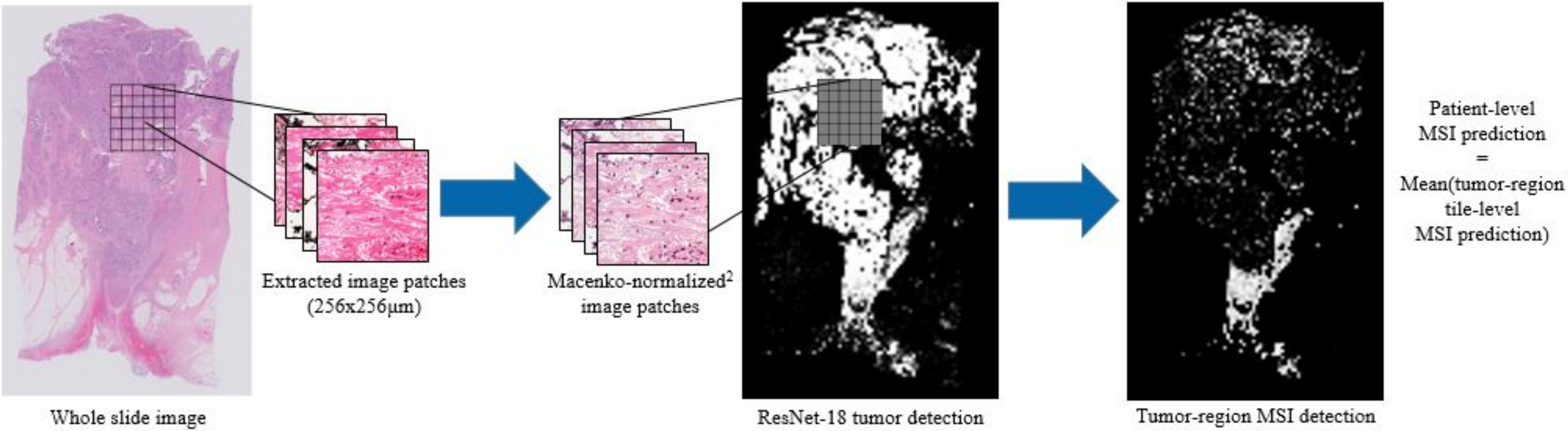
Deep Learning WSI Tutorial

Isaiah Pressman, CCF

Pipeline Outline

1. Libraries & Environment
2. Data Preprocessing
   - Tiling
   - Filtering out background tiles
   - Macenko normalization
   - Tumor detection
3. Training Deep Learning Models
   - Data splitting
   - Model and data loading
   - Main training loop
4. Evaluating & Visualizing Performance
   - Patient-level vs. tile-level evaluation
   - Visualizing performance over time



Whole slide image

Extracted image patches (256x256μm)

Macenko-normalized[2] image patches

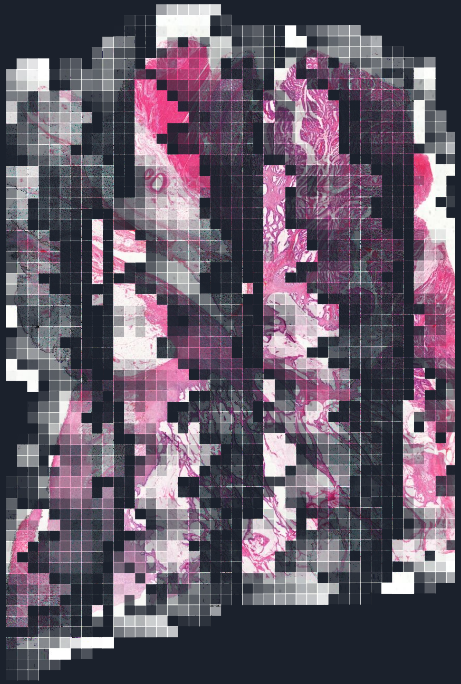
ResNet-18 tumor detection

Tumor-region MSI detection

Patient-level
MSI prediction
=
Mean(tumor-region
tile-level
MSI prediction)

Libraries & Environment

- Most of the packages needed are part of the standard python data science ecosystem. (numpy, scipy, scikit-learn, scikit-image, etc.)
- The only special package is OpenSlide, a library for reading whole slide images in Python.
  - When installing OpenSlide on Linux, it won’t work correctly for some slide formats unless you also install a specific version of libpixman



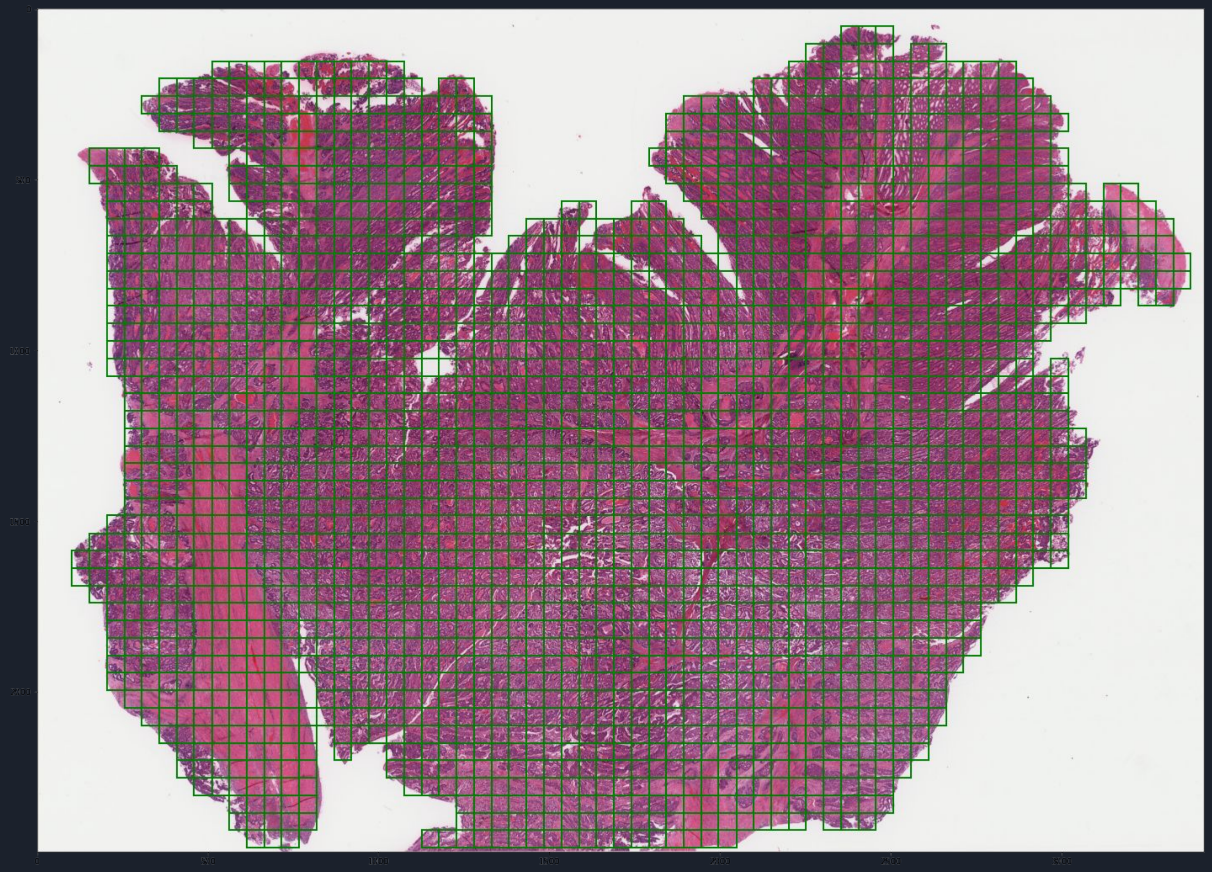
Data Preprocessing

In order to prepare the WSI images for deep learning training and inference, a number of preprocessing steps must be applied:

1. Images are broken into many small tiles (usually 256x256 microns)
2. Tiles are filtered to exclude non-tissue background regions
3. Tiles are Macenko-normalized
4. Tiles are filtered to exclude non-tumorous tissue regions

When applying this pipeline at scale, the implementation should include multiprocessing and/or CuPy (for Macenko normalization) as these additions provide enormous speedups.

Example WSI with tiles



Training Deep Learning Models

The deep learning model pipeline consists of three main steps:

1. Data splitting
2. Model and data loading
3. Training loop
    1. Performing inference
    2. Calculating loss
    3. Backpropagating loss
    4. Updating parameters
    5. Logging results

Evaluating & Visualizing Performance

- Models are trained and validated at a tile level. Ultimately however, the goal is to make decisions at a patient level.
- To convert a tile-level prediction to a patient-level one, we simply take the mean tile-level prediction.
- This is not the only way, nor necessarily the best way, to make a patient-level prediction, and could probably use improvement. (Multi-instance learning, for example, is another way to go about things)
- One should watch model train vs. validation performance to look for overfitting