Cytoarchitectonic Segmentation of Human Brain Areas with Convolutional Neural Networks on HPC Systems

PADC Workshop
12.04.2018 | CHRISTIAN SCHIFFER & HANNAH SPITZER (TEAM BIG DATA ANALYTICS, INM-1)
Aim: High resolution human brain cytoarchitectonic atlas

- Cut brain in ~7400 20µm thick sections
- Stain cell bodies and scan in light microscope at 1µm resolution
- Delineate brain areas in every 60th section of 10 different brains
- Average annotations in common reference space to obtain probabilistic maps
Cytoarchitectonic areas

- Distinguished by variations of cell distribution in cortical laminae and with respect to columnar organization
- Schleicher et al., 1999: Observer independent method for parcellation

Time and labor intensive, does not scale with high throughput imaging

Support and speed up mapping process using Deep Learning

Challenges for automatic brain area segmentation

Methodical challenges

• Limited amount of labelled training data
• Noisy data due to staining and sectioning artifacts, changing angle between sectioning plane and brain surface (oblique cuts)

Technical challenges

• One section: 125,000x90,000px (1µm resolution), approx. 10 GByte per image (8 bit per pixel)
• Training on 100+ sections
• Data I/O strategy? GPU Memory restrictions?
Supervised segmentation using CNNs

- U-Net like architecture (Ronneberger et al, MICCAI 2015)
- Combine high resolution patch input with prior knowledge about topology
Self-supervised geodesic distance task

Leverage available unlabelled data using 3D relationship between sections

Pre-train on self-supervised distance task

\[ f(x_1) \]

\[ f(x_2) \]

\[ \hat{y}_{\text{dist}} \]

Shared weights

Fine-tune on supervised area segmentation task

\[ x_1 \]

\[ x_2 \]

\[ y_{\text{dist}} \]
With atlas prior

Member of the Helmholtz Association

12.04.2018
Pre-trained on self-supervised task

Member of the Helmholtz Association

12.04.2018
Training workflow on HPC systems
Training workflow

GPFS → CPU → GPU
Training workflow

GPFS → CPU → GPU
Training workflow

GPFS → CPU → GPU
Ressource utilization

GPFS

CPU

100%

GPU

25%
Ressource utilization

GPFS

CPU

CPU

CPU

GPU

100%
Ressource utilization

Member of the Helmholtz Association

12.04.2018
Multi-experiment training

experiment 1

experiment 2

experiment 3

experiment 4
Single-experiment training

experiment
Synchronized training

batch 1 ➔ GPU 1 ➔ gradients 1 ➔ gradients 1-4

batch 2 ➔ GPU 2 ➔ gradients 2 ➔ gradients 1-4

batch 3 ➔ GPU 3 ➔ gradients 3 ➔ gradients 1-4

batch 4 ➔ GPU 4 ➔ gradients 4 ➔ gradients 1-4

Average Allreduce
Synchronized training

- **Node 1**
- **Node 3**
- **Node 2**
- **Node 4**

**Average Allreduce (e.g. MPI)**

- **Individual gradients**
- **Averaged gradients**
Performance analysis

- Measure processed patches per second
- Implementation using TensorFlow and MPI
- Horovod adds MPI operations to TensorFlow
- Comparison of JURON and JURECA

Sources: www.tensorflow.org, www.github.com/uber/horovod
Processed images per second compared to number of GPUs

- **Fast** (green dots)
- **Slow** (red dots)
- **Optimal** (dashed red line)

**JURON**

**JURECA**

# Processed images per second vs. # GPUs.
I/O performance on JURON

Time to read a 2025x2025 Byte patch from disk in seconds

- **Mean**
- **Min**

<table>
<thead>
<tr>
<th>Name of node</th>
<th>juronc06</th>
<th>juronc07</th>
<th>juronc09</th>
<th>juronc10</th>
<th>juronc11</th>
<th>juronc15</th>
<th>juronc16</th>
<th>juronc17</th>
<th>juronc18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time for file access in seconds</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Synchronization problem due to I/O performance

measured time (GPU 1)

GPU 1
- IO
- forward pass
- backward pass

GPU 2
- IO
- forward pass
- backward pass

GPU 3
- IO
- forward pass
- backward pass

GPU 4
- IO
- forward pass
- backward pass

time
Synchronization problem due to I/O performance

- **GPU 1**: IO, forward pass, wait, backward pass
- **GPU 2**: IO, forward pass, backward pass
- **GPU 3**: IO, forward pass, wait, backward pass
- **GPU 4**: IO, forward pass, wait, backward pass

**measured time** (GPU 1)

**time**
Conclusion

- JURON significantly outperforms JURECA w.r.t. GPU performance
- I/O problems have to be resolved to allow linear scaling during training
- Results on JURECA show that near linear scaling is possible
- HPC systems help to train large neural networks on high volume input data
Thank you for your attention

Hannah Spitzer
Email: h.spitzer@fz-juelich.de
Phone: +49 2461 61-5960

Christian Schiffer
Email: c.schiffer@fz-juelich.de
Phone: +49 2461 61-5960

Team Big Data Analytics
Institute of Neuroscience and Medicine (INM-1)