Comparison Between a Novel Compressed Sensing-Based Neural Network and Traditional Neural Network Approaches for Electrical Impedance Tomography Reconstruction

Damond M. Li[a], Marco G. Araiza[b], and Long Wang[c,*]
[a]Department of Mechanical Engineering, California Polytechnic State University, 1 Grand Ave, CA, USA 93407;
[b]Department of Computer Science & Software Engineering, California Polytechnic State University, 1 Grand Ave, CA, USA 93407;
[c]Department of Civil and Environmental Engineering, California Polytechnic State University, 1 Grand Ave, CA, USA 93407
[*]E-mail: lwang38@calpoly.edu

ABSTRACT

Electrical impedance tomography (EIT) is a non-destructive and non-radioactive imaging technique used to detect anomalies in a material of interest. Applications of EIT range from medical imaging and early tumor detection to identifying structural damage. Within the past decade, deep learning (DL)-based EIT reconstruction has been an emerging field of study as it shows promise in addressing many of the challenges associated with the non-linear, ill-conditioned nature of EIT inverse problems. The DL-based approach allows for the conductivity of materials to be reconstructed directly through neural networks (NNs) as opposed to iteratively with conventional inverse reconstruction algorithms. So far, the reported DL-based NNs for EIT have mostly been trained by minimizing the mean squared error (MSE) between the predicted and "true" outputs (i.e., conductivity distributions). The performance of these current NNs heavily relies on both the quality and quantity of training data. The NNs trained with simulated data may perform poorly with experimental data. On the other hand, generating sufficient experimental data NN training can be extremely expensive and time-consuming, if feasible at all. To advance the DL-based reconstruction for EIT, this study develops a novel NN architecture, trained with a custom loss function, that serves as a surrogate model for the compressed sensing-based EIT reconstruction algorithm. In other words, the NN is trained to mimic a compressed sensing algorithm that performs the EIT conductivity reconstruction. This approach enables the NN to accurately capture the electrical properties and characteristics of the sensing domain when trained with limited data of varying quality. The performance of the proposed NN was compared to other DL models trained with the traditional MSE loss function by evaluating their reconstruction resolution, accuracy, and other training metrics.

Keywords: Artificial intelligence, compressed sensing, data acquisition system, deep learning, electrical impedance tomography, inverse problem, neural network

1. INTRODUCTION

Electrical impedance tomography (EIT) has garnered increasing attention in recent years as an effective imaging technique for structural health monitoring (SHM) and medical imaging since it is nondestructive, noninvasive, and radiation free. By injecting small electrical currents across an area of interest, its internal conductivity distribution can be characterized. Such information is useful for identifying propagating cracks in concrete, monitoring lung ventilation, and even early diagnosis of tumors [1-3]. Traditional methods for EIT conductivity reconstruction often utilize computationally intensive iterative algorithms. This is mainly due to the ill-posed nature of the EIT inverse problem. With recent advances in computational hardware, deep learning (DL) has been an increasingly attractive approach for EIT reconstruction as it shows potential to enhance reconstruction resolution and quality compared to traditional iterative algorithms, especially for solving non-linear ill-posed problems [4]. However, one major drawback to DL is the reliance on large amounts of data to successfully generalize the problem at hand. Gathering large amounts of data is often extremely impractical, especially for EIT, given the number of variables involved (e.g., shape of sensing domain, number of electrodes, measurement scheme, etc.). This

study aims to leverage a compressed sensing technique to train a neural network (NN) that performs the EIT inverse problem using only simulated training data.

2. TECHANICAL BACKGROUND

2.1. Electrical Impedance Tomography

EIT consists of the forward problem and the inverse problem [5, 6]. The forward problem attempts to solve for the boundary voltage distribution for an assumed conductivity distribution, and the inverse problem solves for the internal conductivity distribution for the given boundary voltage distribution. In practice, the boundary voltage is measured by a data acquisition (DAQ) system and used as input for an EIT reconstruction algorithm to estimate the internal conductivity distribution. Since different internal conductivity distributions can correspond to the same boundary voltage distribution, the EIT inverse problem is severely ill-posed and requires additional optimization techniques to aid reconstruction. One of the optimization algorithms that has been demonstrated effective is based on the compressed sensing (CS) technique [5, 6], which leverages the sparsity of EIT problems to enhance the reconstruction resolution and accuracy. A common CS algorithm used is the iterative shrinkage/thresholding (IST) algorithm, which attempts to solve the following objective function:

$$\delta\sigma = argmin\frac{1}{2}\|J\delta\sigma - \delta V\|_2 + \lambda\|\delta\sigma\|_1 \tag{1}$$

where $\delta\sigma$ is the change in conductivity, J is the Jacobian for the sensing domain, δV is the change in boundary voltage distribution between the homogeneous and inhomogeneous state, λ is the regularization parameter, and $\|\delta\sigma\|_1$ is the L_1-norm of $\delta\sigma$ [5]. This objective function attempts to find a set of $\delta\sigma$ that matches closely with the given δV when transformed by Jacobian, J. The L_1-norm term is included to promote sparsity and tackle the ill-posed nature of the inverse problem. λ controls how much influence the L_1-norm has which can help with noisy reconstructions. This objective function could be leveraged by deep learning to train a neural network.

2.2. Deep Learning

Deep learning is a subset of machine learning and artificial intelligence that utilizes an artificial neural network (NN) comprised of multiple nodes that mimic neurons in a thinking brain. These nodes are organized into layers which are connected to other layers by a set of weights. Essentially, each layer in a NN is represented by a tensor, and a set of tensor operations are performed to go from one layer to another.

Deep learning fundamentally requires three components: input data, expected output data, and a loss function (i.e., a metric to evaluate how well the neural network performs) [7]. During training, the NN output is compared to the expected output and evaluated using a loss function. A common loss function used for deep learning regression models is mean squared error (MSE) which can be calculated by:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2}$$

where y_i is expected output value from the training dataset and $\hat{y}_i$ is the NN output value [4, 8]. During training, the calculated loss is fed into an optimizer that adjusts the weights in each layer of the NN to minimize loss [7].

Since the goal of the optimizer is to minimize a computed numerical value, Equation (1) can be modified to serve as a loss function for NN training. In the loss function:

$$loss = \frac{1}{2}\left\|J\widehat{\delta\sigma} - \delta V\right\|_2 + \lambda\left\|\widehat{\delta\sigma}\right\|_1 \tag{3}$$

where the NN output, $\widehat{\delta\sigma}$, is transformed with J and compared with the NN input, δV. Note that the expected output (i.e., conductivity) is not included in Equation 3, which will render the training independent of the output. Rather than relying on the NN to extract features from the training dataset, features are learned using the custom loss function instead. As a consequence, this approach is expected to make training more resilient to lower quality data, since the expected output in the training dataset has no effect on the calculated loss.

3. METHOD

3.1. Compressed Sensing-Based Neural Network

The proposed CS-based NN incorporates important features from the traditional IST algorithm, namely the architecture and the soft thresholding functions.

3.1.1. Architecture

The IST architecture for EIT can be visually represented by Figure 1.

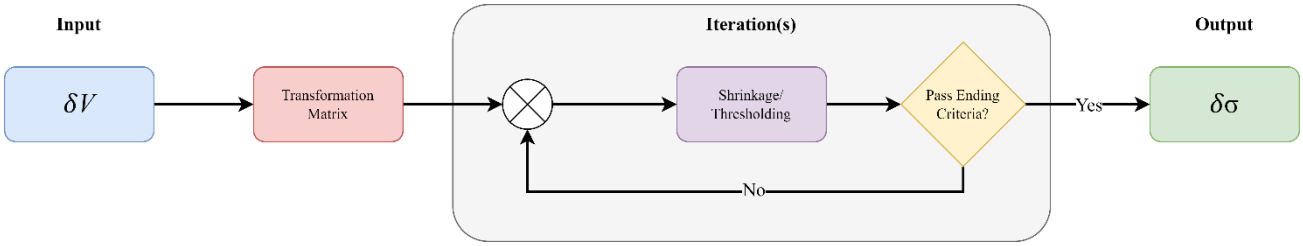

Figure 1. Visual representation of the IST algorithm.

In Figure 1, the input data is mapped using the transformation matrix and it iteratively passes through shrinkage and thresholding functions until the ending criteria is met [9]. Examples of the ending criteria can be a max number of iterations or a converging loss derived from Equation (2). Inspired by the IST architecture, the novel NN developed in this study "unfolds" the iterative portion and represents each individual iteration as a separate series of layers in the NN.

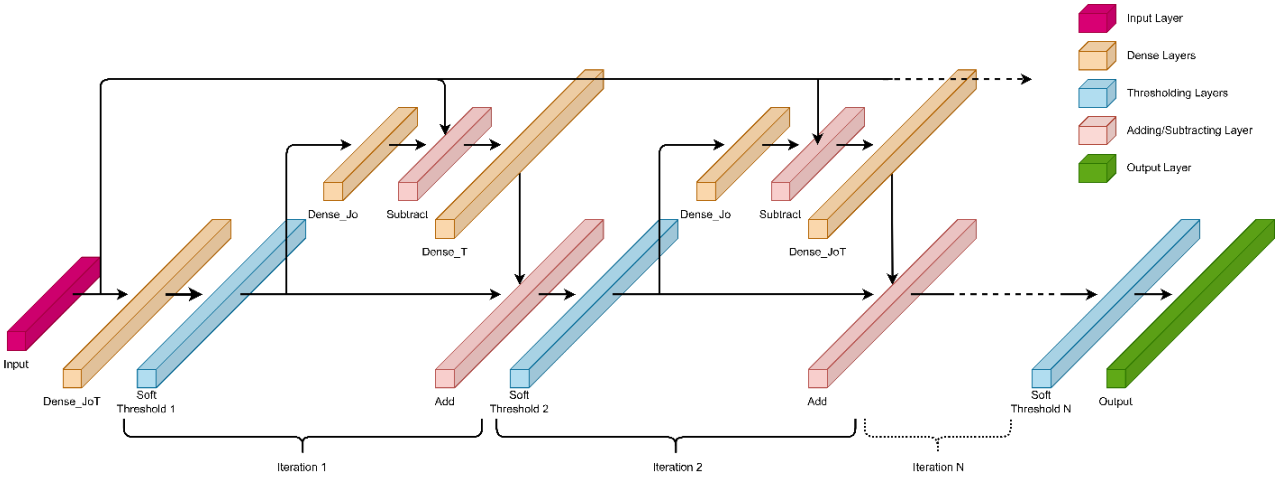

Figure 2. Visual representation of the time-unfolded IST inspired neural network.

Figure 2 shows the proposed CS-based NN architecture. Voltage data is first passed through a mapping layer (i.e., a dense layer that serves as a learned Jacobian) and then branched off to compute the residual and gradient before merging back with the mapped layer. This series of operations represent one iteration of the IST algorithm. Thresholding was implemented through a custom activation function. Although Figure 2 only shows two iterations, the number of iterations could be further increased by continuing the pattern. Incorporating additional iterations has a negligible effect on the overall NN size, since it reuses the same sets of layers with the same weights. The model used in this study was structured to have four iterations.

3.1.2. Soft thresholding activation function

In addition to the L_1-norm, the soft thresholding function used in IST also helps to promote sparsity in the output. It does so by setting values to zero if they fall within a certain threshold. The soft thresholding function can be represented as

$$ST(x) = \begin{cases} x - \tau & \text{if } x > \tau \\ 0 & \text{if } -\tau \leq x \leq \tau \\ x + \tau & \text{if } x < -\tau \end{cases} \quad (4)$$

where τ is the specified threshold hyperparameter. This function yields a graph that resembles a linear profile with a dead-zone centered at zero. As mentioned previously, this thresholding function was implemented through a custom activation function. A parametric analysis was conducted using custom metrics (i.e., comparing pixel error differences in centroid position) to determine the ideal threshold.

3.1.3. Jacobian hyperparameter

The training method proposed in this study suggests that the artificial NN can extract information about the EIT spatial sensing domain through the Jacobian in a custom loss function. There are several factors that can affect the Jacobian, such as number of electrodes, element mesh size, and injection current. The number of electrodes and elements determine the dimensions of the Jacobian, and the stimulation injection current is directly proportional to the magnitude of each value in the Jacobian. A Jacobian was obtained from Electrical Impedance Tomography and Diffuse Optical Tomography Reconstruction Software (EIDORS), an open-source software package used for EIT modeling, using a stimulation amperage of 0.10 Amps. An additional hyperparameter controlled the magnitude of the Jacobian during training [10].

3.1.4. Hyperparameter tuning

Due to the nature of compressed sensing algorithms, the "loss" from each reconstruction is not a consistent metric to evaluate reconstruction quality. For example, a higher loss reconstruction may resemble the ground truth better due to increased sparsity in the output. To properly tune the hyperparameters, two custom metrics were employed that evaluate the position, size, and relative intensity of the reconstructed anomalies. The first metric is with pixel error which can be represented as:

$$PE = \sum_{i=1}^{k} \hat{p}_k \oplus \hat{t}_k \quad (5)$$

where $\hat{p}_k$ and $\hat{t}_k$ are Booleans that represent the pixel's significance for the predicted pixel and the ground truth pixel, respectively. Pixels/elements are considered significant if they are a part of the anomaly. For the predicted pixel, significance is determined by:

$$\hat{p}_k = \begin{cases} 1 & if\ p_k \geq \frac{1}{2} \max\left(\widehat{\delta\sigma}\right) \\ 0 & if\ p_k < \frac{1}{2} \max\left(\widehat{\delta\sigma}\right) \end{cases} \quad (6)$$

where p_k is the predicted value at the corresponding pixel/element of the mesh. The ground truth pixel significance is determined by:

$$\hat{t}_k = \begin{cases} 1 & if\ t_k \neq Base\ Conductivity \\ 0 & if\ t_k = Base\ Conductivity \end{cases} \quad (7)$$

where t_k is the ground truth conductivity of the corresponding pixel/element in the mesh. Essentially, pixels/elements in the reconstruction and ground truth are binarized by significance (i.e., the element is either part of the anomaly or not) and the XOR operator was used to determine how many elements mismatch. This metric provides information regarding the reconstructed anomaly's size and position.

The second metric is evaluating the differences in centroid position between anomalies in the ground truth and reconstruction. The horizontal and vertical centroids are calculated using:

$$\bar{x} = \frac{\sum(p_k \times x)}{\sum p_k} \quad (8)$$

$$\bar{y} = \frac{\sum(p_k \times y)}{\sum p_k} \quad (9)$$

where x and y are the horizontal and vertical positions of the pixels, respectively. The difference in centroid position was then calculated with:

$$\Delta C = \sqrt{(\bar{x}_t - \bar{x}_p)^2 + (\bar{y}_t - \bar{y}_p)^2} \tag{10}$$

where $\bar{x}_t$ and $\bar{x}_p$ are the horizontal centroids of the ground truth and predictions, respectively. $\bar{y}_t$, and $\bar{y}_p$ are the vertical centroids of the ground truth and predictions, respectively. When calculating the centroids using Equations (8) and (9), it is important to subtract all elements by the base conductivity such that only anomalous regions have non-zero elements. This metric provides information regarding the reconstructed anomaly's position and relative intensity.

3.2. Traditional Deep Neural Network Model

Countless NN architectures have been proposed by researchers that utilize different types of layers. For comparison, two neural networks were trained using the traditional approach, including a simple fully-connected feed forward NN and a more complicated long short-term memory (LSTM) convolutional neural network (LSTMConvNet) model.

3.2.1. Fully-connected feed forward neural network

The simple model consists of two hidden dense layers: one with 305 nodes and one with 576. The last layer had 576 nodes to match the 24×24 output and the number of nodes in the first layer was chosen such that the total number of trainable parameters was comparable to the CS-based NN (~ 240,000 trainable parameters). This way, the efficiency of training methods can be evaluated (i.e., how much information the NN can extract for the same number of trainable parameters).

3.2.2. Encoder-decoder neural network model

The LSTMConvNet model utilized a combination of popular architectures that complement each other for EIT reconstructions [11]. The model was structured in an encoder-decoder style, where an LSTM layer of four units was used to extract important features from the input data. The purpose of the LSTM was to extract positional information since the boundary voltage inputs follow a certain order [11]. Such information is often lost when using regular dense layers [12]. The outputs of the LSTM were then flattened and passed into a dense layer of 144 units. The 144 values from the dense layer were reshaped into a 12×12 matrix and upscaled to the final size (i.e., a 24×24 matrix) via convolution and up-sampling layers. Convolutions were utilized to give the network spatial awareness when generating the reconstructions [13]. The architecture diagram of the ED model can be seen in Figure 3.

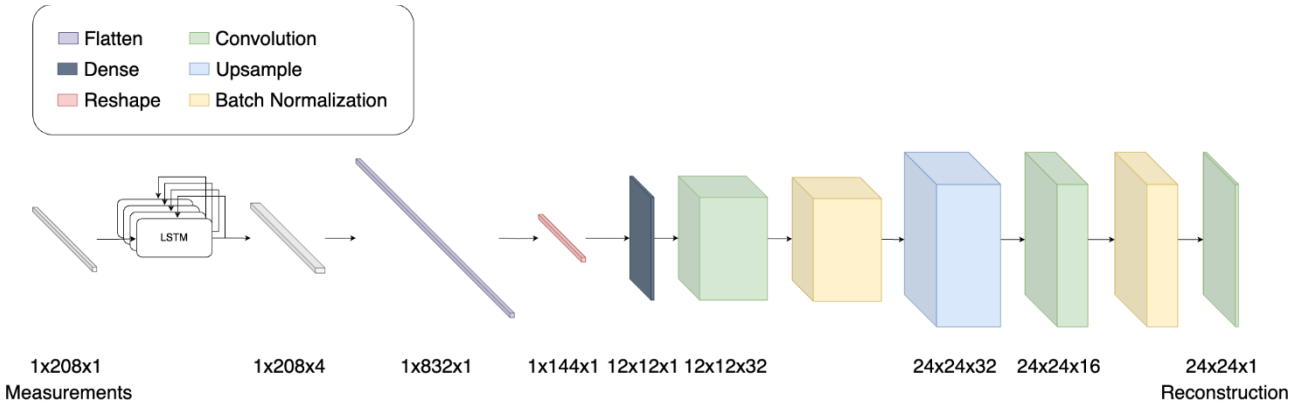

Figure 3. Visual representation of the LSTMConvNet model.

3.3. Training, Validation, and Testing Dataset

All datasets were generated using EIDORS where the sensing domain was modeled using a 24 × 24 element mesh in a 16-electrode configuration. Each datapoint consisted of 1-3 random anomalies which included circular, square, and rectangular anomalies of different sizes, positions, and conductivities. Examples of each anomaly are shown in Figure 4, where blues and reds represent regions of decreased and increased conductivity, respectively, as demonstrated by the colorbar.

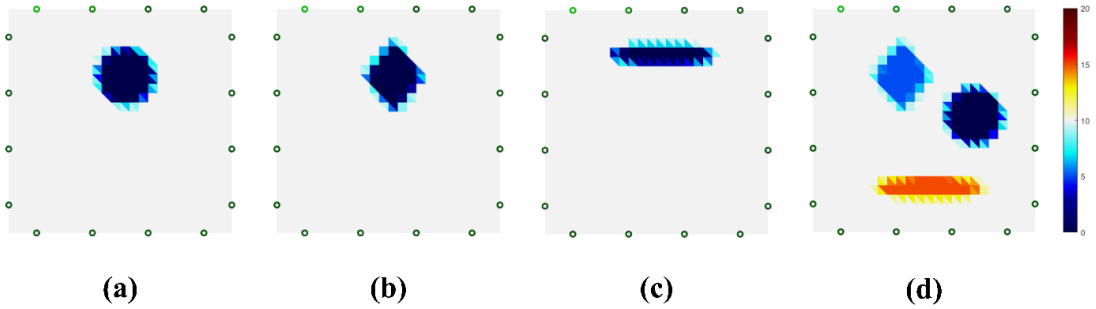

Figure 4. (a) Circular anomaly. (b) Square anomaly. (c) Rectangular anomaly. (d) Combination of all anomaly types, each with their respective conductivities.

For each anomalous scenario, the forward problem was solved to obtain the corresponding boundary voltage distribution. The relative changes in boundary voltage distribution were then determined by subtracting the baseline voltages and normalized to the largest value in the baseline measurements to establish the training data input. The actual conductivities of the anomalous models (in units of Siemens) were saved as the expected output for the traditionally trained neural networks. A total of 10,000 data points were generated and randomly portioned into a 70/20/10 split for training, validation, and testing. White gaussian noise was also added to all datapoints such that the signal-to-noise ratio of the input data was ~45 dB.

Two sets of validation data were generated: one set with only one anomaly present and the other with 1-3 anomalies, similar to the training dataset. The single anomaly dataset was used for hyperparameter tuning and the multiple anomaly dataset was used to monitor/track training losses. Two validation datasets were needed because the custom metrics mentioned previously can only be used to evaluate both types of NNs if one anomaly is present. For example, if a reconstruction contained two anomalies with different conductivities, one of 15 Siemens and the other 50 Siemens, it is possible for the 15 Siemens anomaly to be deemed as insignificant since the threshold is set with the larger 50 Siemens anomaly. In addition, both types of NNs, traditionally trained and CS-based, have different outputs. The traditionally trained NNs output actual conductivity whereas the CS-based NN outputs changes in conductivities. These discrepancies can be accounted for only if one anomaly is present in the reconstructions.

There were also two sets of testing data for the same reason mentioned previously. After NN training, the testing dataset were used to evaluate differences in loss, blur radius, and centroid position, which would provide insight into the actual performance of each NN.

4. RESULTS AND DISCUSSION

4.1. Compressed Sensing-Based Neural Network Training

The compressed sensing-based NN was trained with a 1E-4 learning rate and the Adams optimizer. With the single anomaly validation dataset, multiple training sessions were carried out to determine the optimal set of hyperparameters. For each trial, the NN was set to train for an excessive number of epochs until all metrics stabilized which helped to determine when to end the training. After identifying the optimal set of hyperparameters, training was repeated with the multiple anomaly dataset and the losses were tracked to identify potential signs of overfitting.

4.1.1. Hyperparameter tuning

Using the single anomaly validation dataset, the metrics were tracked and plotted for all training epochs. Figures 5 and 6 show how pixel error and Δ centroid changed over epochs, respectively.

First, one can observe from Figure 5 that reconstructions had the smallest pixel error when $\lambda = 0.0010$. Slightly increasing or decreasing λ by 0.0002 resulted in higher pixel error. Figure 6 shows that for all λ's, there is a bit of instability with the Δ centroid metric for the first 1250 epochs. It should be noted that other λ values ranging from 0.0001–0.0100 were also tested, but the differences in anomaly centroids were noticeably higher and more unstable (excluded from graph for clarity). To achieve a good balance between pixel error and Δ centroid, a λ of 0.0010 was determined to be the optimal hyperparameter. The same process was repeated for other hyperparameters in the NN, such as τ from the soft thresholding

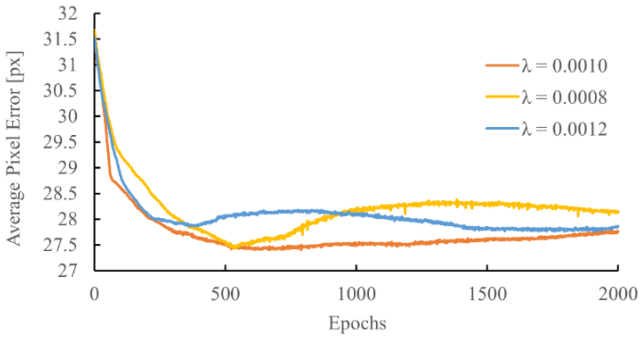

Figure 5. Average pixel error plotted over epochs with varying lambda hyperparameters.

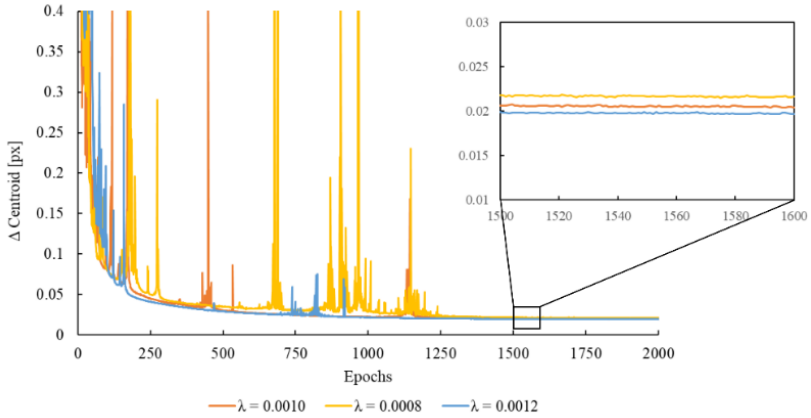

Figure 6. Average ΔCentroid plotted over epochs with varying lambda hyperparameters.

activation layer. Based on the results shown in Figures 5 and 6, all metrics appear to remain stabilize after ~1400 epochs which indicated a good stopping point to end the training.

4.1.2. Training losses

Figure 7 is a plot that tracks how the average batch loss changes over epochs for both training and validation. It was observed from Figure 7a that the validation loss (red line) nearly matched the training loss (blue line) for all epochs during

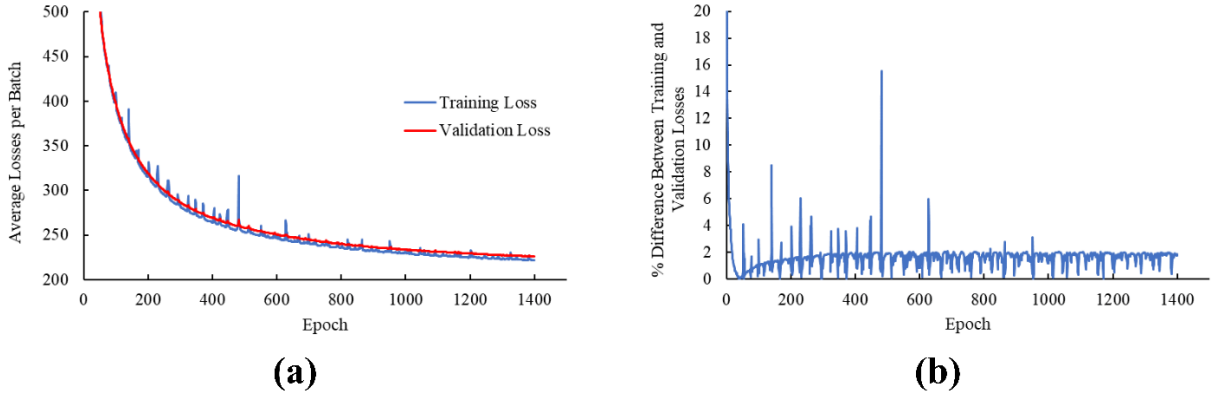

Figure 7. (a) Training and validation loss plotted over epochs for the compressed sensing-based neural network. (b) Percentage differences between training and validation losses corresponding to (a).

training. This suggested that the NN would exhibit similar performance and behaviors when presented with new data it has not seen before. Typically, with traditionally trained NNs, a diverging line is present between training and validation loss which signifies overfitting since newly extracted features only apply to the training set. Since the NN is trained to minimize an objective function, as opposed to directly matching the output to a finite set of data, this trend of matching losses was expected.

4.2. Traditional Neural Network Training

For both the simple dense model and the LSTMConvNet model, training was carried out with the Means Squared Error (MSE) loss function along with a 1E-3 learning rate and the Adams optimizer. The dense layer in the LSTMConvNet used an L_2 regularization with a factor of 0.005 to minimize overfitting. Note that other factors were also tested, but 0.005 appeared to work best. The L_2 regularization penalized larger weights and thus encouraged a simpler model that generalized and created clearer reconstructions [14]. Furthermore, the decoder part of the LSTMConvNet utilized Batch Normalization after each convolution, which standardized the inputs to a layer for each mini batch [13]. This helps the neural network learn more efficiently, often needing fewer epochs. It also helps to prevent the model from relying too much on specific features, which can reduce the necessity for implementing other techniques to prevent overfitting (e.g., dropout).

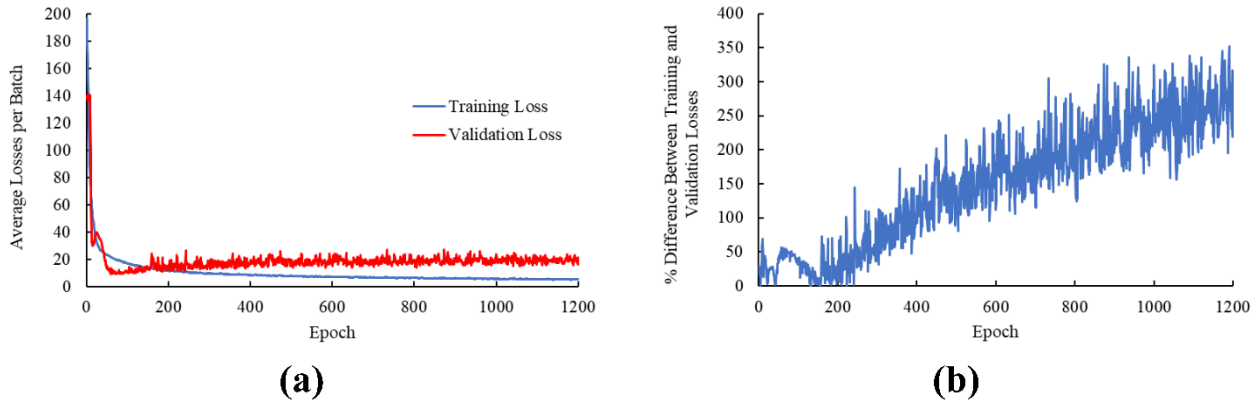

Figure 8. (a) Training and validation loss plotted over epochs for the LSTMConvNet. (b) Percentage differences between training and validation losses corresponding to (a).

As seen in Figure 8, the training and validation losses begin to diverge after ~170 epochs. The validation loss plateaued and remained mostly stationary while training loss continued to decrease steadily. This makes intuitive sense, since the model depends on the MSE loss function which fits to the training data. Training was terminated after 1200 epochs because there was no more improvement with either the training or validation loss.

4.3. Loss and Metrics

The training losses and metrics are summarized in Table 1. For the traditionally trained neural networks (i.e., trained with MSE), validation and testing losses are noticeable greater than the training loss. As mentioned previously, this indicated overfitting, since the NN is extracting information and patterns only present in the training dataset. However, with the CS-based NN, this is not the case. In fact, the loss computed from the testing dataset shows even lower losses compared to training loss. This is likely the case because patterns are extracted from the custom loss function, not the data.

From the results tabulated in Table 1, it is predicted that reconstructions from the CS-based NN will accurately position the anomaly with good relative intensity, as indicated by the low ΔCentroid. However, there will likely be discrepancies in the sizes of the reconstructed anomalies as indicated by the larger pixel error. With the ED model, it is predicted to accurately match the sizes of anomalies, indicated by low MSE loss and pixel error, but there may be discrepancies regarding the relative intensity of reconstructed anomalies. In other words, the overall size of the anomalies will likely

match, but pixels/elements of great intensity may appear scattered or concentrated in specific regions inconsistent with the ground truth.

Table 1. Performance metrics for all neural networks after training.

	Metric						
	Losses			Pixel Error [px]		ΔCentroid [px]	
	Training	Validation	Testing	Validation	Testing	Validation	Testing
Compressed Sensing-Based Neural Network	221.76*	225.85*	202.75*	27.5	27.7	0.0207	0.0219
Simple Dense Layered Neural Network	48.06**	53.89**	55.12**	28.07	29.42	0.1885	0.2655
LSTMConvNet	5.14**	19.36**	41.03**	25.3	28.82	0.2277	.1506

* - Evaluated w/ custom loss function

** - Evaluated w/ Mean Squared Error loss function

4.4. Reconstruction Results

4.4.1. Simulated data

To evaluate the performance of all reconstruction algorithms, a representative dataset was generated, and noise was added to achieve a more reasonable signal-to-noise (SNR) ratio of ~ 86 dB. The reconstruction results from all algorithms are shown in Figure 9.

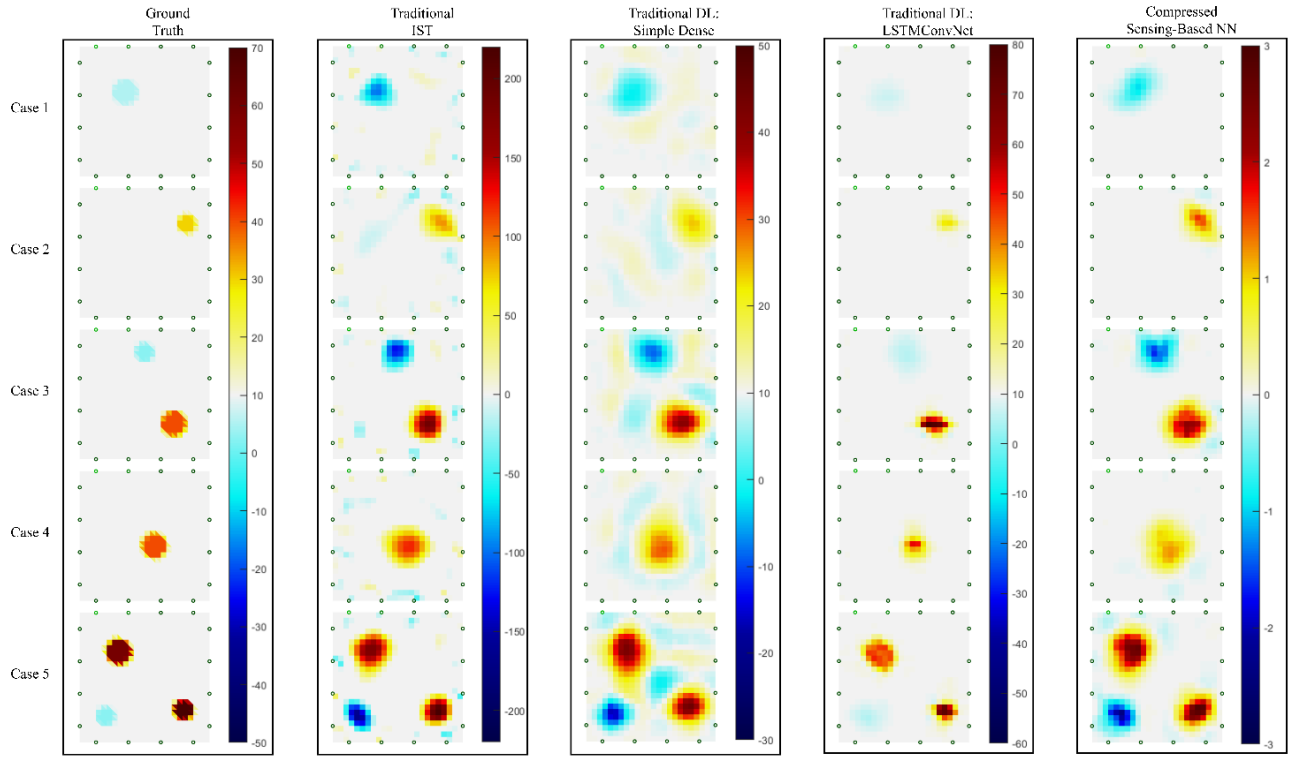

Figure 9. Comparison between reconstruction algorithms for a representative set of data.

4.4.1.1. Traditional IST

The traditional IST algorithm was set for 5000 iterations with a λ of 0.001 (i.e., the same λ used in the CS-based NN). Minor artifacts were present and scattered throughout the reconstruction, which were primarily due to the noise added. Using noise free voltage measurements from the same set of ground truth scenarios showed little to no artifacts (results are not shown here to better justify the comparison).

4.4.1.2. Simple dense neural network

The simple, fully-connected, feed-forward NN did a decent job generalizing the patterns for EIT reconstructions, but the outputs lacked sparsity. In other words, one may still be able to identify the anomalies, but the reconstruction resolution was quite low. Outputs from this NN resembled traditional IST reconstructions with τ set too low. Significantly more artifacts were present compared to traditional IST.

4.4.1.3. Encoder-decoder neural network

The LSTM's and convolution layers in the LSTMConvNet model were able to extract more patterns in the voltage measurements, leading to clearer reconstructions. It should be noted that it was able to do so while having nearly half of the trainable parameters (~125k). This suggested that LSTM's and convolution layers are much more effective and efficient at identifying features/patterns crucial for EIT reconstruction compared to dense layers. As indicated by the low MSE error, this NN did an effective job matching the output to the supposed ground truth.

4.4.1.4. Compressed sensing-based neural network

Compared to traditional IST, the CS-based NN showed arguably better denoising performance, as less artifacts were present in the reconstructions. Given that this was never exposed to the expected outputs in the dataset, these results show that it is possible to train a NN for EIT reconstructions using the CS objective function as a loss function. An additional benefit to using a CS-based NN over the traditional IST algorithm is reconstruction speed. Traditional IST took ~ 0.57 s for each reconstruction (using MATLAB), whereas the CS-based NN took ~ 0.053 s (using Python). Considering differences in programming languages, the stated times are not a perfect comparison, but the NN was still an order of magnitude faster.

4.4.2. Experimentally data

Previously, the performances of all NNs have been evaluated with simulated data from EIDORS. The performances shown in Figure 9 are not necessarily representative of how they would perform with experimental data collected by a data acquisition (DAQ) system. All reconstruction algorithms were evaluated using experimental data previously gathered with a wireless, portable DAQ system developed by the authors. Electrically conductive ultra-high molecular weight polyethylene (UHMWPE) was used as the sensing material and holes were cut to introduce regions of high impedance. The reconstructed results are shown in Figure 10.

There are significantly more factors at play that make reconstructing experimental data challenging. For example, there could be slight discrepancies in electrode spacing and imperfections within the sensing domain that leads to data exhibiting patterns not representable by white Gaussian noise. Based on Case 1 in Figure 10, both the traditional IST and CS-based NN were able to reconstruct the four anomalies in the corners of the UHMWPE. The simple dense network struggled to capture the bottom right anomaly. The LSTMConvNet was able to slightly identify regions of decreased conductivity in the corners, but the overall reconstruction was dominated by a supposed anomaly in the left-center of the sensing domain. However, Case 2 was challenging for all NNs. There was still some resemblance between the traditional IST and CS-based NN. All NNs performed reasonably well for Case 3. The simple dense and CS-based NNs had comparable results to traditional IST and the ground truth. The LSTMConvNet was also able to capture the three anomalies, but there were some discrepancies with the anomalies' relative intensity. Results from Figure 10 show that good performance with simulated data does not necessarily translate into good performance with experimental data, which is especially true for traditional NNs that heavily rely on the training data.

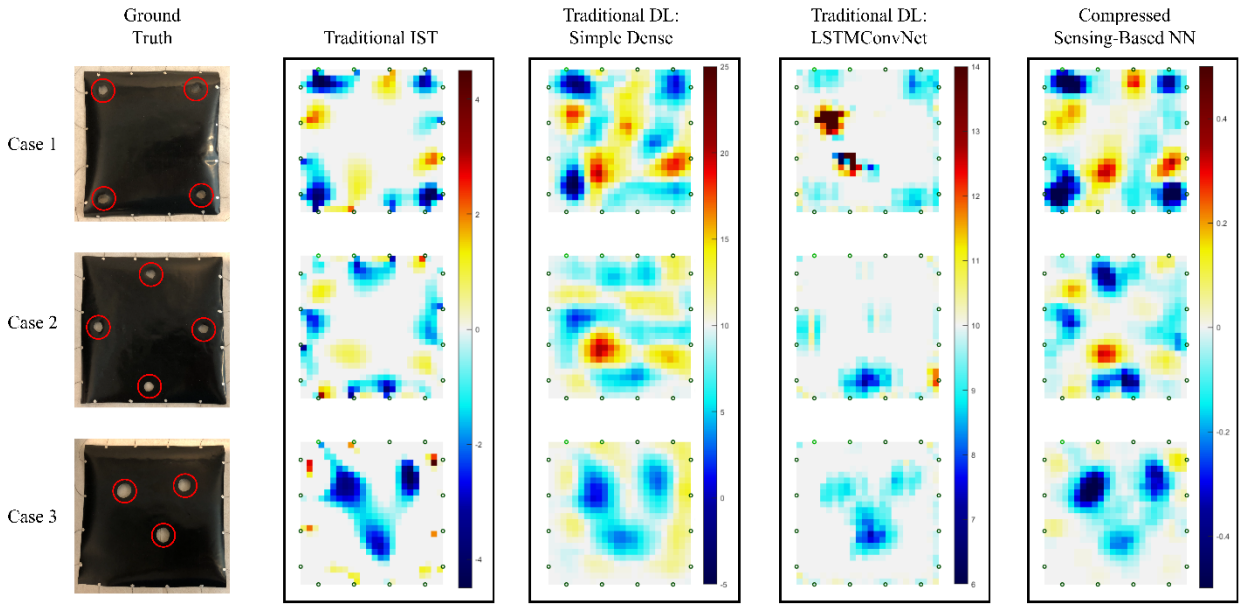

Figure 10. Reconstruction results from all algorithms using experimentally collected data.

4.5. Implications

The results from Figures 7-10 seem to suggest that the unconventionally trained CS-based NN was successful in extracting information from the sensing domain through the Jacobian in the custom loss function. The CS-based NN was able to achieve similar performances compared to the traditional IST while being an order of magnitude faster. It was also more robust compared to traditionally trained NNs as evident by the reconstructions with experimental data. There are, of course, drawbacks to this unconventional training methodology. The sizes of reconstructed anomalies often appear larger than the ground truth which is where the LSTMConvNet tends to perform well. An interesting NN architecture to test would be a combination of the CS-based NN with the LSTMConvNet. For example, the CS-based NN would be used to generalize the sensing domain convolutional/upscaling layers could be used to further refine the reconstruction. The loss function would be the weighted sum from Equations (2) and (3). This way, the resulting NN has the advantage of extracting patterns from the Jacobian, and the reconstruction resolution and quality can be improved via MSE loss.

5. CONCLUSION

This study proposed that aspects from the compressed sensing technique can be leveraged by deep learning to strategically train a neural network for EIT reconstructions. The CS-based NN shares features from compressed sensing, most notably the activation functions, objective/loss function, and the architecture. Two traditionally trained NNs (i.e., trained with the commonly used MSE loss function) of varying complexity served as baselines for comparison. Sets of training, validation, and training data were generated via EIDORS. The hyperparameters in the CS-based NN were tuned using two custom metrics that evaluated the size, position, and relative intensity of reconstructed anomalies. Results show that the CS-based NN could successfully extract patterns from the custom loss function rather than the training data. When trained with the custom loss function, the resulting NN exhibited similar performance with new data (i.e., achieved similar metrics with the testing dataset). This is usually not the case with traditionally trained NN as indicated by the differences in loss between training, validation, and testing data. The LSTMConvNet showed that LSTMs and convolutional layers were much more efficient in extracting patterns for EIT. Future research should consider combining the CS-based NN architecture with upscaling convolutional layers. The resulting NN could then extract patterns and information regarding the sensing domain through the Jacobian and the reconstruction resolution could be refined by means of convolutional layers and MSE loss.

6. ACKNOWLEDGEMENT

This study was supported by the United States National Science Foundation (grant no. 2138756).

7. REFERENCES

1. Hallaji, M. and M. Pour-Ghaz, *A new sensing skin for qualitative damage detection in concrete elements: Rapid difference imaging with electrical resistance tomography.* NDT & E International, 2014. **68**: p. 13-21.
2. Hong, S., et al., *A 10.4 mW Electrical Impedance Tomography SoC for Portable Real-Time Lung Ventilation Monitoring System.* IEEE Journal of Solid-State Circuits, 2015. **50**(11): p. 2501-2512.
3. Zou, Y. and Z. Guo, *A review of electrical impedance techniques for breast cancer detection.* Medical Engineering & Physics, 2003. **25**(2): p. 79-90.
4. Zhang, T., et al., *Advances of deep learning in electrical impedance tomography image reconstruction.* Front Bioeng Biotechnol, 2022. **10**: p. 1019531.
5. Wang, L., *Development of multifunctional nanocomposite sensing systems for structural and human health monitoring*, in *Structural Engineering*. 2019, University of California San Diego: San Diego, California. p. 161.
6. Zhao, Y., et al., *Comparison of electrical impedance tomography inverse solver approaches for damage sensing*, in *Smart Structures and Materials + Nondestructive Evaluation and Health Monitoring*. 2017, Society of Photo-Optical Instrumentation Engineers (SPIE): Portland, Oregon, United States.
7. Chollet, F., *Deep Learning with Python*. 2017, New York, NY: Manning Publications.
8. Yuwei Fan and Lexing, Y., *Solving electrical impedance tomography with deep learning.* Journal of Computational Physics, 2020. **404**: p. 109119.
9. Gregor, K. and Y. LeCun, *Learning fast approximations of sparse coding*, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*. 2010, Omnipress: Haifa, Israel. p. 399–406.
10. Adler, A. and W.R. Lionheart, *Uses and abuses of EIDORS: an extensible software base for EIT.* Physiol Meas, 2006. **27**(5): p. S25-42.
11. Machidon, A.L. and V. Pejović, *Deep learning for compressive sensing: a ubiquitous systems perspective.* Artificial Intelligence Review, 2023. **56**(4): p. 3619-3658.
12. Staudemeyer, R.a.M.E., *Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks.* 2019.
13. Neyshabur, B., *Towards learning convolutions from scratch*, in *Proceedings of the 34th International Conference on Neural Information Processing Systems , articleno = 677 , numpages = 11*. 2020, Curran Associates Inc.
14. *Regularization for Simplicity: L_2 Regularization*. Google for Developers 2022.