ij} + C\alpha^2_{ij} - C\beta^1_{ij} + C\beta^2_{ij}) + (e_i + Cd))$ |
|---|
| $\lambda_1 = 4$ |
| $\lambda_2 = 0$ |
| $\lambda_3 = 3$ |
| $\lambda_4 = 0$ |

**TABLE 9.** Values of the nodes $\mu_{ij}$ of the third layer.

| $\mu_{i1} = \mathrm{ReLU}(C\alpha^1_{i1} - C\alpha^2_{i1} + C\beta^1_{i1} - C\beta^2_{i1} + x_{d+1} - 2C)$ | $\mu_{i2} = \mathrm{ReLU}(C\alpha^1_{i2} - C\alpha^2_{i2} + C\beta^1_{i2} - C\beta^2_{i2} + x_{d+2} - 2C)$ | $\mu_{i3} = \mathrm{ReLU}(C\alpha^1_{i3} - C\alpha^2_{i3} + C\beta^1_{i3} - C\beta^2_{i3} + x_{d+3} - 2C)$ |
|---|---|---|
| $\mu_{11} = 0$ | $\mu_{12} = 0$ | $\mu_{13} = 0$ |
| $\mu_{21} = 5$ | $\mu_{22} = 0$ | $\mu_{23} = 0$ |
| $\mu_{31} = 0$ | $\mu_{32} = 0$ | $\mu_{33} = 0$ |
| $\mu_{41} = 0$ | $\mu_{42} = 1$ | $\mu_{43} = 0$ |

**TABLE 10.** Values of the nodes $\gamma_i$ of the fourth layer.

| $\gamma_i = \mathrm{ReLU}(\sum_{j=1}^{d} \mu_{ij})$ |
|---|
| $\gamma_1 = 0$ |
| $\gamma_2 = 5$ |
| $\gamma_3 = 0$ |
| $\gamma_4 = 1$ |

**TABLE 11.** Values of the nodes $y_i$ of the output layer.

| $y_i = \lambda_i + \gamma_i$ |
|---|
| $y_1 = 4$ |
| $y_2 = 5$ |
| $y_3 = 3$ |
| $y_4 = 1$ |

The fourth layer keeps a copy of $d$ nodes $x_{j+d}$ and performs the computation of $\delta(p_j, i)$ with $4nd$ nodes $\alpha^1_{ij}, \alpha^2_{ij}, \beta^1_{ij}, \beta^2_{ij}, 1 \le i \le n, 1 \le j \le d$ based on Prop. 1. The value of these nodes are:

$$\alpha^1_{ij} = \mathrm{ReLU}((x'_j - i)/\epsilon + 1), \alpha^2_{ij} = \mathrm{ReLU}((x'_j - i)/\epsilon),$$
$$\beta^1_{ij} = \mathrm{ReLU}((i - x'_j)/\epsilon + 1), \beta^2_{ij} = \mathrm{ReLU}((i - x'_j)/\epsilon).$$

The computation of Eq. (21) and $\max(x_{j+d} - C(1 - \delta(p_j, i)), 0)$ in Eq. (22) is performed in the fifth layer with $n$ and $nd$ nodes $\lambda_i$ and $\mu_{ij}$, resp., such that

$$\lambda_i = \mathrm{ReLU}\left(\sum_{j=1}^{d}(-C\alpha^1_{ij} + C\alpha^2_{ij} - C\beta^1_{ij} + C\beta^2_{ij}) + (e_i + Cd)\right),$$
$$\mu_{ij} = \mathrm{ReLU}\left(C\alpha^1_{ij} - C\alpha^2_{ij} + C\beta^1_{ij} - C\beta^2_{ij} + x_{d+j} - 2C\right).$$

The sixth layer keeps a copy of $n$ nodes $\lambda_i$, and is used to perform the computation of Eq. (21) with $n$ nodes as follows

$$\gamma_i = \mathrm{ReLU}\left(\sum_{j=1}^{d} \mu_{ij}\right).$$

Finally, the output layer computes $y_i$ with the following equation

$$y_i = \lambda_i + \gamma_i.$$

Hence the constructed neural network is an $H_d$-generative ReLU with size $\mathcal{O}(nd)$ and constant depth.

*Example 3: Suppose* $\Sigma = \{1, 2, 3, 4, 5\}$, $d = 3$, $e = (4, 1, 3, 2)$ *and* $y = (4, 5, 3, 1)$. *The strings e and y differ at positions* $i = 2, 4$, *and therefore their hamming distance* $d'$ *is* 2. *By using* $x = (2, 4, 2, 5, 1, 3)$ *in Eqs. (20)- (23), we get the vectors* $p = [2, 4, 0]$, $q = [4, 0, 3, 0]$, $r = [0, 5, 0, 1]$, $y = (4, 5, 3, 1)$. *We show that the* $H_d$*-generative ReLU constructed by using Theorem 2 outputs the string y from the input x. The values of each node of the layers are listed in Tables 5- 11.*

*Proof of Theorem 4:* Suppose $e = (e_1, e_2, \ldots, e_n)$ and let $y = (y_1, y_2, \ldots, y_{n+d})$ be a string such that $y$ can be obtained from $e$ by inserting the values $y_{k_j}$ before the positions $k_j$, where $k_j \le m$, $1 \le j \le d$. Note that more than one entry can be inserted before a fixed position, and therefore $\{k_1, k_2, \ldots, k_d\}$ is a multi-set. Furthermore the edit distance of $y$ from $e$ due to insertion is exactly $d$. Construct $x = (x_1, x_2, \ldots, x_{2d})$ such that $x_j \in \{k_1, k_2, \ldots, k_d\}$ and $x_{j+d} = y_{k_j}$ for $1 \le j \le d$. Then we see that $y$ can be obtained from $e$ using $x$ with the following system of

**TABLE 12.** Values of the nodes of the second layer.

| $\alpha^1_{ij} = \text{ReLU}((x_j - i)/\epsilon + 1)$ | $\alpha^2_{ij} = \text{ReLU}((x_j - i)/\epsilon)$ | $\beta^1_{ij} = \text{ReLU}((i - x_j)/\epsilon + 1)$ | $\beta^2_{ij} = \text{ReLU}((i - x_j)/\epsilon)$ |
|---|---|---|---|
| $\alpha^1_{11} = 1$ | $\alpha^2_{11} = 0$ | $\beta^1_{11} = 1$ | $\beta^2_{11} = 0$ |
| $\alpha^1_{12} = 2/\epsilon + 1$ | $\alpha^2_{12} = 2/\epsilon$ | $\beta^1_{12} = 0$ | $\beta^2_{12} = 0$ |
| $\alpha^1_{21} = 0$ | $\alpha^2_{21} = 0$ | $\beta^1_{21} = 1/\epsilon + 1$ | $\beta^2_{21} = 1/\epsilon$ |
| $\alpha^1_{22} = 1/\epsilon + 1$ | $\alpha^2_{22} = 1/\epsilon$ | $\beta^1_{22} = 0$ | $\beta^2_{22} = 0$ |
| $\alpha^1_{31} = 0$ | $\alpha^2_{31} = 0$ | $\beta^1_{31} = 2/\epsilon + 1$ | $\beta^2_{31} = 2/\epsilon$ |
| $\alpha^1_{32} = 1$ | $\alpha^2_{32} = 0$ | $\beta^1_{32} = 1$ | $\beta^2_{32} = 0$ |
| $\alpha^1_{41} = 0$ | $\alpha^2_{41} = 0$ | $\beta^1_{41} = 3/\epsilon + 1$ | $\beta^2_{41} = 3/\epsilon$ |
| $\alpha^1_{42} = 0$ | $\alpha^2_{42} = 0$ | $\beta^1_{42} = 1/\epsilon + 1$ | $\beta^2_{42} = 1/\epsilon$ |

**TABLE 13.** Values of the nodes of the third layer.

| $\eta^1_i = \text{ReLU}(\sum_{j=1}^{d}(-\alpha^1_{ij} - \beta^1_{ij} + \alpha^2_{ij} + \beta^2_{ij})/\epsilon + d/\epsilon + 1)$ | $\eta^2_i = \text{ReLU}(\sum_{j=1}^{d}(-\alpha^1_{ij} - \beta^1_{ij} + \alpha^2_{ij} + \beta^2_{ij})/\epsilon + d/\epsilon)$ |
|---|---|
| $\eta^1_1 = 0$ | $\eta^2_1 = 0$ |
| $\eta^1_2 = 1$ | $\eta^2_2 = 0$ |
| $\eta^1_3 = 0$ | $\eta^2_3 = 0$ |
| $\eta^1_4 = 1$ | $\eta^2_4 = 0$ |

**TABLE 14.** Values of $\gamma_i$ nodes of the fourth layer.

| $\gamma_i = \text{ReLU}(\sum_{k=1}^{i}(\eta^1_k - \eta^2_k))$ |
|---|
| $\gamma_1 = 0$ |
| $\gamma_2 = 1$ |
| $\gamma_3 = 1$ |
| $\gamma_4 = 2$ |

equations.

$$p_j = \sum_{k=1}^{d} H(x_j, x_k) - \sum_{k=j}^{d} \delta(x_j, x_k) \text{ for } j \in \{1, 2, \ldots, d\}, \tag{24}$$

$$q_\ell = \sum_{j=1}^{d} \max(x_j - C \cdot (1 - \delta(\ell, p_j + 1)), 0) \text{ for } \ell \in \{1, 2, \ldots, d\}, \text{ where } C \text{ is a constant with } C \gg \max(m, n), \tag{25}$$

$$q_{d+\ell} = \sum_{j=1}^{d} \max(x_{d+j} - C \cdot (1 - \delta(\ell, p_j + 1)), 0) \text{ for } \ell \in \{1, 2, \ldots, d\}, \tag{26}$$

$$r_i = \sum_{\ell=1}^{d} \delta(q_\ell, i) \text{ for } i \in \{1, \ldots, n+1\}, \tag{27}$$

$$t_i = i + \sum_{k=1}^{i} r_k \text{ for } i \in \{1, 2, \ldots, n\}, \tag{28}$$

$$u^\ell_i = \max(e_i - C \cdot (1 - \delta(t_i, i + \ell - 1)), 0) \text{ for } i \in \{1, 2, \ldots, n\},\ \ell \in \{1, 2, \ldots, d+1\}, \tag{29}$$

$$f_h = \sum_{\forall i, \ell | i+\ell-1=h} u^\ell_i \text{ for } h \in \{1, 2, \ldots, n+d\}, \tag{30}$$

$$v_\ell = q_\ell + \ell - 1 \text{ for } \ell \in \{1, 2, \ldots, d\}, \tag{31}$$

$$w^\ell_i = \max(q_{d+\ell} - C \cdot (1 - \delta(v_\ell, i + \ell - 1)), 0) \text{ for } i \in \{1, \ldots, n+1\}, \ell \in \{1, 2, \ldots, d\}, \tag{32}$$

$$g_h = \sum_{\forall i, \ell | i+\ell-1=h} w^\ell_i \text{ for } h \in \{1, 2, \ldots, n+d\}, \tag{33}$$

$$y_h = f_h + g_h \text{ for } h \in \{1, 2, \ldots, n+d\}. \tag{34}$$

Eqs. (24)-(26) are used to realize the values of indices $x_j$ in ascending order and accordingly the values of $x_{d+j}$. Eqs. (27)-(30) are used to determine the position $h$ of $e_i$ in $y$. Intuitively, the position of $e_i$ will be shifted by the number of insertions to be performed before $e_i$. This implies that we can express the position $h$ of $e_i$ as $h = i + \ell - 1$ for some $\ell$, $1 \le \ell \le d + 1$, where $\ell - 1$ is the insertions before $e_i$. The variable $r_i$ counts the number of insertions between $e_{i-1}$ and $e_i$, if it exists. The term $\sum_{k=1}^{i} r_k$ in Eq. (28) gives the number of all insertions before $e_i$, and therefore $t_i$ is the position of $e_i$ in $y$. The value of the variable $u^\ell_i$ is $e_i$ if and only if the number of insertions before $e_i$ is $\ell - 1$. The variable $f_h$ determines the values from $e$ at position $h$ in $y$, i.e., $f_h = e_i$ for some $i$ if and only if the determined position $i + \ell - 1$ of $e_i$ in $y$ is equal to $h$. Eqs. (31)-(33) are used to determine the position $h$ in $y$ of the new values $q_{\ell+d}$ to be inserted in $e$. It is easy to observe that the position of $q_{\ell+d}$ that should be inserted before $q_\ell = i$ will be $h = i+\ell-1$. Intuitively, $\ell$ is the number of entries to be inserted until $e_i$. The variable $v_\ell$ in Eq. (31) outputs the position of the new entry to be inserted before $q_\ell$. The variable $u^\ell_i$ is $q_{\ell+d}$ if and only if the position of $q_{\ell+d}$ in $y$ is $i + \ell - 1$. The variable $g_h$ determines the value at the position $h$ in $y$, i.e., $g_h = q_{\ell+d}$ if and only if the determined position $i + \ell - 1$ of the new entry $q_\ell$ is equal to $h$. Finally, adding $f_h$ and $g_h$ gives the output entry $y_h$ since $e_i$ and inserted entries cannot have the same position in $y$.

Construct an eight-layer neural network with six hidden layers and ReLU as an activation function to perform the computation of Eqs. (24)-(34), where the first layer is the input layer with $2d$ nodes $x_j$, and the last layer is the output layer with $n + d$ nodes $y_i$. An illustration of the network is given in Figure 7. The second layer keeps a copy of $2d$ nodes $x_j$. By using Prop. 1, this layer also computes $H(x_j, x_k)$ and $\delta(x_j, x_k)$ with $2d^2$ and $4d^2$ nodes, resp., as follows

$$\eta^1_{kj} = \text{ReLU}((x_j - x_k)/\epsilon + 1),\ \eta^2_{kj} = \text{ReLU}((x_j - x_k)/\epsilon),$$
$$\psi^{11}_{kj} = \text{ReLU}((x_j - x_k)/\epsilon + 1),\ \psi^{12}_{kj} = \text{ReLU}((x_j - x_k)/\epsilon),$$
$$\rho^{11}_{kj} = \text{ReLU}((x_k - x_j)/\epsilon + 1),\ \rho^{12}_{kj} = \text{ReLU}((x_k - x_j)/\epsilon).$$

**TABLE 15.** Values of $\mu_i$ and $\lambda_i$ nodes of the fourth layer.

| $\mu_i^1 = \text{ReLU}((\eta_i^1 - \eta_i^2)/\epsilon + 1)$ | $\mu_i^2 = \text{ReLU}((\eta_i^1 - \eta_i^2)/\epsilon)$ | $\lambda_i^1 = \text{ReLU}((-\eta_i^1 + \eta_i^2)/\epsilon + 1)$ | $\lambda_i^2 = \text{ReLU}((-\eta_i^1 + \eta_i^2)/\epsilon)$ |
|---|---|---|---|
| $\mu_1^1 = 1$ | $\mu_1^2 = 0$ | $\lambda_1^1 = 1$ | $\lambda_1^2 = 0$ |
| $\mu_2^1 = 1/\epsilon + 1$ | $\mu_2^2 = 1/\epsilon$ | $\lambda_2^1 = 0$ | $\lambda_2^2 = 0$ |
| $\mu_3^1 = 1$ | $\mu_3^2 = 0$ | $\lambda_3^1 = 1$ | $\lambda_3^2 = 0$ |
| $\mu_4^1 = 1/\epsilon + 1$ | $\mu_4^2 = 1/\epsilon$ | $\lambda_4^1 = 0$ | $\lambda_4^2 = 0$ |

**TABLE 16.** Values of the nodes of the fifth layer.

| $\tau_i = \text{ReLU}(B\gamma_i - C(\mu_i^1 - \mu_i^2 + \lambda_i^1 - \lambda_i^2) + C)$ |
|---|
| $\tau_1 = 0$ |
| $\tau_2 = B$ |
| $\tau_3 = 0$ |
| $\tau_4 = 2B$ |

The third layer keeps a copy of $2d$ nodes $x_j$ and computes $p_j$ with $d$ nodes, as follows

$$\varrho_j = \text{ReLU}(\sum_{k=1}^{d}(\eta_{kj}^1 - \eta_{kj}^2) - \sum_{k=j}^{d}(\psi_{kj}^{11} - \psi_{kj}^{12} + \rho_{kj}^{11} - \rho_{kj}^{12}) + (d - j + 1)).$$

The fourth layer keeps a copy of $2d$ nodes $x_j$. By using Prop. 1, this layer also computes $\delta(\ell, p_j + 1)$ with $4d^2$ nodes, as follows

$$\begin{aligned}\psi_{\ell j}^{21} &= \text{ReLU}((\varrho_j + 1 - \ell)/\epsilon + 1),\\ \psi_{\ell j}^{22} &= \text{ReLU}((\varrho_j + 1 - \ell)/\epsilon),\\ \rho_{\ell j}^{21} &= \text{ReLU}((\ell - \varrho_j - 1)/\epsilon + 1),\\ \rho_{\ell j}^{22} &= \text{ReLU}((\ell - \varrho_j - 1)/\epsilon).\end{aligned}$$

The fifth layer shows the computations of $\max(x_j - C(1 - \delta(\ell, p_j + 1)), 0)$ and $\max(x_{d+j} - C(1 - \delta(\ell, p_j + 1)), 0)$ with $d^2$ nodes, as follows

$$\chi_{\ell j} = \text{ReLU}\left(x_j + C\psi_{\ell j}^{21} - C\psi_{\ell j}^{22} + C\rho_{\ell j}^{21} - C\rho_{\ell j}^{22} - 2C\right),$$
$$\chi_{\ell(d+j)} = \text{ReLU}\left(x_{d+j} + C\psi_{\ell j}^{21} - C\psi_{\ell j}^{22} + C\rho_{\ell j}^{21} - C\rho_{\ell j}^{22} - 2C\right).$$

The sixth layer shows the computations of Eqs. (25) and (26) with $d^2$ nodes, as follows

$$x'_\ell = \text{ReLU}\left(\sum_{j=1}^{d}\chi_{\ell j}\right),$$
$$x'_{d+\ell} = \text{ReLU}\left(\sum_{j=1}^{d}\chi_{\ell(d+j)}\right).$$

The seventh layer keeps a copy of $d$ nodes $x'_{d+\ell}$. By using Prop. 1, this layer also computes $\delta(q_\ell, i)$ and $v_\ell$ used in Eqs. (27) and (31) with $4(n+1)d$, and $d$ nodes, resp., as follows

$$\begin{aligned}\alpha_{i\ell}^1 &= \text{ReLU}((x'_\ell - i)/\epsilon + 1), \alpha_{i\ell}^2 &&= \text{ReLU}((x'_\ell - i)/\epsilon),\\ \beta_{i\ell}^1 &= \text{ReLU}((i - x'_\ell)/\epsilon + 1), \beta_{i\ell}^2 &&= \text{ReLU}((i - x'_\ell)/\epsilon),\\ \gamma_\ell &= \text{ReLU}(x'_\ell + \ell - 1).\end{aligned}$$

The eighth layer keeps a copy of $d$ nodes $x_{d+\ell}$, and computes $r_i$ and $\delta(v_\ell, i + \ell - 1)$ used in Eqs. (27) and (32) with $n+1$, and $4(n+1)d$ nodes, resp., by using Prop. 1 as follows:

$$\begin{aligned}\tau_i &= \text{ReLU}(\sum_{\ell=1}^{d}(\alpha_{i\ell}^1 - \alpha_{i\ell}^2 + \beta_{i\ell}^1 - \beta_{i\ell}^2) - d),\\ \mu_{i\ell}^{11} &= \text{ReLU}((\gamma_\ell - (i + \ell - 1))/\epsilon + 1),\\ \mu_{i\ell}^{12} &= \text{ReLU}((\gamma_\ell - (i + \ell - 1))/\epsilon),\\ \lambda_{i\ell}^{11} &= \text{ReLU}(((i + \ell - 1) - \gamma_\ell)/\epsilon + 1),\\ \lambda_{i\ell}^{12} &= \text{ReLU}(((i + \ell - 1) - \gamma_\ell)/\epsilon).\end{aligned}$$

The computation of $t_i$ and $w_i^\ell$ used in Eqs. (28) and (32) is performed in the ninth layer with $n$ and $(n+1)d$ nodes, resp., such that

$$\tau'_i = \text{ReLU}(i + \sum_{k=1}^{i}\tau_k),$$
$$\omega_{i\ell} = \text{ReLU}(C\mu_{i\ell}^{11} - C\mu_{i\ell}^{12} + C\lambda_{i\ell}^{11} - C\lambda_{i\ell}^{12} + x'_{d+\ell} - 2C).$$

The tenth layer is used to compute $\delta(t_i, i + \ell - 1)$ and $g_h$ used in Eqs. (29) and (33) with $4(d+1)n$ and $n + d$ nodes, resp., by using Prop. 1 as follows

$$\begin{aligned}\mu_{i\ell}^{21} &= \text{ReLU}((\tau'_i - (i + \ell - 1))/\epsilon + 1),\\ \mu_{i\ell}^{22} &= \text{ReLU}((\tau'_i - (i + \ell - 1))/\epsilon),\\ \lambda_{i\ell}^{21} &= \text{ReLU}(((i + \ell - 1) - \tau'_i)/\epsilon + 1),\\ \lambda_{i\ell}^{22} &= \text{ReLU}(((i + \ell - 1) - \tau'_i)/\epsilon),\\ \zeta_h &= \text{ReLU}(\sum_{\forall i,\ell | i+\ell-1=h}\omega_{i\ell}).\end{aligned}$$

The eleventh layer keeps a copy of $n + d$ nodes $\zeta_h$, and computes $u_i^\ell$ with $n(d+1)$ nodes $\omega'_{i\ell}$ that satisfies the following equation

$$\omega'_{i\ell} = \text{ReLU}(C\mu_{i\ell}^{21} - C\mu_{i\ell}^{22} + C\lambda_{i\ell}^{21} - C\lambda_{i\ell}^{22} + e_i - 2C).$$

The twelfth layer again keeps a copy of $n + d$ nodes $\zeta_h$, and computes $f_h$ in Eq. (30) with $(n + d)$ nodes $\zeta'_h$ such that

$$\zeta'_h = \text{ReLU}(\sum_{\forall i,\ell | i+\ell-1=h}\omega'_{i\ell})$$

Finally, the output layer computes $y_h$ as follows

$$y_h = \zeta'_h + \zeta_h.$$

Hence the constructed network is an $EI_d$-generative ReLU of size $\mathcal{O}(nd)$ and constant depth.

**TABLE 17.** Values of the nodes of the sixth layer.

| $\psi^1_{ij} = \text{ReLU}(\tau_{i+j-1}/\epsilon + (-iB/\epsilon + 1))$ | $\psi^2_{ij} = \text{ReLU}(\tau_{i+j-1}/\epsilon + (-iB/\epsilon))$ | $\rho^1_{ij} = \text{ReLU}(-\tau_{i+j-1}/\epsilon + (iB+1)/\epsilon + 1)$ | $\rho^2_{ij} = \text{ReLU}(-\tau_{i+j-1}/\epsilon + (iB+1)/\epsilon)$ |
|---|---|---|---|
| $\psi^1_{11} = 0$ | $\psi^2_{11} = 0$ | $\rho^1_{11} = (B+1)/\epsilon + 1$ | $\rho^2_{11} = (B+1)/\epsilon$ |
| $\psi^1_{12} = 1$ | $\psi^2_{12} = 0$ | $\rho^1_{12} = 1/\epsilon + 1$ | $\rho^2_{12} = 1/\epsilon$ |
| $\psi^1_{13} = 0$ | $\psi^2_{13} = 0$ | $\rho^1_{13} = (B+1)/\epsilon + 1$ | $\rho^2_{13} = (B+1)/\epsilon$ |
| $\psi^1_{21} = 0$ | $\psi^2_{21} = 0$ | $\rho^1_{21} = 1/\epsilon + 1$ | $\rho^2_{21} = 1/\epsilon$ |
| $\psi^1_{22} = 0$ | $\psi^2_{22} = 0$ | $\rho^1_{22} = (2B+1)/\epsilon + 1$ | $\rho^2_{22} = (2B+1)/\epsilon$ |
| $\psi^1_{23} = 1$ | $\psi^2_{23} = 0$ | $\rho^1_{23} = 1/\epsilon + 1$ | $\rho^2_{23} = 1/\epsilon$ |

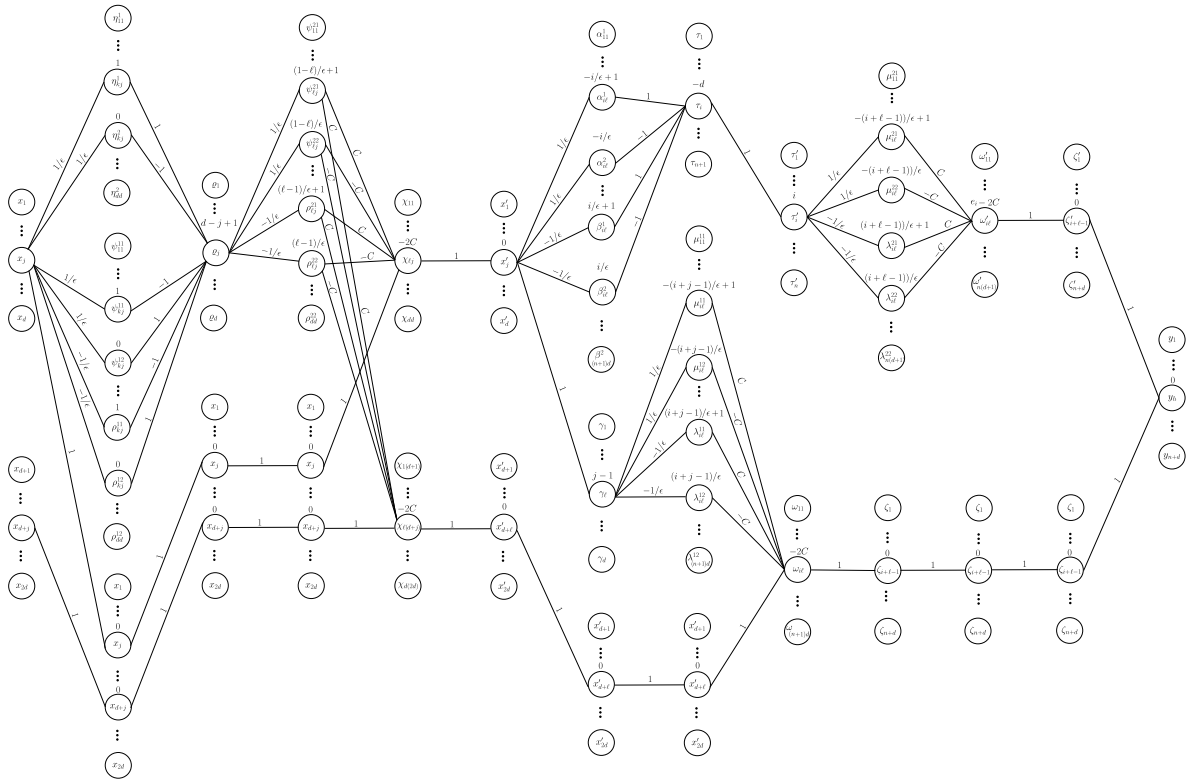

**FIGURE 7.** An illustration of $EI_d$-generative ReLU with the eleven hidden layers, where a few nodes and edges with their bias and non-zero weights are shown, respectively.

**TABLE 18.** Values of the nodes of the seventh layer.

| $\zeta_{ij} = \text{ReLU}(e_{i+j-1} - C(-\psi^1_{ij} + \psi^2_{ij} - \rho^1_{ij} + \rho^2_{ij} + 2))$ |
|---|
| $\zeta_{11} = 0$ |
| $\zeta_{12} = 2$ |
| $\zeta_{13} = 0$ |
| $\zeta_{21} = 0$ |
| $\zeta_{22} = 0$ |
| $\zeta_{23} = 1$ |

**TABLE 19.** Values of the nodes of the output layer.

| $y_i = \sum_{j=1}^{d+1} \zeta_{ij}$ |
|---|
| $y_1 = 2$ |
| $y_2 = 1$ |

**TABLE 20.** Values of the nodes $\eta^1_{kj}, \eta^2_{kj}$ of the second layer.

| $\eta^1_{kj} = \text{ReLU}((x_j - x_k)/\epsilon + 1)$ | $\eta^2_{kj} = \text{ReLU}((x_j - x_k)/\epsilon)$ |
|---|---|
| $\eta^1_{11} = 1$ | $\eta^2_{11} = 0$ |
| $\eta^1_{12} = 0$ | $\eta^2_{12} = 0$ |
| $\eta^1_{13} = 1$ | $\eta^2_{13} = 0$ |
| $\eta^1_{21} = 3/\epsilon + 1$ | $\eta^2_{21} = 3/\epsilon$ |
| $\eta^1_{22} = 1$ | $\eta^2_{22} = 0$ |
| $\eta^1_{23} = 3/\epsilon + 1$ | $\eta^2_{23} = 3/\epsilon$ |
| $\eta^1_{31} = 1$ | $\eta^2_{31} = 0$ |
| $\eta^1_{32} = 0$ | $\eta^2_{32} = 0$ |
| $\eta^1_{33} = 1$ | $\eta^2_{33} = 0$ |

*Example 4: Suppose that* $\Sigma = \{1, 2, 3, 4, 5\}$, $e = (3, 2, 4, 1)$, $d = 3$ *and* $y = (4, 3, 2, 4, 2, 5, 1)$. *By using* $x = (4, 1, 4, 3, 2, 5)$ *in Eqs. (24)- (34), we get the vectors* $p = [1, 0, 2]$, $q = [1, 4, 4, 2, 3, 5]$, $r = [1, 0, 0, 2, 0]$, $t = [2, 3, 4, 7]$, $u = [[0, 3, 0, 0], [0, 2, 0, 0], [0, 4, 0, 0], [0, 0, 0, 1]]$, $f = [0, 3, 2, 4, 0, 0, 1]$, $v = [1, 5, 6]$,

**TABLE 21.** Values of the nodes $\psi_{kj}^{11}, \psi_{kj}^{12}, \rho_{kj}^{11}, \rho_{kj}^{12}$ of the second layer.

| $\psi_{kj}^{11} = \mathrm{ReLU}((x_j - x_k)/\epsilon + 1)$ | $\psi_{kj}^{12} = \mathrm{ReLU}((x_j - x_k)/\epsilon)$ | $\rho_{kj}^{11} = \mathrm{ReLU}((x_k - x_j)/\epsilon + 1)$ | $\rho_{kj}^{12} = \mathrm{ReLU}((x_k - x_j)/\epsilon)$ |
|---|---|---|---|
| $\psi_{11}^{11} = 1$ | $\psi_{11}^{12} = 0$ | $\rho_{11}^{11} = 1$ | $\rho_{11}^{12} = 0$ |
| $\psi_{12}^{11} = 0$ | $\psi_{12}^{12} = 0$ | $\rho_{12}^{11} = 3/\epsilon + 1$ | $\rho_{12}^{12} = 3/\epsilon$ |
| $\psi_{13}^{11} = 1$ | $\psi_{13}^{12} = 0$ | $\rho_{13}^{11} = 1$ | $\rho_{13}^{12} = 0$ |
| $\psi_{21}^{11} = 3/\epsilon + 1$ | $\psi_{21}^{12} = 3/\epsilon$ | $\rho_{21}^{11} = 0$ | $\rho_{21}^{12} = 0$ |
| $\psi_{22}^{11} = 1$ | $\psi_{22}^{12} = 0$ | $\rho_{22}^{11} = 1$ | $\rho_{22}^{12} = 0$ |
| $\psi_{23}^{11} = 3/\epsilon + 1$ | $\psi_{23}^{12} = 3/\epsilon$ | $\rho_{23}^{11} = 0$ | $\rho_{23}^{12} = 0$ |
| $\psi_{31}^{11} = 1$ | $\psi_{31}^{12} = 0$ | $\rho_{31}^{11} = 1$ | $\rho_{31}^{12} = 0$ |
| $\psi_{32}^{11} = 0$ | $\psi_{32}^{12} = 0$ | $\rho_{32}^{11} = 3/\epsilon + 1$ | $\rho_{32}^{12} = 3/\epsilon$ |
| $\psi_{33}^{11} = 1$ | $\psi_{33}^{12} = 0$ | $\rho_{33}^{11} = 1$ | $\rho_{33}^{12} = 0$ |

**TABLE 22.** Values of the nodes $\varrho_j$ of the third layer.

| $\varrho_j = \mathrm{ReLU}(\sum_{k=1}^{d}(\eta_{kj}^1 - \eta_{kj}^2) - \sum_{k=j}^{d}(\psi_{kj}^{11} - \psi_{kj}^{12} + \rho_{kj}^{11} - \rho_{kj}^{12}) + (d - j + 1))$ |
|---|
| $\varrho_1 = 1$ |
| $\varrho_2 = 0$ |
| $\varrho_2 = 2$ |

$w = [[2, 0, 0], [0, 0, 0], [0, 0, 0], [0, 3, 5], [0, 0, 0]]$ $g = [2, 0, 0, 0, 3, 5, 0]$, *and* $y = (2, 3, 2, 4, 3, 5, 1)$. *We show that the* $EI_d$*-generative ReLU constructed by using Theorem 4 outputs* $y$ *for the input* $x$. *The values of each node of the layers are listed in Tables 20- 36.*

*Proof of Theorem 5:* Suppose $e = (e_1, e_2, \ldots, e_n)$ is a string over $\Sigma$, and $x = (x_1, \ldots, x_j, \ldots, x_{5d})$ be an arbitrary string over $[0, 1)$ with $x_j = i \cdot \Delta$, where $\Delta$ is a small constant. We consider that the entries $x_j$, $1 \le j \le 2d$, (resp., $2d + 1 \le j \le 3d$ and $3d + 1 \le j \le 5d$) correspond to substitution (resp., deletion and insertion) operations. These operations are performed in the following six steps which consist of pre-processing of $x$, padding on $e$ followed by substitution, deletion, insertion, and trimming operations.

(i) Perform a pre-processing to convert $x$ into an integer string such that the real value $x_j$ with $1 \le j \le d$, $2d+1 \le j \le 3d$, $3d+1 \le j \le 4d$ (resp., $d+1 \le j \le 2d$, $4d+1 \le j \le 5d$) in the intervals $0, (0, 1/(n+1)], \ldots, (n/(n+1), (n+1)/(n+1)]$ (resp., $[0, 1/m], (1/m, 2/m], \ldots, ((m-1)/m, m/m]$ is converted into $0, 1, \ldots, n+1$ (resp., $1, 2, \ldots, m$). This is performed by first identifying the range and then applying the threshold function by using Eqs. (35)- (38). The variable $p_i^j$ (resp., $q_\ell^i$) in Eq. (35) (resp., Eq. (36)) are used to identify the range of $x_j$. More precisely $p_i^j$ (resp., $q_\ell^i$) is 1 if and only if $x_j$ is in the $i$-th interval from $0, (0, 1/(n+1)], \ldots, (n/(n+1), (n+1)/(n+1)]$ (resp., $[0, 1/m], (1/m, 2/m], \ldots, ((m-1)/m, m/m]$). The variables $p_j'$ and $q_j'$ store the integer value corresponding to $x_j$, i.e., $p_j' = i$ (resp., $q_j' = \ell$) if and only if $p_i^j = 1$ (resp., $q_\ell^i = 1$).

$$p_i^j = \big[(i-1)/(n+1) \le x_j \le i/(n+1)\big] - \delta(x_j, (i-1)/(n+1)), \quad \text{for } i \in \{0, 1, \ldots, n+1\}, j \in \{1, \ldots, d, 2d+1, \ldots, 4d\}, \tag{35}$$

For $j \in \{d+1, \ldots, 2d, 4d+1, \ldots, 5d\}$,

$$q_\ell^j = \begin{cases} \big[(\ell-1)/m \le x_j \le \ell/m\big] & \text{if } \ell = 1, \\ \big[(\ell-1)/m \le x_j \le \ell/m\big] - \delta(x_j, (\ell-1)/m). & \text{if } \ell \in \{2, \ldots, m\}, \end{cases} \tag{36}$$

$$p_j' = \sum_{i=0}^{n+1} p_i^j \cdot i \text{ for } j \in \{1, \ldots, d, 2d+1, \ldots, 4d\}. \tag{37}$$

$$q_j' = \sum_{\ell=1}^{m} q_\ell^j \cdot \ell \text{ for } j \in \{d+1, \ldots, 2d, 4d+1, \ldots, 5d\}. \tag{38}$$

Let $x$ be the converted string. To avoid performing redundant operations, ignore $x_j$ if it is repeated for $1 \le j \le d$, $2d+1 \le j \le 3d$, or set $x_j := 0$ if it has at least $(d+1)$-th index among the non-zero and non-repeating values for $1 \le j \le d$, $2d+1 \le j \le 4d$. Furthermore for the insertion indices $3d+1 \le j \le 4d$, set $x_j := n+1$ if $x_j = 0$ since, the insertion operation cannot handle the index zero. These computations are performed using Eqs. (39)- (42) as follows

$$t_j = \begin{cases} \max(1 - (\delta(p_j', 0) + \sum_{k=1}^{j-1} \delta(p_j', p_k')), 0) \\ \quad \text{if } j \in \{1, \ldots, d\}, \\ \max(1 - (\delta(p_j', 0) + \sum_{k=2d+1}^{j-1} \delta(p_j', p_k')), 0) \\ \quad \text{if } j \in \{2d+1, \ldots, 3d\}, \\ 1 - \delta(p_j', 0) \quad \text{if } j \in \{3d+1, \ldots, 4d\}. \end{cases} \tag{39}$$

$$u_j = \left[\sum_{k=1}^{j} t_k \ge d+1\right] \text{ for } j \in \{1, \ldots, d, 2d+1, \ldots, 4d\}. \tag{40}$$

$$v_j = \max(p_j' - C \cdot u_j, 0) \text{ for } j \in \{1, \ldots, d, 2d+1, \ldots, 4d\}. \tag{41}$$

$$w_j = \begin{cases} v_j \quad \text{if } j \in \{1, \ldots, d, 2d+1, \ldots, 3d\}, \\ \max(v_j + (n+1) \cdot \delta(v_j, 0), 0) \text{ if } j \in \{3d+1, \ldots, 4d\}. \end{cases} \tag{42}$$

**TABLE 23.** Values of the nodes $\psi_{\ell j}^{21}, \psi_{\ell j}^{22}, \rho_{\ell j}^{21}, \rho_{\ell j}^{22}$ of the fourth layer.

| $\psi_{\ell j}^{21} = \text{ReLU}((\varrho_j + 1 - \ell)/\epsilon + 1)$ | $\psi_{\ell j}^{22} = \text{ReLU}((\varrho_j + 1 - \ell)/\epsilon)$ | $\rho_{\ell j}^{21} = \text{ReLU}((\ell - \varrho_j - 1)/\epsilon + 1)$ | $\rho_{\ell j}^{22} = \text{ReLU}((\ell - \varrho_j - 1)/\epsilon)$ |
|---|---|---|---|
| $\psi_{11}^{21} = 1/\epsilon + 1$ | $\psi_{11}^{22} = 1/\epsilon$ | $\rho_{11}^{21} = 0$ | $\rho_{11}^{22} = 0$ |
| $\psi_{12}^{21} = 1$ | $\psi_{12}^{22} = 0$ | $\rho_{12}^{21} = 1$ | $\rho_{12}^{22} = 0$ |
| $\psi_{13}^{21} = 2/\epsilon + 1$ | $\psi_{13}^{22} = 2/\epsilon$ | $\rho_{13}^{21} = 0$ | $\rho_{13}^{22} = 0$ |
| $\psi_{21}^{21} = 1$ | $\psi_{21}^{22} = 0$ | $\rho_{21}^{21} = 1$ | $\rho_{21}^{22} = 0$ |
| $\psi_{22}^{21} = 0$ | $\psi_{22}^{22} = 0$ | $\rho_{22}^{21} = 1/\epsilon + 1$ | $\rho_{22}^{22} = 1/\epsilon$ |
| $\psi_{23}^{21} = 1/\epsilon + 1$ | $\psi_{23}^{22} = 1/\epsilon$ | $\rho_{23}^{21} = 0$ | $\rho_{23}^{22} = 0$ |
| $\psi_{31}^{21} = 0$ | $\psi_{31}^{22} = 0$ | $\rho_{31}^{21} = 1/\epsilon + 1$ | $\rho_{31}^{22} = 1/\epsilon$ |
| $\psi_{32}^{21} = 0$ | $\psi_{32}^{22} = 0$ | $\rho_{32}^{21} = 2/\epsilon + 1$ | $\rho_{32}^{22} = 2/\epsilon$ |
| $\psi_{33}^{21} = 1$ | $\psi_{33}^{22} = 0$ | $\rho_{33}^{21} = 1$ | $\rho_{33}^{22} = 0$ |

**TABLE 24.** Values of the nodes $\chi_{\ell j}$ and $\chi_{\ell(d+j)}$ of the fifth layer.

| $\chi_{\ell j} = \text{ReLU}(x_j + C\psi_{\ell j}^{21} - C\psi_{\ell j}^{22} + C\rho_{\ell j}^{21} - C\rho_{\ell j}^{22}) - 2C)$ | $\chi_{\ell(d+j)} = \text{ReLU}(x_{d+j} + C\psi_{\ell j}^{21} - C\psi_{\ell j}^{22} + C\rho_{\ell j}^{21} - C\rho_{\ell j}^{22}) - 2C)$ |
|---|---|
| $\chi_{11} = 0$ | $\chi_{14} = 0$ |
| $\chi_{12} = 1$ | $\chi_{15} = 2$ |
| $\chi_{13} = 0$ | $\chi_{16} = 0$ |
| $\chi_{21} = 4$ | $\chi_{24} = 3$ |
| $\chi_{22} = 0$ | $\chi_{25} = 0$ |
| $\chi_{23} = 0$ | $\chi_{26} = 0$ |
| $\chi_{31} = 0$ | $\chi_{34} = 0$ |
| $\chi_{32} = 0$ | $\chi_{35} = 0$ |
| $\chi_{33} = 4$ | $\chi_{36} = 5$ |

**TABLE 25.** Values of the nodes $x'_\ell$ and $x'_{d+\ell}$ of the sixth layer.

| $x'_\ell = \text{ReLU}(\sum_{j=1}^{d} \chi_{\ell j})$ | $x'_{d+\ell} = \text{ReLU}(\sum_{j=1}^{d} \chi_{\ell(d+j)})$ |
|---|---|
| $x'_1 = 1$ | $x'_4 = 2$ |
| $x'_2 = 4$ | $x'_5 = 3$ |
| $x'_3 = 4$ | $x'_6 = 5$ |

Here $t_j$ identifies if $x_j$ is non-zero and non-repeated, i.e., $t_j = 1$ if and only if $x_j = p'_j$ is non-zero and non-repeated (resp., $x_j = 0$) for $1 \le j \le d$, $2d + 1 \le j \le 3d$ (resp., $3d + 1 \le j \le 4d$). The variable $u_j$ identifies if $x_j$ has index greater than $d$. More precisely, $u_j = 1$ if and only if $x_j$ has index greater than $d$ among the non-zero and non-repeating values and $v_j = 0$ if and only if $u_j = 1$. Finally, $w_j$ is used to replace 0 with $n+1$ to perform insertion operation, i.e., $w_j = n+1$ if and only if $v_j = 0$ for the insertion indices for $3d + 1 \le j \le 4d$. Basically, $w_j$ and $q'_j$ are the resultant values that are obtained by converting $x_j$ into integers, where $w_j$ is used to identify positions to make changes and $q'_j$ is used to store the value to be substituted or inserted. Finally, get a processed string $x'$ by concatenating $w_j$ and $q'_j$ as $x'_j := w_j$ (resp., $q'_j$) if $1 \le j \le d, 2d+1 \le j \le 4d$ (resp., $d + 1 \le j \le 2d, 4d + 1 \le j \le 5d$).

(ii) Due to the different number of substitution, deletion, and insertion operations, the resultant strings can have different lengths. To handle this issue, increase the length of $e$ by padding $d$ zeros at the end so that the total length of the string becomes $n + d$.

(iii) Apply substitution operations on $e$ using $x'_j$, $1 \le j \le 2d$, as discussed in Theorem 2 and get a string of size $n + d$.

(iv) Apply deletion on the output of (iii) using $x'_j$, $2d+1 \le j \le 3d$, as discussed in Theorem 3 and get a string of size $n$.

(v) Apply insertion operations on the output of (iv) using $x'_j$, $3d + 1 \le j \le 5d$, as discussed in Theorem 4 and get a string of size $n + d$.

(vi) Finally, obtain the resultant string $y$ with the first $n - n_D + n_I$ entries of the output of (v), where $n_D$ is the number of distinct non-zero entries among $x'_j$, $2d + 1 \le j \le 3d$, and $n_I$ is the number of non-zero entries among $x'_j$, $3d+1 \le j \le 4d$, that are in the range $(0, (n + 1 - n_D)/(n + 1)]$ with index less than $d+1$. It is easy to notice that $y$ has edit distance at most $d$ from $e$. Furthermore, the number of substitution, deletion and insertion operations can be controlled in $x$, and therefore any string $y$ over $\Sigma$ with at most distance $d$ from $e$ can be obtained by appropriately selecting $x$ and applying steps (i)-(vi).

Construct a twenty-two-layer neural network with ReLU as an activation function to perform the steps (i)-(v). The first layer is the input layer with $5d$ nodes $x_j$, and the last layer is the output layer of size $n + d$ with output $y'$. The second layer corresponds to Eqs. (35) and (36) which computes $\lceil x_j \ge (i-1)/(n+1) \rceil$, $\lceil -x_j \ge -i/(n+1) \rceil$, $\delta(x_j, (i-1)/(n+1))$, $\lceil x_j \ge (\ell - 1)/m \rceil$, $\lceil -x_j \ge -\ell/m \rceil$ and $\delta(x_j, (\ell - 1)/m)$, by using Props. 1 and 2 with $2(n + 2)(5d)$ nodes $\gamma_{ij}^{11}$, $\gamma_{ij}^{12}$, $2(n + 2)(5d)$ nodes $\gamma_{ij}^{21}$, $\gamma_{ij}^{22}$, $4(n + 2)(5d)$ nodes $\alpha_{ij}^{11}$, $\alpha_{ij}^{12}$, $\beta_{ij}^{11}$, $\beta_{ij}^{12}$, $2(m)(5d)$ nodes $\omega_{\ell j}^{11}$, $\omega_{\ell j}^{12}$, $2(m)(5d)$ nodes $\omega_{\ell j}^{21}$, $\omega_{\ell j}^{22}$ and $4(m)(5d)$ nodes $\alpha_{\ell j}^{21}$, $\alpha_{\ell j}^{22}$, $\beta_{\ell j}^{21}$, $\beta_{\ell j}^{22}$, respectively:

$$
\begin{aligned}
\gamma_{ij}^{11} &= \text{ReLU}(x_j/\epsilon - (i-1)/\epsilon(n+1) + 1),\\
\gamma_{ij}^{12} &= \text{ReLU}(x_j/\epsilon - (i-1)/\epsilon(n+1)),\\
\gamma_{ij}^{21} &= \text{ReLU}(-x_j/\epsilon + i/\epsilon(n+1) + 1),\\
\gamma_{ij}^{22} &= \text{ReLU}(-x_j/\epsilon + i/\epsilon(n+1)),\\
\alpha_{ij}^{11} &= \text{ReLU}(x_j/\epsilon - (i-1)/\epsilon(n+1) + 1),\\
\alpha_{ij}^{12} &= \text{ReLU}(x_j/\epsilon - (i-1)/\epsilon(n+1)),\\
\beta_{ij}^{11} &= \text{ReLU}((i-1)/\epsilon(n+1) - x_j/\epsilon + 1),\\
\beta_{ij}^{12} &= \text{ReLU}((i-1)/\epsilon(n+1) - x_j/\epsilon),\\
\omega_{\ell j}^{11} &= \text{ReLU}(x_j/\epsilon - (\ell-1)/\epsilon m + 1),\\
\omega_{\ell j}^{12} &= \text{ReLU}(x_j/\epsilon - (\ell-1)/\epsilon m),\\
\omega_{\ell j}^{21} &= \text{ReLU}(-x_j/\epsilon + \ell/\epsilon m + 1),\\
\omega_{\ell j}^{22} &= \text{ReLU}(-x_j/\epsilon + \ell/\epsilon m),\\
\alpha_{\ell j}^{21} &= \text{ReLU}(x_j/\epsilon - (\ell-1)/\epsilon m + 1),
\end{aligned}
$$

**TABLE 26.** Values of the nodes $\alpha^1_{i\ell}, \alpha^2_{i\ell}, \beta^1_{i\ell}, \beta^2_{i\ell}$ of the seventh layer.

| $\alpha^1_{i\ell} = \mathrm{ReLU}((x'_\ell - i)/\epsilon + 1)$ | $\alpha^2_{i\ell} = \mathrm{ReLU}((x'_\ell - i)/\epsilon)$ | $\beta^1_{i\ell} = \mathrm{ReLU}((i - x'_\ell)/\epsilon + 1)$ | $\beta^2_{i\ell} = \mathrm{ReLU}((i - x'_\ell)/\epsilon)$ |
|---|---|---|---|
| $\alpha^1_{11} = 1$ | $\alpha^2_{11} = 0$ | $\beta^1_{11} = 1$ | $\beta^2_{11} = 0$ |
| $\alpha^1_{12} = 3/\epsilon + 1$ | $\alpha^2_{12} = 3/\epsilon$ | $\beta^1_{12} = 0$ | $\beta^2_{12} = 0$ |
| $\alpha^1_{13} = 3/\epsilon + 1$ | $\alpha^2_{13} = 3/\epsilon$ | $\beta^1_{13} = 0$ | $\beta^2_{13} = 0$ |
| $\alpha^1_{21} = 0$ | $\alpha^2_{21} = 0$ | $\beta^1_{21} = 1/\epsilon + 1$ | $\beta^2_{21} = 1/\epsilon$ |
| $\alpha^1_{22} = 2/\epsilon + 1$ | $\alpha^2_{22} = 2/\epsilon$ | $\beta^1_{22} = 0$ | $\beta^2_{22} = 0$ |
| $\alpha^1_{23} = 2/\epsilon + 1$ | $\alpha^2_{23} = 2/\epsilon$ | $\beta^1_{23} = 0$ | $\beta^2_{23} = 0$ |
| $\alpha^1_{31} = 0$ | $\alpha^2_{31} = 0$ | $\beta^1_{31} = 2/\epsilon + 1$ | $\beta^2_{31} = 2/\epsilon$ |
| $\alpha^1_{32} = 1/\epsilon + 1$ | $\alpha^2_{32} = 1/\epsilon$ | $\beta^1_{32} = 0$ | $\beta^2_{32} = 0$ |
| $\alpha^1_{33} = 1/\epsilon + 1$ | $\alpha^2_{33} = 1/\epsilon$ | $\beta^1_{33} = 0$ | $\beta^2_{33} = 0$ |
| $\alpha^1_{41} = 0$ | $\alpha^2_{41} = 0$ | $\beta^1_{41} = 3/\epsilon + 1$ | $\beta^2_{41} = 3/\epsilon$ |
| $\alpha^1_{42} = 1$ | $\alpha^2_{42} = 0$ | $\beta^1_{42} = 1$ | $\beta^2_{42} = 0$ |
| $\alpha^1_{43} = 1$ | $\alpha^2_{43} = 0$ | $\beta^1_{43} = 1$ | $\beta^2_{43} = 0$ |
| $\alpha^1_{51} = 0$ | $\alpha^2_{51} = 0$ | $\beta^1_{51} = 4/\epsilon + 1$ | $\beta^2_{51} = 4/\epsilon$ |
| $\alpha^1_{52} = 0$ | $\alpha^2_{52} = 0$ | $\beta^1_{52} = 1/\epsilon + 1$ | $\beta^2_{52} = 1/\epsilon$ |
| $\alpha^1_{53} = 0$ | $\alpha^2_{53} = 0$ | $\beta^1_{53} = 1/\epsilon + 1$ | $\beta^2_{53} = 1/\epsilon$ |

**TABLE 27.** Values of the nodes $\tau_j$ of the seventh layer.

| $\gamma_\ell = \mathrm{ReLU}(x'_\ell + \ell - 1)$ |
|---|
| $\gamma_1 = 1$ |
| $\gamma_2 = 5$ |
| $\gamma_2 = 6$ |

**TABLE 28.** Values of the nodes $\tau_i$ of the eighth layer.

| $\tau_i = \mathrm{ReLU}(\sum_{j=1}^{d}(\alpha^1_{i\ell} - \alpha^2_{i\ell} + \beta^1_{i\ell} - \beta^2_{i\ell}) - d)$ |
|---|
| $\tau_1 = 1$ |
| $\tau_2 = 0$ |
| $\tau_3 = 0$ |
| $\tau_4 = 2$ |
| $\tau_5 = 0$ |

$$\alpha^{22}_{\ell j} = \mathrm{ReLU}(x_j/\epsilon - (\ell-1)/\epsilon m),$$
$$\beta^{21}_{\ell j} = \mathrm{ReLU}((\ell-1)/\epsilon m - x_j/\epsilon + 1),$$
$$\beta^{22}_{\ell j} = \mathrm{ReLU}((\ell-1)/\epsilon m - x_j/\epsilon).$$

The third layer corresponds to Eqs. (37)-(38) with $5d$ and $5d$ nodes $\rho_j$ and $\psi_j$, respectively, as follows

$$\rho_j = \mathrm{ReLU}(\sum_{i=0}^{n+1}(\gamma^{11}_{ij} - \gamma^{12}_{ij} + \gamma^{21}_{ij} - \gamma^{22}_{ij} - \alpha^{11}_{ij} + \alpha^{12}_{ij} - \beta^{11}_{ij} + \beta^{12}_{ij}) \cdot i),$$

$$\psi_j = \mathrm{ReLU}(\sum_{\ell=1}^{m}(\omega^{11}_{\ell j} - \omega^{12}_{\ell j} + \omega^{21}_{\ell j} - \omega^{22}_{\ell j} - 1) \cdot \ell - \sum_{\ell=2}^{m}(\alpha^{21}_{\ell j} - \alpha^{22}_{\ell j} + \beta^{21}_{\ell j} - \beta^{22}_{\ell j} - 1) \cdot \ell).$$

The fourth layer keeps copies of $\rho_j$ and $\psi_j$, and computes $\delta(p'_j, 0)$ and $\delta(p'_j, p'_k)$ with $4(5d)$ and $4(9d^2)$ nodes, respectively. These nodes are:

$$\mu^{11}_{0j} = \mathrm{ReLU}(\rho_j/\epsilon + 1),$$
$$\mu^{12}_{0j} = \mathrm{ReLU}(\rho_j/\epsilon),$$
$$\lambda^{11}_{0j} = \mathrm{ReLU}(-\rho_j/\epsilon + 1),$$
$$\lambda^{12}_{0j} = \mathrm{ReLU}(-\rho_j/\epsilon),$$
$$\mu^{21}_{kj} = \mathrm{ReLU}((\rho_j - \rho_k)/\epsilon + 1),$$
$$\mu^{22}_{kj} = \mathrm{ReLU}((\rho_j - \rho_k)/\epsilon),$$
$$\lambda^{21}_{kj} = \mathrm{ReLU}((-\rho_j + \rho_k)/\epsilon + 1),$$
$$\lambda^{22}_{kj} = \mathrm{ReLU}((-\rho_j + \rho_k)/\epsilon).$$

The fifth layer keeps copies of $\rho_j$ and $\psi_j$ nodes and computes $\tau_j$ with $5d$ nodes as follows

$$\tau_j = \begin{cases} \mathrm{ReLU}(j + 1 - \mu^{11}_{0j} + \mu^{12}_{0j} - \lambda^{11}_{0j} + \lambda^{12}_{0j} - \sum_{k=1}^{j-1}(\mu^{21}_{kj} - \mu^{22}_{kj} + \lambda^{21}_{kj} - \lambda^{22}_{kj})) & \text{if } 1 \le j \le d, \\ \mathrm{ReLU}(j - 2d + 1 - \mu^{11}_{0j} + \mu^{12}_{0j} - \lambda^{11}_{0j} + \lambda^{12}_{0j} - \sum_{k=2d+1}^{j-1}(\mu^{21}_{kj} - \mu^{22}_{kj} + \lambda^{21}_{kj} - \lambda^{22}_{kj})) & \text{if } 2d+1 \le j \le 3d, \\ \mathrm{ReLU}(2 - \mu^{11}_{0j} + \mu^{12}_{0j} - \lambda^{11}_{0j} + \lambda^{12}_{0j}) & \text{if } 3d+1 \le j \le 4d. \end{cases}$$

The sixth layer keeps copies of $\rho_j$ and $\psi_j$ nodes and performs the computation of Eq. (40) with $10d$ nodes $\eta^1_j$ and $\eta^2_j$ as follows:

$$\eta^1_j = \mathrm{ReLU}(\sum_{k=1}^{j}\tau_k/\epsilon - (d+1)/\epsilon + 1),$$
$$\eta^2_j = \mathrm{ReLU}(\sum_{k=1}^{j}\tau_k/\epsilon - (d+1)/\epsilon).$$

The seventh layer keeps a copy of $\psi_j$ and performs the computation of Eq. (41) with $5d$ nodes $\zeta_j$ such that

$$\zeta_j = \mathrm{ReLU}(\rho_j - C \cdot (\eta^1_j - \eta^2_j)).$$

The eighth layer keeps copies of $\psi_j$ and $\zeta_j$. It also computes $\delta(v_j, 0)$ with $4(5d)$ nodes $\mu^{31}_{0j}$, $\mu^{32}_{0j}$, $\lambda^{31}_{0j}$ and $\lambda^{32}_{0j}$

$$\mu^{31}_{0j} = \mathrm{ReLU}(\zeta_j/\epsilon + 1), \quad \mu^{32}_{0j} = \mathrm{ReLU}(\zeta_j/\epsilon),$$
$$\lambda^{31}_{0j} = \mathrm{ReLU}(-\zeta_j/\epsilon + 1), \quad \lambda^{32}_{0j} = \mathrm{ReLU}(-\zeta_j/\epsilon).$$

**TABLE 29.** Values of the nodes $\mu_{i\ell}^{11}$, $\mu_{i\ell}^{12}$, $\lambda_{i\ell}^{11}$ and $\lambda_{i\ell}^{12}$ of the eighth layer.

| $\mu_{i\ell}^{11} = \text{ReLU}(\gamma_\ell - (i+\ell-1))/\epsilon + 1)$ | $\mu_{i\ell}^{12} = \text{ReLU}(\gamma_\ell - (i+\ell-1))/\epsilon)$ | $\lambda_{i\ell}^{11} = \text{ReLU}((i+\ell-1) - \gamma_\ell)/\epsilon + 1)$ | $\lambda_{i\ell}^{12} = \text{ReLU}((i+\ell-1) - \gamma_\ell)/\epsilon)$ |
|---|---|---|---|
| $\mu_{11}^{11} = 1$ | $\mu_{11}^{12} = 0$ | $\lambda_{11}^{11} = 1$ | $\lambda_{11}^{12} = 0$ |
| $\mu_{12}^{11} = 3/\epsilon + 1$ | $\mu_{12}^{12} = 3/\epsilon$ | $\lambda_{12}^{11} = 0$ | $\lambda_{12}^{12} = 0$ |
| $\mu_{13}^{11} = 3/\epsilon + 1$ | $\mu_{13}^{12} = 3/\epsilon$ | $\lambda_{13}^{11} = 0$ | $\lambda_{13}^{12} = 0$ |
| $\mu_{21}^{11} = 0$ | $\mu_{21}^{12} = 0$ | $\lambda_{21}^{11} = 1/\epsilon + 1$ | $\lambda_{21}^{12} = 1/\epsilon$ |
| $\mu_{22}^{11} = 2/\epsilon + 1$ | $\mu_{22}^{12} = 2/\epsilon$ | $\lambda_{22}^{11} = 0$ | $\lambda_{22}^{12} = 0$ |
| $\mu_{23}^{11} = 2/\epsilon + 1$ | $\mu_{23}^{12} = 2/\epsilon$ | $\lambda_{23}^{11} = 0$ | $\lambda_{23}^{12} = 0$ |
| $\mu_{31}^{11} = 0$ | $\mu_{31}^{12} = 0$ | $\lambda_{31}^{11} = 2/\epsilon + 1$ | $\lambda_{31}^{12} = 2/\epsilon$ |
| $\mu_{32}^{11} = 1/\epsilon + 1$ | $\mu_{32}^{12} = 1/\epsilon$ | $\lambda_{32}^{11} = 0$ | $\lambda_{32}^{12} = 0$ |
| $\mu_{33}^{11} = 1/\epsilon + 1$ | $\mu_{33}^{12} = 1/\epsilon$ | $\lambda_{33}^{11} = 0$ | $\lambda_{33}^{12} = 0$ |
| $\mu_{41}^{11} = 0$ | $\mu_{41}^{12} = 0$ | $\lambda_{41}^{11} = 3/\epsilon + 1$ | $\lambda_{41}^{12} = 3/\epsilon$ |
| $\mu_{42}^{11} = 1$ | $\mu_{42}^{12} = 0$ | $\lambda_{42}^{11} = 1$ | $\lambda_{42}^{12} = 0$ |
| $\mu_{43}^{11} = 1$ | $\mu_{43}^{12} = 0$ | $\lambda_{43}^{11} = 1$ | $\lambda_{43}^{12} = 0$ |
| $\mu_{51}^{11} = 0$ | $\mu_{51}^{12} = 0$ | $\lambda_{51}^{11} = 4/\epsilon + 1$ | $\lambda_{51}^{12} = 4/\epsilon$ |
| $\mu_{52}^{11} = 0$ | $\mu_{52}^{12} = 0$ | $\lambda_{52}^{11} = 1/\epsilon + 1$ | $\lambda_{52}^{12} = 1/\epsilon$ |
| $\mu_{53}^{11} = 0$ | $\mu_{53}^{12} = 0$ | $\lambda_{53}^{11} = 1/\epsilon + 1$ | $\lambda_{53}^{12} = 1/\epsilon$ |

**TABLE 30.** Values of the nodes $\tau_i'$ of the ninth layer.

| $\tau_i' = \text{ReLU}(i + \sum_{k=0}^{i-1} \tau_k)$ |
|---|
| $\tau_1' = 2$ |
| $\tau_2' = 3$ |
| $\tau_3' = 4$ |
| $\tau_4' = 7$ |

The ninth layer keeps a copy of $\psi_j$ and computes $w_j$ corresponding to Eq. (42) with $4(5d)$ nodes $\varrho_j$ such that

$$\varrho_j = \begin{cases} \zeta_j & \text{if } 1 \le j \le d, 2d+1 \le j \le 3d, \\ \text{ReLU}(\zeta_j + (n+1)(\mu_{0j}^{31} - \mu_{0j}^{32} + \lambda_{0j}^{31} - \lambda_{0j}^{32} - 1)) & \\ \quad \text{if } 3d+1 \le j \le 4d. & \end{cases}$$

The tenth layer computes $x_j'$ with $5d$ nodes $\chi_j$ as follows:

$$\chi_j = \text{ReLU}(\varrho_j + \psi_j).$$

Next apply substitution (resp., deletion, and insertion) on the nodes $\chi_j$, with $1 \le j \le 2d$ (resp., $2d+1 \le j \le 3d$, and $3d+1 \le j \le 5d$) using the networks defined in Theorems 2, 3 and 4 with a little alteration, and get $y'$. For this, the eleventh layer of this neural network is the second layer of substitution (resp., deletion, and insertion). The sixteenth layer of this neural network gives the output of substitution operation in the form of nodes $e_{Sh}$ for $h \in \{1, \ldots, n+d\}$, by using Theorems 2 whereas this layer copies the sixth layer of deletion operation from the fifteenth layer as an identity map and has the seventh layer of insertion operation. To construct the seventeenth layer, in the equation

$$\zeta_{ij} = \text{ReLU}\left(e_{i+j-1} - C(-\psi_{ij}^1 + \psi_{ij}^2 - \rho_{ij}^1 + \rho_{ij}^2 + 2)\right).$$

of the seventh layer of deletion, $e_{i+j-1} = e_{Si+j-1}$ is a node of the previous layer connected with $\zeta_{ij}$ with weight 1. In this case the bias of $\zeta_{ij}$ is only $-2C$. The eighteenth layer gives the output of deletion operation in the form of nodes $e_{Di}$ for $i \in \{1, \ldots, n\}$, by using Theorem 3 and also contains the ninth layer of insertion. The nineteenth layer of this neural network copies $e_{Di}$ nodes as an identity map and contains the nodes of the tenth layer of insertion. To construct the twentieth layer, in the equation

$$\omega_{i\ell}' = \text{ReLU}(C\mu_{i\ell}^{21} - C\mu_{i\ell}^{22} + C\lambda_{i\ell}^{21} - C\lambda_{i\ell}^{22} + e_i - 2C).$$

of the eleventh layer of insertion, $e_i = e_{Di}$ is a node of previous layer connected with $\omega_{i\ell}'$ with weight 1. In this case the only bias of $\omega_{i\ell}'$ is $-2C$. The twenty-second layer gives the output in the form of nodes $e_{Ih} = y_h'$ for $h \in \{1, \ldots, n+d\}$ after the insertion operation by using Theorem 4. Finally, get $y$ by applying step (vi) on $y'$. Thus, network has a constant depth and size $\mathcal{O}(\max(dn, dm))$, which completes the proof.□

*Example 5: Let* $\Sigma = \{1, 2, 3, 4, 5, 6, 7\}$, $e = (4, 1, 3, 7, 5)$, *and* $d = 3$. *Take* $\Delta = 0.01$ *and* $x = (0.03, 0, 0.03, 0.27, 0, 0.6, 0, 0.55, 0.55, 0, 0.25, 0.9, 0, 0.7, 0.24)$. *By using* $x$ *in Eqs. (35)- (42), we get the vectors*:

$p = [[0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],$
$[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0]],$

$q = [[0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],$
$[0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],$
$[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0],$
$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]],$

$p' = [1, 0, 1, 0, 0, 0, 0, 4, 4, 0, 2, 6, 0, 0, 0],$
$q' = [0, 0, 0, 2, 1, 5, 0, 0, 0, 0, 0, 0, 1, 5, 2],$
$t = [1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0],$
$u = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],$

**TABLE 31.** Values of the nodes $\omega_{i\ell}$ of the ninth layer.

| $\omega_{i1} = \text{ReLU}(C\mu_{i1}^{11} - C\mu_{i1}^{12} + C\lambda_{i1}^{11} - C\lambda_{i1}^{12} + x'_{d+1} - 2C)$ | $\omega_{i2} = \text{ReLU}(C\mu_{i2}^{11} - C\mu_{i2}^{12} + C\lambda_{i2}^{11} - C\lambda_{i2}^{12} + x'_{d+2} - 2C)$ | $\omega_{i3} = \text{ReLU}(C\mu_{i3}^{11} - C\mu_{i3}^{12} + C\lambda_{i3}^{11} - C\lambda_{i3}^{12} + x'_{d+3} - 2C)$ |
|---|---|---|
| $\omega_{11} = 2$ | $\omega_{12} = 0$ | $\omega_{13} = 0$ |
| $\omega_{21} = 0$ | $\omega_{22} = 0$ | $\omega_{23} = 0$ |
| $\omega_{31} = 0$ | $\omega_{32} = 0$ | $\omega_{33} = 0$ |
| $\omega_{41} = 0$ | $\omega_{42} = 3$ | $\omega_{43} = 5$ |
| $\omega_{51} = 0$ | $\omega_{52} = 0$ | $\omega_{53} = 0$ |

**TABLE 32.** Values of the nodes $\mu_{i\ell}^{21}$, $\mu_{i\ell}^{22}$, $\lambda_{i\ell}^{21}$ and $\lambda_{i\ell}^{22}$ of the tenth layer.

| $\mu_{i\ell}^{21} = \text{ReLU}((\tau'_i - (i+\ell-1))/\epsilon + 1)$ | $\mu_{i\ell}^{22} = \text{ReLU}((\tau'_i - (i+\ell-1))/\epsilon)$ | $\lambda_{i\ell}^{21} = \text{ReLU}(((i+\ell-1) - \tau'_i)/\epsilon + 1)$ | $\lambda_{i\ell}^{22} = \text{ReLU}(((i+\ell-1) - \tau'_i)/\epsilon)$ |
|---|---|---|---|
| $\mu_{11}^{21} = 1/\epsilon + 1$ | $\mu_{11}^{22} = 1/\epsilon$ | $\lambda_{11}^{21} = 0$ | $\lambda_{11}^{22} = 0$ |
| $\mu_{12}^{21} = 1$ | $\mu_{12}^{22} = 0$ | $\lambda_{12}^{21} = 1$ | $\lambda_{12}^{22} = 0$ |
| $\mu_{13}^{21} = 0$ | $\mu_{13}^{22} = 0$ | $\lambda_{13}^{21} = 1/\epsilon + 1$ | $\lambda_{13}^{22} = 1/\epsilon$ |
| $\mu_{14}^{21} = 0$ | $\mu_{14}^{22} = 0$ | $\lambda_{14}^{21} = 2/\epsilon + 1$ | $\lambda_{14}^{22} = 2/\epsilon$ |
| $\mu_{21}^{21} = 1/\epsilon + 1$ | $\mu_{21}^{22} = 1/\epsilon$ | $\lambda_{21}^{21} = 0$ | $\lambda_{21}^{22} = 0$ |
| $\mu_{22}^{21} = 1$ | $\mu_{22}^{22} = 0$ | $\lambda_{22}^{21} = 1$ | $\lambda_{22}^{22} = 0$ |
| $\mu_{23}^{21} = 0$ | $\mu_{23}^{22} = 0$ | $\lambda_{23}^{21} = 1/\epsilon + 1$ | $\lambda_{23}^{22} = 1/\epsilon$ |
| $\mu_{24}^{21} = 0$ | $\mu_{24}^{22} = 0$ | $\lambda_{24}^{21} = 2/\epsilon + 1$ | $\lambda_{24}^{22} = 2/\epsilon$ |
| $\mu_{31}^{21} = 1/\epsilon + 1$ | $\mu_{31}^{22} = 1/\epsilon$ | $\lambda_{31}^{21} = 0$ | $\lambda_{31}^{22} = 0$ |
| $\mu_{32}^{21} = 1$ | $\mu_{32}^{22} = 0$ | $\lambda_{32}^{21} = 1$ | $\lambda_{32}^{22} = 0$ |
| $\mu_{33}^{21} = 0$ | $\mu_{33}^{22} = 0$ | $\lambda_{33}^{21} = 1/\epsilon + 1$ | $\lambda_{33}^{22} = 1/\epsilon$ |
| $\mu_{34}^{21} = 0$ | $\mu_{34}^{22} = 0$ | $\lambda_{34}^{21} = 2/\epsilon + 1$ | $\lambda_{34}^{22} = 2/\epsilon$ |
| $\mu_{41}^{21} = 3/\epsilon + 1$ | $\mu_{41}^{22} = 3/\epsilon$ | $\lambda_{41}^{21} = 0$ | $\lambda_{41}^{22} = 0$ |
| $\mu_{42}^{21} = 2/\epsilon + 1$ | $\mu_{42}^{22} = 2/\epsilon$ | $\lambda_{42}^{21} = 0$ | $\lambda_{42}^{22} = 0$ |
| $\mu_{43}^{21} = 1/\epsilon + 1$ | $\mu_{43}^{22} = 1/\epsilon$ | $\lambda_{43}^{21} = 0$ | $\lambda_{43}^{22} = 0$ |
| $\mu_{44}^{21} = 1$ | $\mu_{44}^{22} = 0$ | $\lambda_{44}^{21} = 1$ | $\lambda_{44}^{22} = 0$ |

**TABLE 33.** Values of the nodes $\zeta_h$ of the tenth layer.

| $\zeta_h = \text{ReLU}(\sum_{\forall i,\ell \mid i+\ell-1=h} \omega_{i\ell})$ |
|---|
| $\zeta_1 = 2$ |
| $\zeta_2 = 0$ |
| $\zeta_3 = 0$ |
| $\zeta_4 = 0$ |
| $\zeta_5 = 3$ |
| $\zeta_6 = 5$ |
| $\zeta_7 = 0$ |

$$v = [1, 0, 1, 0, 0, 0, 0, 4, 4, 0, 2, 0, 0, 0, 0],$$
$$w = [1, 0, 1, 0, 0, 0, 0, 4, 4, 6, 2, 6, 0, 0, 0],$$
$$x' = (1, 0, 1, 2, 1, 5, 0, 4, 4, 6, 2, 6, 1, 5, 2).$$

*By applying substitution, deletion, and insertion operations on* $e = (4, 1, 3, 7, 5, 0, 0, 0)$ *using* $x'$ *we get* $(2, 1, 3, 7, 5, 0, 0, 0)$*;* $(2, 1, 3, 5, 0)$*; and* $y' = (2, 5, 1, 3, 5, 0, 0, 0)$*, respectively. Finally, by selecting first* $n - n_D + n_I = 5 - 1 + 1 = 5$ *we get* $y = (2, 5, 1, 3, 5)$.

### EXAMPLES OF CODE EXECUTION

All codes are freely available at https://github.com/MGANN-KU/ReLU_Networks. An explanation of the program codes is given below.

The file `Binary_Hamming_distance_substitution.py` contains an implementation of binary $H_d$-generative ReLU to generate strings with a given Hamming distance.

**Input:**
`e`:= Input string of length `n`
`d`:= Hamming distance
`x`:= The binary conversion string of length `n`
**Output:**
`y`:= The string obtained by applying substitution operation on `e` following `x` and has at most distance `d`.

**An example:** `e = (1,1,0), d = 2, x = (1, 3), y = (0, 1, 1).`

The file `Hamming_distance_substitution.py` contains an implementation of $H_d$-generative ReLU to generate strings with a given Hamming distance.

**Input:**
`e`:= Input string of length `n`
`d`:= Hamming distance
`m`:= The size of the symbol set
`x`:= The conversion string of length `2n`
**Output:**
`y`:= The string obtained by applying substitution operation on `e` following `x` and has at most distance `d`.

**An example:** `e = (4,1, 3, 2), d = 3, m = 5, x = (2, 4, 2, 5, 1, 3), y = (4, 5, 3, 1).` The file `Edit_distance_deletion.py` contains an implementation of R $ED_d$-generative ReLU to generate strings with a given edit distance due to deletion operation only.

**Input:**
`e`:= Input string of length `n`

**TABLE 34.** Values of the nodes $\omega'_{i\ell}$ of the eleventh layer.

| $\omega'_{i1} = \text{ReLU}(C\mu_{i1}^{21} - C\mu_{i1}^{22} + C\lambda_{i1}^{21} - C\lambda_{i1}^{22} + e_i - 2C)$ | $\omega'_{i2} = \text{ReLU}(C\mu_{i2}^{21} - C\mu_{i2}^{22} + C\lambda_{i2}^{21} - C\lambda_{i2}^{22} + e_i - 2C)$ | $\omega'_{i3} = \text{ReLU}(C\mu_{i3}^{21} - C\mu_{i3}^{22} + C\lambda_{i3}^{21} - C\lambda_{i3}^{22} + e_i - 2C)$ | $\omega'_{i4} = \text{ReLU}(C\mu_{i4}^{21} - C\mu_{i4}^{22} + C\lambda_{i4}^{21} - C\lambda_{i4}^{22} + e_i - 2C)$ |
|---|---|---|---|
| $\omega'_{11} = 0$ | $\omega'_{11} = 3$ | $\omega'_{11} = 0$ | $\omega'_{11} = 0$ |
| $\omega'_{21} = 0$ | $\omega'_{21} = 2$ | $\omega'_{21} = 0$ | $\omega'_{21} = 0$ |
| $\omega'_{31} = 0$ | $\omega'_{31} = 4$ | $\omega'_{31} = 0$ | $\omega'_{31} = 0$ |
| $\omega'_{41} = 0$ | $\omega'_{41} = 0$ | $\omega'_{41} = 0$ | $\omega'_{41} = 1$ |

**TABLE 35.** Values of the nodes $\zeta'_h$ of the seventh layer.

| $\zeta'_h = \text{ReLU}(\sum_{\forall i,\ell \mid i+\ell-1=h}(\omega'_{i\ell})$ |
|---|
| $\zeta'_1 = 0$ |
| $\zeta'_2 = 3$ |
| $\zeta'_3 = 2$ |
| $\zeta'_4 = 4$ |
| $\zeta'_5 = 0$ |
| $\zeta'_6 = 0$ |
| $\zeta'_7 = 1$ |

**TABLE 36.** Values of the nodes $y_h$ of the output layer.

| $y_h = \zeta'_h + \zeta_h$ |
|---|
| $y_1 = 2$ |
| $y_2 = 3$ |
| $y_3 = 2$ |
| $y_4 = 4$ |
| $y_5 = 3$ |
| $y_6 = 5$ |
| $y_6 = 1$ |

d:= Hamming distance
x:= The conversion string of length n
**Output:**
y:= The string obtained by applying deletion operation on e following x and has at most distance d.

**An example:** `e = (3, 2, 4, 1) d = 2, x = (1, 3), y = (2, 1).`

The file `Edit_distance_insertion.py` contains an implementation of $EI_d$-generative ReLU to generate strings with a given edit distance due to insertion operation only.

**Input:**
e:= Input string of length n
d:= Hamming distance
m:= The size of the symbol set
x:= The conversion string of length n
**Output:**
y:= The string obtained by applying insertion operation on e following x and has at most distance d.

**An example:** `e = (3, 2, 4, 1) d = 3, m = 5, x = (4, 1, 4, 3, 2, 5), y = (2, 3, 2, 4, 3, 5, 1).`

The file `Edit_distance_unified.py` contains an implementation of $E_d$-generative ReLU to generate strings with a given edit distance due to substitution, deletion and insertion operations simultaneously.

**Input:**
e:= Input string of length n
d:= Hamming distance
m:= The size of the symbol set
Δ:= The small number
x:= The conversion string of length 2n
**Output:**
y:= The string obtained by applying substitution, deletion, and insertion operations simultaneously on e following x and has at most distance d.

**An example:** e:= (4, 1, 3, 7, 5) d:= 3, m:= 7, Δ:= 0.01 x:= (0.03, 0, 0.03, 0.27, 0, 0.6, 0, 0.55, 0.55, 0, 0.25, 0.9, 0, 0.7, 0.24), y:= (2, 5, 1, 3, 5).

## ACKNOWLEDGMENT

The authors would like to thank Dr. Naveed Ahmed Azam, Quaid-i-Azam University, for the useful technical discussions.

## REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, 2006, pp. 1–4.

[3] D. P Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[4] S. Thrun, *Exploring Artificial Intelligence in the New Millenium, Chapter Robotic Mapping: A Survey*. San Mateo, CA, USA: Morgan Kaufmann, 2005, pp. 1–35.

[5] B. A. Kindhi, M. A. Hendrawan, D. Purwitasari, T. A. Sardjono, and M. H. Purnomo, "Distance-based pattern matching of DNA sequences for evaluating primary mutation," in *Proc. 2nd Int. Conf. Inf. Technol., Inf. Syst. Electr. Eng. (ICITISEE)*, Nov. 2017, pp. 310–314.

[6] T. Buschmann and L. V. Bystrykh, "Levenshtein error-correcting barcodes for multiplexed DNA sequencing," *BMC Bioinf.*, vol. 14, no. 1, pp. 1–10, Dec. 2013.

[7] R. Cotterell and J. Eisner, "Probabilistic typology: Deep generative models of vowel inventories," 2017, *arXiv:1705.01684*.

[8] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[9] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.

[10] D. Klein and D. M. Christopher, "Fast exact inference with a factored model for natural language parsing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 1–15.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[12] H. P. Pinheiro, A. de Souza Pinheiro, and P. K. Sen, "Comparison of genomic sequences using the Hamming distance," *J. Stat. Planning Inference*, vol. 130, nos. 1–2, pp. 325–339, Mar. 2005.
[13] S. Kumano and T. Akutsu, "Comparison of the representational power of random forests, binary decision diagrams, and neural networks," *Neural Comput.*, vol. 34, no. 4, pp. 1019–1044, 2022.
[14] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.
[15] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.
[16] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein, "On the expressive power of deep neural networks," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, Aug. 2017, pp. 2847–2854.
[17] M. Telgarsky, "Representation benefits of deep feedforward networks," 2015, *arXiv:1509.08101*.
[18] L. Szymanski and B. McCane, "Deep networks are effective encoders of periodicity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 10, pp. 1816–1827, Oct. 2014.
[19] V. Chatziafratis, S. G. Nagarajan, I. Panageas, and X. Wang, "Depth-width trade-offs for ReLU networks via Sharkovsky's theorem," 2019, *arXiv:1912.04378*.
[20] B. Hanin and D. Rolnick, "Complexity of linear regions in deep networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 2596–2604.
[21] Y. Bengio, O. Delalleau, and C. Simard, "Decision trees do not generalize to new variations," *Comput. Intell.*, vol. 26, no. 4, pp. 449–467, Nov. 2010.
[22] G. Biau, E. Scornet, and J. Welbl, "Neural random forests," *Sankhya A*, vol. 81, no. 2, pp. 347–386, Dec. 2019.
[23] C. Wang, W.-H. Kao, and C. K. Hsiao, "Using Hamming distance as information for SNP-sets clustering and testing in disease association studies," *PLoS ONE*, vol. 10, no. 8, Aug. 2015, Art. no. e0135918.
[24] M. Mohammadi-Kambs, K. Hölz, M. M. Somoza, and A. Ott, "Hamming distance as a concept in DNA molecular recognition," *ACS Omega*, vol. 2, no. 4, pp. 1302–1308, Apr. 2017.
[25] M. Hasan, A. S. M. Miah, M. M. Hossain, and M. S. Hossain, "LL-PMS8: A time efficient approach to solve planted motif search problem," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3843–3850, Jun. 2022.

**MAMOONA GHAFOOR** received the M.Sc. and M.Phil. degrees in mathematics from Quaid-i-Azam University Islamabad, Pakistan, in 2011 and 2013, respectively. She is currently pursuing the Ph.D. degree with the Bioinformatics Center, Institute for Chemical Research, Kyoto University. Her research interests include computational mathematics and machine learning algorithms.

**TATSUYA AKUTSU** (Senior Member, IEEE) received the B.E. and M.E. degrees in aeronautics and the D.E. degree in information engineering from The University of Tokyo, in 1984, 1986, and 1989, respectively. Since 2001, he has been a Professor with the Bioinformatics Center, Institute for Chemical Research, Kyoto University. His research interests include bioinformatics, complex networks, and discrete algorithms.