Domain independent post-processing with graph U-nets: Applications to Electrical Impedance Tomographic Imaging

William Herzberg,
Department of Mathematical and Statistical Sciences; Marquette University, Milwaukee, WI 53233 USA

Andreas Hauptmann [Senior Member, IEEE],
Research Unit of Mathematical Sciences, University of Oulu, Finland and also with the Department of Computer Science, University College London, UK.

Sarah J. Hamilton
Department of Mathematical and Statistical Sciences; Marquette University, Milwaukee, WI 53233 USA

Abstract

Objective: To extend the highly successful U-Net Convolutional Neural Network architecture, which is limited to rectangular pixel/voxel domains, to a graph-based equivalent that works flexibly on irregular meshes; and demonstrate the effectiveness on Electrical Impedance Tomography (EIT).

Approach: By interpreting the irregular mesh as a graph, we develop a graph U-Net with new cluster pooling and unpooling layers that mimic the classic neighborhood based max-pooling important for imaging applications.

Main Results: The proposed graph U-Net is shown to be flexible and effective for improving early iterate Total Variation (TV) reconstructions from EIT measurements, using as little as the first iteration. The performance is evaluated for simulated data, and on experimental data from three measurement devices with different measurement geometries and instrumentations. We successfully show that such networks can be trained with a simple two-dimensional simulated training set, and generalize to very different domains, including measurements from a three-dimensional device and subsequent 3D reconstructions.

Significance: As many inverse problems are solved on irregular (e.g. finite element) meshes, the proposed graph U-Net and pooling layers provide the added flexibility to process directly on the computational mesh. Post-processing an early iterate reconstruction greatly reduces the computational cost which can become prohibitive in higher dimensions with dense meshes. As the graph structure is independent of 'dimension', the flexibility to extend networks trained on 2D domains to 3D domains offers a possibility to further reduce computational cost in training.

Keywords

conductivity; electrical impedance tomography; finite element method; graph convolutional networks; unet; post-processing; deep learning

1. INTRODUCTION

Nonlinear inverse problems are often described by partial differential equations (PDEs) and measurements are taken directly on the boundary of the domain, resulting in varying domain shapes [1]. Consequently, reconstruction algorithms need to offer the flexibility to operate on these varying domains. The finite element method (FEM), and in particular the corresponding meshes, offers this flexibility with respect to domain shapes and hence the tomographic image is usually computed on a target specific mesh. A popular class of reconstruction algorithms for this task are optimization-based variational methods [2], where the reconstructions are iteratively updated on the mesh by some optimization algo-rithm, such as a Gauss-Newton type method. Unfortunately, these methods tend to be expensive due to costly Jacobian computations resulting in a tradeoff between cost and image quality for increasing iterations. Additionally, reconstructions can be sensitive to modeling of the domains or measurement devices, potentially causing severe reconstruction artifacts [3].

One way to improve the image quality from an early iterate is to perform a post-processing step to provide image quality comparable to the full iterative algorithm, but with a substantial reduction in computational cost. For this specific task Deep Learning approaches have been immensely popular in recent years [4]. Here, given the suboptimal reconstruction from an early iterate one trains a neural network with representative training data to produce an improved reconstruction. For this specific purpose of post-processing, U-Net architectures [5], [6] have been immensely successful. U-nets use a multi-scale convolutional neural network (CNN) to process images on multiple resolutions by extracting edge information as well as long range features. The main limitation of CNN U-nets lies in the strict geometric assumptions on the mesh, i.e., the application of the convolutional filters requires a regular rectangular mesh with ideally isotropic pixel dimensions, preventing their direct application to the aforementioned reconstruction problem on FE meshes. A simple remedy would be to interpolate between the FE meshes and the rectangular grid for application of the CNN, losing the flexibility that FEM provides [7].

Alternatively, to retain the flexibility one can interpret the FE mesh as a graph and process the image directly using graph convolutional neural networks [8]. Here we propose an extension of the CNN based U-Net architecture to a graph U-Net. New graph-based pooling operations are required to move between the multiple resolutions of a U-net such that the local relations are preserved. To achieve this we propose a cluster based pooling and unpooling that provides comparable down and up-sampling to the classic max, or average, pooling on CNNs. For computational feasibility, the clusters are pre-computed for each mesh and can be efficiently applied during training and inference. The proposed graph U-Net is then applied in the context of electrical impedance tomography (EIT) a highly nonlinear inverse problem that requires strong regularization to obtain good image

quality. In this work, we compute an initial reconstruction with only a few iterations of total variation regularized Gauss-Newton method and then train the network to improve image quality providing excellent reconstruction quality with a considerable reduction in processing time.

The primary advantage of a graph U-net lies in the flexibility with respect to measurement domains. We note that, while the initial reconstructions still require the careful modeling of each device and domain, the networks can be trained independently on general, simple, measurement setups. This overcomes drawbacks, 1) the creation of general enough training data is a time intensive task and 2) we cannot predict all possible encountered domains in the measurement process, e.g., varying chest shapes for physiological measurements of human subjects [9]. Furthermore, the graph based nature of the network is dimension independent as neighboring nodes are described by a dimension independent adjacency matrix. This allows one to train the network in 2D and apply it to measurements from 3D domains.

This paper addresses the challenging task to process EIT reconstructions from a diverse set of measurement devices with a single network trained in a single 2D chest-shaped measurement domain with elliptical inclusions. We can show that the network successfully generalizes to measurement data under varying domain shapes as well as to reconstructions from three different EIT devices (KIT4, ACT3, and ACT5). The data from the KIT4 device [10], [11] was taken on a chest-shaped tank with 16 electrodes using bipolar current injection whereas the ACT3 data was taken on a circular tank with simultaneous injection/measurement across all 32 electrodes [12], [13]. Both the KIT4 and ACT3 datasets corresponded to 2D cross-sectional imaging. However, the ACT5 datasets [14], [15] used a fully 3D box geometry, with 32 large electrodes and simultaneous current injection. We emphasize that all tests are performed on either simulated or experimental tank data, not human data and the clinical viability of these methods is outside the scope of this current work.

This work is the first to show that a single network can be successfully used in a variety of instrumentation and measurement setups, combining the flexibility of FEM with graph neural networks (GNN) and thus making deep learning techniques more accessible to inverse problems and imaging applications that heavily rely on the use of FE meshes. For this purpose, we also provide a code package GN4IP[1]. It should be noted that this paper reuses some content from thesis [16], with permission. The target audience of this manuscript is readers with some familiarity with neural networks and deep learning. References [4], [17], [18] help provide such background knowledge.

The remainder of the paper is organized as follows. Section 2 develops the graph U-net and novel pooling layers, a brief overview of EIT, the experimental data, training data, and metrics to be used to assess reconstruction quality. Results are given in Sec. II and discussion follows in Sec. IV where modifications and extensions are explored, including the 3D data with reconstructions from different algorithms than the network was trained on. Conclusions are drawn in Sec. V.

---

[1]Graph Networks for Inverse Problems (GN4IP) is available at github.com/herzberg/GN4IP

II. METHOD

A. Learned Reconstruction

The main focus of this work is post-processing tasks, however the modified graph U-net and new pooling layers could easily be used in place of a residual network in a model-based learning framework such as [8]. The problem at hand is to improve a fast, reliable image that has predictable artifacts that could be removed via post-processing. Post-processing EIT reconstructions with the traditional U-net architecture was shown highly effective for d-bar based reconstructions [19], [20] and the dominant current scheme [21], but each of those works applied the networks to rectangular pixel image data, despite measurements obtained on different domains. However, many image reconstruction methods are performed on irregular meshes (e.g. FE meshes), which then require the solution (reconstructed image) to be interpolated from the computational mesh to the pixel grid. In some large scale cases, this may be cost-prohibitive or less desirable when solutions are needed at high precision. Thus, it is of interest to have an alternative network structure for learned reconstruction (e.g. post-processing and model-based learning) on the computational mesh.

Graph Neural Networks have been around for years and recently have garnered renewed interest for their scalability and data flexibility. As with traditional CNNs, several options for convolutional and pooling layers exist. However, at the time of this work, the existing pooling layers, in particular were inadequate for mirroring the maxpooling in classic CNN U-nets. Therefore, we developed new layers here based on spatially clustering neighboring nodes in the computational mesh and then performing the maxpooling over the clusters. The structure of the modified graph U-net used here is shown in Figure 1.

A graph U-net with cluster pooling: As the name suggests, graph networks act on *graph* input data. Historically, graph convolutional networks have been applied to citation networks, social networks, or knowledge graphs. However, whenever you have a computational mesh, you inherently have a graph representing the connections between the elements or nodes in your computational mesh. In particular, for irregular meshes commonly associated with FEM there are two natural options for the associated graph: the mesh elements, or the mesh nodes (Fig. 2). The *adjacency matrix* $\mathscr{A}$ is a sparse matrix listing the edges (connections) between the graph nodes. The basic form of the sparse adjacency matrix is $\mathscr{A}_{ij} = 1$ if there is an edge connected nodes n_i and n_j. Self-loops are recorded separately.

As opposed to convolutional layers which require regular n-dimensional regular input data, *graph convolutional* layers are designed to work on simple, homogeneous graph-type data. While the list of graph convolutional layers is ever expanding, we used the layer proposed by Kipf and Welling [22] designed to be analogous to the CNN setting:

$$H^{(i+1)} = \widetilde{D}^{-\frac{1}{2}}(\mathscr{A} + I)\widetilde{D}^{-\frac{1}{2}}H^{(i)}W^{(i)}, \tag{1}$$

where the inputs to the layer are the graph's feature matrix $H^{(i)} \in \mathbb{R}^{N \times f(i)}$ and adjacency matrix $\mathscr{A} \in \mathbb{R}^{N \times N}$, and the output is a new feature matrix $H^{(i+1)} \in \mathbb{R}^{N \times f(i+1)}$ for the graph with the same structure [22]. Self loops, or edges between a graph node and itself,

are represented by the identity matrix I, of the same size as $\mathscr{A}$, and are added to the adjacency matrix. Then, that sum is multiplied on the left and right by $\widetilde{D}^{-\frac{1}{2}}$ to account for the number of edges each node has. The diagonal matrix $\widetilde{D}$, which does not contain trainable parameters and is only determined by $\mathscr{A}$, is defined by $\widetilde{D}_{ii} = 1 + \sum_j \mathscr{A}_{ij}$. Finally, multiplication of the scaled adjacency matrix $\widetilde{D}^{-\frac{1}{2}}(\mathscr{A} + I)\widetilde{D}^{-\frac{1}{2}}$ by the input feature matrix $H^{(i)}$ aggregates information within local neighborhoods and multiplication by the weight matrix $W \in \mathbb{R}^{f^{(i)} \times f^{(i+1)}}$ takes linear combinations of the aggregated features to form the output features.

Bias parameters $b \in \mathbb{R}^{f^{(i+1)}}$ can be included in a graph convolution by adding them to the output feature vector of each node (each row of the output feature matrix). The weight matrix W (and optional bias vector) are the trainable parameters that are optimized during training. One significant difference between convolutional layers and graph convolutional layers is in how they aggregate information within a pixel or node's neighborhood. Convolutional layers learn linear aggregation functions via the kernel parameters while graph convolutions aggregate information according to a fixed linear function, $\widetilde{D}^{-\frac{1}{2}}(\mathscr{A} + I)\widetilde{D}^{-\frac{1}{2}}$, determined by the graph's adjacency matrix $\mathscr{A}$. The non-learned aggregation function of such a graph convolution has raised questions of the learning capacity of graph convolutional layers [23]. Despite those concerns, graph convolutional layers have been used successfully for a variety of graph and node classification tasks [22], [24]. GCNs have also been used for model-based learning directly on irregular mesh data [8]. In the down-sampling path, graph convolutional layers are used like in the original graph U-net [6] and in previous work on images represented as graphs [8].

After graph convolutions, pooling layers are used to move down and up the U-net. Several node-dropping, hierarchical pooling layers (layers that gradually coarsen a graph by removing nodes) for GNNs have been proposed including self-attention graph pooling [25], adaptive structure aware pooling [26], and gPool [6]. The gPool layer selects graph nodes to preserve by first learning a projection of nodes' feature vectors. For each node, its feature vector is projected by a vector $p \in \mathbb{R}^{(i)}$ to return a scalar score. Then, the nodes with the top projection scores are preserved. The adjacency matrix is also sliced to preserve only the rows and columns for the preserved nodes. The parameters in the projection vector p are optimized during training.

One significant difference between the gPool layer and the max-pooling layer used in CNNs is that the gPool layer performs global node selection while the max-pooling layer performs local selection. That is, the max-pooling layer only considers features within a subregion or window of the input when preserving pixel data, while the gPool layer considers the projection scores from the entire graph when deciding which nodes to preserve. Self-attention graph pooling, adaptive structure aware pooling, and other node-dropping, hierarchical pooling layers also consider the whole graph when selecting which nodes to preserve, which gives way to the possibility that entire regions of a graph could be discarded

when pooling [27]; something that is not possible with CNN-based max-pooling. Therefore a new graph pooling layer, the *k-means cluster (kMC) max pool layer*, was developed with local pooling in mind.

To create local windows in the input graph, the k-means++ algorithm[2] [28] was used to cluster the graph nodes. The inputs to the k-means++ algorithm include the locations of the N graph nodes and the number of clusters N_c desired. The output of the algorithm was the cluster assignments $c \in \{1, ..., N_c\}^N$ and the locations of the clusters. Locations of the input nodes are determined from the FE mesh that the input graph was representing. That is, for data on the mesh nodes, the locations of those nodes are used, and, for data on the elements, the element centroid locations can be used. The locations of the output clusters are taken as the centroid of the graph nodes in the cluster. As the k-means++ algorithm is stochastic, it was repeated multiple times, and the cluster assignments with the minimum within-cluster spacing selected.

With clusters defined, max-pooling is performed within each cluster along each feature of the input graph. Therefore, the output of the pooling operation is a set of N_c nodes at the cluster locations and with the maximal features from the input nodes within each cluster. The adjacency matrix of the output graph is determined by the cluster assignments as well. If any edge connected two input nodes assigned to separate clusters, an edge was drawn between the output nodes representing those clusters. Figure 3 (top) depicts the structure of the input and output graphs of the kMC max pooling operation.

In encoder-decoder type GNNs that have a down-sampling path of node-dropping, hierarchical pooling layers and a symmetrical up-sampling path, the up-sampling often uses the gUnpooling layer [6], [27]. Nodes and edges that were removed in the associated pooling layer are restored in the gUnpooling layer. The features of the restored nodes are set to zero. There are no trainable parameters in the gUnpool layer. The gUnpool layers could not be used here as pairing it with the k-means cluster max pool layer, would result in the output graph having all nodes' features set to zero because all output nodes are restored. Like the gUnpool layer, a new *clone cluster unpool layer*, shown in Figure 3 (bottom) was designed to restore the pre-pooled graph structure. Instead of returning the previously removed nodes with features set to 0, the nodes are restored with feature vectors equal to the node representing the cluster to which the nodes belong. That is, each output node is restored as a clone/copy of the input node representing the cluster including the output node. The proposed layer was developed to act like a nearest-neighbor up-sampling layer, or a transpose convolutional layer with equal stride and kernel size and parameters fixed to 1

For each graph, the adjacency matrix and downsampled clustering assignments can be computed offline prior to the training or processing through the network. The resulting graph U-net Λ_θ takes as inputs $(x_{in}, \mathcal{A}, \mathbf{c})$ where $\mathbf{c} = (c_1, ..., c_{N_p})$ are the cluster assignments for the N_p pooling (and unpooling) layers in the network. The post-processed image $\hat{x}$ is then the output of

---

[2]The MATLAB R2021b implementation kmeans() was used.

$$\hat{x} = \Lambda_\theta(x_{in}, \mathscr{A}, \mathbf{c}). \tag{2}$$

Note that each cluster assignment vector, c_j, is of a different length, and $\mathbf{c}$ represents the list of vectors. Figure 1 provides a diagram of the proposed graph U-net with $N_p = 3$ pooling layers. The *k-means cluster max pooling* and *clone cluster unpooling* layers used in this graph U-net allow the cluster assignments to be mesh specific and computed prior to training or predicting. In the experiments conducted, no problems were noticed by training or prediction samples having different meshes and cluster assignments. Computing cluster assignments can take several minutes depending on the number of input nodes, clusters, and repetitions, thus computing them ahead of time keeps both training and prediction fast. In addition, the number of clusters used at each pooling layer can be tuned similarly to how the kernel size of a CNN max pooling layer can be adjusted. Overall, the new proposed graph pooling layer was a fast and flexible alternative to the existing graph pooling layers, and it behaves more like the traditional max pooling layer used in convolutional U-nets. The proposed clone cluster unpooling layer works naturally with the cluster-based pooling in the graph U-net architecture. Alternatively, regional downsampling of the graph and adjacency matrix could be performed via mesh coarsening as in [29].

Training: Training data consists of $\{x_{recon}^i, x_{true}^i\}$ pairs where xi recon is x_{recon}^i produced via the user's chosen solution method on measurement data y^i. The computational mesh associated to x_{recon}^i leads to an adjacency matrix $\mathscr{A}^i$ and corresponding cluster assignments ci. Then, the network is trained using x_{recon}^i, A^i, and $\mathbf{c}^i$ as inputs and x_{true}^i as the desired outputs. A loss function involving the network outputs $\hat{x}^i$ and the known true images x_{true}^i can be used. Either MSE or an ℓ^1 loss function are natural choices. One could also weight the loss function differently for each sample or for each graph node. Once the training of the network is complete, the parameters $\hat{\theta}$ are saved for use in the online prediction stage.

Note that training data could have all different computational meshes, all the same, or any mix thereof. In this work, a single reconstruction mesh was used per network trained to demonstrate a simple case. Note that this does not restrict later passing reconstructions on different meshes through the trained network (i.e. a network trained on reconstructions from mesh^j, adjacency matrix $\mathscr{A}^j$, with clusters $\mathbf{c}^j$ is not restricted only to meshj as the adjacency matrix and clusters are inputs to the network). Several test cases are presented in Section III demonstrating the flexibility of the networks to input data coming from different domain shapes, experimental setups, and even higher dimensional data (3D when trained on 2D).

In the online prediction stage, an updated reconstruction $\hat{x}$ is estimated from measurement data y by first computing an initial reconstruction x_{recon} from y and passing x_{recon}, its adjacency matrix $\mathscr{A}$, and corresponding cluster assignments $\mathbf{c}$ through the trained network using (2) where the trained parameters $\hat{\theta}$, which were saved during the offline stage, are used in the network.

B. Case Study: Electrical Impedance Tomography

EIT is an imaging modality that uses electrodes attached to the surface of a domain to inject harmless current and measure the resulting electrical potential. From the known current patterns and resulting measured potential, the conductivity distribution of the interior of the domain can be estimated [30]. The mathematical problem of recovering the conductivity is a severely ill-posed inverse problem as large changes in the internal conductivity can present as only small changes in the boundary measurements [31], [32].

The recovered conductivity distribution can be visualized as an image and/or useful metrics extracted. Applications of EIT are wide-ranging from nondestructive evaluation to several medical imaging applications (see [33], [34] for a more comprehensive list). Here we focus on absolute, also called static, EIT imaging which uses recovers the static conductivity at the time the data was collected from a single frame of experimental data. Absolute/static imaging is important in applications such as nondestructive evaluation, breast cancer, or stroke classification where a pre-injury/illness dataset is unavailable. Alternatively, time-difference EIT imaging recovers the change in conductivity between two frames of data. Such time-difference data is useful in monitoring settings such are thoracic imaging of heart and lung function or stroke monitoring. Commercial EIT systems for monitoring heart and lung function are available and used in Europe and South America. Alternatively, frequency sweep data can be used in absolute imaging scenarios or difference imaging scenarios to further identify tissue based on the electrical properties and how they change with the frequency of the applied current.

The conductivity equation [30]

$$\nabla \cdot \sigma(x) \nabla u(x) = 0 \quad x \in \Omega \subset \mathbb{R}^n, \quad n = 2, 3, \tag{3}$$

models the relationship between the electric potential $u(x)$ and conductivity $\sigma(x)$ in a domain $\sigma \subset \mathbb{R}^n$ with Lipschitz boundary. In the *forward problem* of EIT, the voltage measurements at the electrodes are simulated for a known current pattern T and bounded conductivity distribution $0 < c \leq \sigma(x) \leq C < \infty$ for some constants c and C. Boundary conditions are given by the Complete Electrode Model (CEM) [35] which takes into account both the shunting effect and contact impedance when modeling the electrodes. The CEM is given by

$$
\begin{aligned}
\int_{e_\ell} \sigma \frac{\partial u}{\partial v} dS &= T_\ell, \ \ell = 1, 2, ..., L, \\
\left(u + z_\ell \sigma \frac{\partial u}{\partial v} \right)\Big|_{e_\ell} &= U_\ell, \ \ell = 1, 2, ..., L, \\
\sigma \frac{\partial u}{\partial v}\Big|_{\partial\Omega / \cup e_\ell} &= 0, \quad \ell = 1, 2, ..., L,
\end{aligned}
\tag{4}
$$

where L denotes the number of electrodes, e_ℓ the ℓth electrode; z_ℓ, T_ℓ, and U_ℓ, are the contact impedance, current injected, and electric potential on the ℓth electrode, respectively; and v is the outward unit vector normal to the boundary. Furthermore, ensuring $\sum_{\ell=1}^{L} T_\ell = 0$ and

LP $\sum_{\ell=1}^{L} U_\ell = 0$ enforces conservation of charge and guarantees existence and uniqueness [35]–[37].

The *inverse problem*, determining the interior conductivity distribution $\sigma \in \Omega$ that led to the measured voltages for the known applied current patterns, was solved using the well-established Total Variation (TV) method. The total variation of a discrete conductivity distribution

$$TV(\sigma) = \sum \left| \mathbf{L}\sigma \right|, \tag{5}$$

is often computed using the sparse difference matrix $\mathbf{L}$ which approximates the gradient of the conductivity distribution. It has one row $\mathbf{L}_i \in \mathbb{R}^{N_M}$ for each edge segment separating two elements in the mesh with N_M elements. Each row of $\mathbf{L}$ has two nonzero elements; d_i and $-d_i$ are the entries in the columns n_i and m_i for the ith edge segment with length di that separates mesh elements n_i and m_i. TV regularized methods often use a smoothed approximation of (5) to simplify the minimization task of the absolute value term by making it differentiable. In this work TV regularization is implemented by solving an optimization problem to obtain the iterate

$$\sigma_{k+1} = \sigma_k + \alpha_k(\delta\sigma_k), \tag{6}$$

where α_k is a step length, computed via a line search, that minimizes the objective function $F(\sigma_{k+1})$ where

$$F(\sigma) = \frac{1}{2}\|U(\sigma) - V\|_2^2 + \lambda \sum_i \sqrt{(\mathbf{L}_i\sigma)^2 + \gamma}, \tag{7}$$

and the update

$$\delta\sigma_k = -\left(J_k^T J_k + \lambda B_k\right)^{-1}\left(J_k^T(U_k - V) + \lambda B_k \sigma_k\right), \tag{8}$$

where the subscript T denotes the nonconjugate transpose, $J_k = J(\sigma_k)$ is the Jacobian for iterate σ_k, $U_k = U(\sigma_k)$, and $B_k = \mathbf{L}^T E_k^{-1} \mathbf{L}$ where $E_k = \text{diag}(\eta_i)$ with $\eta_i = \sqrt{(\mathbf{L}_i\sigma_k)^2 + \gamma}$. The TV method was used here as it is a commonly used reconstruction method for EIT that recovers sharp target boundaries. See [36]–[38] for further details on derivation and implementation of the TV method for the EIT problem.

C. Metrics for Success & Experimental Setups

Improvement in image/reconstruction quality will be assessed by several metrics and furthermore compared to results from a classic CNN architecture. The training data as well as various test sets (simulated and experimental) are also described here.

CNN Comparison Method: An alternative to the graph U-net presented in this work was to interpolate image data defined on a FE mesh to a pixel grid so that a typical convolutional U-net can be used. Thus, the reconstruction x_{in} was computed, as before, on a FE mesh from

measurement data y, then interpolated to a pixel grid $\tilde{x}_{in} = f(x_{in})$. Next, a post-processing CNN Λ_θ^{CNN} is used to estimate the reconstruction on the pixel grid: $\tilde{x} = \Lambda_\theta(\tilde{x}_{in})$. If desired, the network output can then be interpolated back to the FE computational mesh: $\hat{x} = f^\dagger(\tilde{x})$.

In this setting, the CNN networks are trained using a set of initial reconstruction and truth image pairs, where both have been interpolated from the computational mesh to the pixel grid. It may be desirable to utilize a CNN to post-process images, as opposed to a GNN, as there is precedent for using CNNs on image data. Still, the loss of fine detail in refined portions of the mesh and the errors induced by interpolating to and from a pixel grid could be prohibitive or time intensive in certain imaging applications.

Experimental Data: Six experimental tank datasets with conductive agar targets, taken on three different EIT machines will be used to evaluate the graph U-net method presented in Section II-A1. The first set, denoted KIT4[3], comes from the 16 electrode KIT4 system at the University of Eastern Finland [10], [11]. Conductive agar targets (pink 0.323 S/m, white 0.061 S/m) were placed in a saline bath, with measured conductivity 0.135 S/m, in a chest shaped tank with perimeter 1.02 m and 16 evenly spaced electrodes of width 20 mm. The computational reconstruction mesh used for the KIT4 datasets contained 3,984 elements. Three experiments were performed to mimic heart (pink) and lung (white) imaging (Fig. 4-left). Sample KIT4-S.1 shows a 'healthy' setup with two low conductivity targets (lungs) and one high conductivity target (heart). Sample KIT4-S.2 has a cut in the "lung" target on the viewer's left while KIT4-S3 replaces the missing portion with a more conductive agar (e.g. possibly a pleural effusion). For each setup, 16 adjacent current patterns were applied with an amplitude of 3 mA and current frequency of 10 kHz, and measurements were recorded on all electrodes. The regularization parameters for TV were selected as $\lambda = 5 \cdot 10^{-5}$ and $\gamma = 10^{-14}$ based on testing from simulated datasets.

Next, archival data from the 32 electrode ACT3 system [12], [13] was used. In Sample ACT3-S.1 ((Fig. 4-right), a circular heart target (0.750 S/m) and two lung targets (0.240 S/m) were placed in a saline bath (0.424 S/m) in a tank of radius 0.15m. Trigonometric current patterns with maximum amplitude 0.2mA and frequency 28.8kHz were applied on the 32 equally spaced electrodes of width 25mm.

Lastly, data from the ACT5 system [14], was used for extension testing to 3D data. Samples ACT5-S.1 and ACT5-S.2 (Fig 5) were collected on plexiglas box with interior dimensions 17.0cm x 25.5cm x 17.0cm with 32 electrodes of size 8cm x 8cm [15]. The top of the tank is removable and has small holes allowing for filling and resealing between experiments. Spherical agar targets of measured conductivity 0.290 S/m were placed in tap water measuring 0.024 S/m. Optimal current patterns for the saline only tank were obtained and used for ACT5-S.1 and S.2.[4]

Training Data: Simulated data using the chest-shaped domain corresponding the the KIT4 data, with 16 electrodes and adjacent current pattern injection, was used to generate the

---

[3]The experimental KIT4 data is freely available at https://github.com/sarahjhamilton/open2Deit_agar.
[4]The experimental ACT5 data is freely available at https://github.com/sarahjhamilton/open3D_EIT_data.

training data. The measurement mesh contained 10,274 elements, and the reconstruction mesh was the same as the one used for the experimental KIT4 datasets (3,984 elements). The simulated true conductivity distributions had an equal chance of either three or four randomly placed elliptical targets that were not allowed to overlap or touch the boundary. For each ellipse, the major axis was between [0.03–0.07] meters and the minor axis was [50 – 90]% of the value of the major axis. Each target had an equal chance of having a constant conductivity in the range of [0.04, 0.07] S/m or [0.25, 0.35] S/m, while the background conductivity values were constant in the range of [0.11, 0.17] S/m. The measured voltage data was simulated using all 16 possible adjacent current patterns with an amplitude of 3 mA and 0.5% relative noise added to the voltages prior to reconstruction with TV. This corresponds to the simulated data having an SNR of about 51 dB, well below the SNR of the experimental data (KIT4 65.52 dB [39], and ACT3/ACT5 96 dB).

Eight separate U-nets were trained, four based on GNNs and four on CNNs. The four graph U-nets are named GNN-TVx and the four convolutional U-nets are named CNN-TVx where the "x" represents the TV method iterate used as input to the network. That is GNN-TV2 and CNN-TV2 are a graph U-net and classic U-net, respectively, that use the second iteration σ_2 of the TV method as input. The networks were trained using 5,000 training samples and 500 validation samples. The number of training samples used was greater than the number used for the model-based methods in [8] since the networks have many more trainable parameters. The number of samples was in line with other implementations of the U-net architecture for EIT [11] and did not result in severe or harmful over-fitting of the networks trained here. More testing could be done to determine if even fewer samples could be used for training. The ADAM optimizer [40] with an initial learning rate of 5 10^{-4} and mini batches of 32 samples were used to optimize the parameters. Other learning rates and batch sizes were also tested and resulted in similar minimum loss values and training times. Training of each network was stopped if the validation loss, MSE, failed to decrease over the course of 50 epochs and the parameters (weights and biases) that resulted in the lowest validation loss were saved.

The loss plots for the GNNs and CNNs using different iterates as input are shown in Figure 6. As expected, the U-nets using later iterates as input achieved lower minimum validation losses than the networks using TV1 inputs. There was a greater difference in the minimum validation loss values between graph U-nets with different inputs compared to classic U-nets with different inputs. Still, for both types, there was not a large difference between the U-nets using the third and fourth iterates as inputs. The CNNs reached lower raw minimum validation loss values but they cannot be compared directly to the graph U-nets because the loss was computed over different domains and discretizations.

The shapes of the loss curves are different between the types of U-nets but consistent across input iteration. The graph U-nets required 130–280 epochs to reach a minimum validation loss while the convolutional U-nets required only 20–30 epochs. After reaching the minimum validation loss, the graph U-nets validation loss values leveled out in later epochs, while the CNN validation loss values slightly increased. For both network types, the training loss values continued to decrease. All of these characteristics were consistent across repetitions of training independent networks, and more research is needed to determine

why the differences exist or what the effects are in the final reconstructions. In addition, determining if the network weights resulting in the lowest validation loss produce the "best" reconstructions is also the topic of future research as metrics other than MSE are also critical in assessing EIT reconstruction quality.

Metrics: As there is no universally accepted metric for assessing the quality of EIT reconstructions, the metrics listed below, along with visual inspection, will be used collectively to assess reconstruction quality:

$$\textbf{Dynamic Range}: \text{DR} = \frac{\max(\hat{\sigma}) - \min(\hat{\sigma})}{\max(\sigma_{\text{true}}) - \min(\sigma_{\text{true}})} \times 100\% .$$

$$\textbf{Total Variation Ratio}: \text{TVR} = \frac{\sum |\mathbf{L}\hat{\sigma}|}{\sum |\mathbf{L}\sigma_{\text{true}}|} \times 100\% .$$

$$\textbf{Relative } l_2 \textbf{ Voltage Error}: RE_V^{l_2} = \frac{\|U(\hat{\sigma}) - V\|_2}{\|V\|_2}$$

where 100% is a perfect score for DR and TVR while 0 is ideal for relative voltage error.

Note that these metrics above do not account for the varying sizes of the mesh elements. If desired, the metrics can be scaled relative to element size as well and the networks even trained based on such weighted metrics. Further work is needed to determine if the weighted or unweighted metrics and/or using weighted loss functions during training are more correlated with visually high quality reconstructions. Such work, while interesting is left for future studies. Here, only unweighted metrics were used and reported. Additionally, region of interest (ROI) metrics may be of higher interest than global image metrics. Where appropriate, e.g. lung imaging,

ROI metrics are also presented and represent the mean recon- structed conductivity value in the indicated region.

III. RESULTS

We first explore results for data consistent with the training data described in Sec. II-C3. Figure 7 compares the results for a simulated dataset consistent with the network data. The top row displays the truth as well as the non-learned TV reconstruction after 20 iterations. The second row displays the input images for the networks which vary from iteration 1 to 4 of TV. The third and fourth rows show the post-processed reconstructions from the GNN and CNN networks. Note that the inputs (row 2) are interpolated to a pixel grid, processed by the CNN networks, then interpolated back to the computational mesh on which they are displayed in row four. As expected the CNNs display excellent sharpening even from a first TV iteration input image. The GNNs outputs are similarly sharpened with early iterate TV inputs but the targets are better separated when using at least the second TV iterate. Metrics averaged over 50 simulated test samples consistent with the training data are also shown in

Fig 7. We see that overall the learned methods outperform the classic TV in all metrics aside from the relative voltage error fit, which is not unexpected as TV optimizes specifically for this whereas the network solutions do not. The GNN and CNN based U-nets perform similarly across the remaining metrics with the GNNs slightly outperforming the CNNs for the DR and TVR. Note, also that the first iteration of TV performed significantly worse than later TV iterates for the GNNs indicating a second iterate starting point may be advisable.

Reconstructions from the experimental KIT4 S.1-S-3 datasets are shown in Fig. 8, with corresponding mean conductivity ROI metrics in Table I. As with the simulated data, we see remarkable sharpening even with a single iterate of TV used as input for both the CNN and GNN. Recall that the training data for the networks was simple ellipses and thus the target shapes even in KIT4-S.1 are slightly different than that and the networks have not seen 'cut' data as in KIT4-S.2 and S.3. The post-processed images from the CNNs have slightly deformed 'lungs' when compared to the GNN output. Small conductive artifacts appear many of the GNN and CNN reconstructions at the bottom center. None of the methods, learned or TV, were able to recover the sharp cut in both the top and bottom portion of the viewer's left lung in Sample KIT4-S.3. The bottom portion of the lung did have a sharper dividing line for the GNNs as well as CNN-TV4. Moving to the metrics, we see that TV slightly outperforms the networks, as expected. However, in the DR and TVR we again see the GNNs better approximate the true dynamic range and total variation ratio, in particular for the earlier TV iterates. For the mean conductivity ROI metrics in Table I, overall, the GNNs slightly outperform both the full TV and CNNs. In each case the full TV reconstruction produced the visually most similar reconstruction to the truth, but needed approximately 20 iterations, compared to 1–4 iterations with the post-processed setting. Depending on the application, in particular for 3D, it may be important to balance the reconstruction quality versus computational cost.

Next, further out-of-distribution data is tested by using the ACT3-S.1 dataset which comes from a circular domain, 32 electrodes, applied trigonometric current patterns, and has targets with different conductivity values than in the training data. In order to process the ACT3-S.1 sample through the trained networks, the input images were scaled to have the background value in the expected window, processed, then scaled back. Figure 9 shows the resulting conductivity reconstructions and Table I the corresponding ROI metrics. Here, both the CNN and GNN networks required at least TV2 input to resolve the targets, though the GNN-TV1 image is more recognizable than the CNN-TV1, albeit the contrast is worse. The CNN-TV2 did the best at separating the lungs but underestimated the size of the heart. The ROI metrics show that the mean target ROIs are quite close overall to the truth.

To this end, in Fig. 10, we study how well the networks handle incorrect domain modeling, a notoriously challenging problem in absolute/static EIT imaging [41]. Here, the true domain is the chest-shaped domain but we naively reconstruct on the elliptical domain shown. Again, we see that TV1 is not informative enough for GNN-TV1 to recover all four targets. Notably, the boundary artifacts common from the domain mismodeling are significantly reduced in the post-processed images for both the GNN and CNN networks.

IV.   Discussion

To further test test the robustness of the graph U-net we explored how well the networks trained on 2D EIT data worked when the input data came from 3D measurement data that was significantly different than the networks were trained on. Additional modifications to the network structure (layers and inputs) are discussed. Lastly, a free python package GN4IP developed for this project is presented.

A.   Testing 3D data in 2D networks

The data is defined over a graph, not a pixel grid, as such the convolutions and pooling in the graph U-net are not dependent on the 2D geometry on which the data was trained. This flexibility allows us to input data from different dimensions. We take the networks trained on the 2D EIT data in a chest-shaped 16 electrode tank with adjacent current pattern injection, and moderate conductivity contrasts and test how well they generalize to 3D EIT reconstructions from the experimental ACT5 tank data with 32 large electrodes, with different current patterns, and high contrast targets ($0.290/0.024 \approx 12x$ contrast). Here, the inputs to the network are coming from the first iteration of a Levenberg-Marquardt (LM) algorithm, with update term

$$\delta\sigma_k^{(LM)} = -\left(J_k^T J_k + \lambda_{LM} I\right)^{-1} J_k^T (U_k - V),$$

using $\lambda_{LM} = 1e - 6$. The computational mesh for the 3D box tank shown in Fig. 5 had 85,699 elements and 18,569 nodes. Solutions were computed on the elements, using linear basis functions, and thus the associated graph had 85,699 nodes. Next, the LM iteration 1 reconstructions were scaled by 1/5 to bring the background conductivity value into the window expected by the network. The 3D reconstructions were then processed through GNN-TV1 and scaled back by 5 yielding the images in Fig. 11 (rows 1–2) resulting in significantly sharper images. The 3D reconstructions are visualized here by stacking transparent slices in the xy-plane to render a transparent 3D image. The images do not achieve the contrast of the true targets, but as the network was not expecting data at 12x contrast, this is not unexpected and in fact is in line with the regularized TV results in Fig. 3 of [15]. Computational cost of the single iterate was under 10 minutes, not optimized and the post-processing negligible.

Next, we further test the limits by post-processing reconstructions from a direct Complex Geometrical Optics (CGO) based method, the $\mathbf{t}^{\mathrm{exp}}$ approximation [15], [42], based on the full nonlinear direct solution method [43]. The method is fast (a few seconds) and is based on scattering transforms for the associated Schrödinger problem, essentially nonlinear Fourier transforms tailor made for the EIT problem. For simplicity, the $\mathbf{t}^{\mathrm{exp}}$ conductivity reconstructions in Fig. 3 of [15], which can be computed on any type of mesh, were interpolated to the same 85,699 element FEM mesh, scaled again by 1/5, and then used as input to the network GNN-TV1. As the $\mathbf{t}^{\mathrm{exp}}$ reconstructions were able to achieve the correct experimental 12x contrast before post-processing, the contrast of the targets input to the network were very far out (6x higher) from what was expected. As such, the network

struggled with what to do with this contrast as can be seen in Fig. 11 (row 3). However, row 4 suggests that the contrast really was the problem, as $\mathbf{t}^{exp}$ reconstructions from simulated noisy voltage data corresponding to the contrast expected by the network is post-processed extremely well. Note how different the artifacts are in the input $\mathbf{t}^{exp}$ images when compared to the 2D TV input images. Nevertheless, the graph U-net is able to sharpen this image remarkably well and adjust the contrast.

This underlines the flexibility of the graph structure, where training a graph U-net on 2D data and using on 3D data may be particularly advantageous for computationally demanding problems with dense 3D meshes. Testing the limits of this approach is the subject of ongoing work.

B. Modifying the Network Architecture

The growing number of applications utilizing graph data has motivated the increased interest in GNNs over the past several years [44]. Consequently, a variety of network architectures have been proposed that leverage different geometric aspects of the available graph data. While the presented study here considers the suitability of the graph U-net for the post-processing task, we also note, that a possible shortcoming of the graph convolutional layer [22] is a reduced fitting capacity compared to a standard convolutional layer. Specifically, the graph convolutional layer aggregates information according to a function of the adjacency matrix in contrast to a *learned* linear combination of neighboring pixels in the standard convolution. While we presented a convolutional U-Net architecture here, other architectures and layers can be considered. For instance, further improvements to the presented application here could be achieved by using more expressive layers, such as graph attention networks [45] and variants [46], or layers exploiting specific geometric relations [29].

C. Further Applications

Aside from the post-processing tasks in the EIT reconstruction problem, other imaging (or non-imaging) inverse problems that utilize irregular and sample-specific domains may benefit from using GNNs. One example is in omnidirectional or 360° imaging tasks where placing the image on a rectangular pixel grid causes distortion [47]. The use of GNNs instead of CNNs would remove the need for projections and allow the images to remain in their natural spherical domains. In particular, if only the improved solution at the boundary of an N-dimensional object is desired, the solution could be post-processed on a graph consisting only of boundary mesh elements removing the need to project to lower dimensional pixel grids and using determine wrapping conditions in the projected domain. Additionally, ResNets in model-based learning (e.g. [8]) may be replaced by graph U-nets counterparts, offering larger receptive fields which may be needed to negate nonlocal artifacts in the image or data domain [48], [49].

D. Python Package GN4IP

A Python package, Graph Networks for Inverse Problems (GN4IP), was developed to more easily implement learned model-based [8] and post-processing reconstruction methods

[16]. In general, it contains methods for loading datasets from .mat files; building a GNN and CNN with simple sequential or U-net architectures; training and saving model parameters; and predicting; among other things. The package utilizes the PyTorch and PyTorch Geometric libraries for their neural network capabilities. Also, the package is capable of calling on EIDORS[5], a set of open source algorithms for EIT implemented in MATLAB, to solve the EIT forward problem and compute LM and TV updates needed for the model-based methods. The GN4IP package is currently available on github[6].

V. CONCLUSION

A new graph U-net alternative to the convolutional U-net that has been used extensively for imaging tasks was presented here. The presented architecture is distinguished by the pro- posed *k-means cluster max pool layer* and *clone cluster unpool layer*. The k-means cluster max pool layer behaves similarly to the max pool layer of the convolutional U-net in that it aggregates features within local windows of the input's data structure. This was different from previously used hierarchical, node-dropping layers. The clone cluster unpool layer works naturally with the cluster assignments of the associated k-means cluster max pool layer to up-sample the input graph by restoring the original graph structure. The main advantage of the presented graph U-net over the CNN U-Net is given by the flexibility provided by the graph framework, allowing application to irregular data defined over FEM meshes and being dimension agnostic.

Using EIT as a case study, the new graph U-net was tested on six different experimental datasets coming from three different EIT machines both in 2D and 3D. Compared to the classic CNN alternative, the proposed network shows comparable performance and the k-means cluster maxpool layers provide similar behaviour. The advantage of the graph framework comes with the added flexibility to process on irregular data where interpolation between meshes is undesirable or computationally expensive. A significant advantage of the graph, as presented, is the ability to train on lower dimensional data (e.g. 2D) and application to higher dimensional data (e.g. 3D). This is a conceptual difference to the CNN pixel/voxel based setting, where filters are dimension dependent.

Compared to the full iterative TV method used as baseline, the presented post-processing uses only the first few iterates, effectively reducing the inference time. On average, each iteration of the TV method took about 1.7 seconds per sample in 2D, while the application of a trained U-net of either type took only 0.01 seconds per sample. Therefore, eliminating a fraction of the required iterations reduces the inference time by about the same fraction, i.e., 20 versus 4 iterates or less. The time savings become even more valuable in 3D applications where the meshes contain more elements and each iteration of the classical method of choice takes considerably longer, often even computationally prohibitive.

In regards to inverse problems in general, this work indicates that GNNs are a flexible option for applying deep learning and are a viable alternative to other network types. Continued

---

[5]Electrical Impedance Tomography and Diffuse Optical Tomography Re- construction Software (EIDORS) is available at eidors3d.sourceforge.net

[6]Graph Networks for Inverse Problems (GN4IP) is available at github.com/wherzberg/GN4IP

research on novel GNN layers, architectures, and applications is encouraging for the future of GNNs for inverse problems.

Acknowledgement

References

[1]. Mueller JL and Siltanen S, Linear and nonlinear inverse problems with practical applications. SIAM, 2012.

[2]. Kaltenbacher B, Neubauer A, and Scherzer O, "Iterative regularization methods for nonlinear ill-posed problems," in Iterative Regularization Methods for Nonlinear Ill-Posed Problems. de Gruyter, 2008.

[3]. Lionheart W. and Adler A, "Comparing d-bar and common regularization-based methods for electrical impedance tomography," Physiological measurement, vol. 40, no. 4, p. 044004, 2019.

[4]. Arridge S, Maass P, Ö ktem O, and Schönlieb C-B, "Solving inverse problems using data-driven models," Acta Numerica, vol. 28, pp. 1–174, 2019.

[5]. Ronneberger O, Fischer P, and Brox T, "U-net: Convolutional networks for biomedical image segmentation," in International Conference on Medical image computing and computer-assisted intervention. Springer, 2015, pp. 234–241.

[6]. Gao H. and Ji S, "Graph u-nets," in International Conference on Machine Learning, 2019, pp. 2083–2092.

[7]. Mozumder M, Hauptmann A, Nissila I ̈, Arridge SR, and Tar- vainen T, "A model-based iterative learning approach for diffuse optical tomography," arXiv: 2104.09579, 2021.

[8]. Herzberg W, Rowe DB, Hauptmann A, and Hamilton SJ, "Graph convolutional networks for model-based learning in nonlinear inverse problems," IEEE Transactions on Computational Imaging, vol. 7, pp. 1341–1353, 2021. [PubMed: 35873096]

[9]. Sousa TH, Camargo E, Martins AR, Biasi C, Pinto A, and Lima RG, "In vivo measurements for construction an anatomical thoracic atlas for electrical impedance tomography (eit): Methods for eit regularizations," in 2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings. IEEE, 2014, pp. 802–805.

[10]. Kourunen J, Savolainen T, Lehikoinen A, Vauhkonen M, and Heikkinen L, "Suitability of a pxi platform for an electrical impedance tomography system," Measurement Science and Technology, vol. 20, p. 015503, 12 2008.

[11]. Hamilton SJ, Ha A¨nninen, A. Hauptmann, and V. Kolehmainen, "Beltrami-net: domain-independent deep d-bar learning for absolute imaging with electrical impedance tomography (a-EIT)," Physiological Measurement, vol. 40, no. 7, p. 074002, jul 2019.

[12]. Cook RD, Saulnier GJ, Gisser DG, Goble JC, Newell JC, and Isaacson D, "ACT3: A high-speed, high-precision electrical impedance tomograph," IEEE Transactions on Biomedical Engineering, vol. 41, no. 8, pp. 713–722, August 1994. [PubMed: 7927393]

[13]. Isaacson D, Mueller JL, Newell JC, and Siltanen S, "Reconstructions of chest phantoms by the D-bar method for electrical impedance tomography," IEEE Transactions on Medical Imaging, vol. 23, pp. 821–828, 2004. [PubMed: 15250634]

[14]. Rajabi Shishvan O, Abdelwahab A, and Saulnier G, "ACT5 EIT system," in Proceedings of the 21st International Conference on Biomedical Applications of Electrical Impedance Tomography. Zenodo, Jun. 2021. [Online]. Available: 10.5281/zenodo.4635480

[15]. Hamilton S, Muller P, Isaacson D, Kolehmainen V, Newell J, Shishvan OR, Saulnier G, and Toivanen J, "Fast absolute 3d cgo-based electrical impedance tomography on experimental data," Physiological Measurement, vol. 43, no. 12, p. 124001, 2022.

[16]. Herzberg W, "Graph neural networks for inverse problems with flexible meshes," Ph.D. dissertation, Marquette University, 2022.

[17]. Goodfellow I, Bengio Y, and Courville A, Deep learning. MIT press, 2016.

[18]. Prince SJ, UNDERSTANDING DEEP LEARNING. MIT PRESS, 2023.

[19]. Hamilton SJ and Hauptmann A, "Deep D-Bar: Real-time electrical impedance tomography imaging with deep neural networks," IEEE Transactions on Medical Imaging, vol. 37, no. 10, p. 2367–2377, Oct 2018. [PubMed: 29994023]

[20]. Hamilton S, Hauptmann HA,A, and Kolehmainen V, "Beltrami-net: domain-independent deep d-bar learning for absolute imaging with electrical impedance tomography (a-eit)," Physiological Measurement, vol. 40, p. 074002, 2018.

[21]. Wei Z, Liu D, and Chen X, "Dominant-current deep learning scheme for electrical impedance tomography," IEEE Transactions on Biomedical Engineering, vol. 66, no. 9, pp. 2546–2555, 2019. [PubMed: 30629486]

[22]. Kipf TN and Welling M, "Semi-supervised classification with graph convolutional networks," CoRR, vol. abs/1609.02907, 2016. [Online]. Available: http://arxiv.org/abs/1609.02907

[23]. NT H. and Maehara T, "Revisiting graph neural networks: All we have is low-pass filters," 2019. [Online]. Available: https://arxiv.org/abs/1905.09550

[24]. Zhang S, Tong H, Xu J, and Maciejewski R, "Graph convolutional networks: a comprehensive review," Computational Social Networks, vol. 6, no. 1, pp. 1–23, 2019.

[25]. Lee J, Lee I, and Kang J, "Self-attention graph pooling," 2019.

[26]. Ranjan E, Sanyal S, and Talukdar PP, "Asap: Adaptive structure aware pooling for learning hierarchical graph representations," 2019. [Online]. Available: https://arxiv.org/abs/1911.07979

[27]. Liu C, Zhan Y, Li C, Du B, Wu J, Hu W, Liu T, and Tao D, "Graph pooling for graph neural networks: Progress, challenges, and opportunities," 2022. [Online]. Available: https://arxiv.org/abs/2204.07321

[28]. Arthur D. and Vassilvitskii S, "K-means++: The advantages of careful seeding," in Proceedings of the Eighteenth Annual ACM-SIAM Sym-posium on Discrete Algorithms, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[29]. Suk J, Brune C, and Wolterink JM, "Se(3) symmetry lets graph neural networks learn arterial velocity estimation from small datasets," 2023.

[30]. Caldero A-P , "On an inverse boundary value problem," in Seminar on Numerical Analysis and its Applications to Continuum Physics (Rio de Janeiro, 1980). Rio de Janeiro: Soc. Brasil. Mat, 1980, pp. 65–73.

[31]. Alessandrini G, "Stable determination of conductivity by boundary measurements," Applicable Analysis, vol. 27, no. 1–3, pp. 153–172, 1988.

[32]. Nachman AI, "Global uniqueness for a two-dimensional inverse boundary value problem," Annals of Mathematics, vol. 143, pp. 71–96, 1996.

[33]. Mueller J. and Siltanen S, Linear and Nonlinear Inverse Problems with Practical Applications. SIAM, 2012.

[34]. Borcea L, "Electrical impedance tomography," Inverse Problems, vol. 18, pp. 99–136, 2002.

[35]. Somersalo E, Cheney M, and Isaacson D, "Existence and uniqueness for electrode models for electric current computed tomography," SIAM Journal on Applied Mathematics, vol. 52, no. 4, pp. 1023–1040, 1992.

[36]. Vauhkonen M, "Electrical impedance tomography and prior information," 1997.

[37]. Kaipio J, Kolehmainen V, Somersalo E, and Vauhkonen M, "Statistical inversion and monte carlo sampling methods in electrical impedance tomography," Inverse Problems, vol. 16, 07 2000.

[38]. Borsic A, Graham B, Adler A, and Lionheart W, "In vivo impedance imaging with total variation regularization," IEEE Transactions on Medical Imaging, vol. 29, pp. 44–54, 2010. [PubMed: 20051330]

[39]. Nissinen A, Kolehmainen VP, and Kaipio JP, "Compensation of modelling errors due to unknown domain boundary in electrical impedance tomography," IEEE Transactions on Medical Imaging, vol. 30, no. 2, pp. 231–242, 2011. [PubMed: 20840893]

[40]. Kingma DP and Ba J, "Adam: A method for stochastic optimization," 2015.

[41]. Hamilton SJ, Mueller JL, and Santos TR, "Robust computation in 2d absolute eit (a-eit) using d-bar methods with the 'exp' approximation," Physiological Measurement, vol. 39, no. 6, p. 064005, jun 2018.

[42]. Hamilton SJ, Isaacson D, Kolehmainen V, Muller PA, Toivainen J, and Bray PF, "3D electrical impedance tomography reconstructions from simulated electrode data using direct inversion texp and Calderón methods," Inverse Problems & Imaging, 2021.

[43]. Nachman A, Sylvester J, and Uhlmann G, "An n-dimensional Borg–Levinson theorem," Communications in Mathematical Physics, vol. 115, pp. 595–605, 1988.

[44]. Wu Z, Pan S, Chen F, Long G, Zhang C, and Yu PS, "A comprehensive survey on graph neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 32, no. 1, pp. 4–24, 2021. [PubMed: 32217482]

[45]. Veli kovic P´, Cucurull G, Casanova A, Romero A, Liò P, and Bengio Y, "Graph attention networks," 2017. [Online]. Available: https://arxiv.org/abs/1710.10903

[46]. Wang G, Ying R, Huang J, and Leskovec J, "Multi-hop attention graph neural network," 2020. [Online]. Available: https://arxiv.org/abs/2009.14332

[47]. Su Y-C and Grauman K, "Kernel transformer networks for compact spherical convolution," pp. 9434–9443, 06 2019.

[48]. Hauptmann A, Cox B, Lucka F, Huynh N, Betcke M, Beard P, and Arridge S, "Approximate k-space models and deep learning for fast photoacoustic reconstruction," in Machine Learning for Medical Image Reconstruction: First International Workshop, MLMIR 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Proceedings 1. Springer, 2018, pp. 103–111.

[49]. Hauptmann A. and Poimala J, "Model-corrected learned primal-dual models for fast limited-view photoacoustic tomography," arXiv preprint arXiv:2304.01963, 2023.
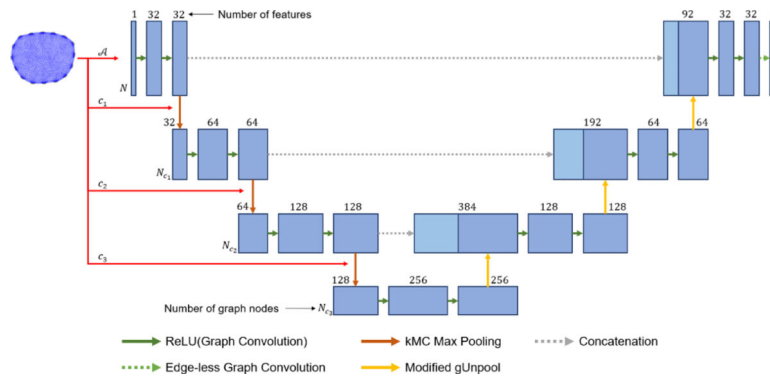
Fig. 1.

A diagram of the proposed graph U-net with three pooling layers. The input to the network is an initial reconstruction along with the adjacency matrix $\mathscr{A}$ and cluster assignments (c_1, c_2, c_3) for each of the pooling layers, determined from the mesh. The output is defined on the same graph structure as the input and can be taken, for example, as the final reconstruction.
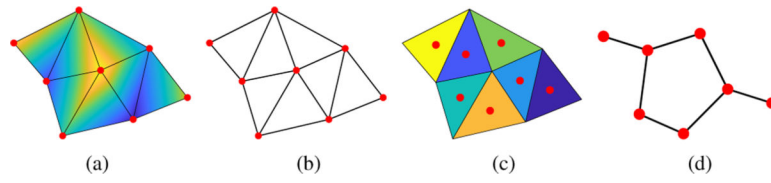
Fig. 2.
A mesh with data defined over mesh nodes (a) and (b) the corresponding graph. A mesh with data defined over mesh elements (c) and (d) the corresponding graph.
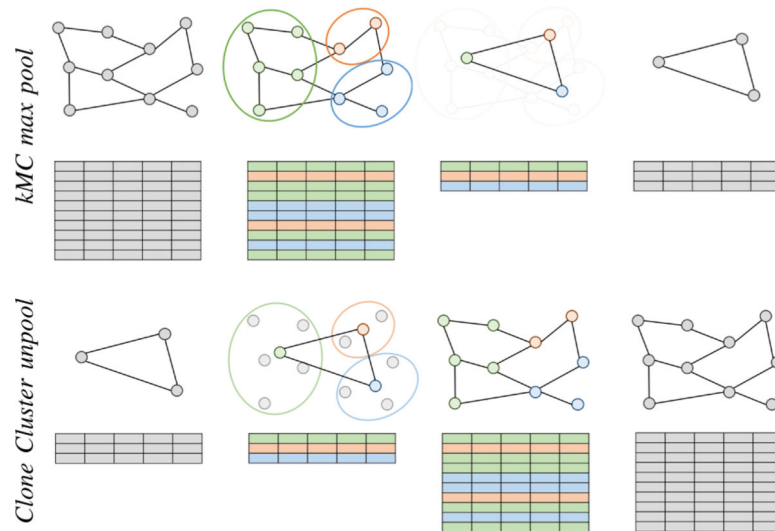
Fig. 3.
Top: A diagram of a graph (top) and its feature matrix (bottom) being pooled using the novel **k-means cluster max pool layer**. The k-means++ algorithm is used to cluster the input nodes (left) according to their spatial location. For each cluster (middle left), an output node is placed at the centroid of the cluster (middle right), and the node's features are determined using the maximum within the cluster. The edges of the output graph (right) connect previously connected clusters. Bottom: A diagram of a graph (top) and its feature matrix (bottom) being unpooled by **a clone cluster unpool layer** after a previous cluster-based pooling layer. The previous structure of the graph (node locations and edges) are restored (middle left) and the features of the input nodes are cloned/copied to the output nodes within each cluster (middle right).
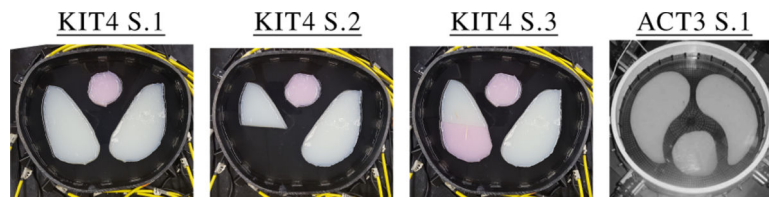
Fig. 4.
Photographs of the KIT4 and ACT3 experimental data [13], [20].

Fig. 5.
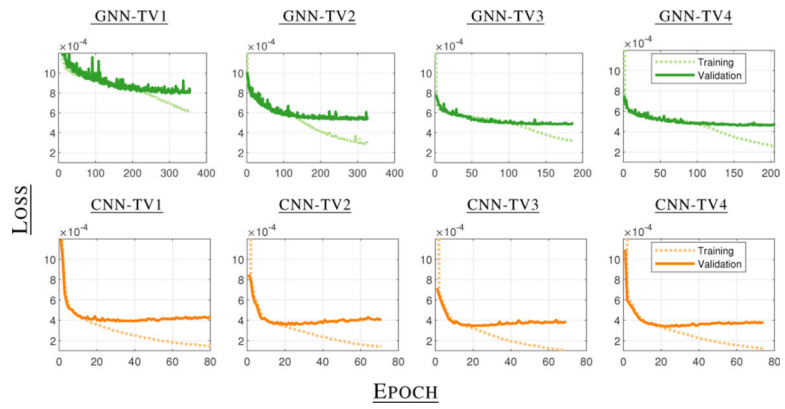ACT5 experimental setups and side view showing target height.

Fig. 6.
Loss plots (training and validation) for the **GNN-TVx** and **CNN-TVx** networks that were trained with the **Chest-16** dataset. The losses were computed as the mean squared error between the networks' outputs and the true conductivity distributions computed over the mesh for the GNNs (top) and over the pixel grid for the CNNs (bottom).
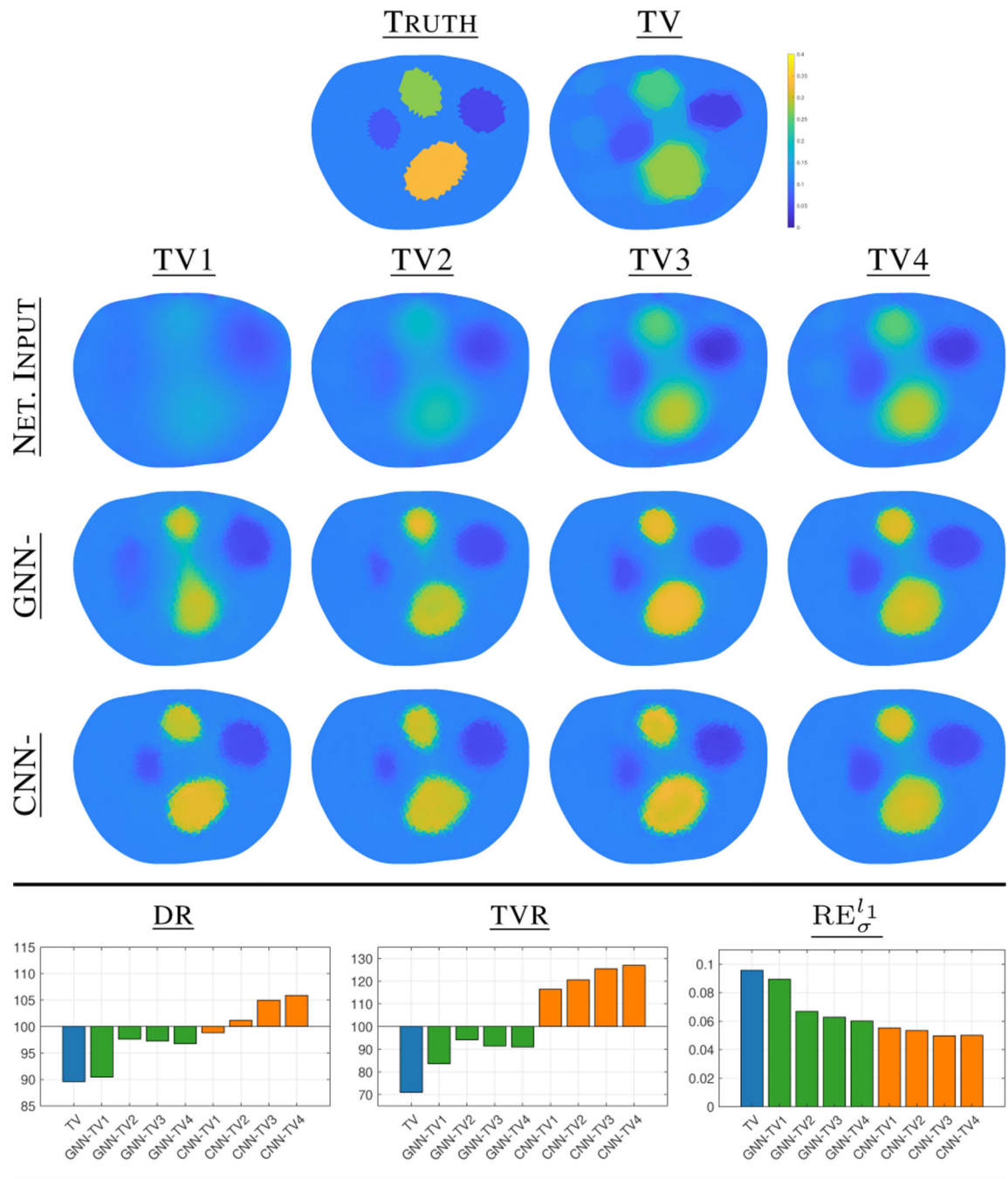
Fig. 7.
Demonstration of the **GNN-TVx** and **CNN-TVx** networks on simulated data consistent with training data as well as average metric scores for reconstructions of such 50 test samples compared to the **TV** method. All conductivity reconstructions are on the same color scale.
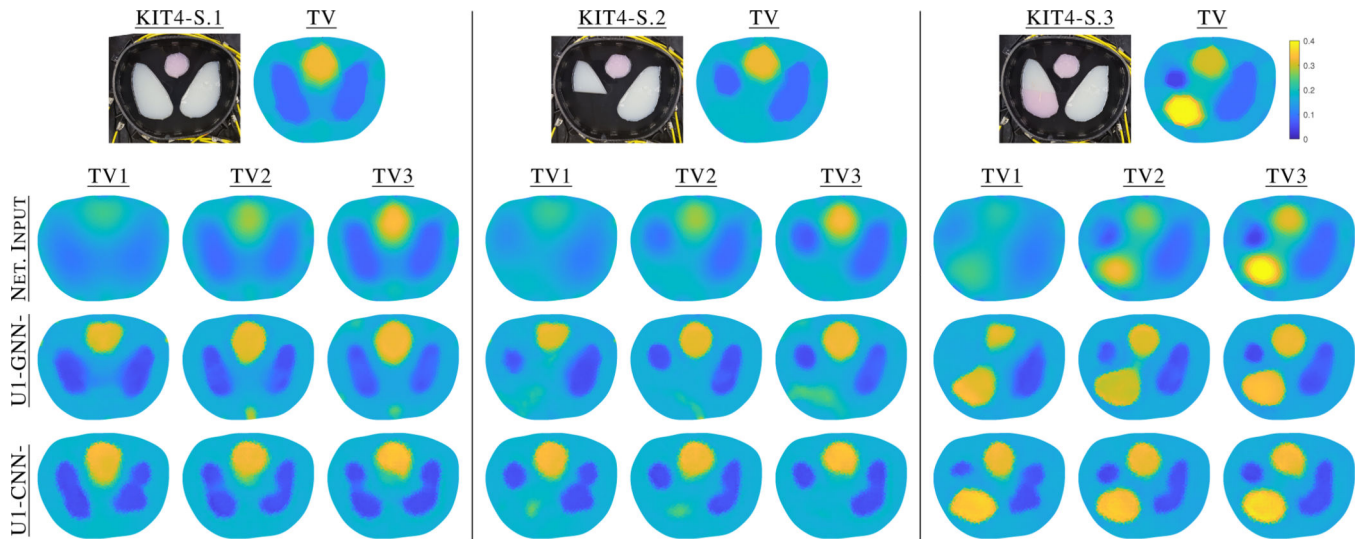
Fig. 8.
Results for KIT4-S.1, S.2, and S.3 for the first three iterations of TV as input to the networks GNN-TVx and CNN-TVx, all on the same color scale.
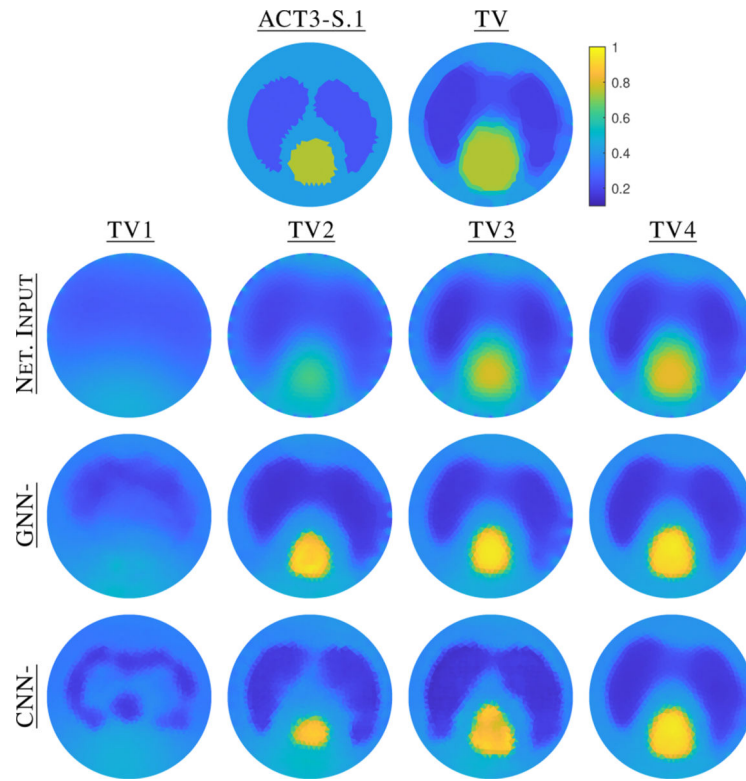
Fig. 9.
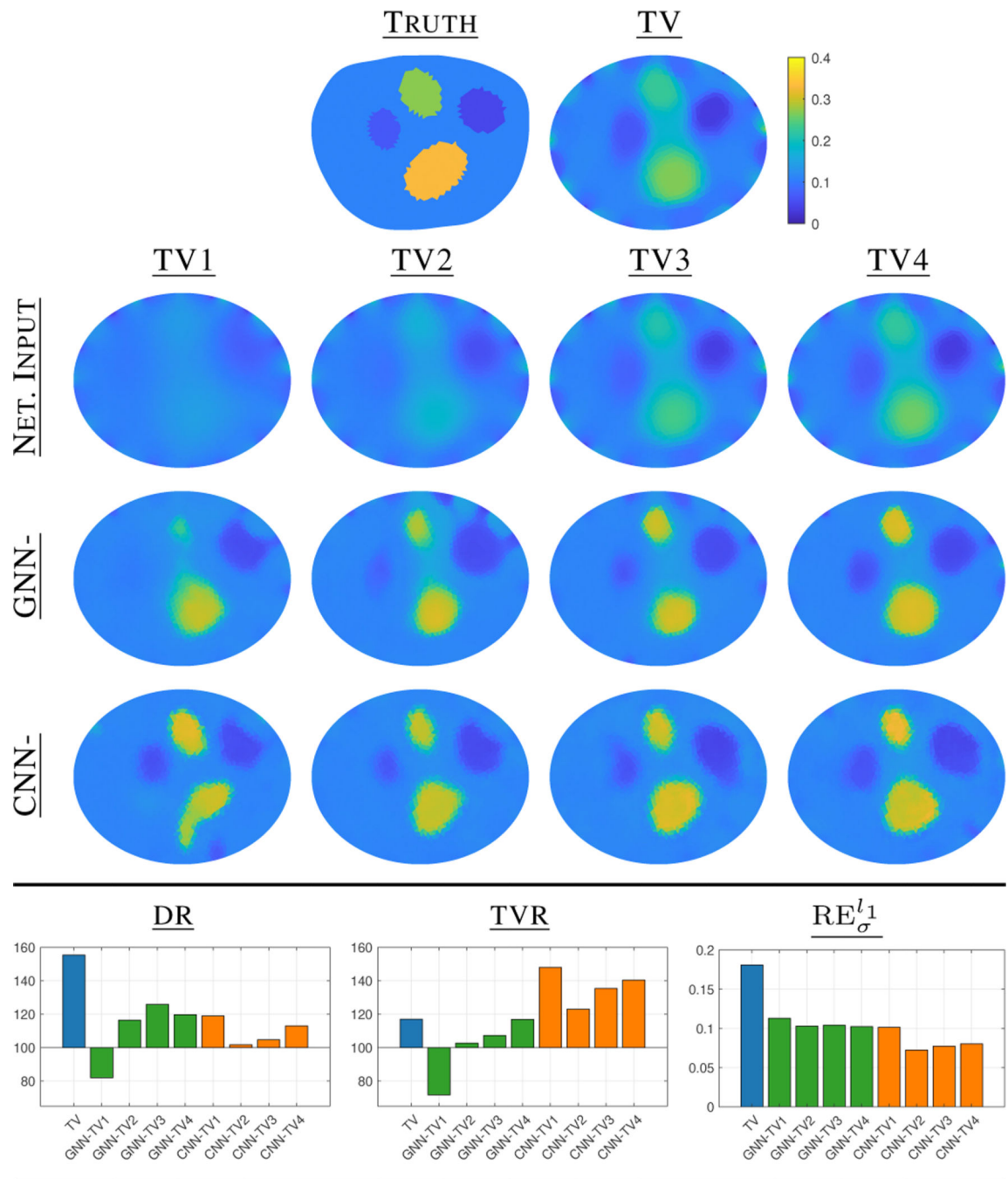Demonstration of the networks on the ACT3-S.1 data.

Fig. 10.
Robustness to domain modeling errors. Results of the GNN-TVx and CNN-TVx networks on simulated data, assuming the true domain was an ellipse instead of chest-shape as well as average metric scores for reconstructions of such 50 test samples compared to the TV methods.
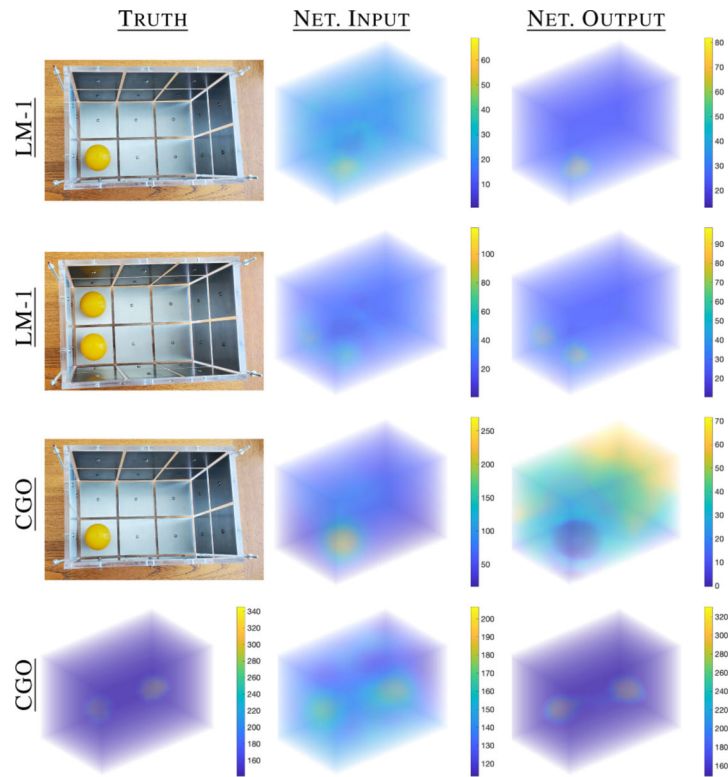
Fig. 11.

Post-processed reconstructions from experimental ACT5 data from LM iteration 1 and $\mathbf{t}^{exp}$ are shown in rows 1–3. Row 4 shows the result of post-processing a CGO reconstruction in line with the contrast expected by the U1 network using simulated, noisy voltage data.

TABLE I

Mᴇᴀɴ Cᴏɴᴅᴜᴄᴛɪᴠɪᴛʏ ROI ᴍᴇᴛʀɪᴄs (S/ᴍ) ғᴏʀ KIT4 ᴀɴᴅ ACT3 ʀᴇᴄᴏɴsᴛʀᴜᴄᴛɪᴏɴs.

		Truth	TV	GNN-				CNN-			
				TV1	TV2	TV3	TV4	TV1	TV2	TV3	TV4
KIT4-S.1	Heart	0.323	**0.324**	0.309	0.319	0.326	0.321	0.325	0.326	0.336	0.332
	R. Lung	0.061	0.105	**0.100**	0.116	0.118	0.118	0.106	0.113	0.104	0.106
	L. Lung	0.061	0.099	0.101	0.109	0.110	0.111	0.098	0.104	0.106	**0.099**
KIT4-S.2	Heart	0.323	0.318	0.291	0.317	**0.324**	**0.322**	0.326	**0.324**	0.334	0.327
	R. Lung	0.061	0.105	**0.098**	0.112	0.116	0.120	0.106	0.114	0.105	0.108
	L. Lung	0.061	**0.101**	0.122	0.105	0.107	0.103	0.109	0.105	0.109	**0.101**
KIT4-S.3	Heart	0.323	0.299	0.271	0.309	0.320	0.318	0.320	**0.323**	0.327	0.320
	R. Lung	0.061	0.099	**0.097**	0.105	0.103	0.106	0.106	0.106	0.100	0.105
	L. Lung	0.061	**0.118**	0.176	0.136	0.125	0.144	0.141	0.128	0.123	0.120
	Injury	0.323	0.350	0.302	0.295	**0.320**	0.316	0.337	0.328	0.344	0.357
ACT3-S.1	Heart	0.750	**0.748**	0.434	0.780	0.788	0.851	0.426	0.656	0.765	0.808
	R. Lung	0240	0.216	0.277	0.200	0.225	0.217	0.292	**0.231**	0.204	0.207
	L. Lung	0.240	0.202	0.271	0.205	0.223	0.215	0.280	**0.230**	0.196	0.200