FIXED-POINT ALGORITHMS FOR INVERSE OF RESIDUAL RECTIFIER NEURAL NETWORKS

RUHUA WANG∗
School of Electrical Engineering, Computing and Mathematical Sciences
Curtin University
Bentley, WA, Australia

SENIAN AN, WANQUAN LIU AND LING LI
School of Electrical Engineering, Computing and Mathematical Sciences
Curtin University
Bentley, WA, Australia

(Communicated by Xiaochun Cao)

Abstract. A deep neural network with invertible hidden layers has a nice property of preserving all the information in the feature learning stage. In this paper, we analyse the hidden layers of residual rectifier neural networks, and investigate conditions for invertibility under which the hidden layers are invertible. A new fixed-point algorithm is developed to invert the hidden layers of residual networks. The proposed inverse algorithms are capable of inverting some residual networks which cannot be inverted by existing inverting algorithms. Furthermore, a special residual rectifier network is designed and trained on MNIST so that it can achieve comparable performance with the state-of-art performance while its hidden layers are invertible.

1. Introduction. One of the fundamental difficulties in understanding the behaviour of deep neural networks is the loss of information due to the rectifier activation wherein the information of the negative components is discarded or reduced. As a consequence, the transformations of hidden layers may not be invertible, and the images cannot be recovered from their hidden layers. Many works have investigated the invertibility of visual representations [7, 14, 18] to open the black box of deep neural networks and understand deep image representations, but have observed significant information loss of the input images with increasing depth.

Recently, several invertible neural networks have been designed with special structures and have achieved impressive performances in image generation [13], image translation [19] and natural language processing [16]. An invertible network named as i-RevNet [12] is proposed for the analysis of deep residual networks and some new insights on the success of deep residual networks have been found. Although progressively discarding uninformative features and abstraction is widely believed to be the key factor for the successes of deep networks, [12] demonstrates that the loss of information is not a necessary condition, by enforcing one-to-one

2020 Mathematics Subject Classification. Primary: 68T07.
Key words and phrases. Invertible neural networks, fixed-point algorithms, invertible residual networks, invertible rectifier neural networks, inverse algorithm of neural networks.

∗ Corresponding author: Ruhua Wang.
mappings in the hidden layers in i-RevNet which performs similarly to the standard residual network, and suggests an alternative explanation for the success of deep networks by a progressive contraction and linear separation with increasing depth. The i-RevNet is constructed through a cascade of homeomorphic layers which can be fully inverted, with an explicit formula, up to the last hidden layer and therefore no information is discarded. The phenomenon of progressive separation and contraction in non-invertible networks is also observed in several other works \[15,21\].

In this paper, instead of designing inverting structures to recover the inputs from the hidden layers, we investigate the conditions under which the hidden layers of rectifier neural networks are invertible. Under these conditions there is no information loss in hidden layer transformations, and the inputs are recoverable from hidden layers. A recent work \[2\] presents a contractive condition on the Lipschitz-constant of the convolution path under which the residual unit is invertible. In this paper, we present some weaker conditions and propose new inverse algorithms for the invertible residual networks.

The main contributions of this paper include i) some weaker conditions than the contractive conditions presented in \[2\] are provided under which a residual unit is invertible; ii) new fixed-point algorithms are proposed to compute the inputs of residual units from their outputs; and iii) experimental results are provided to show that the proposed new fixed-point inverse algorithms can be used to invert more general residual units which cannot be inverted by the existing fixed-point algorithm presented in \[2\].

**Related work:** Understanding the transformations of hidden layers of deep neural networks is of great importance for understanding the evolution of the learned features with increasing depth. To achieve this, several works \[4,7,14,21\] have proposed to invert the representations by means of learned or hand-engineered priors but significant information of the input images is lost with depth in many successful deep neural network architectures. Recently, \[12\] constructs an invertible type of neural network architecture, namely RevNet, to investigate the evolution of hidden layer features. This work is closely related to reversible residual network (RevNet) \[8\], NICE \[5\] and Real-NVP architectures \[6\]. However, the structures of these neural networks are very different from standard neural networks in image classification. For standard residual networks, \[2\] provide a contractive condition on the Lipschitz-constant of the convolution path under which the residual unit is invertible. In this paper, we provide a weaker condition in terms of the weight matrix under which the residual unit is invertible. Other than the residual unit, the invertibility of other network components such as batch normalization and convolution are also critical for building an invertible network. \[13\] introduces \(1 \times 1\) invertible convolution and the reverse function of batch normalization named as actnorm in their generative model. In this paper, we also use \(1 \times 1\) invertible convolution as part of the proposed invertible convolution networks.

**Organization.** The rest of this paper is organised as follows. In Section 2, we introduce the concept of nonsingularity for rectifier linear transform and present the conditions under which the rectifier linear transform is nonsingular and thus invertible. Section 3 investigates the nonsingularity of the residual units. Section 4 presents a new algorithm to invert a special convolutional layer whereas Section 5 exhibits the architecture of our proposed network. Section 6 provides experimental results while Section 7 concludes the paper.
2. Nonsingularity of rectifier linear transform. In this section, we consider the rectifier linear equation

\[ \max(0, x) = Ax + b \quad (1) \]

and investigate the conditions on \( A \) under which Eq. (1) has a unique solution for any given \( b \). Similar to linear transforms, these conditions are closely related to the nonsingularity of the rectifier linear transform \( \{\max(0, x) - Ax\} \). A transform from \( \mathbb{R}^n \) to \( \mathbb{R}^m \), namely \( f(x) \), is called nonsingular if \( f(x_1) \neq f(x_2) \) holds for any \( x_1 \neq x_2 \). Similarly, it is called singular if there exists \( x_1 \neq x_2 \) such that \( f(x_1) = f(x_2) \). Note that a linear transform \( f(x) = Ax + b \) is nonsingular if and only if \( A \) is a nonsingular matrix. Apparently, if \( \{\max(0, x) - Ax\} \) is nonsingular, the solution to (1) is unique for any \( b \). On the other hand, if the solution to (1) is unique for any \( b \), \( \{\max(0, x) - Ax\} \) must be a nonsingular transform.

Next we present a sufficient condition on \( A \) such that Eq. (1) has a unique solution for any given \( b \).

**Theorem 1.** If

\[ \frac{1}{2} (A + A^T) < AA^T \quad (2) \]

then the solution of \( x \) to Eq. (1) is unique for any given \( b \).

Theorem 1 shows that when \( A \) is large enough, the solution of (1) is unique. For instance, let \( A \) be a nonsingular matrix, if \( (A + A^T) \) is negative definite, then (2) holds and the solution to Eq. (1) is unique. Otherwise, let \( \sigma_1 > 0 \) be the largest eigenvalue of \( (A + A^T) \) and \( \sigma_2 > 0 \) be the smallest eigenvalue of \( A^T A \). Then for \( \alpha > \sigma_1/(2\sigma_2) \), \( \alpha A \) satisfies (2) and thus \( \{\max(0, x) - \alpha Ax\} \) is nonsingular. This shows that for any non-singular \( A \), if \( \alpha \) is sufficiently large, \( \{\max(0, x) - \alpha Ax\} \) is then nonsingular.

**Proof.** Note that \( \max(0, x) = \max(0, -x) + x \). Equation (1) implies that

\[ \max(0, -x) = (A - I)x + b. \quad (3) \]

Suppose Equation (1) has two different solutions, namely \( x_1 \) and \( x_2 \). Then we have

\[
\begin{bmatrix}
\max(0, x_1) - \max(0, x_2) \\
\max(0, -x_1) - \max(0, -x_2)
\end{bmatrix}
= \begin{bmatrix} A \\ A - I \end{bmatrix} (x_1 - x_2). \quad (4)
\]

From Proposition 8 of [1], it follows that

\[
\left\| \begin{bmatrix}
\max(0, x_1) - \max(0, x_2) \\
\max(0, -x_1) - \max(0, -x_2)
\end{bmatrix} \right\| \leq \|x_1 - x_2\|. \quad (5)
\]

which, from Eq. (4), implies

\[
\left\| \begin{bmatrix} A \\ A - I \end{bmatrix} (x_1 - x_2) \right\| \leq \|x_1 - x_2\|. \quad (6)
\]

Since \( A^T A \geq \frac{1}{2} (A + A^T) \), we have

\[
\begin{bmatrix} A \\ A - I \end{bmatrix}^T \left[ \begin{bmatrix} A \\ A - I \end{bmatrix} \right] = 2A^T A - A - A^T + I \geq I \quad (7)
\]
and therefore
\[ \left\| \begin{bmatrix} A \\ A - I \end{bmatrix} (x_1 - x_2) \right\| > \|x_1 - x_2\|, \forall x_1 \neq x_2. \quad (8) \]
Hence (6) holds only when \( x_1 = x_2 \), which implies that the solution to Eq (1) is unique.

\[ \square \]

2.1. **The proposed inverse algorithm.** This section presents an algorithm to solve the rectifier linear equation (1), that is, compute \( x \) with given \( A \) and \( b \).

From Eq.(1) and note that \( \max(0, -x) = \max(0, x) - x \), we have
\[ \begin{bmatrix} \max(0, x) \\ \max(0, -x) \end{bmatrix} = \begin{bmatrix} A \\ A - I \end{bmatrix} x + \begin{bmatrix} b \\ b \end{bmatrix} \quad (9) \]
and therefore
\[ x = \begin{bmatrix} A \\ A - I \end{bmatrix}^\dagger \begin{bmatrix} \max(0, x) \\ \max(0, -x) \end{bmatrix} - \begin{bmatrix} A \\ A - I \end{bmatrix}^\dagger \begin{bmatrix} b \\ b \end{bmatrix}. \quad (10) \]

The proposed fixed-point inverse algorithm is described as below:

1) Let \( x_0 = b \);
2) for \( k = 1, 2, \cdots \), do
\[ x_k = \begin{bmatrix} A \\ A - I \end{bmatrix}^\dagger \begin{bmatrix} \max(0, x_{k-1}) \\ \max(0, -x_{k-1}) \end{bmatrix} - \begin{bmatrix} A \\ A - I \end{bmatrix}^\dagger \begin{bmatrix} b \\ b \end{bmatrix}. \quad (11) \]
until \( \|x_k - x_{k-1}\| < \epsilon \) where \( \epsilon \) is threshold for convergence.

Note that the maximum singular value of \( \begin{bmatrix} A \\ A - I \end{bmatrix}^\dagger \) is less than 1 under the condition in Theorem 1, and
\[ x_{k+1} - x_k = \begin{bmatrix} A \\ A - I \end{bmatrix}^\dagger \begin{bmatrix} \max(0, x_k) - \max(0, x_{k-1}) \\ \max(0, -x_k) - \max(0, -x_{k-1}) \end{bmatrix} \quad (12) \]
we have
\[ \|x_{k+1} - x_k\| < \left\| \begin{bmatrix} \max(0, x_k) - \max(0, x_{k-1}) \\ \max(0, -x_k) - \max(0, -x_{k-1}) \end{bmatrix} \right\| < \|x_k - x_{k-1}\|. \quad (13) \]

Hence, this algorithm converges if the condition (2) in Theorem 1 is satisfied. From Theorem 1, the solution is unique and therefore the algorithm converges to its unique solution.

3. **Invertibility of residual units.** For residual networks with rectifier as the nonlinear activation function, we have
\[ x_{k+1} = \max(0, W_k x_k + b_k) + x_k \quad (14) \]
where \( W_k \) is the linear transformation matrix which may include convolution and batch normalization since both of them are essentially linear transforms.
Next, we consider the conditions under which $x_k$ is recoverable from $x_{k+1}$. Let $z_k = W_k x_k + b_k$ and assume that $W_k$ is nonsingular, we have

$$x_k = W_k^{-1} z_k + \hat{b}_k$$

(15)

where $\hat{b}_k = -W_k^{-1} b_k$. Rewrite (14) in terms of $z_k$, we have

$$x_{k+1} = \max(0, z_k) + W_k^{-1} z_k + \hat{b}_k$$

(16)

that is,

$$\max(0, z_k) = -W_k^{-1} z_k + x_{k+1} - \hat{b}_k.$$  

(17)

From Theorem 1, if

$$W_k^{-T} W_k^{-1} \geq \frac{1}{2} (-W_k^{-1} - W_k^{-T}),$$

(18)

i.e,

$$I \geq \frac{1}{2} (-W_k - W_k^T),$$

(19)

then $z_k$ is recoverable from $x_{k+1}$. Since $x_k$ is recoverable from $z_k$, $x_k$ is recoverable from $x_{k+1}$ as well.

Hence, we have

**Corollary 2.** *If $W_k$ is nonsingular and

$$-(W_k + W_k^T) \leq 2I,$$

then $x_k$ is recoverable from $x_{k+1}$ from (14). That is, the transform from $x_k$ to $x_{k+1}$ in a residual hidden layer is nonsingular.*

3.1. **Fixed-point inverse algorithm of the residual unit.** For the inverse of a residual unit, one needs to solve the following rectifier equation:

$$\max(0, W x + b) = -x + c$$

(21)

which is equivalent to

$$\max(0, -W x - b) = -(I + W)x + c - b.$$  

(22)

By combining these two equations, we have

$$\begin{bmatrix} I \\ I + W \end{bmatrix} x = - \begin{bmatrix} \max(0, W x + b) \\ \max(0, -W x - b) \end{bmatrix} + \begin{bmatrix} c \\ c - b \end{bmatrix}$$

(23)

and therefore

$$x = - \begin{bmatrix} I \\ I + W \end{bmatrix}^\dagger \begin{bmatrix} \max(0, W x + b) \\ \max(0, -W x - b) \end{bmatrix} + \begin{bmatrix} I \\ I + W \end{bmatrix}^\dagger \begin{bmatrix} c \\ c - b \end{bmatrix}$$

(24)

The proposed inverse algorithm is as below

1) Let $x_0 = \begin{bmatrix} I \\ I + W \end{bmatrix}^\dagger \begin{bmatrix} c \\ c - b \end{bmatrix}$;

2) for $k = 1, 2, \cdots$, do

$$x_k = - \begin{bmatrix} I \\ I + W \end{bmatrix}^\dagger \begin{bmatrix} \max(0, W x_{k-1} + b) \\ \max(0, -W x_{k-1} - b) \end{bmatrix} + \begin{bmatrix} I \\ I + W \end{bmatrix}^\dagger \begin{bmatrix} c \\ c - b \end{bmatrix}$$

(25)

until $\|x_k - x_{k-1}\| < \epsilon$ where $\epsilon$ is threshold for convergence.
For convolution layers, we need to carefully consider the implementation of the pseudo-inverse of \[
\begin{bmatrix}
I \\
I + W
\end{bmatrix}
\] since the size of \(W\) is extremely large. We need to find an efficient way to compute the pseudo-inverse through convolutions/deconvolutions. Note that the pseudo-inverse of \[
\begin{bmatrix}
I \\
I + W
\end{bmatrix}
\] equals to:

\[
\left(\begin{bmatrix}
I \\
I + W
\end{bmatrix}^T \begin{bmatrix}
I \\
I + W
\end{bmatrix}\right)^{-1} \begin{bmatrix}
I \\
I + W^T
\end{bmatrix}
\]

Then Eq. \((25)\) can be rewritten as

\[
x_k = (2I + W + W^T + W^TW)^{-1} \begin{bmatrix}
I \\
I + W^T
\end{bmatrix} \begin{bmatrix}
y_1 \\
y_2
\end{bmatrix}
\]

where \(y_1 = \mathbf{c} - \max(0, Wx_{k-1} + \mathbf{b})\) and \(y_2 = \mathbf{c} - \mathbf{b} - \max(0, -Wx_{k-1} - \mathbf{b})\).

Let \(F\) be the filter for the convolution associated with \(W\). \(WT\) can be implemented using convolution with filter \(F^T\). To obtain \((I + WT)\mathbf{y}_2\), one can implement it as a convolution \(F_1 \ast \mathbf{y}_2\). Here, \(F_1\) is the associated filter, that is \(F_1 = (C + F^T)\), while \(C\) is a matrix whose central element is 1 and other elements are zero.

Then the Eq. \((27)\) can be simplified as

\[
(I + (I + W)^T (I + W)) \mathbf{x}_k = \mathbf{y}
\]

where \(\mathbf{y}\) is \((y_1 + (I + W^T) \mathbf{y}_2)\). It can be implemented as a convolution as well, that is \(F_2 \ast \mathbf{x}_k = \mathbf{y}\). \(F_2\) is the associated convolution filter of \((I + (I + W)^T (I + W))\), which can be easily obtained via \(F_2 = C_2 + (C_1 + F^T) \ast (C_1 + F)\). Note \(C_1\) and \(C_2\) are matrices whose central element is 1 and other elements are zero. Finally, we can compute \(\mathbf{x}_k\) through deconvolution once \(F_2\) and \(\mathbf{y}\) are obtained. The details of deconvolution operation are shown in section 4.1.

3.2. Comparison to [2]. In [2], another fixed-point algorithm is presented to inverse a residual. When applied to Eq. \((21)\), this algorithm can be described as below.

1) Let \(\mathbf{x}_0 = \mathbf{c}\);
2) for \(k = 1, 2, \cdots\), do

\[
x_k = \mathbf{c} - \max(0, Wx_{k-1} + \mathbf{b})
\]

until \(\|x_k - x_{k-1}\| < \epsilon\) where \(\epsilon\) is threshold for convergence.

For the convergence of this algorithm, the maximum singular value of \(W\) should be less than 1, that is

\[
-1 < \frac{x^T W x}{x^T x} < 1, \forall x
\]

and therefore

\[
-2I < W + W^T < 2I
\]

which is much stronger than the condition for the convergence of the proposed algorithm, that is, \((-W - W^T) \geq 2I\).
4. Invertibility of convolutional layers. In this section, we illustrate how to invert the convolution layer in our specially designed network.

A convolution layer usually consists of multiple channels and attempts to learn multiple filters in both spatial dimension (width and height) and depth dimension (channel). Consequently, the task of a single convolution kernel is to map cross-channel correlations and spatial correlations simultaneously. To make the convolution layer easier to be inverted in our proposed network, we use depthwise separable convolutions which are commonly used in deep learning [3, 10] instead of standard convolution. The depthwise separable convolutions consist of two steps: depthwise convolutions followed by pointwise convolutions, which operates as a spatial feature learning step and a channel combination step. In our specially designed residual network, depthwise convolutions are used in the residual units while the pointwise convolutions are employed out of the residual units. In this case, instead of inverting the depthwise convolutions independently, we invert the whole residual unit as proposed in Section 3. Thus, the objective of inverting a convolutional layer can be formulated as two steps as well: restore the input image from the pointwise convolution layer and depthwise convolution involved residual units sequentially. Besides, an important process called deconvolution operation is involved to invert the first layer of the proposed residual network in our study. Therefore, in the first part of this section, we introduce the method for deconvolution operation, while in the second part inverse process of depthwise convolutions is explained.

4.1. Deconvolution. Given a 2-dimensional single channel image $I$, and a 2-dimensional kernel $K$, the convolution operation [9] can be mathematically represented as

$$S(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m, n), \quad (32)$$

where $S$ is the convolved image, i.e., obtained feature map.

For our problem, to restore the input image $I$ from the convolved image $S$ with the kernel $K$, a deconvolution technique called Inverse Filtering [17] is used to reverse the effects of convolution operation on the input image. Deconvolution with inverse filtering is usually performed in frequency domain by computing the Fourier transform of $I$ and $S$.

Finally applying the inverse Fourier transform on $F\{I\}$ we can get the estimated deconvolved image $\hat{I}$.

For our problem, to restore the input image $I$ from the convolved image $S$ with the kernel $K$, a deconvolution technique called Inverse Filtering [17] is used to reverse the effects of convolution operation on the input image. Deconvolution with inverse filtering is usually performed in frequency domain by computing the Fourier transform of $K$ and $S$. We can obtain the Fourier transform of $I$ by

$$F\{I\} = F\{S\}/F\{K\}. \quad (33)$$

Finally applying the inverse Fourier transform on $F\{I\}$ we can get the estimated deconvolved image $\hat{I}$.

However, many neural network libraries, such as Tensorflow and Keras, implement the cross-correlation instead of convolution as below.

$$S(i, j) = (K \ast I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n), \quad (34)$$

It is the same as the convolution operation but without flipping the kernel. Therefore, in our implementation, we first flip the kernel and then perform the above-mentioned deconvolution process to recover the convolved image.

4.2. Pointwise deconvolution. A pointwise convolution is also called $1 \times 1$ convolution in the deep learning community. It learns cross-channel correlation by projecting the channels outputs by previous layer (e.g. depthwise convolutions)
onto a new channel space. This operation in a $1 \times 1$ convolutional layer can be rewritten into a linear transformation form as:

$$y = Wx,$$  \hspace{1cm} (35)

where $x$ is the input coming from the residual unit in our designed network, $W$ is the weights learned by the filter and $y$ is the output to next layer. For simplicity, we omit the bias term here.

Thus, the deconvolution of the $1 \times 1$ convolution operation can be formulated as the following equation.

$$\hat{x} = W^{-1}y$$  \hspace{1cm} (36)

We implement this equation as a reverse transform function for restoring input features from the $1 \times 1$ convolution layer.

5. Invertible network architecture. Section 3 and Section 4 demonstrate the invertibility of the residual unit with a rectifier activation and a convolution layer, which makes it possible to build an invertible network. Besides, the invertibility of other components, such as batch normalization (BN) and bidirectional ReLU used in our network is also of great importance. In this section, we first introduce the architecture of the proposed invertible network, followed by the inverting operation of batch normalization and bidirectional ReLU.

5.1. Network architecture. As proposed in Eq.(14), $W_k$ in a residual unit is the linear transformation matrix which may include convolution and batch normalization. However, batch normalization in this algorithm actually works as an image size convolution while the convolution is a kernel size (e.g. $3 \times 3$) convolution. These two convolutions cannot be inverted together in the residual unit. Therefore, we move the batch normalization out of the residual unit when designing the network architecture. In our experiments, only one residual unit is used in the network. The architecture of the specially designed residual network is illustrated in Figure 1, in which the layers in green are considered invertible.

The network starts with a regular $3 \times 3 \times 5$ convolution layer, which expands the input to a higher dimension. Then, the feature dimension is doubled by a bidirectional ReLU activation. For a regular ReLU activation function, it is defined as $f(x) = x^+ = \max(0, x)$, where $x$ is the input to a neuron. Only the positive part of $x$ is preserved, hence it is impractical to reverse this activation operation in a normal layer. To address this issue, we introduce a bidirectional ReLU that preserves all the information of $x$. It can be defined as

$$f(x) = \begin{bmatrix} x^+ \\ x^- \end{bmatrix} = \begin{bmatrix} \max(0, x) \\ \max(0, -x) \end{bmatrix},$$  \hspace{1cm} (37)

where the square brackets denote a matrix concatenation in channel dimension. That is why the feature dimension is doubled after a bidirectional ReLU activation. An invertible residual unit is added afterwards. Here we use a $3 \times 3 \times 10$ filter in depthwise conv layer in the residual unit after which a $1 \times 1 \times 10$ in pointwise convolution layer and a batch normalization layer are added. Finally one dropout layer and two fully-connected layers are introduced for a classification problem. The proposed residual network achieves a comparable performance for classification in our experiments. Experiment results are presented in Section VI.
5.2. Invertibility of batch normalization. Batch normalization was first introduced by [11] that standardizes the input to a neural network layer for every mini-batch. It stabilizes and accelerates the training of neural networks. Assuming that we have a mini-batch of inputs: \( \mathcal{X} = \{x_1, x_2, ..., x_m\} \), the variance \( \sigma^2 \) and mean \( \mu \) of this mini-batch are computed to perform normalization as below:

\[
y_i = \gamma \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta,
\]

where \( \gamma \) and \( \beta \) performs scaling and shifting on standardized \( x_i \), and \( \epsilon \) is a small value added to variance to avoid dividing by zero.

To reverse the effect on data \( x_i \), we implement the following equation and call it de-normalization:

\[
x_i = \frac{y_i - \beta}{\gamma \sqrt{\sigma^2 + \epsilon + \mu}}.
\]

5.3. Invertibility of bidirectional ReLU. The bidirectional ReLU used after the first conv layer in our paper preserves all the information of \( x \) (Eq.(37)). Thus it has the property of recovering the original input data, which can be achieved by

\[
x = x^+ - x^- = \max(0, x) - \max(0, -x).
\]

6. Experimental results. Experiments are conducted to validate the performance of the proposed fixed-point inverse algorithms for the rectifier linear transform in Eq.(11) and the residual units in Eq.(25). We compared our inverse algorithms with the existing inverse algorithm presented in [2], which also inverts a residual layer via a fixed-point method. The experimental results demonstrate that the proposed inverse algorithms are more widely applicable than the method we compared as claimed theoretically in Section 3. In other words, the proposed algorithm works on more cases under weaker conditions. Details of the experiments are described in the following sections.

6.1. Evaluating invertibility of rectifier linear transforms. To evaluate the performance of the proposed inverse algorithm in Section 2.1, we randomly generate some synthetic data and recover it through the proposed method. The existing method presented in [2] is compared under the same condition. Details of the inverse processes are presented as follows.

Firstly, we randomly generate \( A \) and \( x \), and then obtain \( b \) by Eq.(1). \( x \) is used as the ground truth. After that, we recover \( x \) through both the proposed inverse
algorithm and the existing algorithm with the given $A$ and $b$. We denote the recovered input by $\hat{x}$. To determine the performance of the inverse algorithms, we calculate the relative error between the ground truth $x$ and the recovered $\hat{x}$ based on $E_r = \frac{\|\hat{x} - x\|}{\|x\|} \times 100\%$. $\hat{x}$ is considered as fully recovered when $E_r = 0$.

In the experiment, we randomly generate $x$ of dimension 10, $M$ of size $10 \times 10$ and set $A = \gamma \ast M$ where $\gamma$ is used to control the norm of $A$ so that we can investigate how the inverse performance is affected by the norm of $A$. $b$ is obtained by Eq.(1). Each element of $M$ and $x$ is randomly selected from the standard normal distribution. We choose 10 different scales of $\gamma$, that is $\gamma = 1, 2, 3, \ldots, 10$. For each $\gamma$, 500 cases are evaluated to determine the convergence of the inverse algorithms. When $\hat{x}$ is fully recovered, that case is considered invertible.

Figure 2 shows the performance of the proposed inverse algorithm and the existing algorithm. The percentages of invertible cases increase with the increment of $\gamma$ for both methods. The performance of the proposed inverse algorithm consistently outperforms the existing algorithm as demonstrated in Figure 2. For the proposed inverse algorithm, more than 73% of $x$ (out of 500 cases) have been inversely successfully when the scale $\gamma$ is 1, while only 42.7% cases work for the existing algorithm. Besides, the proposed algorithm consistently outperforms the existing method with the increment of $\gamma$.

As observed from the experiment, the proposed inverse algorithm works for more cases than the existing algorithm when $A$ and $b$ are randomly generated.

6.2. Evaluating invertibility of residual units: Fully-connected layer. To evaluate the performance of the proposed inverse algorithm for residual units in Section 3.1, two situations are investigated. The first one is that the linear transform matrix $W$ works as a fully-connected layer. Again, the synthetic data is used in this experiment.

Similarly, we randomly generate $W$, $b$, and $x$, and then obtain $c$ by Eq.(14). Again, we recover $x$ through both the proposed inverse algorithm and the existing algorithm. The inverse performance is determined via the relative error between the recovered input $\hat{x}$ and the ground truth $x$.

The generation process for $W$, $x$ and $b$ is similar to the above experiment. A scale variable $\gamma$ that controls the norm of $W$ is also applied (to investigate how the inverse performance is affected by the norm of $W$). Here, we choose 10 different scales with $\gamma = 0.1, 0.2, 0.3, \ldots, 1.0$. 500 cases are evaluated to determine the convergence of the inverse algorithms for each $\gamma$.

Figure 3 demonstrates the performance of the proposed fixed-point inverse algorithm and the existing algorithm. Both algorithms work for all cases when $\gamma = 0.1$. The percentages of invertible cases decrease with the increment of $\gamma$ for both two methods. However, the proposed method consistently works on more cases. That is because when $\gamma$ is small enough (e.g. 0.1), the fully-connected layer is contractive and the residual unit can be inverted by both methods. Otherwise, the fully-connected may not be contractive, but may still satisfy the new conditions for the convergence of the proposed algorithm in Section 3.1. Thus, the proposed fixed-point algorithm is still more widely applicable than the existing fixed-point algorithm.

6.3. Evaluating invertibility of residual units: Convolutional layer. The other situation is when the linear transform matrix $W$ is obtained through a convolution layer in the residual units. To evaluate the performance of the proposed
Figure 2. Inverse of the rectifier linear transform: Invertible percentage of 500 cases changes along with $\gamma$ when the dimension of $x$ is 10.

inverse algorithm for residual units in Eq.(25), we conducted some experiments on a special residual network using only one residual unit. The architecture of the proposed residual network is described in Section 5. As shown in Figure 1, the layers in green are all invertible. Based on Theorem 1, if the weights in a residual layer are small enough, the residual unit is guaranteed invertible. Thus, a max norm constraint (e.g. max norm is 0.5) is enforced on the kernels of the depthwise convolution layer in our implementation. The value is selected based on the validation dataset. Experiments on MNIST\(^1\) data set for digit classification demonstrate that the proposed residual network shows good invertibility under this condition while achieves a good performance for classification. MNIST data set consists of 70,000 handwritten digits examples. The whole database is split into three subsets: 50,000 for training, 10,000 for validation and another 10,000 for testing. Each digit is a $28 \times 28$ grey scale image. 60 epochs are executed while Adam optimizer with learning rate 0.001 is used during the training stage. The test classification error of the proposed residual network is 0.88\%, which is comparable to the state-of-art performance on MNIST without data augmentation [20], despite using a small network with weight constraints enforced.

The invertibility of the proposed architecture is then validated based on the trained classification model. The layers shown in green in Figure 1 are all invertible as discussed above. We use the inverse algorithm presented in Eq.(25) to recover the original images from the residual unit. Again, the proposed inverse algorithm for residual units is compared with the existing fixed-point method. Six samples are randomly selected from the training set for recovering (Figure 4). It can be seen

\(^1\)http://yann.lecun.com/exdb/mnist/
Figure 3. Inverse of the residual unit with the fully-connected layer: Invertible percentage of 500 cases changes along with $\gamma$ when the dimension of $x$ is 10.

Figure 4. Comparison of recovered images to original digit images. The 1st row illustrates the original images, whereas the 2nd and 3rd rows show the recovered images from the proposed fixed-point method and the existing fixed-point method, respectively.

that each sample is successfully reconstructed from the invertible model using the proposed method, much better than using the existing method.

To further study the applicability of our model, we conducted this recovering on 1000 MNIST images and the relative error is shown in Figure 5. 100 samples per class are randomly chosen and recovered. We calculate the relative errors between
the original images $x$ and the recovered images $\hat{x}$ based on $E_r = \frac{\|\hat{x} - x\|}{\|x\|} \times 100\%$. The relative errors for all recovered images using the proposed method are zeros. The same validation is conducted on the existing fixed-point method, and the relative error for all cases are above 68%. Thus it can be concluded that the existing method fails to invert the trained residual network model.

In summary, our proposed network with only one residual block performs well in classification and its invertibility is guaranteed at the same time. Besides, the proposed fixed-point inverse algorithm is more widely applicable, in the sense that it works on more cases under a weaker condition comparing to the existing fixed-point method.

7. Conclusion. The invertibility of the residual networks with rectifier as activation functions has been investigated and conditions are presented for the residual units to be invertible. The conditions are addressed for general rectifier linear units and can be used to analyse the invertibility of other deep rectifier networks. A new algorithm is also proposed to invert hidden layers of residual networks, and a special residual neural network is designed in our experiments to validate the invertibility without sacrificing the classification performance. The experimental results demonstrate that the proposed fixed-point inverse algorithms can be widely applicable since it works on more cases under weaker weight constraints comparing to the existing fixed-point method.
REFERENCES

[1] S. An, F. Boussaid and M. Bennamoun, How can deep rectifier networks achieve linear separability and preserve distances?, International Conference on Machine Learning, 2015, 514–523.

[2] J. Behrmann, W. Grathwohl, R. T. Q. Chen, D. Duvenaud and J.-H. Jacobsen, Invertible residual networks, International Conference on Machine Learning, 2019, 573–582.

[3] F. Chollet, Xception: Deep learning with depthwise separable convolutions, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, 1251–1258.

[4] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin and N. Usunier, Parseval networks: Improving robustness to adversarial examples, in Proceedings of the 34th International Conference on Machine Learning-Volume 70, 2017, 854–863.

[5] L. Dinh, D. Krueger and Y. Bengio, NICE: Non-linear independent components estimation, preprint, arXiv:1410.8516.

[6] L. Dinh, J. Sohl-Dickstein and S. Bengio, Density estimation using real NVP, preprint, arXiv:1605.08803.

[7] A. Dosovitskiy and T. Brox, Inverting visual representations with convolutional networks, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 4829–4837.

[8] A. N. Gomez, M. Ren, R. Urtasun and R. B. Grosse, The reversible residual network: Backpropagation without storing activations, in Advances in Neural Information Processing Systems, 2017, 2214–2224.

[9] I. Goodfellow, Y. Bengio and A. Courville, Deep Learning. Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, 2016. Available from: http://www.deeplearningbook.org.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto and H. Adam, MobileNets: Efficient convolutional neural networks for mobile vision applications, preprint, arXiv:1704.04861.

[11] S. Ioffe and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, preprint, arXiv:1502.03167.

[12] J.-H. Jacobsen, A. Smeulders and E. Oyallon, i-RevNet: Deep invertible networks, preprint, arXiv:1802.07088.

[13] D. P. Kingma and P. Dhariwal, Glow: Generative flow with invertible 1x1 convolutions, in Advances in Neural Information Processing Systems, 2018, 10215–10224.

[14] A. Mahendran and A. Vedaldi, Understanding deep image representations by inverting them, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, 5188–5196.

[15] E. Oyallon, Building a regular decision boundary with deep networks, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, 5106–5114.

[16] R. Prenger, R. Valle and B. Catanzaro, Waveglow: A flow-based generative network for speech synthesis, ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 2019, 3617–3621.

[17] A. Saberi, A. A. Stoorvogel and P. Sannuti, Inverse filtering and deconvolution, Internat. J. Robust Nonlinear Control, 11 (2001), 131–156.

[18] R. Shwartz-Ziv and N. Tishby, Opening the black box of Deep Neural Networks via Information, preprint, arXiv:1703.00810.

[19] T. F. van der Ouderaa and D. E. Worrall, Reversible GANS for memory-efficient image-to-image translation, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, 4720–4728.

[20] J. Wang and L. Perez, The effectiveness of data augmentation in image classification using deep learning, preprint, arXiv:1712.04621.

[21] M. D. Zeiler and R. Fergus, Visualizing and understanding convolutional networks, in Computer Vision–ECCV 2014, Lecture Notes in Computer Science, 8089, Springer, Cham, 2014, 818–833.

Received August 2020; revised September 2020.

E-mail address: ruhua.wang@postgrad.curtin.edu.au
E-mail address: s.an@curtin.edu.au
E-mail address: w.liu@curtin.edu.au
E-mail address: l.li@curtin.edu.au