Cancer Detection Using Histopathological Images

Overview

This Jupyter notebook implements a binary image classification task using PyTorch and a Convolutional Neural Network (CNN). The dataset used is the Histopathologic Cancer Detection dataset from Kaggle, which contains labeled image data for detecting metastatic cancer in histopathology scans.

Key Steps and Workflow

1. Import Libraries

The notebook begins by importing essential libraries:

- `numpy`, `pandas`, `matplotlib`, `plotly` for data handling and visualization
- `torch`, `torchvision` for deep learning and image transformations
- `PIL` for image processing
- Custom dataset and training utilities from `torch.utils.data`

2. Install and Use `torchsummary`

Installs `torchsummary` to visualize the model architecture (similar to Keras/TF).

3. Load and Explore Data

- Loads `train_labels.csv` which contains image IDs and binary labels (0: non-malignant, 1: malignant).
- Checks for duplicates (none found).
- Examines class distribution: 130,908 non-malignant vs. 89,117 malignant cases.

4. Data Visualization

- Defines helper functions to display sample images from both classes.
- Images are displayed with colored borders (green for non-malignant, red for malignant).
- Observations: Distinguishing between classes is challenging without expert knowledge.

5. Dataset Preparation

- Defines a custom `Dataset` class to load and preprocess images.
- Uses transforms (e.g., resizing, normalization, augmentation) for preprocessing.
- Splits data into training and validation sets.

6. Model Architecture

- Defines a CNN model using PyTorch's `nn.Module`.
- Uses convolutional layers, pooling, dropout, and fully connected layers.

- Prints model summary using `torchsummary`.

7. Training Setup

- Defines loss function (binary cross-entropy) and optimizer (Adam).
- Uses learning rate scheduling with `ReduceLROnPlateau`.
- Implements training and validation loops with metrics tracking (loss, accuracy).

8. Training and Evaluation

- Trains the model over multiple epochs.
- Logs training/validation accuracy and loss.
- Visualizes learning curves to monitor performance and overfitting.

9. Inference and Results

- Evaluates the model on the test set.
- Generates predictions and saves results in a submission-ready format.

Key Techniques Used

- Data Augmentation: Random rotations, flips, and normalization to improve generalization.
- Transfer Learning: Option to use a pre-trained model (not shown explicitly, but common in such notebooks).
- Class Imbalance Handling: May use weighted loss or oversampling (not explicitly implemented here).
- Early Stopping / LR Scheduling: To avoid overfitting and improve convergence.