Abstract—The fused lasso signal approximator (FLSA) is a vital optimization problem with extensive applications in signal processing and biomedical engineering. However, the optimization problem is difficult to solve since it is both non-smooth and nonseparable. The existing numerical solutions implicate the use of several auxiliary variables in order to deal with the nondifferentiable penalty. Thus, the resulting algorithms are both time- and memory-inefficient. This paper proposes a compact neural network to solve the FLSA. The neural network has a one-layer structure with the number of neurons proportionate to the dimension of the given signal, thanks to the utilization of consecutive projections. The proposed neural network is stable in the Lyapunov sense and is guaranteed to converge globally to the optimal solution of the FLSA. Experiments on several applications from signal processing and biomedical engineering confirm the reasonable performance of the proposed neural network.

Index Terms—Fused lasso, global convergence, Lyapunov, neural network.

I. INTRODUCTION

THE FUSED lasso signal approximator (FLSA) has diverse applications in several disciplines. The FLSA enforces sparsity both in the coefficients and the differences between consecutive elements in the coefficient vector. A class of the corresponding optimization problem is

$$\min_{x} \frac{1}{2} \|x-y\|_2^2 + M_{i=1}^{N+1} \|x_i - x_{i+1}\|_1$$

where $k_1, k_2 > 0$ are the regularization parameters, $y \in \mathbb{R}^n$ is the observed signal, and $x \in \mathbb{R}^n$ is the noise-free approximation of $y$. If $x_1 = 0$, FLSA has the closed-form solution by applying the steepest-descending operator [1], and for $x_1 = 0$, FLSA boils down to the total variation denoising problem [2]. As a result, the solution to this minimization yields the solution to the total variation denoising as well.

Neural networks have been long used for finding the optimal solution to optimization problems since the Hopfield’s pioneering works on solving the combinatorial traveling salesman problem [3] and linear programming [4]. Ever since, problems with respect to manifold types of constraints [5]-[14]. Solving optimization problems using neural networks have several salient advantages over traditional numerical methods in real-time processing. First and foremost, the structure of the neural network can be effectively implemented using very-large-scale integration and optical technologies [15]. Thus, when there is a demand for real-time optimization, neural networks offer the practical solution. In addition, ordinary differential equations (ODEs) representing a recurrent neural network can be solved by using different methods so they can be implemented on digital computers as well. Another advantage of neural networks is that they can converge globally to the exact optimal solution of the given minimization regardless of their initial values.

Zhang and Constantinides [6] presented a Lagrangian neural network for solving general nonlinear programming. Bouzerdoum and Pattison [5] developed a neural network for quadratic programming with box constraints only by writing the partial dual of the given minimization. Quadratic programming has been considered in numerous research studies and, thus, many networks exist to solve a particular case of quadratic minimization. One case is the strictly convex quadratic programming for which many neural solutions have been tailored [9]-[11], [16]-[18]. Another neural network is for the case when the second derivative of the objective function equals the identity matrix [19], and it has been shown that the neural network has a simpler structure when other neural networks are applied to the same minimization. Several neural solutions are also presented to solve general quadratic programming [20]-[22].

Aside from neural networks for general optimization problems, several well-known engineering problems have also been solved by using projection neural networks. For instance, there are several neural networks for support vector machine training [23], [24]. There are also several neural solutions for regression analysis [25], [26], robot control [27], image restoration [28], image fusion [29], and non-negative matrix factorization [30]. One of the essential problems which has been considered recently is the $l_1$ minimization, for which several neural networks have been presented [31]-[36]. The optimization problem has a similar challenge to FLSA since it implicates the minimization of the $l_1$-norm. However, the FLSA is more complicated as it entails two nonsmooth terms, one of which minimizes the difference among consecutive elements in the coefficient vector.
As FLSA has two nondifferentiable terms, finding its optimal solution is not straightforward. Techniques in the literature usually need to employ several auxiliary variables in order to deal with the nonsmoothness and nonseparability of the problem [37]-[40]. As a result, the resulting algorithms have higher time and memory complexity. In contrast to those solvers, the Condat’s method is a noniterative method which solves FLSA for $A_1 = 0$ [41]. The complexity of algorithm on the most real problems is $O(n)$ and its worst case is $O(n^2)$. The method, though very fast, is sequential by nature since the optimal value of $x_1$ is reliant on the optimal value $x_{n-1}$. As a result, the algorithm is not prone to parallel execution.

In this paper, a projection neural network is presented to solve FLSA. The proposed neural solution has a simple one-layer structure and is efficient in terms of both the component required for its circuit implementation and the number of operations in each iteration. The neural model deals with the two nonsmooth terms in FLSA by using two consecutive projections, which results in a network with $n - 1$ neurons, where $n$ is the length of the given data. As a result, the proposed network is compact since the dimension of the network is linear with respect to the dimension of the given data. The neural model is guaranteed to be stable in the sense of Lyapunov, and its trajectory is assured to converge to the optimal solution of FLSA given any arbitrary initial point. The neural solution is further applied to several problems in signal processing and biomedical engineering and the results are compared to the state-of-the-art solvers. In contrast to the Condat’s method, the proposed network can also be applied to other problems, such as weighted total variation minimization [42] and trend filtering [43]. The complexity of the solving is of $O(n)$ for a finite number of iterations.

In summary, the contributions of this paper are as follows.

1) A neural model is tailored for FLSA, which is, to the best of my knowledge, the first attempt to solve this problem using neurodynamic optimization. The neural model has a straightforward structure as it uses two successive projections.

2) The network is proved to be stable in the sense of Lyapunov, and it is assured that it converges globally to the optimal solution of FLSA.

3) The proposed model is simplified in the case where $A_1 = 0$ in FLSA. In this case, the neural network boils down to a well-known projection neural network, where its exponential convergence could also be guaranteed.

The remainder of this paper is structured as follows. The neural network model is developed in Section II, and its stability is assured in Section III. In Section IV, the proposed model is simplified for circumstances when $A_1 = 0$ in FLSA, and it is shown that the proposed neural network boils down to a well-known projection model with guaranteed exponential convergence. The experiments regarding the proposed neural network are presented in Section VI. Finally, this paper is concluded in Section VII.

II. NEURODYNAMIC MODEL

In this section, a neurodynamic model is developed for FLSA. To this end, FLSA can be equivalently rewritten as

$$\min_{x} \frac{1}{2} x^T \lambda - y^T \lambda + A^T \lambda + k \lambda D x \lambda$$  \hspace{1cm} (FLSA2)

where $\lambda_i$ is the $i$-norm and $D = R^{n-1 \times n}$ is defined as

$$D =\begin{bmatrix}
1 & 0 & \cdots & 0 \\
0 & 1 & \cdots & 0 \\
0 & 0 & \cdots & 0
\end{bmatrix}$$

Since FLSA2 is convex, the Karush-Kuhn-Tucker (KKT) conditions [44] are necessary and sufficient for the optimality. On the other hand, it is required to use the property of the subgradients for FLSA2 due to its nonsmoothness. Thus, $x^*$ is the optimal solution of FLSA2 if there are $w^*$ and $z^*$ such that $x = y + w + D^T z = 0$, where $w = A_1^T (|x|_1)$ and $z = A_2^T (|D x|_1)$. The derivative of $|x|_1$ is not defined at zero, but the derivative is simply one if $x_j > 0$, and 0 otherwise. Thus, one can write

$$M \begin{cases}
e -A_1 & \text{if } x_j = 0 \\
0 & \text{if } x_j < 0.
\end{cases}$$  \hspace{1cm} (1)

The conditions in (1) can be restated using the well-known projection operator as [45]

$$w = P x (w + x)$$  \hspace{1cm} (2)

where $P(x) = [P^x (a_1), P^x (a_2), \ldots, P^x (a_n)]$, and

$$P (a_j) = \begin{cases}
a_1 & \text{if } a_j > A_1 \\
0 & \text{if } |a_j| < A_1 \\
-A_1 & \text{if } a_j < -A_1.
\end{cases}$$

Similarly, one obtains the same projection equation for $z$ as

$$z = P z (z + D x).$$

As a result, the KKT conditions can be summarized as

$$x = y + w + D^T z = 0$$

$$w = P x (w + x)$$

$$z = P z (z + D x).$$

From the first equation in (3), it follows that $x = y - w - D^T z$. Replacing it in the second equation, one arrives at $x = y - D^T z - P A_2 (y - D^T z)$. As a result, the conditions in (3) can be rewritten simply as

$$z = P z (z - D P A_2 (y - D^T z) - Dy + DD^T z).$$  \hspace{1cm} (4)

Based on (4), a neural network is proposed with the dynamic equation as being given by

$$\frac{dz}{dt} = P z (z - h (z)) - z$$  \hspace{1cm} (5)

output Equation

$$x = y - D^T z - P A_2 (y - D^T z)$$  \hspace{1cm} (6)

where $h (z) = DP A_2 (y - D^T z) - Dy + DD^T z$. The dynamic system given in (5) can be easily recognized by a simple one-layer neural network, as shown in Fig. 1 for a general matrix $D$. The number of neurons for FLSA is $k = n - 1$, hence
the structure of the network grows linearly with respect to the dimension of the given data. According to Fig. 1 with $D^{\text{nn}}$, the circuit realizing the proposed neural network consists of $k$ integrators, $n + k$ piecewise activation functions, $2n + 2k$ summers, $2nk$ multipliers, and some weights.

### III. Stability Analysis

This section contains the stability analysis of the proposed neural network. First, some required definitions and lemmas are presented.

**Definition 1:** $x^*$ is the equilibrium point of the dynamic system

$$X = f(x), \quad x(t_0) = x_0 \in R^n$$  \hspace{1cm} (7)

where $f : R^n \rightarrow R^n$ is a function, if $f(x^*) = 0$.

**Definition 2 (Lyapunov Stability):** Let $x(t)$ be a solution of the dynamic system in (7). An equilibrium point $x^*$ is stable if, for $t > t_0$

$$V(x(t) - x^*) < V(x(0) - x^*)$$

The dynamic system is asymptotically stable if $x(t) \rightarrow x^*$ as $t \rightarrow \infty$ holds as well.

**Definition 3 (Monotonicity):** A mapping $H$ is called monotone at $y \in Q$ if

$$(x - y)^T (H(x) - H(y)) > 0 \quad Vx \in Q.$$  

If the above inequality holds for any $x, y \in Q$, then the mapping $H$ is said to be monotone in $Q$.

**Definition 4 (Variational Inequality):** The variational inequality problem is about finding $x^* \in Q$ such that

$$(x - x^*)^T H(x) > 0 \quad Vx \in Q$$

where $H$ is a function from $R^n$ into itself. It was shown that $x^*$ is the solution of the above inequality if and only if the following projection equation holds [46]:

$$x = P_Q(x - H(x))$$

where $P_Q(.)$ is the projection operator defined as

$$P_Q(z) = \arg \min \{||z - y|| \mid y \in Q\}.$$

**Lemma 1 (Projection Properties [46]):** Let $Q \subset R^n$ be a closed convex set, then:

1) $(a - P_Q(a))^T (P_Q(a) - b) > 0, \quad a \in R^n, \quad b \in Q$;
2) $(P_Q(a) - P_Q(b))^T (a - b) < ||a - b||^2, \quad a, b \in R^n$.

The stability and the convergence of the proposed neural network in (5) is now examined. Without loss of generality, we assume that $D \in R^{(n-1) \times n}$. Before stating the main results, it is first shown that the function $h(z)$ in (6) is monotone.

**Lemma 2:** The mapping $h(z) = D(\phi_k (y - D^T z) - y + D^T z)$ is monotone in $R^{n-1}$.

**Proof:** For any $u, v \in R^{n-1}$, one obtains

$$(h(u) - h(v))^T (u - v) = (Ph (y - D'u) - y + D'u - P_Q (y - D^T v) + y - D^T v) \in D(D^T u - D^T v)$$

$$+ \|D^T u - D^T v\|^2.$$  \hspace{1cm} (8)

Let $a = y - D^T u$ and $b = y - D^T v$ in Lemma 1-2, then ($P_Q$)

$$< ||D^T u - D^T v||^2$$

$$\|^T u - D^T v\|^2 + (P_Q (y - D^T u) - P_Q (y - D^T v))^T$$

$$x (D^T u - D^T v) > 0.$$  \hspace{1cm} (9)

Considering (8) and (9), it follows that:

$$(h(u) - h(v))^T (u - v) > 0$$

and $h$ is thus monotone.

Now, consider the function $H$ with the definition

$$H(z) = 1y - D^T z - P_d (y - D^T z)f.$$  \hspace{1cm} (10)

The following two lemmas are about the features of this function.

**Lemma 3 [47]:** The derivative of the function $H(z)$ given in (10) is $h(z)$, for example, $VH(z) = h(z)$.

**Lemma 4:** The function $H$ given above is convex.

**Proof:** The convexity of $H(z)$ can be simply proved using its first derivative. In particular, one obtains

$$(z_1 - z_2)^T (V H(z_1) - VH(z_2)) = (z_1 - z_2)^T (h(z_1) - h(z_2)) > 0$$
where the last inequality holds true since $h$ is monotone. Thus, $H$ is convex.

**Lemma 5**: There exists a unique continuous solution trajectory $z(t)$ for the dynamic system (5) with any given initial point for $t \in [t_0, T]$.

**Proof**: It is first required to show that the right-hand side of the dynamic system is Lipschitz. To do so, let $z_1, z_2 \in \mathbb{R}^{n+1}$ be two arbitrary variables, then

$$
\begin{align*}
&\|z_1 - Ph (Z1 - h(Z1)) - Z2 + Ph (Z2 - h(Z2))\| \\
&< 2\|z_1 - z_2\| + \|h(z_1) - h(z_2)\| \\
&< \|z_1 - z_2\| + \|h(z_1) - h(z_2)\| \\
&< \|z_1 - z_2\| + \|D\| \|y - D^T z_1\| - y + D^T z_1 + P_{xi} \|y - D^T z_2\|.
\end{align*}
$$

(11)

Now, consider the following inequality for any arbitrary $x, y,$ and $PQ$:

$$
\begin{align*}
&\|x - PQ(X) - y + PQO\| \leq \|x - y\| + \|PQ(X) - Pa(y)\| + \\
&- 2\|PQ(X) - Pn(y)\| (x - y) \\
&< (1) \|x - y\| - \|PQ(X) - Pn(y)\| \leq \|x - y\| \\
&< \|x - y\|.
\end{align*}
$$

(12)

where (1) is obtained by using Lemma 1-2. Let $x = b - D^T z_1$ and $y = b - D^T z_2$ in (12), then one can rewrite (11) as

$$
\begin{align*}
&\|z_1 - P_{xi} (z_1 - h(z_1)) - z_2 + P_{xi} (z_2 - h(z_2))\| \\
&< 2\|z_1 - z_2\| + \|D\| \|z_1 - z_2\| \\
&< \|z_1 - z_2\| + \|D\| \|z_1 - z_2\|.
\end{align*}
$$

Therefore, the right-hand side of the dynamic equation is Lipschitz. According to the Peano’s theorem for ODEs [48], a unique continuous solution $z(t)$ exists for the dynamic system in (5) in the interval $[t_0, T]$.

The stability of the dynamic system in (5) is now explored.

**Theorem 1**: The neural network governed by the dynamic equation in (5) is stable in the sense of Lyapunov and converges globally to a unique equilibrium $z^*$. The unique solution to FLSA is then obtained by the output equation in (6).

**Proof**: With any arbitrary initialization $z_0$, the trajectory of the dynamic system in (5) has a unique continuous solution trajectory, thanks to Lemma 5. Consider the following Lyapunov function [19], [49]:

$$
V(z) = H(z) - H(z^*) + h(z^*)^T (z^* - z) + 1 \|z - z^*\|^2
$$

where $z^*$ is the equilibrium of the dynamic system and $u \in \mathbb{R}^{n+1}$. Since $H$ is convex, it simply follows that [46]:

$$
H(z) - H(z^*) + h(z^*) (z^* - z) > 0.
$$

Thus, $V(z) > \|z - z^*\|^2/2$, which means that $V(z) \rightarrow 0$ as $\|z\| \rightarrow 0$ [49]. The time derivative of the Lyapunov function is then obtained as [19], [49]:

$$
\frac{dV}{dt} (z) = h(z) - h(z^*) + z - z^* \frac{dL}{dz}.
$$

(13)

Now, let $a = z - h(z)$ and $b = z^*$ in Lemma 1-1), one obtains [19], [49]

$$
\{P_{xi} (z - h(z)) - z^*\}^T (z - h(z) - P_{xi} (z - h(z))) > 0.
$$

(14)

Further, using Definition 4 with $x = P_{xi} (z - h(z))$ results in

$$
\{P_{xi} (z - h(z)) - z^*\}^T h(z^*) > 0, \quad V(z) \rightarrow 0.
$$

(15)

Adding the inequalities in (14) and (15), we obtain

$$
\{P_{xi} (z - h(z)) - z^* + z - z^*\}^T (z - h(z) + (z^*)) > 0.
$$

It follows [49]:

$$
\{z - z^* + h(z) - (h(z^*))^T (P_{xi} (z - h(z)) - z)
$$

$$
< 0. \quad (16)
$$

where the last inequality is obtained as $(z - z^*)^T (h(z) - h(z^*)) > 0$ due to the monotonicity of $h(z)$, and $\|z\|^2$ is evidently non-negative. Plugging (16) into (13), it follows:

$$
dV(z) < 0. \quad (17)
$$

and the dynamic system in (5) is thus stable in the sense of $<0$ Lyapunov.

According to the invariant set theorem [50], all trajectory-ries of the proposed neural network converge to the largest invariant set $n$, where $dV(z)/dt = 0$. Now, it is shown that $dV(z)/dt = 0$ if and only if $dz/dt = 0$. If $dz/dt = 0$, then

$$
dV(z) = 0 \Rightarrow dz/dt = 0.
$$

Conversely, $dV(z)/dt = 0$ implies $dz/dt = 0$.

Finally, the above inequalities show that $z(t)$ is bounded. The boundedness of trajectories of the dynamic system in (5) means that a subsequence $\{z(t_k)\}$ exists such that

$$
limit_{k \rightarrow \infty} z(t_k) = z
$$

where $z$ is an equilibrium of (5). Consider the following Lyapunov function:

$$
V(z) = H(z) - H(z) + (z - z^* + z - z^*)^T (z - z^*) + 1 \|z - z^*\|^2
$$

Then, $limit_{k \rightarrow \infty} z(t_k) = V(u) = 0$. Hence, there exists $\epsilon > 0$ for any $e > 0$ such that $\epsilon(t(t)) < V(z(t_k)) < \epsilon$.

Therefore, $\limsup_{z(t) \rightarrow z} -2 \lim_{t \rightarrow \infty} V(z) = z$. As a result, the dynamic system in (5) is globally convergent to one of its equilibrium points. According to Lemma 5, the state trajectory of (5) is unique, thus it converges to a unique equilibrium point $z^*$, where the unique solution to FLSA is obtained by $x^* = y - D^2 z^* - P_{xi} (y - D^2 z^*)$. ■
IV. SIMPLIFICATION FOROTHER PROBLEMS

This section presents the simplification of the proposed neural network in order to be applied to two well-known problems:

1) total variation denoising and 2) trend filtering.

If \( X_1 = 0 \), FLSA can be restated as

\[
\min_{x} \|x - y\|_\infty + \frac{k}{2} \|Dx\|_1
\]

which is the total variation denoising problem [2]. The KKT conditions for this minimization can be written as

\[
\begin{align*}
    x - y + D^T z &= 0 \\
    z &= Px_1 (z + Dx)
\end{align*}
\]

Thus, the neural model can be simply rewritten as

\[
\begin{align*}
    d_i &= Px_2 (z - h(z)) - z \\
    x &= y - D^T z
\end{align*}
\]

where \( h(z) = DD^T z - Dy \). In comparison to the neural network for FLSA, the neural network in (18) does not have the projection to \( X_i \), since it is zero. The neural model in (18) is a linear projection equation, which has been extensively studied in different research [11], [49], [51], [52]. Since \( DD^T \) is positive definite, it has been investigated that the trajectory of this neural network is exponentially convergent in finite time to the unique equilibrium [51].

The approximation of the trend in a given time series is a fundamental problem that arises in a variety of disciplines. Trend filtering has been also modeled similar to (17), in which matrix \( D \) is replaced with \( D \in \mathbb{R}^{p(n-1)x1} \) defined as [43]

\[
\begin{bmatrix}
    1 & 2 & \cdots & 1 \\
    -1 & -1 & \cdots & 0 \\
    0 & 0 & \cdots & 0 \\
    0 & 0 & \cdots & 0
\end{bmatrix}
\]

Since the neural model in (18) does not impose any assumption on \( D \), it can be simply applied to this problem as well. On the other hand, matrix \( D \) is also full row rank, thus \( DD^T \) is positive definite, and the dynamic system in (18) is thus exponentially convergent in finite time.

Another important problem is the proposed model can solve the weighted total variation, that is, [42]

\[
\begin{align*}
    \min_{x} & \|x - y\|_\infty + \lambda \|X^\top 2 \|_1 + \lambda \|X_2 \cap a\lambda x - x_{i+1}\|_1 \\
    \text{s.t.} & \quad i = 1, \ldots, n
\end{align*}
\]

where \( a \in \mathbb{R}^{n-1} \), \( a_i > 0 \) penalizes the difference between the associated elements in the vector. One can define \( D = \text{diag}(a)D \), where \( \text{diag} \) is a diagonal matrix whose diagonal elements are \( a \), making the problem turn into FLSA2. Thus, the proposed neural network can also solve this problem.

V. COMPLEXITY OF THE NEURAL SOLUTION

For a general matrix \( D \in \mathbb{R}^{p \times x} \), the proposed neural network in (5) requires \( 2nk \) multiplications and \( 2nk + n + k \) additions/subtractions. However, the computations can be more economical in some special cases, such as FLSA or the total variation denoising (17).

For FLSA, the value of \( y - D^T z \) is

\[
\begin{align*}
    y_1 &= z_1 \\
    y_2 &= z_2 \\
    y_3 &= z_3 \\
    y &= D^T z
\end{align*}
\]

Having the value of \( X_1 \), the value of \( x = y - D^T z - Px_1 (y - D^T z) \) can be obtained by \( 3n+3 \) additions/subtractions (and no multiplication). The time complexity is also linear in \( n \) since we only need to iterate over the length of \( y - D^T z \). Similarly, \( Px_1 (z + Dx) - z \) can be computed in linear time with respect to \( n \). Overall, the time complexity of recurrent neural network is linear in \( n \) for a number of iterations. Algorithm 1 summarizes the procedure of the FLSA computations in each iteration. The algorithm can also be written by one loop, but we deliberately use two loops so that they can be executed in parallel. Each iteration in these loops can be executed independently with respect to other iterations of the loop. Therefore, the algorithm is ideal for parallel computing.

### Algorithm 1 FLSA Computations

**Input:** \( y \in \mathbb{R}^n \), \( z \in \mathbb{R}^{n-1} \), \( X_1 \), \( X_2 \)

**for iter=1:n do** if \( i == 1 \) then

\[
\text{temp} = y_1 - z_1
\]

end if
For the total variation denoising, the order is also linear in $n$, and the number of iterations is finite [11], [49], [51], [52]. Thus, the overall time complexity of the neural network is of $O(n)$ as well, similar to the Condat’s method. However, the difference is that the Condat’s method is of order $n^2$ in the worst case and not prone to parallelism. In the proposed neural network, aside from its possibility to be implement using circuits, the computations of $x$ and $P(z + Dx) — z$ can be conducted in parallel.

VI. EXPERIMENTS

This section presents the experiments regarding the proposed neural network. The neural network is implemented using the ODE solvers of MATLAB with the tolerance $10^{-5}$. The implementation of the proposed network is publicly available.\(^1\) First, we study how parameters in FLSA can be tuned, and then the convergence of the dynamic system in (5) was empirically investigated. Then, the proposed neural network was applied to several real-world problems, including array comparative genomic hybridization (aCGH or CGH array) data recovery and trend filtering. The performance of the neural solution was compared to those of the state-of-the-art methods in each application.

A. Parameter Tuning

There are two parameters in (FLSA) whose values can significantly impact the behavior and the outcome of the neural network. Consequently, it is vital to appropriately tune the parameters. There are several strategies for identifying the appropriate values of such parameters, including, but not limited to genetic algorithms [53], sequential minimization [54], L-curve method [55], and variational Bayes [56]. Another simple yet practical strategy is grid search where we build a model for every possible combination of the parameters $X_1$ and $A_2$, and then select the parameters with the best model fit gauged by some fitness function.

As there are two parameters in (FLSA), we need to apply a 2-D grid search, which is time consuming in general. However, following a related study [57], we restricted the choice of the parameters to the set $0.1, 0.3, 0.5, 0.7,$ and $0.9$. We further gauged the fitness of an estimated model, shown by $X$, as

\[
\|y — XI\|^2
\]

where $s$ is the number of nonzero elements in $X$. Finally, we selected the model with the maximum fitness value.

B. Convergence

The theoretical analysis showed that the neural network in (5) converges globally to the optimal solution of FLSA, regardless of the initial point. In this experiment, the convergence of the network is empirically explored by performing a recovery on aCGH data. In this regard, a sample from the Pollack et al. dataset [58] was randomly selected, and it was subjected to the recurrent neural network for recovery.

The sensitivity of the proposed model was investigated by initializing the network with distinct values. The network was initialized with a vector of one, zero, and a random vector, respectively, so that the convergence of the neural solution could be explored in practice. Fig. 2 plots the trajectory of each element of $z$ in the dynamic system in (5) against different time slots. The trajectory of each element is displayed by distinct colors so they can be distinguished in three plots. Based on this figure, the trajectory of the dynamic system is convergent to the same value, regardless of the initialization. This experiment corroborates the global convergence of the proposed neural network, which was theoretically assured in Section III.

C. aCGH Data Analysis

aCGH or CGH array is arguably the first important application of FLSA [59], [60]. The aCGH data helps the diagnosis and prognosis of various diseases such as cancer by finding the aberrational regions in the DNA genome of a given sample. The major impediment to find such regions is that aCGH data is highly corrupted by various sources of noise. Therefore, a recovery method is required to approximate the noise-free data from contaminated observations. It was shown that the resulting aCGH data of a sample is both sparse and smooth [60]. Hence, FLSA is the minimization which is deemed to recover the noise-free data.

For this experiment, several synthesized and real aCGH data were subjected to the proposed neural solution, and its

\[1\] https://github.com/Majeed7
performance was compared with those of more sophisticated methods.

For the synthesized data, 50 samples with the length of 120 were generated. The elements of each aCGH sample were assumed normal if it is zero, and the aberrant regions were created by distorting the elements of each sample with the value 1 or -1. The length of each aberrant region was randomly selected from the set {5, 10, 20, 30}. Then, a Gaussian noise was added to each sample to produce the observation. To compare the methods in different corruption levels, several ratios of the data to the noise were considered. These ratios are called signal-to-noise ratios (SNRs). To affect the SNRs, one can simply write

\[ y = x + \text{SNR} \cdot e \]  

(20)

where \( e \) is the noise distributed according to the standard normal distribution. As a result, the higher values of SNR would result in less corrupted observations, and the smaller values would indicate that the data is highly contaminated with the noise \( e \). The noisy samples were then subjected to the neural network, total variation and spectral regularization (TVSp) [61], piecewise and low-rank approximation (PLA) [62], low-rank approximation based on half-quadratic programming (LRHQ) [63], and group fused lasso segmentation (GFLseg) [64]. The parameters of methods, when not determined by the method itself, are identically identified similar to the proposed neural network.

Since the ground truth is available in the synthetic experiments, one can simply compare the performance of each method by juxtaposing real and recovered data. For this experiment, the receiver operator characteristic (ROC) curve is used to contrast different methods. The ROC curve plots the true positive rate (TPR) against the false positive rate (FPR), and

deviation from diagonal is the indicator of the goodness of a method.

Fig. 3 plots the ROC curve of various methods over the synthesized data with different SNRs. As the level of corruption decreases, the performance of all methods increases significantly. In particular, the performance of the proposed neural network is competitive with TVSp and LRHQ in all levels of corruptions. On the other hand, it is significantly superior to GFLseg and PLA in all scenarios.

Although the performance of the proposed network is competitive with TVSp and LRHQ, the neural solution is more time- and memory-efficient. TVSp and LRHQ (and also PLA) use the nuclear norm regularization, which needs to compute the singular value decomposition in each iteration. For \( m \) samples with the length of \( n \), the singular value decomposition has the complexity of order \( O(mn^2) \). Hence, the methods based on the nuclear norm are of higher time complexity.

Regarding memory complexity, the proposed neural network does not need to use any auxiliary variables. The size of the network is commensurate with the length of the given data.
However, TVSp and LRHQ require the use of several auxiliary variables in order to solve their corresponding optimization problem. As a result, they occupy more space in the main memory compared with the proposed neural network.

To show the efficiency of the proposed neural network in practice, we generate synthesized data with varying lengths. The proposed neural solution is then compared with other methods based on the execution time required for each generated data. Fig. 4 plots the execution time in seconds of the foregoing methods on data with differing lengths.

The neural network is significantly superior to PLA and LRHQ regarding execution time. The neural solution is further competitive with TVSp, and GFLseg is significantly faster than other methods. The reason for this difference is that the proposed neural network is solely implemented in MATLAB while different parts of TVSp and GFLseg are implemented in C/C++. Generally, the proposed neural network is efficient concerning execution time.

The neural network is further applied to a real aCGH data to verify its efficiency. The Pollack et al. dataset [58] is a well-known aCGH dataset on breast cancer which contains 44 samples of 6691 human mapped genes. The dataset was subjected to the neural network, TVSp, and LRHQ, and three recovered samples are displayed in Fig. 5. In this figure, each row is dedicated to each sample, and each column corresponds to a method. The red dots in this figure are the real data, and the blue line is the data recovered by the corresponding techniques. Based on this figure, the recovered data by the proposed neural network is way smoother than those of TVSp and LRHQ, even though the neural network has less time and memory complexity in comparison to competing methods. To quantify the difference between methods in terms of smoothness, we gauge the total variation of each sample from the Pollack et al. dataset and compare the average of the total variation of methods using the Friedman test and the corresponding post-hoc analysis with Nemenyi’s correction method [65]. The null hypothesis of the Friedman test was rejected with p-value was close to zero. We further visualize the critical difference (CD) diagram to compare the methods based on the smoothness. Fig. 6 plots the rank of each method obtained by the Friedman test; the lower the rank, the better the method. Also, the methods that are not significantly different from each other are connected with red dots. According to this figure, GFLseg and the proposed neural network are identical top methods in terms of smooth recovery, followed by TVSp, LRHQ, and PLA. The reason that GFLSeg performs well is that, rather than conducting the recovery, it first finds the mutated points and then connects the points by a line. Thus, it has a very smooth recovery. This experiment corroborates the efficiency of the neural network in real-world situations.

D. Trend Filtering

In this section, the proposed neural network in (18) is applied to a fictitious dataset for the so-called trend filtering, and its results are compared with HP filtering [66]. The synthesized data is created based on the following equation:

\[
X_{t+1} = x_t + \frac{V_t}{V_t},
\]  

\[t = 1, 2, \ldots, n\]  

Fig. 5. Recovery of three samples from the Pollack et al. dataset by three methods. Each row is dedicated to a particular sample, while each column corresponds to each method. The methods are the proposed neural network, TVSp [61], and LRHQ [63]. The blue lines are the data recovered by the corresponding methods and the red dots denote the data before recovery.

Fig. 6. CD diagram to compare methods with respect to the smoothness of samples on the Pollack et al. dataset.
The proposed neural network can also be applied to several other problems as well. For instance, the total variation-regularized problem has diverse applications, such as image restoration, image deblurring, and hot-spot detection. The neural network is thus possible to solve these problems as well, which is left for the future research. Also, as Algorithm 1 suggests, the neural network can be executed highly in parallel. Therefore, another avenue for future research is to take advantage of the graphics processing units (GPUs) and execute the program in parallel with those technologies.

ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for their constructive comments. He would also like to thank the efforts of A. A. Atashin, M.Sc., for implementing the proposed neural network.

REFERENCES

quadratic programming problems and its k-winners-take-all application,” 


