Electron microscopes (EM) produce nanometer-scale images.
SERIAL BLOCK-FACE IMAGING

Serial block-face scanning electron microscopy (SBF-SEM): Image huge 3D samples by repeated cutting and scanning.

(Denk, Horstmann 2004)
SBF-SEM APPLICATIONS

SBF-SEM images provide new insight into the organization of complex biological systems.

Connectomics

Systems biology

(vimeo.com/101018819)

(Pokrovskaya et al., 2016)
Image segmentation: Partition image pixels into labeled regions corresponding to image content.

Natural image

EM image
**IMAGE SEGMENTATION**

**Image segmentation**: Partition image pixels into labeled regions corresponding to image content.

- **Natural image**
- **EM image**
Manual segmentation is infeasible for large SBF-SEM images.

**Automated segmentation**: algorithmically classify each pixel, manually correct.

**Practical** segmentation algorithm: Manual correction of algorithm output is much faster than manual segmentation.
A practical segmentation algorithm requires high (> 99.9%) accuracy despite:

- **Noise + small objects**
- **Difficult label assignment**
DEEP LEARNING FOR SEGMENTATION

Sliding window network: Convolutional neural network classifies one pixel at a time.

(Ciresan et al., 2012)

Encoder-decoder network (U-net): Convolutional encoding and decoding paths classify large image patches at once.
**ENCODER-DECODER NETWORKS**

**Encoding path:** Convolution, pooling operations decompose an image into a multiscale collection of features.

**Decoding path:** Convolution, transposed convolution operations synthesize a new image from encoder features.

(Ronneberger et al., 2015)
Many design choices required for building encoder-decoder network architectures.
- Convolution kernel size
- Convolution kernels per layer
- Convolution layers per stack
- Use dropout?
- Use batch normalization?
- Convolution layer regularization

Design choices can be represented as numeric hyperparameters (HPs).

Architecture design ⇔ HP space search.
Two optimization problems when applying neural networks to a problem domain.

**Learning**: Optimize network weight parameters. Parameter ranges are continuous, objective function is (sub)differentiable.
- Evaluation is cheap, optimize with backpropagation.

**Architecture design**: Optimize network HPs. Mix of continuous and discrete ranges, objective function is not differentiable.
- Evaluation is expensive, optimization is an unstructured search.
**THE GENENET LIBRARY**

**genenet**: Build, train, and deploy encoder-decoder networks for segmentation using Python and TensorFlow.

**Goal**: simple network design for humans and algorithms.

Build computation graphs from **Gene graphs**.
**Computation graph**: sequence of functions mapping network input to output.

**Gene graph**: A height-\( n \) tree of Genes that builds a computation graph.

**Gene**: Gene graph node. Each builds a subgraph (module) in the computation graph.
THE GENE GRAPH

**Leaf** Genes (height 0) build small modules.

**Internal** Genes (height $i > 0$) assemble their child Gene constructions into larger modules.

**Root** Gene (height $n$) assembles a full computation graph in TensorFlow.
PathGene
NetGene
Height-\(i\) Gene \(g_i\) with ancestors \(g_{i+1}, \ldots, g_n\).

HP \(h_{(n_{convlayers})}\) has value \(h_i\) at Gene \(g_i\).

Each \(g_i\) tracks a \textit{delta} value \(\Delta h_i\).

\[
h_i = \Delta h_i + \Delta h_{i+1} + \cdots + \Delta h_n.
\]
Changing $\Delta h_n$ affects $h$ for the whole Gene graph.

Changing $\Delta h_0$ affects $h$ for $g_0$ and its descendants.

Allows for easy random network generation.

Choose feasible regions for HPs (one for height $n$, another for height $i < n$).

Sample values from height $n$ downward.
**Classifier ensemble**: Take some classifiers, average their predictions.

For EM segmentation, form an ensemble from high-performing neural networks.

Diverse network architectures contribute to high ensemble ambiguity, improving performance (Krogh, Vedelsby 1995).
PRELIMINARY SBF-SEM RESULTS

Our lab imaged a human platelet sample with a Gatan 3View.

**Goal:** Segment cells and 5 organelle types in a $250 \times 2000 \times 2000$ volume.
**Biowulf**: NIH high-performance computing cluster.

Train networks on NVIDIA K80 GPUs.

Create training jobs with Bash, load *Singularity* containers on Biowulf nodes, run *genenet* scripts.
Lab members manually segmented a $50 \times 800 \times 800$ subvolume.

We trained 80 random networks for 100000 iterations on Biowulf.

**Mutable HPS:**
- $n_{\text{convkernels}}$
- $n_{\text{convlayers}}$
- $\log_{\text{learning_rate}}$
- $n_{\text{stacks}}$
- $\text{input\_size}$
- Regularization HPs
RANDOM NETWORK PERFORMANCE

Below: Comparison of random network validation performance (adjusted Rand score) with the original (Ronneberger et al., 2015) u-net.

Nine networks outperformed the original u-net.
RANDOM NETWORK PERFORMANCE
**ENSEMBLE PERFORMANCE**

**Strategy:** Make an ensemble of the best $N$ networks, evaluate on validation data. $N = 4$ is best.
The point: Is correcting the algorithm faster than manual segmentation?

First ensemble segmentation correction: \( \sim 2 \times \) speedup for a \( 10 \times 800 \times 800 \) volume.

Second ensemble segmentation correction: \( \sim 3 \times \) speedup for a \( 20 \times 800 \times 800 \) volume.

A good start, but much more is needed.
FUTURE WORK

Use 3D encoder-decoder nets instead of 2D.

Train more networks for longer.

Explore evolutionary strategies for network design, instead of random sampling.

**The big challenge:** Robust segmentation that generalizes between tasks.