Understanding Neural Networks for Image Segmentation

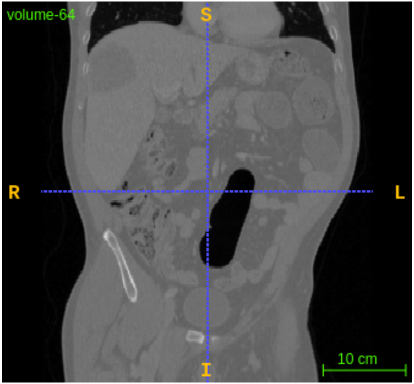
Jonas Actor

Rice University

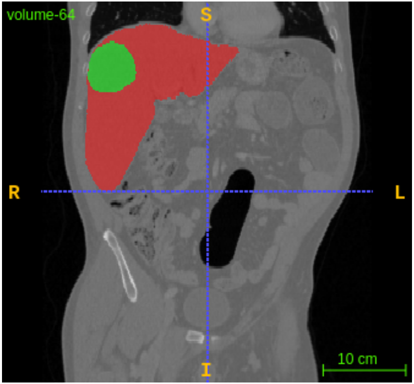
6 November 2019

jonasactor@rice.edu

Target application: medical image segmentation



Abdominal CT scan



Liver and tumor segmentation

Why automate?

- needed for treatment plans
- costly (time+effort) to perform by hand
- less interobserver variability $\rightarrow$ better accuracy

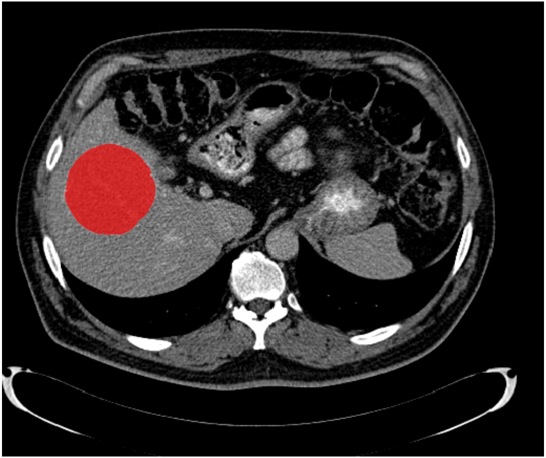
errors in segmentation = errors in radiation treatment

Goal: understand why CNNs work so well

1. Compare PDEs to CNNs
   - Aside: Upwind schemes
2. Build CNNs like PDEs
3. Analyze kernels and explain performance
   - Entrywise comparison
   - SVD comparison

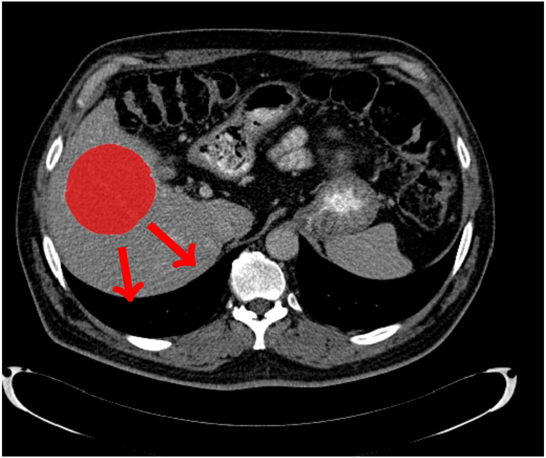
Table of Contents

Classical Approach: Solve a PDE



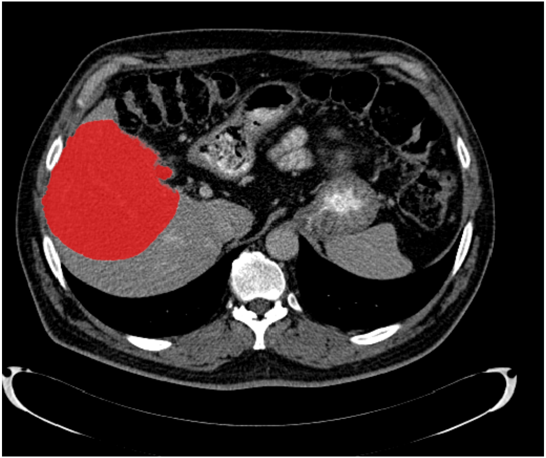
time = 200

Classical Approach: Solve a PDE



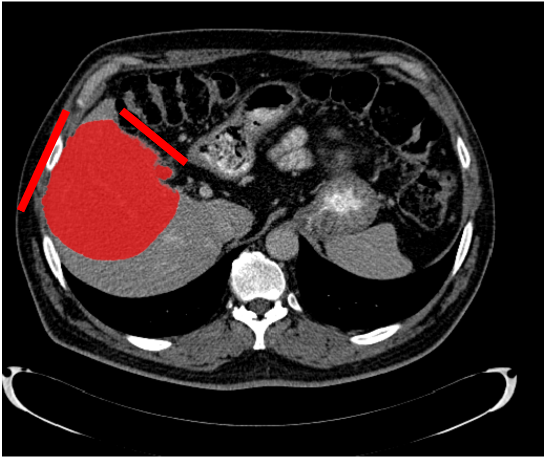
time = 200

Classical Approach: Solve a PDE



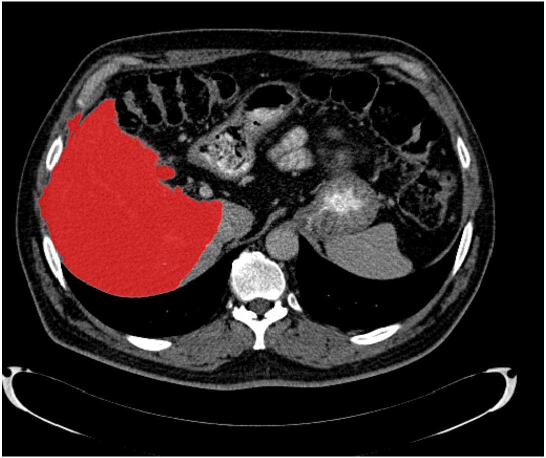
time = 500

Classical Approach: Solve a PDE



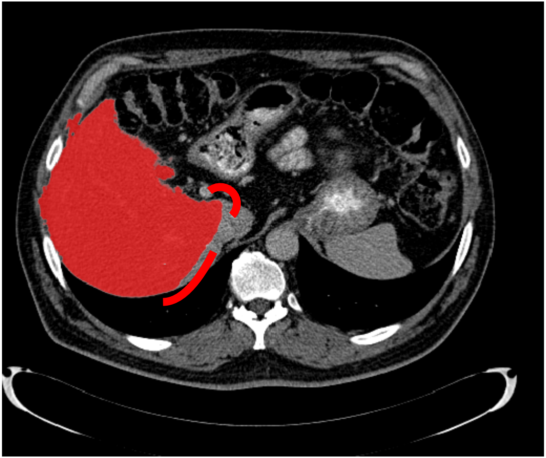
time = 500

Classical Approach: Solve a PDE



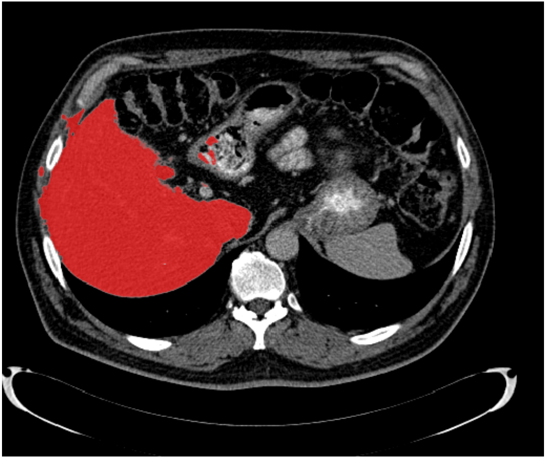
time = 1100

Classical Approach: Solve a PDE



time = 1100

Classical Approach: Solve a PDE



time = 1500

PDE of choice: Level Set Equation

$$\partial_t u - \alpha \underbrace{g_I}_{\text{balloon force}} - \beta \underbrace{g_I \kappa(u)}_{\text{mean curvature}} - \gamma \underbrace{\nabla g_I \cdot \nabla u}_{\text{convection}} = 0$$

- Well-established theory to analyze approximation, stability
- Upwind finite differences + fast marching method
- Semiautomated : requires initialization by user
- Works for simple problems only : relies on edge information

Why upwind schemes?

- convergence like finite differences
- improved stability

M-Matrices

Definition

A matrix $A \in \mathbb{R}^{n \times n}$ is a *M-Matrix* if:

1. $a_{ij} \leq 0$ for all $i \neq j$
2. A^{-1} exists
3. $A^{-1} \geq 0$

Checking for M-matrices

Theorem

The matrix $A \in \mathbb{R}^{n\times n}$ is a M-matrix if and only if:

1. $a_{ij} \leq 0$ *for* $i \neq j$
2. $\exists \mathbf{x} \in \mathbb{R}^n$ *such that* $A\mathbf{x} > 0$.

The vector $\mathbf{x}$ *is called the majorizing element of* A.

M-matrices provide stable numerical methods

An example: convection

Consider on $\Omega = [0, 1]$ the convection equation

$$\partial_t u = a(x)\partial_x u \qquad\qquad + \text{ Dirichlet BC's}$$

convection *a* both positive and negative

Discretization

Discretize in space with a finite difference stencil:

$$\partial_x u_i \approx \frac{1}{h} \sum_{j=-1}^{1} w_{i,j} u_{i+j}$$

Examples:

- $w_{i,-1} = -1$, $w_{i,0} = 1$, $w_{i,1} = 0$: forward FD
- $w_{i,-1} = \frac{-1}{2}$, $w_{i,0} = 0$, $w_{i,1} = \frac{1}{2}$: centered FD

Discretization

$$\partial_t u = a(x) \partial_t u$$

$$\partial_t u_i \approx a_i \sum_{j=-1}^{1} \frac{1}{h} w_{i,j} u_{i+j}$$

$$\partial_t \mathbf{u} = L \mathbf{u}$$

Upwind scheme

Choose our **upwind scheme**:

$$\begin{aligned} w_{i,-1} &= -\max\{0, sgn(a_i)\} \\ w_{i,0} &= \quad 1 \\ w_{i,1} &= \quad \min\{0, sgn(a_i)\} \end{aligned}$$

Upwind scheme

$$
\begin{aligned}
L_{i,i-1} &= -\frac{1}{h}\max\{0, a_i\} \\
L_{i,i} &= \quad \frac{1}{h}|a_i| \\
L_{i,i+1} &= \quad \frac{1}{h}\min\{0, a_i\}
\end{aligned}
$$

Stability of our discretization

Lemma

L is an M-matrix.

Proof.

- $L_{i,j} \leq 0$ for $i \neq j$
- The vector $\mathbf{x}$ with $x_i = ih$ is a majorizing element.

Comparison: PDE vs CNN

	Can analyze?	Accurate?
PDE	yes	sometimes
CNN	no	yes

Goal: accurate method we can analyze

Similarities between LSE and CNN

	LSE	CNN
Convolution	finite difference kernel $\frac{1}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	learned kernel $\begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix}$
ReLU	upwind scheme $\max(0, D^+ * u) + \min(0, D^- * u)$	activation function $\max(0, K * x + b)$

Table of Contents

Build a CNN like a LSE solver

Level Set Equation $\longrightarrow$ Level Set Network

1. Discretize Level Set Equation
   - Explicit forward Euler in time
   - Upwind finite differences in space
2. Forward Euler $\longrightarrow$ residual skip connections
3. Upwind finite differences $\longrightarrow$ convolutions and ReLU

Forward Euler time discretization

$$\frac{u^{(t+1)} - u^{(t)}}{\delta_t} - \alpha g_I - \beta g_I \kappa(u^{(t)}) - \gamma \nabla g_I \cdot \nabla u^{(t)} = 0$$

Edge indicator g_I

$$g_I(x) = \frac{1}{1 + \|\nabla I(x)\|^2}$$

becomes

$$g_I(x) = \frac{1}{1 + \sum_{j=1}^{N_{conv}} (\sigma_j * I(x))^2}$$

Convection term: Upwind Finite Differences

$$
\begin{aligned}
\nabla g_I \cdot \nabla u^{(k)} &= \max\{0, \partial_x g_{\mathcal{I}}\} D_x^+ * u^{(k)} - \max\{0, -\partial_x g_{\mathcal{I}}\} D_x^- * u^{(k)} \\
&+ \max\{0, \partial_y g_{\mathcal{I}}\} D_y^+ * u^{(k)} - \max\{0, -\partial_y g_{\mathcal{I}}\} D_y^- * u^{(k)} \\
&= \mathrm{ReLU}(\partial_x g_{\mathcal{I}}) D_x^+ * u^{(k)} - \mathrm{ReLU}(-\partial_x g_{\mathcal{I}}) D_x^- * u^{(k)} \\
&+ \mathrm{ReLU}(\partial_y g_{\mathcal{I}}) D_y^+ * u^{(k)} - \mathrm{ReLU}(-\partial_y g_{\mathcal{I}}) D_y^- * u^{(k)}
\end{aligned}
$$

Curvature term $\kappa(u)$

$$\kappa(u^{(k)}) = \nabla \cdot \left(\frac{\nabla u^{(k)}}{\left\| \nabla u^{(k)} \right\|} \right)$$

becomes

$$\begin{aligned}
\nabla \hat{u}_i^{(k)} &= \rho_i * u^{(k)} \\
\kappa(u^{(k)}) &= \sum_{j=1}^{N_{conv}^2} \sigma_j * \left(\sum_{i=1}^{N_{conv}^1} \frac{\nabla \hat{u}_i^{(k)}}{\left\| \nabla \hat{u}_i^{(k)} \right\| + \varepsilon} \right)
\end{aligned}$$

Implementation

- If convolution kernels are finite difference kernels, we recover LSE
- If convolution kernels are learned, we construct LSN
- Python + Tensorflow / Keras implementation
- Our LSE agrees with ITK-SNAP, a common LSE-segmentation program

LSN: Results

K-Fold	LSE	LSN Test	LSN Validation	UNet
0	0.736	0.837	0.619	0.912
1	0.600	0.847	0.729	0.919
2	0.483	0.116	0.005	0.874
3	0.730	0.827	0.606	0.895
4	0.643	0.831	0.596	0.915
Avg	0.604	0.692	0.511	0.903
Avg$-\{2\}$	0.640	0.837	0.638	0.911

Table: DSC scores for each fold, from training the level set network.

Table of Contents

Identification of Kernels

- Are CNN convolution kernels finite difference stencils?
- Are they *close* to finite difference stencils?
- What about other standard image processing kernels?

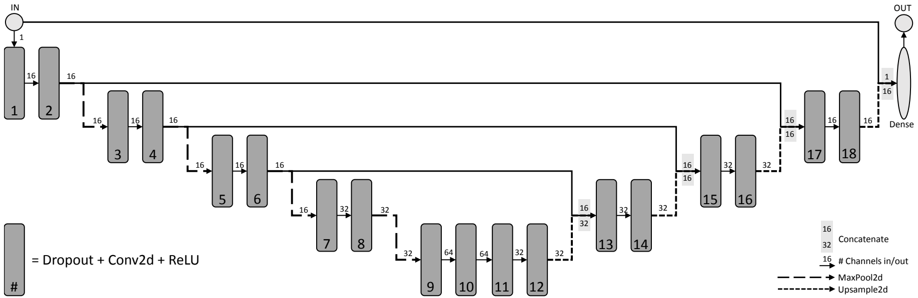
Numerical analysis kernels

- Laplacian
- Edge detection
- Identity

Image processing kernels

- Gaussian blur
- Local mean
- Sharpen

Setup of CNN



Trained on MICCAI LiTS 2017 dataset for liver segmentation

Kernel Analysis: Comparing entries

- For each layer, separate each channel's 3×3 convolution kernel
- Flatten each 3×3 kernel into a vector $\in \mathbb{R}^9$
- Cluster with k-means
- Project down using PCA
- Project known numerical analysis and image processing kernels

Kernel Clustering Results

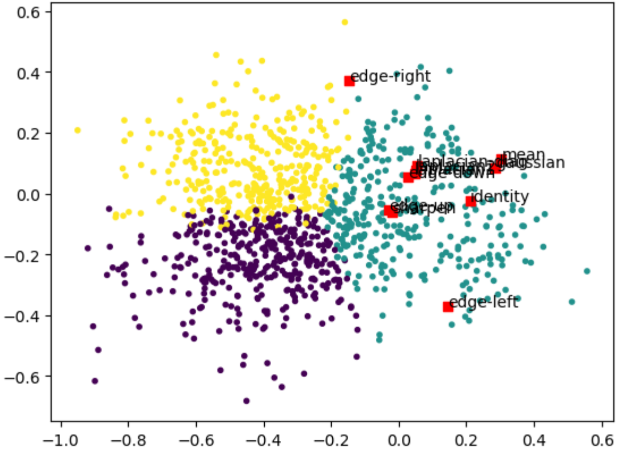


Figure: Convolution Layer 11 (encoder)

Kernel Clustering Results

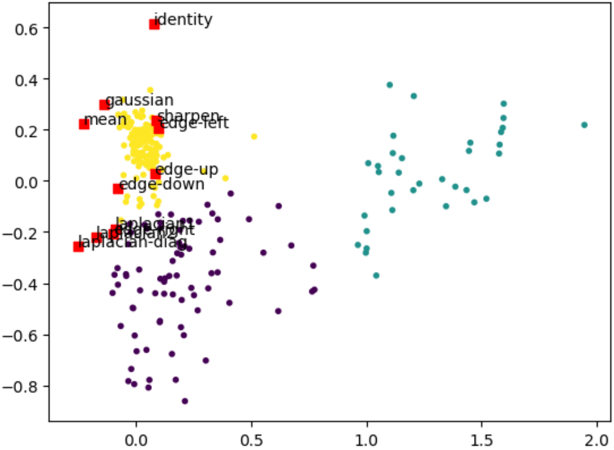


Figure: Convolution Layer 14 (decoder)

Kernel Analysis: Comparing actions

1. Construct matrix $A_{[K]} \in \mathbb{R}^{n_x n_y \times n_x n_y}$ describing convolution with K
2. Compute singular values of linear operator
3. Compute singular values of clinical image processing kernels
4. Assign closest clinical feature F that has smallest spectral distance to K

Kernel Analysis: Comparing actions

$$\text{Kernel } K = \begin{bmatrix} k_{-1,-1} & k_{-1,0} & k_{-1,1} \\ k_{0,-1} & k_{0,0} & k_{0,1} \\ k_{1,-1} & k_{1,0} & k_{1,1} \end{bmatrix}$$

Kernel Analysis: Comparing actions

For $i \in \{-1, 0, 1\}$,

$$T_i = \begin{bmatrix} k_{i,0} & k_{i,1} & & & \\ k_{i,-1} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & k_{i,1} \\ & & & k_{i,-1} & k_{i,0} \end{bmatrix} \in \mathbb{R}^{n_x \times n_x}$$

Kernel Analysis: Comparing actions

$$A_{[K]} = \begin{bmatrix} T_0 & T_1 & & & \\ T_{-1} & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & T_1 \\ & & & T_{-1} & T_0 \end{bmatrix} \in \mathbb{R}^{n_x n_y \times n_x n_y} \qquad n_y \times n_y \text{ blocks}$$

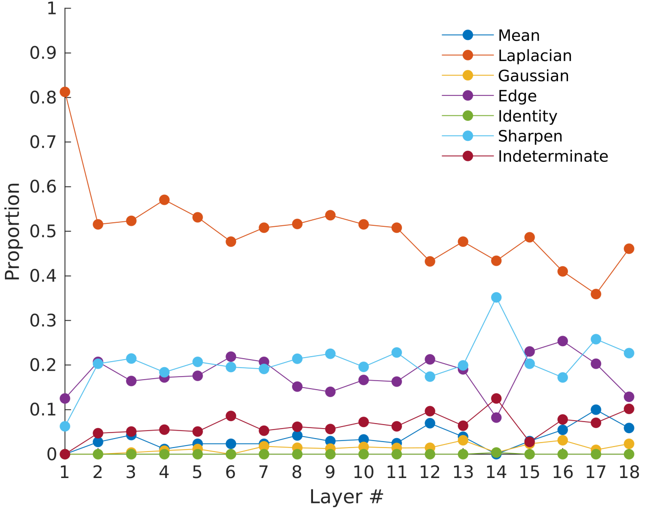
Kernel Analysis: Comparing actions

$$A_{[K]} = U_K \Sigma_K V_K^T \qquad \forall K \in \{\text{layers}\}$$
$$A_{[F]} = U_F \Sigma_F V_F^T \qquad \forall F \in \{\text{features}\}$$

$$\text{find} \quad \underset{F \in \{\text{features}\}}{\arg\min} \; \|\Sigma_K - \Sigma_F\|_1$$

Label as ‘indeterminate’ if not within 10% of largest singular value

Kernel Analysis Results



Conclusions

- Level Set Equation $\neq$ Level Set Network $\neq$ UNet
- Framework for using same operations (convolutions + ReLU) for both NNs and PDEs
- Framework for GPU-supported finite differences in Tensorflow
- Examine how learned CNN kernels change across different layers

JAA is supported by the NLM Training Program in Biomedical Informatics & Data Science (T15LM007093), supplemented by the Ken Kennedy Institute Computer Science & Engineering Enhancement Fellowship, funded by the Rice Oil & Gas HPC Conference.