**HIMLT Development Guide**This work was supported by the National Science Foundation under Grant No. NSF- 2409704.

**Project Overview**

**What is HIMLT?**

It is a modern web-based system for analyzing whole-slide images (WSI) of tissue samples Display gigapixel medical images smoothly in the browser.

* Detect and highlight cell nuclei automatically.
* Load precomputed nuclei coordinates from HDF5 files.
* Let pathologists/Researcher’s select positive/negative examples.
* Train AI models with Random Forest classifier to classify nuclei across entire slides.
* Visualize predictions in real-time.

**System Architecture**

***WEBAPP-BROWSER***

***React Frontend (TypeScript)***

* + - OpenSeadragon viewer
    - Dataset/slide selection
    - SVG nuclei visualization overlay (dots)
    - Interactive labeling interface
    - Training & prediction controls
    - Prediction visualization (magenta/cyan dots)

HTTP/REST API

**APACHE WEB SERVER (port 80)**

* Serves React build
* Reverse proxy to Flask

**FLASK API SERVER (Port 5000)**

• /api/datasets - List all datasets

• /api/slides - Get slides for dataset

• /api/train-histomics - Submit training job

• /api/predict-histomics - Get predictions

• /api/job-status/{job\_id} - Check job status

• /api/list-models - List trained models

**histomics\_bridge.py**

* Redis connection
* Job Submission
* Feature file handling
* Status monitoring

**MySQL Database**

* datasets
* slides
* dataset\_slides
* nuclei\_boundary

**Redis Queue**

• Job queue management

• Task distribution

**run\_model\_server.py**  
• Random Forest training

• Feature extraction

• Prediction generation

• HDF5 file processing

**Explanation:**

**For Image Viewing:**

1. User selects dataset and slide in React
2. Flask API fetches slide info from MySQL
3. OpenSeadragon requests tiles from IIP Server
4. Smooth image viewing on UI.

**For Nuclei Visualization:**

1. User clicks "Show Nuclei"
2. React fetches nuclei coordinates via Flask API
3. Flask loads data from HDF5 files using dataset.py
4. SVG overlay draws dots at centroid positions

**For ML Training:**

1. User selects 4 positive + 4 negative nuclei
2. React sends labeled samples to Flask
3. histomics\_bridge.py creates training job
4. Job submitted to Redis queue
5. Run\_model\_server.py picks up job
6. Loads Loads 64D features from HDF5 for 8 samples and go on.
7. Trains Random Forest
8. Saves model to checkpoints/
9. Returns job status

**For Predictions:**

1. User may/may not select the trained model. Users can view the predictions automatically once the training is done for the first time, otherwise we can select the trained model and requests predictions.
2. Flask API submits prediction job via bridge
3. Redis queue is enabled
4. run\_model\_server loads trained model
5. Generates predictions for all nuclei
6. Returns results with confidence scores
7. React displays magenta (positive) and cyan (negative) dots along with heatmaps

**Development Environment Setup**

**Directory Structure**

Location- /opt/new\_histomics/

├── checkpoints

│ ├── TCGA-06-5410-01Z-00-DX1.pkl

│ └── TCGA-19-2620-01Z-00-DX1.pkl

├── dataset.py

├── networks.py

├── run\_model\_server.py

├── settings.py

├── .env

Location ->/var/www/html/new\_histomics/backend/

├── app.py

├── histomics\_bridge.py

Location ->/data

GBM/ # Dataset-specific features

└── gbm\_features.h5 # HDF5 file with nuclei data

Location ->/var/www/html/new\_histomics (React Frontend )

├── index.tsx

├── assets # build

**Prerequisites**

**System Requirements:**

* Ubuntu/Linux server
* Python 3.10+
* Node.js 20+
* MySQL 8.0+
* Redis 6+
* IIP Image Server

**Installed Services:**

sudo systemctl status apache2

sudo systemctl status mysql

sudo systemctl status redis-server

sudo systemctl status iipsrv

*# Check Python version.*

python3 --version

/usr/bin/python3.10 –version  
  
  
**Image Files**

Whole slide images (TIF format) must be placed in /data/GBM/:

/data/GBM/

├── gbm\_features.h5

├── TCGA-06-5410-01Z-00-DX1.\*.tif

└── TCGA-19-2620-01Z-00-DX1.\*.tif  
  
--- TIF files are pyramidal TIFFs generated from SVS files using:  
```bash

vips tiffsave input.svs output.tif \ --compression=jpeg --Q=99 \ --tile --tile-width=256 --tile-height=256 \ --pyramid  
  
**Database setup:**

Four tables are created automatically with sample data:

```sql

datasets        -- GBM dataset metadata

slides          -- Slide information and pyramid paths

dataset\_slides  -- Junction table linking slides to datasets

nuclei\_boundary -- Nuclei x,y coordinates (populated from H5)

```

Update the above tables(datasets,slides,dataset\_slides) information with respect to the input image data. The system reads nuclei data from an HDF5 file and imports it into MySQL in the table “nuclei\_boundary”  
  
1. Firstly convert gbm\_features.h5 to binary file and extract the below details (slide\_name,x\_centroid,y\_centroid and boundary details to ‘nuclei\_boundary’ table)

gbm\_features.h5

├── slide\_name # Slide identifier strings

├── x\_centroid # X coordinates of nuclei

├── y\_centroid # Y coordinates of nuclei

└── boundary # Boundary coordinate data

**Training Workflow**

This documentation demonstrates the interactive machine learning workflow in HIMLT for training a Random Forest classifier on whole-slide images.

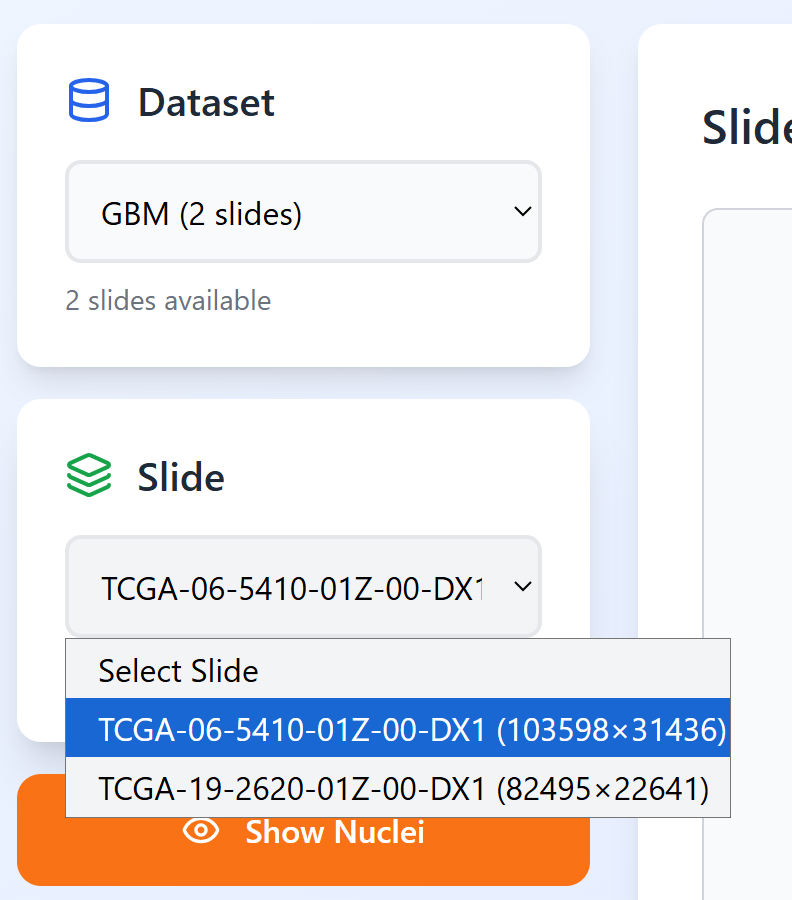
**Starting a Training Session**

Navigate to <http://10.80.10.111/new_histomics> access the HIMLT interface.

The interface displays available datasets in the left panel. Select a dataset from the dropdown list. After selecting the dataset, the slides panel will populate with available slides.

Select a slide from the list to load it in the OpenSeadragon viewer with smooth zoom and pan capabilities.

**IMAGE: interface showing dataset/slide selection**



**Loading Nuclei Visualization**

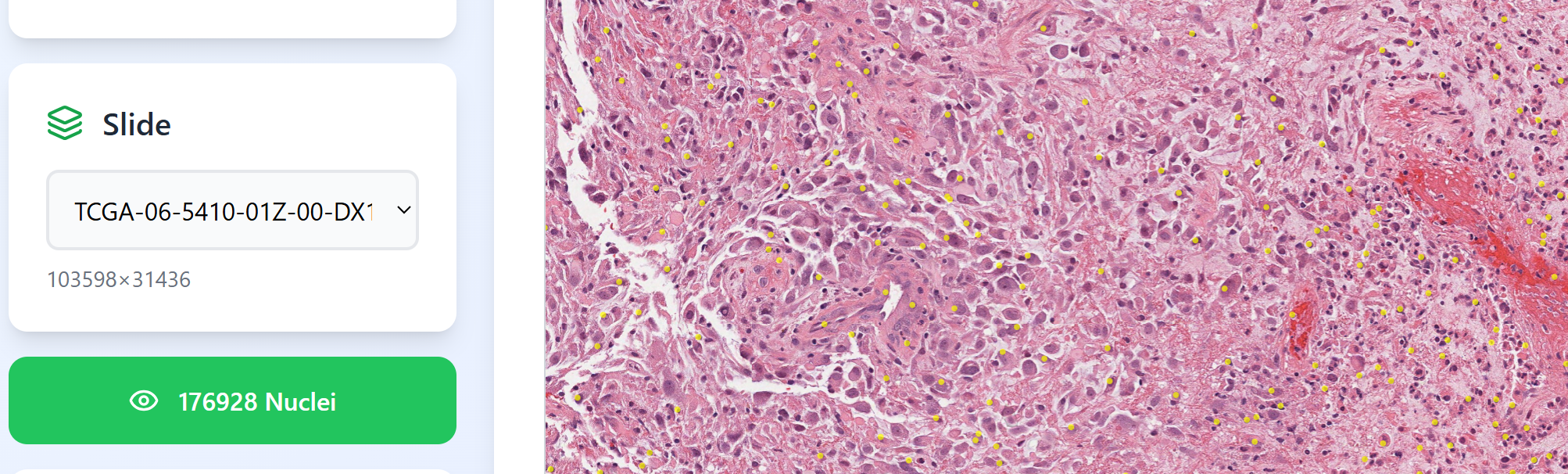
Once a slide is loaded, click the **"Show Nuclei"** button to display precomputed nuclei from HDF5 files.

Nuclei appear as small yellow dots at their centroid locations overlaid on the slide image. The system loads nuclei coordinates (x\_centroid, y\_centroid) from HDF5 feature files.

The button text changes to show the count of nuclei in the WSI.

**Note:** For performance, approximately 10% of nuclei are displayed.Zoom into regions to see nuclei more clearly.

**IMAGE: Slide viewer with "Show Nuclei" button and nuclei dots overlay**

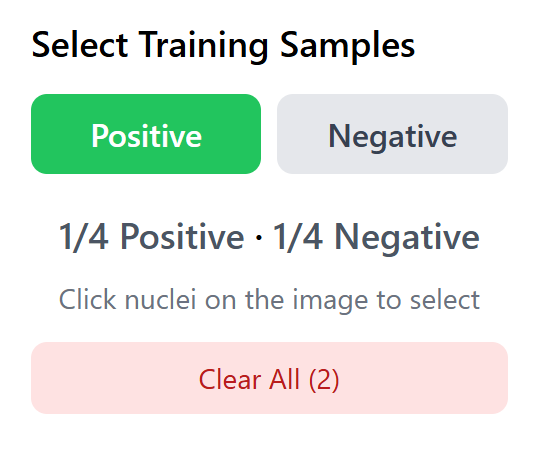


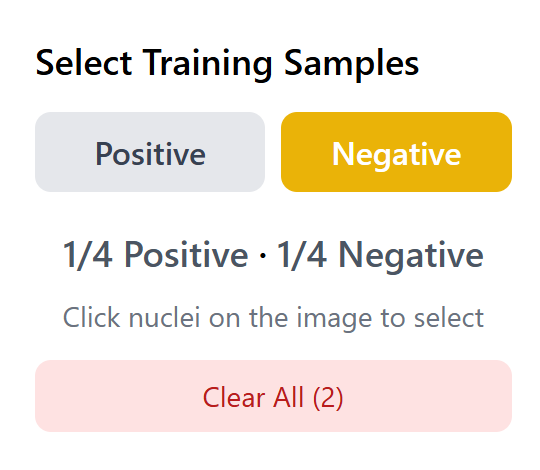
**Selecting Training Samples - Sample Selection Interface**

After nuclei are displayed, the **"Select Training Samples"** panel appears with:

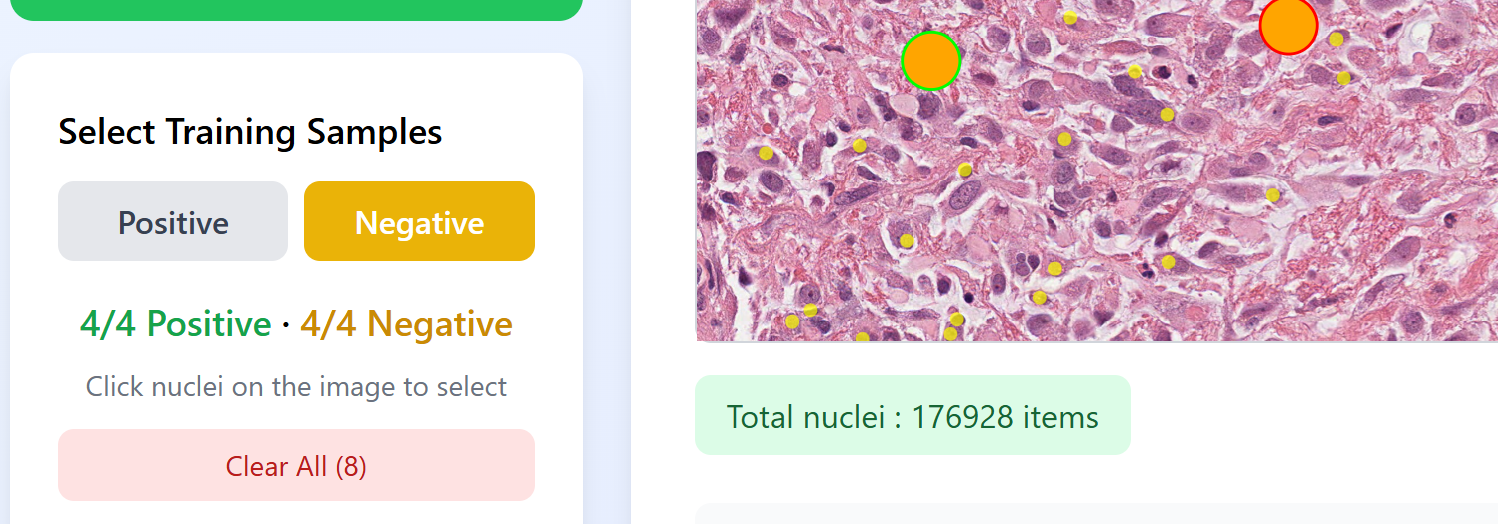
* **Positive** button (green when active)
* **Negative** button (yellow when active)
* Counter showing: "X/4 Positive · Y/4 Negative"
* Instructions: "Click nuclei on the image to select"
* Click the **"Positive"** button
* Click on 4 nuclei dots in the slide viewer that represent positive. Selected nuclei highlight in Orange with **green** border with larger dots.
* Click the **"Negative"** button
* Click on 4 nuclei that represents negative examples Selected nuclei highlight in orange with **red border** with larger dots.
* The counter updates as you select: "4/4 Positive · 4/4 Negative"

**IMAGE: Selection panel with Positive/Negative buttons and counter.**

****

****

**IMAGE: Slide viewer showing positive and negative highlighted selected nuclei**



**Clear Selection**

If you make a mistake, click the **"Clear All (X)"** button below the counter to deselect all samples and start over.

**Note:**

* You can zoom and pan while selecting
* Same nucleus cannot be selected twice
* Selected nuclei stay highlighted when you zoom/pan

**Training the Model**

Once you have 4 positive and 4 negative samples selected (8 total), the **"Train Model"** button becomes enabled (changes from gray to purple).

Click **"Train Model"** to submit the training job.

**Training Process**

1. Sends selected nuclei indices to Flask API (/api/train-histomics)
2. Submits job to Redis queue (REQUEST\_QUEUE)
3. run\_model\_server.py processes the job:
   * Loads 64-dimensional features from HDF5 for your 8 samples
   * Cleans any NaN values (replaces with 0)
   * Trains Random Forest classifier
   * Generates predictions for ALL nuclei .
4. Returns results via Redis

**Progress Indicator**

During training, the text button changes to "Training... X%" with a progress bar. once Training is completed in less than few seconds. **WAIT** until you see the prediction Alert arrives.

**Viewing Predictions**

**Automatic Prediction Generation**

After training is completed, predictions are automatically generated for ALL nuclei in the slide.

An alert appears: "Training complete! Generated X predictions (Y positive, Z negative)"

**Prediction Display**

The interface automatically displays prediction results as colored SVG dots:

* **Magenta (#FF00FF) dots:** Nuclei predicted as positive
* **Cyan (#00FFFF) dots:** Nuclei predicted as negative

**Orange dots with borders:** Your original 8 training samples

* Green border = positive training sample
* Red border = negative training sample

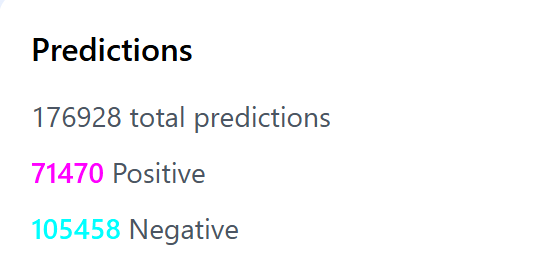
Note: Zoom into different regions to verify predictions. Magenta dots should cluster where positive examples are expected, cyan where negative examples are expected.

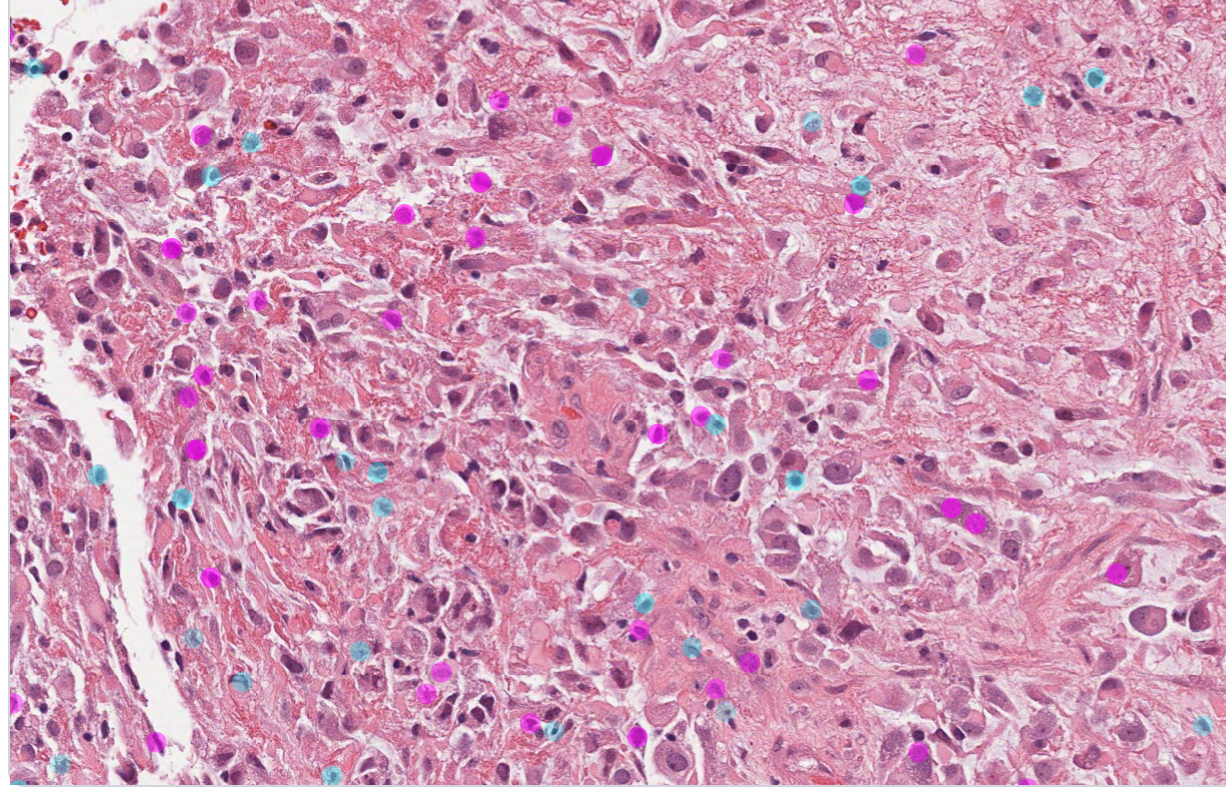
**Prediction Statistics**

**Below the viewer, prediction statistics are displayed:**

* Total Nuclei Predicted: Count of all nuclei
* Positive Predictions: Count and percentage
* Negative Predictions: Count and percentage

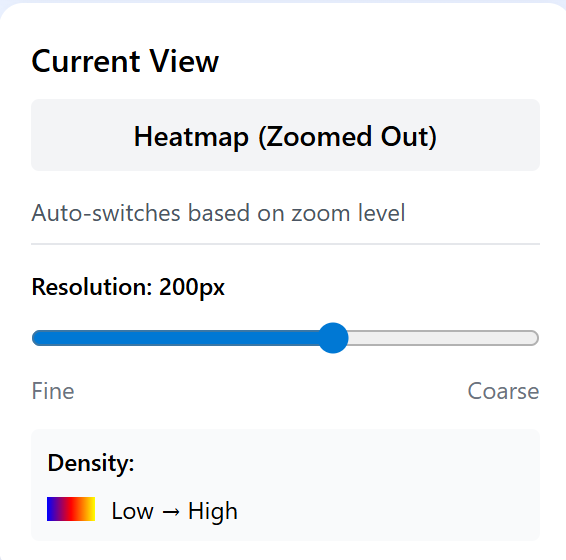
**IMAGE: Slide viewer showing magenta and cyan prediction dots with orange training samples**

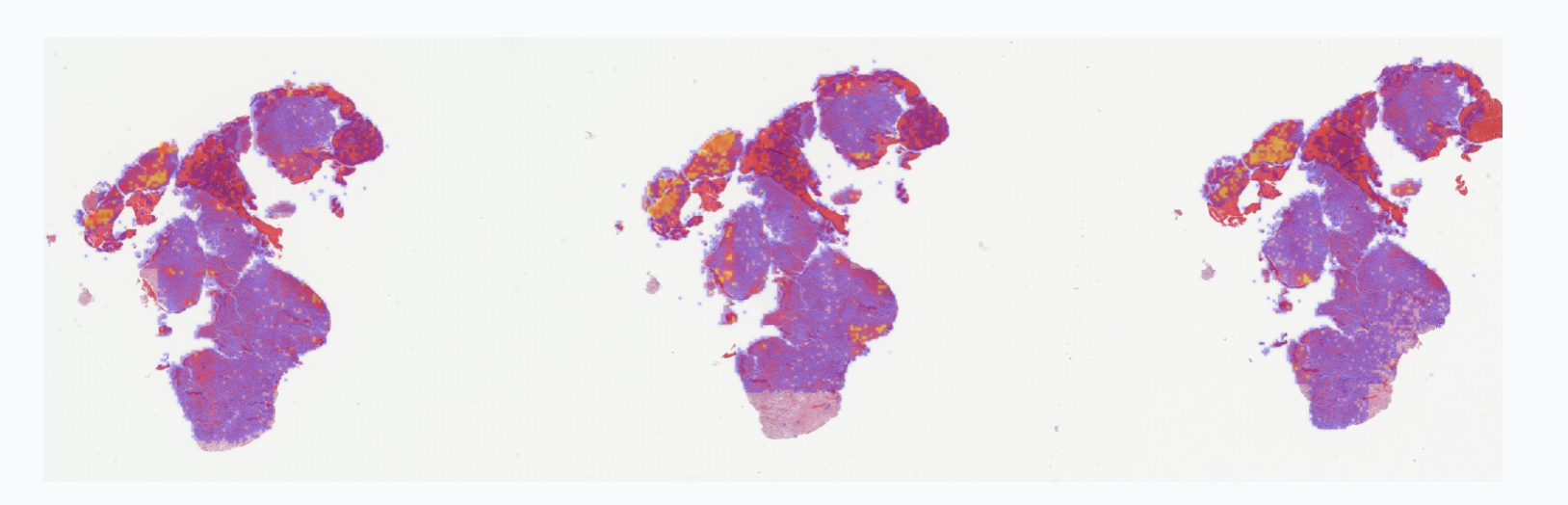
****

****

**Heat Map image viewer**

* System automatically shows heatmap
* User identifies hotspot (yellow region) -High Tumor Cell Density
* User zooms into that region
* System automatically switches to predictions
* User sees individual nuclei with classification
* User can change the Resolution to view the heat spots with density as below with a bar.
* Density ranges from Blue (low) → Red (moderate) → Yellow (high)

****



**Model Management**

**Saving Models**

Trained models are automatically saved to /opt/new\_histomics/checkpoints/ as .pkl files named after the slide.

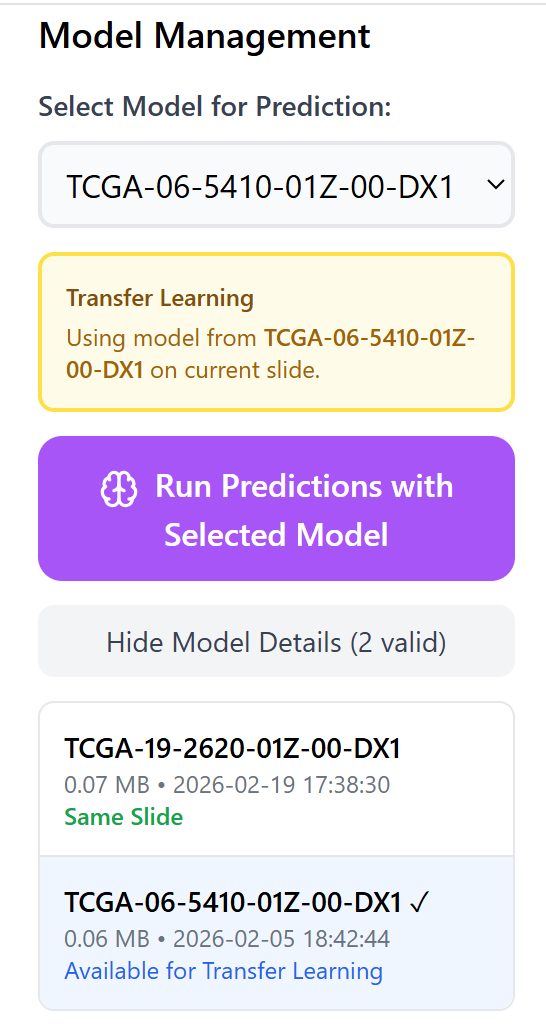
**Loading Previous Models**

Click on the dropdown to view all available trained models. A dropdown appears showing saved models .

Select a model from the dropdown and click **" Run Predictions with Current Model"** to apply it to the current slide or click **" Run Predictions with Selected Model"** to apply it to the different slide

This allows you to:

* Apply models trained on one slide to another slide
* Compare different training approaches
* Reload models from previous sessions



**Iterative Refinement**

**When Predictions Need Improvement**

If predictions are not satisfactory:

**Step 1:** The system tracks all training samples across iterations in “allTrainingSamples” state

**Step 2:** Training samples from previous iterations are displayed as **orange dots** with colored borders

**Step 3:** Select 4 new positive and 4 new negative samples (additional corrections)

**Step 4:** Click **"Refine Model(Iteration x)"** again - the new model trains on ALL samples (original 8 + new 8 = 16 total)

**Iteration Tracking**

The interface displays:

* **Iteration number** (e.g., "Iteration: 2")
* **Total training samples** accumulated across all iterations

Each iteration improves the model by incorporating corrections.

