Electrical impedance tomography, enclosure method and machine learning

Samuli Siltanen
Department of Mathematics and Statistics
University of Helsinki
samuli.siltanen@helsinki.fi
www.siltanen-research.net
YouTube: Samun tiedekanava

Colorado State University
March 5, 2020
This is a joint work with

Takanori Ide, Aisin AW Corp., Japan
One of these texts is by Hemingway, and the other is Google-translated to Japanese and back

Kilimanjaro is a snow-covered mountain 19,710 feet high, and is said to be the highest mountain in Africa. Its western summit is called the Masai “Ngaje Ngai,” the House of God. Close to the western summit there is the dried and frozen carcass of a leopard. No one has explained what the leopard was seeking at that altitude.

Kilimanjaro is a mountain of 19,710 feet covered with snow and is said to be the highest mountain in Africa. The summit of the west is called “Ngaje Ngai” in Masai, the house of God. Near the top of the west there is a dry and frozen dead body of leopard. No one has ever explained what leopard wanted at that altitude.

Link to the New York Times article: The Great A.I. Awakening
Outline

Introduction to neural networks

Convolutional neural networks

EIT and the enclosure method

The enclosure method: least-squares computation

The enclosure method: deep learning
Neural network models are based on biological neurons
A computational “neuron” is a simple formula forming an output number from several inputs.

Input dendrites \( x_1 \) \( w_{11} \) \( f \) \( b_1 \) Output to axon \( f(w_{11}x_1 + w_{12}x_2 + b_1) \)
We will make use of an activation function $f$. It can be for example a sigmoid like this:

$$f(t) = \left(1 + e^{-t}\right)^{-1}$$
We will make use of an activation function $f$. Rectified linear unit (ReLU) is for big networks.

\[ f(t) = \max\{0, t\} \]
Let us build a very simple neural network. Here we add another output.
Let us build a very simple neural network. Now we consider the output as a vector.
Let us build a very simple neural network.
The output can be written in matrix notation.

\[
\begin{align*}
\mathbf{x}^\top & = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \\
\mathbf{w} & = \begin{pmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{pmatrix}, \\
\mathbf{b} & = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \\
\mathbf{f} & = f(\mathbf{Ax} + \mathbf{b}).
\end{align*}
\]
We introduce a shorthand notation for our activated matrix-vector formula

\[ F_\theta(x) := f(Ax + b), \]

where the vector \( \theta \) contains all the parameters that will be learned:

\[ \theta = [w_{11}, w_{12}, w_{21}, w_{22}, b_1, b_2]^T \in \mathbb{R}^6 \]
Let us build a slightly more complicated model. Namely, we add one more hidden layer.

The output vector can be written in the form
\[ F_{\theta}^{(2)}(F_{\theta}^{(1)}(x)) = F_{\theta}^{(2)}(f(A^{(1)}x + b^{(1)})) = f(A^{(2)}(f(A^{(1)}x + b^{(1)})) + b^{(2)}) \]
This is the shorthand notation for the output vector of the network with two hidden layers:

$$\mathcal{F}_{\theta}(x) := F_{\theta}^{(2)}(F_{\theta}^{(1)}(x)),$$

where the vector $\theta$ contains 12 parameters that will be learned:

$$\theta = [w_{11}^{(1)}, w_{12}^{(1)}, w_{21}^{(1)}, w_{22}^{(1)}, b_1^{(1)}, b_2^{(1)}, w_{11}^{(2)}, w_{12}^{(2)}, w_{21}^{(2)}, w_{22}^{(2)}, b_1^{(2)}, b_2^{(2)}]^T$$
In practice we use a large number of neurons and group them into $L$ layers.

Output layer: $m$ numbers

Hidden layer

Hidden layer

Input layer: $n$ numbers
The network is a nonlinear function $\mathcal{F}_\theta : \mathbb{R}^n \rightarrow \mathbb{R}^m$

The function $\mathcal{F}_\theta$ is defined recursively like this:

\[
\begin{align*}
y^{(0)} & = x, \\
y^{(1)} & = f(A^{(1)}y^{(0)} + b^{(1)}), \\
y^{(2)} & = f(A^{(2)}y^{(1)} + b^{(2)}), \\
& \vdots \\
y^{(L)} & = f(A^{(L)}y^{(L-1)} + b^{(L)}), \\
\mathcal{F}_\theta(x) & = y^{(L)}.
\end{align*}
\]

The matrix $A^{(\ell)}$ has size $m^{(\ell)} \times n^{(\ell)}$, and $b^{(\ell)} \in \mathbb{R}^{m^{(\ell)}}$ is the bias vector. So the total number of parameters in the vector $\theta \in \mathbb{R}^T$ is

\[
T = \sum_{\ell=1}^{L} (m^{(\ell)} + 1)n^{(\ell)}.
\]
Training data consists of many pairs of correct input→output pairs

\[(x^{\{j\}}, v^{\{j\}}), \quad j = 1, 2, \ldots, J\]

\[x^{\{j\}} = \begin{bmatrix} x_1^{\{j\}} \\ x_2^{\{j\}} \\ \vdots \\ x_n^{\{j\}} \end{bmatrix}, \quad v^{\{j\}} = \begin{bmatrix} v_1^{\{j\}} \\ v_2^{\{j\}} \\ \vdots \\ v_m^{\{j\}} \end{bmatrix}.\]
How to use training data for learning?

We wish to find such values for the parameters $\theta_1, \theta_2, \ldots, \theta_T$ that the following equation holds as well as possible for all $j = 1, 2, \ldots, J$:

$$F_{\theta}(x^{\{j\}}) = v^{\{j\}}.$$  

In other words, we wish to use optimization to find the minimizer

$$\tilde{\theta} = \arg\min_{\theta} \left\{ \sum_{j=1}^{J} \| F_{\theta}(x^{\{j\}}) - v^{\{j\}} \|_2^2 \right\}$$

of the loss function. We can also regularize to avoid overfitting:

$$\tilde{\theta}_\alpha = \arg\min_{\theta} \left\{ \sum_{j=1}^{J} \| F_{\theta}(x^{\{j\}}) - v^{\{j\}} \|_2^2 + \alpha \| R\theta \|_2^2 \right\};$$

here $\alpha > 0$. 
Introduction to neural networks

Convolutional neural networks

EIT and the enclosure method

The enclosure method: least-squares computation

The enclosure method: deep learning
\[
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 \\
0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 \\
\end{bmatrix}
\]
\[
\begin{bmatrix}
0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 \\
0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 & 0.05 \\
0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 & 0 \\
0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 & 0.05 \\
0.10 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.10 \\
0.05 & 0.05 & 0 & 0 & 0 & 0 & 0 & 0.05 & 0.30 & 0.50 & 0.50
\end{bmatrix}
\]
Grayscale colors are represented as numbers; in the 8-bit case zero is black and 255 is white.
Two-dimensional convolution step-by-step

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>250</td>
<td>251</td>
<td>253</td>
</tr>
<tr>
<td>251</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>251</td>
<td>252</td>
<td>253</td>
<td>252</td>
</tr>
<tr>
<td>239</td>
<td>240</td>
<td>242</td>
<td>238</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
<td>61</td>
<td>65</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
<td>44</td>
<td>56</td>
</tr>
</tbody>
</table>

Original image and convolution kernel

<table>
<thead>
<tr>
<th></th>
<th>250</th>
<th>251</th>
<th>252</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>46</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Result of convolution
### Two-dimensional convolution step-by-step

#### Original image and convolution kernel

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>250</td>
<td>1</td>
<td>251</td>
<td>252</td>
</tr>
<tr>
<td>251</td>
<td>252</td>
<td>0</td>
<td>253</td>
<td>0</td>
</tr>
<tr>
<td>239</td>
<td>240</td>
<td>-1</td>
<td>242</td>
<td>238</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
<td>61</td>
<td>65</td>
<td>81</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
<td>44</td>
<td>56</td>
<td>77</td>
</tr>
</tbody>
</table>

#### Result of convolution

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>46</td>
<td>42</td>
</tr>
</tbody>
</table>

- **Original image and convolution kernel**
- **Result of convolution**
### Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th></th>
<th>250</th>
<th>250</th>
<th>251</th>
<th>252</th>
<th>253</th>
</tr>
</thead>
<tbody>
<tr>
<td>251</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>250</td>
<td>252</td>
<td>253</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>251</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>239</td>
<td>240</td>
<td>242</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
<td>61</td>
<td>65</td>
<td>81</td>
<td>70</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
<td>44</td>
<td>56</td>
<td>77</td>
<td>71</td>
</tr>
<tr>
<td>42</td>
<td>41</td>
<td>43</td>
<td>50</td>
<td>72</td>
<td>75</td>
</tr>
</tbody>
</table>

**Original image and convolution kernel**

|     |     |     |     |     |
|-----|-----|-----|-----|
| 43  | 46  | 42  | 19  |

**Result of convolution**
Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th>250</th>
<th>250</th>
<th>251</th>
<th>253</th>
<th>252</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>252</td>
<td>253</td>
<td>252</td>
</tr>
<tr>
<td>251</td>
<td>252</td>
<td>1</td>
<td>253</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>240</td>
<td>242</td>
<td>238</td>
</tr>
<tr>
<td>239</td>
<td>240</td>
<td>242</td>
<td>238</td>
<td>238</td>
<td>239</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>54</td>
<td>61</td>
<td>65</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
<td>61</td>
<td>65</td>
<td>81</td>
<td>70</td>
</tr>
<tr>
<td>70</td>
<td>71</td>
<td>71</td>
<td>71</td>
<td>72</td>
<td>72</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
<td>44</td>
<td>56</td>
<td>77</td>
<td>77</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>43</th>
<th>46</th>
<th>42</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>46</td>
<td>42</td>
<td>19</td>
</tr>
<tr>
<td>600</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th></th>
<th>250</th>
<th>250</th>
<th>251</th>
<th>253</th>
<th>252</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>251</td>
<td>1</td>
<td>252</td>
<td>1</td>
<td>253</td>
<td>1</td>
<td>252</td>
</tr>
<tr>
<td>239</td>
<td>0</td>
<td>240</td>
<td>0</td>
<td>242</td>
<td>0</td>
<td>238</td>
</tr>
<tr>
<td>54</td>
<td>-1</td>
<td>54</td>
<td>-1</td>
<td>61</td>
<td>-1</td>
<td>65</td>
</tr>
<tr>
<td>38</td>
<td>39</td>
<td>44</td>
<td>56</td>
<td>77</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>43</th>
<th>46</th>
<th>42</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>581</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th>Original image and convolution kernel</th>
<th>Result of convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>250 250 251 253 252 251</td>
<td>43 46 42 19</td>
</tr>
<tr>
<td>251 252 253 1 252 250 1 250</td>
<td>600 581 541 511</td>
</tr>
<tr>
<td>239 240 242 0 238 0 238 0</td>
<td></td>
</tr>
<tr>
<td>239 240 242 0 238 0 238 0</td>
<td></td>
</tr>
<tr>
<td>54 61 -1 65 -1 81 -1 70</td>
<td></td>
</tr>
<tr>
<td>54 61 -1 65 -1 81 -1 70</td>
<td></td>
</tr>
<tr>
<td>38 39 44 56 77 71</td>
<td></td>
</tr>
<tr>
<td>38 42 39 44 56 71</td>
<td></td>
</tr>
<tr>
<td>38 42 39 44 56 71</td>
<td></td>
</tr>
<tr>
<td>42 41 43 50 72 75</td>
<td></td>
</tr>
</tbody>
</table>
Two-dimensional convolution step-by-step

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

Original image and convolution kernel

Result of convolution
## Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th></th>
<th>250</th>
<th>250</th>
<th>251</th>
<th>253</th>
<th>252</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>251</td>
<td>252</td>
<td>253</td>
<td>252</td>
<td>250</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>239</td>
<td>240</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>238</td>
<td>239</td>
</tr>
<tr>
<td>54</td>
<td>0</td>
<td>61</td>
<td>0</td>
<td>65</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>38</td>
<td>-1</td>
<td>44</td>
<td>-1</td>
<td>56</td>
<td>-1</td>
<td>77</td>
</tr>
<tr>
<td>42</td>
<td>41</td>
<td>43</td>
<td>50</td>
<td>72</td>
<td>75</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>43</th>
<th>46</th>
<th>42</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>581</td>
<td>541</td>
<td>511</td>
<td></td>
</tr>
<tr>
<td>587</td>
<td>577</td>
<td>548</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Original image and convolution kernel**

**Result of convolution**
Two-dimensional convolution step-by-step

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th>Original image and convolution kernel</th>
<th>Result of convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>250 250 251 253 252 251</td>
<td>43 46 42 19</td>
</tr>
<tr>
<td>251</td>
<td>600 581 541 511</td>
</tr>
<tr>
<td>239 240 242 238 238 239</td>
<td>587 577 548 536</td>
</tr>
<tr>
<td>1 54</td>
<td>30</td>
</tr>
<tr>
<td>1 54 1 61 65 81 70</td>
<td></td>
</tr>
<tr>
<td>0 39 0 44 56 77 71</td>
<td></td>
</tr>
<tr>
<td>-1 -1 -1</td>
<td></td>
</tr>
<tr>
<td>42 41 43 50 72 75</td>
<td></td>
</tr>
</tbody>
</table>
### Two-dimensional convolution step-by-step

<table>
<thead>
<tr>
<th>Original image</th>
<th>Convolution kernel</th>
<th>Result of convolution</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Original image" /></td>
<td><img src="image" alt="Convolution kernel" /></td>
<td><img src="image" alt="Result of convolution" /></td>
</tr>
</tbody>
</table>

**Original image and convolution kernel**

**Result of convolution**
### Two-dimensional convolution step-by-step

**Original image and convolution kernel**

<table>
<thead>
<tr>
<th>250</th>
<th>250</th>
<th>251</th>
<th>253</th>
<th>252</th>
<th>251</th>
</tr>
</thead>
<tbody>
<tr>
<td>251</td>
<td>252</td>
<td>253</td>
<td>252</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>239</td>
<td>240</td>
<td>242</td>
<td>238</td>
<td>238</td>
<td>239</td>
</tr>
<tr>
<td>54</td>
<td>54</td>
<td>61</td>
<td>65</td>
<td>81</td>
<td>70</td>
</tr>
<tr>
<td>39</td>
<td>39</td>
<td>44</td>
<td>56</td>
<td>0</td>
<td>71</td>
</tr>
<tr>
<td>42</td>
<td>42</td>
<td>43</td>
<td>50</td>
<td>0</td>
<td>72</td>
</tr>
</tbody>
</table>

**Result of convolution**

<table>
<thead>
<tr>
<th>43</th>
<th>46</th>
<th>42</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>581</td>
<td>541</td>
<td>511</td>
</tr>
<tr>
<td>587</td>
<td>577</td>
<td>548</td>
<td>536</td>
</tr>
<tr>
<td>30</td>
<td>34</td>
<td>38</td>
<td></td>
</tr>
</tbody>
</table>

Original image and convolution kernel

Result of convolution
Two-dimensional convolution step-by-step

Original image and convolution kernel

Result of convolution
The convolution operation picked out the horizontal border between gray shades.
Pooling layers offer ways to flexibly summarize information coming from an image area.
Pooling layers offer ways to flexibly summarize information coming from an image area.
Pooling layers offer ways to flexibly summarize information coming from an image area
Pooling layers offer ways to flexibly summarize information coming from an image area.
Pooling layers offer ways to flexibly summarize information coming from an image area.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>30</th>
<th>30</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
</tr>
</tbody>
</table>
Pooling layers offer ways to flexibly summarize information coming from an image area.

\[
\begin{array}{ccccccc}
0 & 0 & 30 & 30 & 30 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
\end{array}
\]
Pooling layers offer ways to flexibly summarize information coming from an image area
Pooling layers offer ways to flexibly summarize information coming from an image area.

$$
\begin{array}{ccccccc}
0 & 0 & 30 & 30 & 30 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 \\
0 & 0 & 0 & 0 & 30 & 0 & 0 \\
\end{array}
$$
Pooling layers offer ways to flexibly summarize information coming from an image area.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>30</th>
<th>30</th>
<th>30</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Faces and their parts
Representing an eye with $3 \times 3$ filters is like a puzzle with $3 \times 3$ pieces.
Stacked convolutional and pooling layers learn both the puzzle pieces and their combinations.
How does a CNN model learn?
Training data and optimization are needed
How does a CNN model learn?
Training data and optimization are needed.
The resulting neural net by Google could not only recognize cats but also draw AI-cats!

Alexander Mordvintsev: *Father Cat*, 2015.
Outline

Introduction to neural networks

Convolutional neural networks

EIT and the enclosure method

The enclosure method: least-squares computation

The enclosure method: deep learning
The mathematical model of EIT is the inverse conductivity problem introduced by Calderón

Let $\Omega \subset \mathbb{R}^2$ be the unit disc and let conductivity $\sigma : \Omega \to \mathbb{R}$ satisfy

$$0 < M^{-1} \leq \sigma(z) \leq M.$$  

Applying voltage $f$ at the boundary $\partial \Omega$ leads to the elliptic PDE

$$\begin{cases} 
\nabla \cdot \sigma \nabla u = 0 & \text{in } \Omega, \\
|_{\partial \Omega} = f. 
\end{cases}$$

Boundary measurements are modelled by the Dirichlet-to-Neumann map

$$\Lambda_{\sigma} : f \mapsto \sigma \frac{\partial u}{\partial n}|_{\partial \Omega}.$$  

Calderón’s problem is to recover $\sigma$ from the knowledge of $\Lambda_{\sigma}$. It is a nonlinear and ill-posed inverse problem.
The inclusion detection problem

Let $\Omega \subset \mathbb{R}^2$ be a bounded, simply connected $C^\infty$ domain and

$$\sigma = 1 + \chi_D h.$$ 

Here $D \subset \Omega$ and $h \in L^\infty(D)$ is such that $\sigma$ is bounded away from zero and has a jump along $\partial D$. We assume that the conductivity $\sigma$ is strictly positive.

The goal of inclusion detection is to use EIT measurements, here $\Lambda_\sigma$, to extract information about $D$.

[Ikehata 2000], [Ikehata and S 2000], [Brühl and Hanke 2000]
The support function $h_D(\omega)$

Define the support function of inclusion $D \subset \mathbb{R}^2$ with direction $\omega \in S^1$ by

$$h_D(\omega) := \sup_{x \in D} x \cdot \omega.$$ 

Here $\Omega$ is the unit disc.
The jump condition

For \( \omega \in S^1 \) and \( \delta > 0 \), define

\[
D_\omega(\delta) = \{ x \in D \mid h_D(\omega) - \delta < x \cdot \omega \leq h_D(\omega) \}.
\]

The conductivity \( \sigma = 1 + \chi_D h \) satisfies the jump condition if for each \( \omega \in S^1 \), there exist positive constants \( C_\omega \) and \( \delta_\omega \) such that

\[
\begin{align*}
    h(x) &\geq C_\omega \quad \text{for almost all } x \in D_\omega(\delta_\omega), \text{ or} \\
    -h(x) &\geq C_\omega \quad \text{for almost all } x \in D_\omega(\delta_\omega).
\end{align*}
\]
Illustration of the jump condition

$\text{OK.}$
Illustration of the jump condition

Not OK!
Calderón's exponential

Let $\omega$ be a unit vector, and denote by $\omega^\perp$ the vector $\omega$ rotated counterclockwise by angle $\pi/2$. Then $\omega \cdot \omega^\perp = 0$.

For each $\tau > 0$ and $x \in \mathbb{R}^2$ set $f_\omega(x; \tau) := e^{\tau x \cdot \omega + i \tau x \cdot \omega^\perp}$. 
Calderón’s exponential

Let \( \omega \) be a unit vector, and denote by \( \omega^\perp \) the vector \( \omega \) rotated counterclockwise by angle \( \pi/2 \). Then \( \omega \cdot \omega^\perp = 0 \).

For each \( \tau > 0 \) and \( x \in \mathbb{R}^2 \) set \( f_\omega(x; \tau) := e^{\tau x \cdot \omega + i \tau x \cdot \omega^\perp} \).

In the plot we have \( \omega = \left[ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]^T \).
Calderón’s exponential

Let $\omega$ be a unit vector, and denote by $\omega^\perp$ the vector $\omega$ rotated counterclockwise by angle $\pi/2$. Then $\omega \cdot \omega^\perp = 0$.

For each $\tau > 0$ and $x \in \mathbb{R}^2$ set $f_\omega(x; \tau) := e^{\tau x \cdot \omega} + i \tau x \cdot \omega^\perp$.

In the plot we have $\omega = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$. 

$\tau=3$

$\text{Re}(f_\omega(x; \tau))$

$\text{Im}(f_\omega(x; \tau))$
Calderón’s exponential

Let $\omega$ be a unit vector, and denote by $\omega^\perp$ the vector $\omega$ rotated counterclockwise by angle $\pi/2$. Then $\omega \cdot \omega^\perp = 0$.

For each $\tau > 0$ and $x \in \mathbb{R}^2$ set $f_\omega(x; \tau) := e^{\tau x \cdot \omega + i \tau x \cdot \omega^\perp}$.

In the plot we have $\omega = \left[ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]^T$. 

\[
\begin{align*}
\tau &= 5 \\
\text{Re}(f_\omega(x; \tau)) &\quad \text{Im}(f_\omega(x; \tau))
\end{align*}
\]
Calderón’s exponential

Let \( \omega \) be a unit vector, and denote by \( \omega^\perp \) the vector \( \omega \) rotated counterclockwise by angle \( \pi/2 \). Then \( \omega \cdot \omega^\perp = 0 \).

For each \( \tau > 0 \) and \( x \in \mathbb{R}^2 \) set \( f_\omega(x; \tau) := e^{\tau x \cdot \omega + i \tau x \cdot \omega^\perp} \).

In the plot we have \( \omega = \left[ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]^T \).
Calderón’s exponential

Let $\omega$ be a unit vector, and denote by $\omega^\perp$ the vector $\omega$ rotated counterclockwise by angle $\pi/2$. Then $\omega \cdot \omega^\perp = 0$.

For each $\tau > 0$ and $x \in \mathbb{R}^2$ set $f_\omega(x; \tau) := e^{\tau x \cdot \omega + i \tau x \cdot \omega^\perp}$.

In the plot we have $\omega = [0, 1]^T$. 

![Graph of the exponential function $f_\omega(x; \tau)$ with $\tau = 20$. The plot shows the real and imaginary parts of the function for $x \in \mathbb{R}^2$. The unit vector $\omega$ is plotted as $[0, 1]^T$.](image)
Calderón’s exponential and Ikehata’s indicator function

Let \( \omega \) be a unit vector, and denote by \( \omega^\perp \) the vector \( \omega \) rotated counterclockwise by angle \( \pi/2 \). Then \( \omega \cdot \omega^\perp = 0 \).

For each \( \tau > 0 \) and \( x \in \mathbb{R}^2 \) set

\[
f_\omega(x; \tau) := e^{\tau x \cdot \omega + i \tau x \cdot \omega^\perp}.
\]

The indicator function \( I_\omega(\tau) \) is defined by

\[
I_\omega(\tau) = \langle (\Lambda_\sigma - \Lambda_1) \overline{f_\omega(\cdot; \tau)}, f_\omega(\cdot; \tau) \rangle.
\]

The fundamental theorem of enclosure method says that

\[
h_D(\omega) = \lim_{\tau \to \infty} \frac{\log |I_\omega(\tau)|}{2\tau},
\]
Measurement noise makes the evaluation of the indicator function unstable for large values of $\tau$

true values of $\frac{1}{2} \log |I_\omega(\tau)|$

noisy values of $\frac{1}{2} \log |I_\omega(\tau)|$
Outline

Introduction to neural networks

Convolutional neural networks

EIT and the enclosure method

The enclosure method: least-squares computation

The enclosure method: deep learning
Here are a few examples of convex hulls calculated with 45 uniformly distributed vectors $\omega$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$
We recover the support function approximately using least-squares fitting of a line to \( \frac{1}{2} \log |I_\omega(\tau)| \)
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to \( \frac{1}{2} \log |l_\omega(\tau)| \)
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$
How about a more difficult case?
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to \( \frac{1}{2} \log |l_\omega(\tau)| \).
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to \( \frac{1}{2} \log |I_\omega(\tau)| \)
We recover the support function approximately using least-squares fitting of a line to \( \frac{1}{2} \log |I_\omega(\tau)| \)
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_{\omega}(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |I_{\omega}(\tau)|$. 
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$.
We recover the support function approximately using least-squares fitting of a line to $\frac{1}{2} \log |l_\omega(\tau)|$
Outline

Introduction to neural networks

Convolutional neural networks

EIT and the enclosure method

The enclosure method: least-squares computation

The enclosure method: deep learning
We give the indicator functions to the neural net in the form of a $21 \times 50$ image.

Horizontal axis: angles 1, 2, \ldots, 44, 45, 1, 2, 3, 4, 5.
Vertical axis: 7 values of indicator function thrice.
We give the indicator functions to the neural net in the form of a $21 \times 50$ image.

Horizontal axis: angles $1, 2, \ldots, 44, 45, 1, 2, 3, 4, 5$.
Vertical axis: 7 values of indicator function thrice.
After many experiments we ended up choosing this network architecture.

There are six layers in our convolutional neural network. Layers that involve learnable parameters are marked with this color.

**Image Input**
- 21x50 images with zero mean normalization

**Convolution**
- 120 filters of size 6x4, three-pixel zero padding

**Leaky ReLU**
- Leaky ReLU with scale 0.01

**Fully Connected**
- Fully connected layer with 45 outputs

**Tanh**
- Hyperbolic tangent layer

**Regression Output**
Training data: 19 000 cases with round inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

There are 1–4 disc-shaped inclusions, random centers.

Disc radii are random, uniformly distributed in the interval $[0.05, 0.2]$.

The conductivity inside each inclusion is random with uniform distribution in either $[1.5, 5]$ or in $[0.2, 0.667]$. 
Training data: 19,000 cases with round inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

There are 1–4 disc-shaped inclusions, random centers.

Disc radii are random, uniformly distributed in the interval $[0.05, 0.2]$.

The conductivity inside each inclusion is random with uniform distribution in either $[1.5, 5]$ or in $[0.2, 0.667]$.
Training data: 19 000 cases with round inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

There are 1–4 disc-shaped inclusions, random centers.

Disc radii are random, uniformly distributed in the interval $[0.05, 0.2]$.

The conductivity inside each inclusion is random with uniform distribution in either $[1.5, 5]$ or in $[0.2, 0.667]$. 
Training data: 19 000 cases with round inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

There are 1–4 disc-shaped inclusions, random centers.

Disc radii are random, uniformly distributed in the interval \([0.05, 0.2]\).

The conductivity inside each inclusion is random with uniform distribution in either \([1.5, 5]\) or in \([0.2, 0.67]\).
Relative error measurement for enclosure method

Denote the true convex hull by $\mathcal{C} \subset \mathbb{R}^2$ and the reconstructed convex hull by $\mathcal{B} \subset \mathbb{R}^2$. How to measure the quality of $\mathcal{B}$ as an approximation to $\mathcal{C}$ quantitatively?

We use the relative error

$$\frac{|\mathcal{C} \setminus \mathcal{B}| + |\mathcal{B} \setminus \mathcal{C}|}{|\Omega|} \cdot 100\%,$$

where $|\cdot|$ is area. Here $|\Omega| = \pi$ since we work in the unit disc.
Reconstruction examples

LS fit error 1235%  AI error 7%  True convex hull
Reconstruction examples

LS fit error 59%

AI error 32%

True convex hull
Reconstruction examples

LS fit error 110%

AI error 21%

True convex hull
Reconstruction examples

LS fit error 116%  Al error 16%  True convex hull
Reconstruction examples

LS fit error 66%

AI error 7%

True convex hull
Test data: 1000 cases with elliptic inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

Five randomly placed and oriented elliptic inclusions.

Each semi-major axis random with half-length $R$ uniformly distributed in the interval $[0.04, 0.22]$; each semi-minor axis $r$ in the interval $[0.04, R]$

The conductivity inside each inclusion is random with uniform distribution in either $[1.3, 7]$ or in $[0.14, 0.77]$. 
Test data: 1000 cases with elliptic inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

Five randomly placed and oriented elliptic inclusions.

Each semi-major axis random with half-length $R$ uniformly distributed in the interval $[0.04, 0.22]$; each semi-minor axis $r$ in the interval $[0.04, R]$

The conductivity inside each inclusion is random with uniform distribution in either $[1.3, 7]$ or in $[0.14, 0.77]$. 
Test data: 1000 cases with elliptic inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

Five randomly placed and oriented elliptic inclusions.

Each semi-major axis random with half-length $R$ uniformly distributed in the interval $[0.04, 0.22]$; each semi-minor axis $r$ in the interval $[0.04, R]$.

The conductivity inside each inclusion is random with uniform distribution in either $[1.3, 7]$ or in $[0.14, 0.77]$. 
Test data: 1000 cases with elliptic inclusions

Background: homogeneous conductivity 1 defined in the unit disc.

Five randomly placed and oriented elliptic inclusions.

Each semi-major axis random with half-length $R$ uniformly distributed in the interval $[0.04, 0.22]$; each semi-minor axis $r$ in the interval $[0.04, R]$.

The conductivity inside each inclusion is random with uniform distribution in either $[1.3, 7]$ or in $[0.14, 0.77]$. 
Reconstruction examples: test data

LS fit error 99%

AI error 12%

True convex hull
Reconstruction examples: test data

LS fit error 71%  AI error 9%  True convex hull
Reconstruction examples: test data

LS fit error 83%

AI error 8%

True convex hull
Reconstruction examples: test data

LS fit error 89%  AI error 12%  True convex hull
Reconstruction examples: test data

LS fit error 64%

AI error 12%

True convex hull
Error histograms for least-squares approach and AI

**Least-squares fit relative errors (%)**

**Deep learning relative errors (%)**
Error histograms for false negatives

Least-squares fit false negatives (%)

Deep learning false negatives (%)

0 10 20 30 40 50 60

0 5 10 15

0 10 20 30 40 50 60

0 5 10 15
Error histograms for false positives

Least-squares fit false positives (%)

Deep learning false positives (%)

0 10 20 30 40 50 60

0 50 100
Conclusion

When solving ill-posed inverse problems with neural networks, use a classical inversion method for extracting robust features (as opposed to learning from raw data).

Analytic understanding of the features leads to gray-box learning (instead of black-box learning).

Future work

- Add comparison to enhanced least-squares approach
- Compare the above approach to learning directly from raw data
- Use the complete electrode model
- Work with measured data
- Apply to other PDE where enclosure method works
Links to open computational resources

Open EIT datasets:
- Finnish Inverse Problems Society (FIPS) dataset page

Reconstruction algorithms: FIPS Computational Blog
- The D-bar Method for EIT—Simulated Data
- The D-bar Method for EIT—Experimental Data

For these slides see: http://www.siltanen-research.net/talks.html
Thank you for your attention!

← Slime mold called Lycogala conicum