**Home Assignment - Senior Data Scientist - Samueli Institute**

**Goal**:

The overall goal is to be able to classify pathology images as positive (1) or negative   
(0) using last-layer(s) re-training.

**Installed CUDA 12.1 from here:**

<https://pytorch.org/get-started/locally/>

pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu121

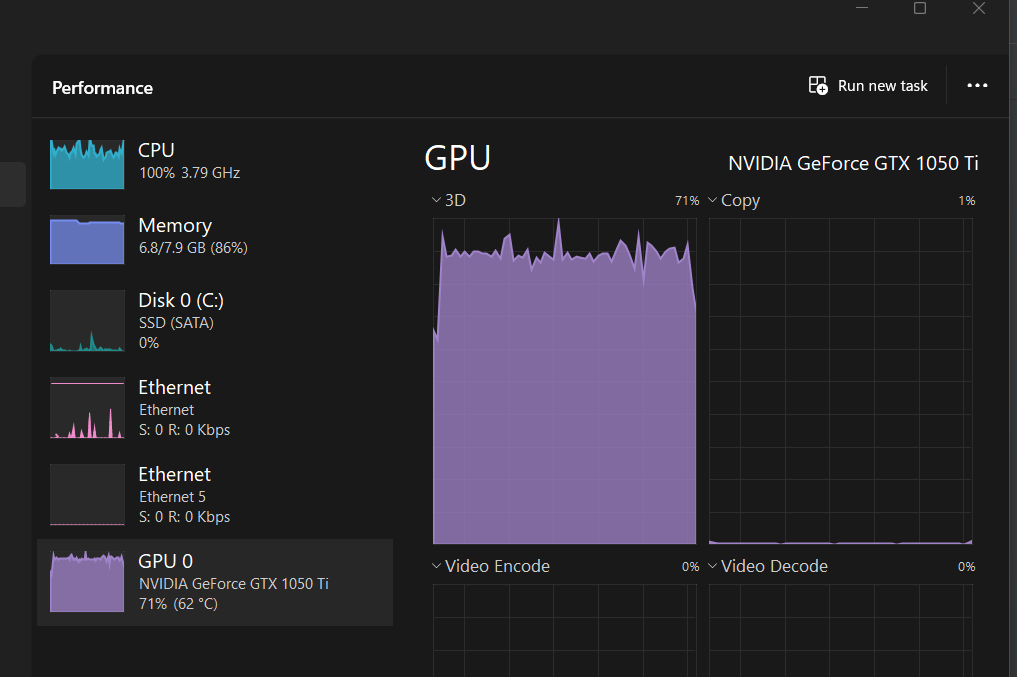
pip install openslide-python opencv-python torch torchvision pandas scikit-learn

**SVS file labels before extraction: labels.csv**

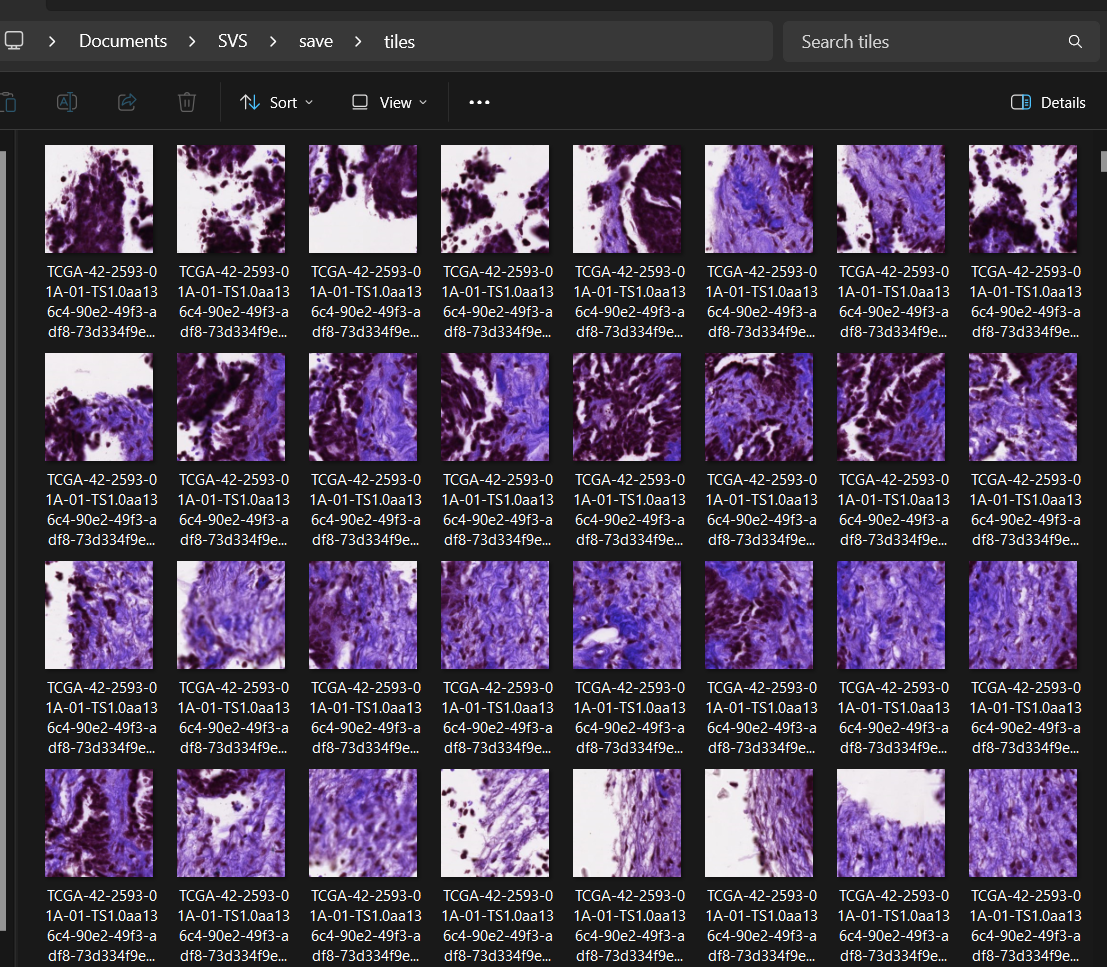
A screenshot of a computer

Description automatically generated

Training with GPU GTX1050 TI:



Extract tiles without black / white are less than 50% so we will not be on the borders of slide:



Output structure:

A screenshot of a computer

Description automatically generated

**Tiles info after tile extraction: labels.csv**

A screenshot of a computer

Description automatically generated

**Code results:**

**Labels loaded successfully.**

Tiles info file already exists at C:\Users\User\Documents\SVS\save\tiles\_info.csv. Loading existing data...

**Total tiles created: 2413**

Epoch 1: Val Accuracy = 0.9876

Epoch 2: Val Accuracy = 0.9862

Epoch 3: Val Accuracy = 0.9834

Epoch 4: Val Accuracy = 0.9890

Epoch 5: Val Accuracy = 1.0000

Epoch 6: Val Accuracy = 0.9489

Epoch 7: Val Accuracy = 0.9959

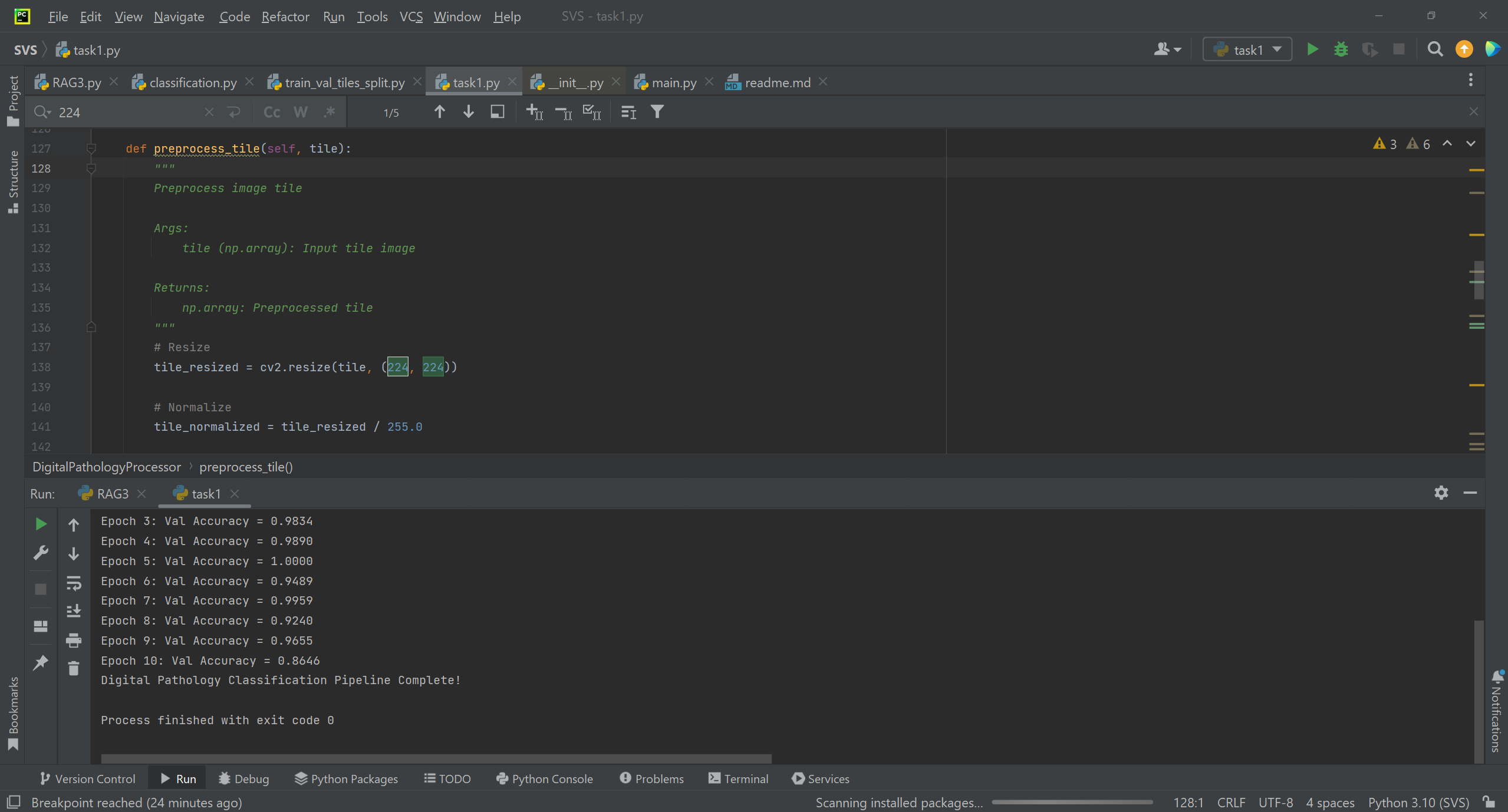
Epoch 8: Val Accuracy = 0.9240

Epoch 9: Val Accuracy = 0.9655

Epoch 10: Val Accuracy = 0.8646

**Digital Pathology Classification Pipeline Complete!**

Process finished with exit code 0



**Summary of the Code:**

This project implements a digital pathology classification pipeline for analyzing whole-slide images (WSI) in SVS format.

**Steps in the Code:**

1 Configuration and Initialization:

Specify paths for SVS files, output directory, and labels CSV file.

Create necessary directories for saving tiles and processed outputs.

2 Loading Labels:

Load slide labels from a CSV file into a Pandas DataFrame. Handles both comma-separated and whitespace-separated formats.

3 Tiling SVS Images:

Each WSI is divided into smaller image tiles of a specified size (default: 256x256 pixels).

**Tiles with >50% white or black areas are skipped to reduce noise and irrelevant data.**

Preprocessed tiles are saved as PNG files, and metadata is stored in tiles\_info.csv.

4 Preprocessing Tiles:

**Resize tiles to 224x224 pixels.**

Normalize pixel values for input into deep learning models.

5 Dataset Preparation:

Split the data into training and validation sets, maintaining class balance using stratified sampling.

Implement a PyTorch Dataset class for efficient loading and transformation of tile images.

6 Model Training:

Use a pre-trained ResNet50 model for binary classification.

Replace the fully connected layer to adapt the model to the number of classes (binary in this case).

**Train for 10 epochs using CrossEntropyLoss and Adam optimizer. Save the best-performing model based on validation accuracy.**

7 Metrics:

Compute and print validation accuracy after each epoch.

Save the model weights with the highest validation accuracy.

8 Pipeline Execution:

Integrate the steps into a main() function, ensuring pipeline is executed in sequence.

**Performance Analysis:**

Strengths:

**Efficient Data Handling**: Skipping tiles with high white/black ratios reduces irrelevant data, improving model focus and training time.

**Transfer Learning**: Using a pre-trained ResNet50 model leverages existing feature extraction capabilities, improving performance with limited data.

**Reproducibility**: Saves preprocessed tiles and metadata for reuse, avoiding redundant computations.

**Validation Accuracy Monitoring**: Ensures model improvement by saving only the best-performing weights.

Limitations:

**Tile Overlap**: The current overlap parameter is static (=0), and its effect on performance is not analyzed.

**Class Imbalance**: If the dataset is imbalanced, performance might favor the majority class, which could affect generalizability.

**Performance Metrics**: The pipeline tracks only accuracy during validation. Metrics like F1-score, precision, and recall are not computed, and we can add them later on.

**Next Steps for Improvement:**

**Performance Enhancements:**

1 Use data augmentation (e.g., flips, rotations) to increase diversity in training data.

2 Experiment with larger or overlapping tiles to capture more contextual information.

3 Experiment with different learning rates to improve model convergence.

**Advanced Metrics:**

Incorporate detailed performance metrics (e.g., confusion matrix, F1-score) for better evaluation of class-specific predictions.

**Multi-GPU Training:**

Enable distributed training on multiple GPUs for faster training of large datasets.

**Model Customization:**

Fine-tune additional layers of the ResNet50 backbone instead of training only the fully connected layer.

Explore other architectures like Vision Transformers or EfficientNet for improve.

**Inference Pipeline:**

Add functionality for deploying the trained model to classify new WSI data, including visualization of predictions on the slides.

**Elnatan**

**ID. 305465866**

**PHONE. 0506-998223**