1. Preprocessing Plugin

Purpose

To **prepare raw images** for deep-learning workflows. Offers both **single image preview** and **bulk** operations. Allows you to correct illumination, filter noise, extract channels, overlay separate channels, and more.

Major Sections

1.1 Single-Image Preprocessing

1. **Load Image for Preview**
    - **Browse Image...**: Opens a file dialog to select one image.
    - **Next/Prev Image**: Navigates through images in the same folder (if multiple exist).
    - **Original / Processed Preview Windows**: Shows side-by-side previews.
2. **Basic Options**
    - **Grayscale**: Convert the image from BGR to a single grayscale channel.
    - **Invert**: Invert pixel intensities (useful for certain microscopic images).
3. **Thresholding**
    - **None**: No threshold is applied.
    - **Manual**: Applies a user-specified threshold value.
    - **Adaptive**: Uses adaptive thresholding (per-channel for color images, or single-channel if grayscale).
    - **Otsu**: Computes the optimal threshold via Otsu's method (per-channel if color).
4. **Histogram Equalization & CLAHE**
    - **Histogram Equalization**: Globally enhances contrast by equalizing the pixel intensity histogram.
    - **CLAHE** (Contrast Limited Adaptive Histogram Equalization): More localized enhancement than global hist-eq, especially beneficial for medical images.
5. **Morphological Operations**
    - **Operation**: Erode, Dilate, Opening, Closing, TopHat, BlackHat.
    - **Kernel Size**: Size of the structuring element (e.g., 3×3).
6. **Denoising**
    - **Gaussian / Median / Bilateral / NLM**: Various noise reduction filters.
    - **Kernel**: Controls the filter's neighborhood size or parameters.
7. **Channel Extraction**
    - **B, G, R**: Optionally extract a single color channel if needed.
    - If "None," no specific channel extraction is performed.
8. **Resize**
    - **Width & Height**: If non-zero, resizes images to those dimensions.
9. **Enhance Red Channel**

    - **Enable**: Toggles whether to apply brightness/contrast specifically to the red channel.
    - **Brightness Factor & Contrast Factor**: Fine-tunes how the red channel is brightened or how its contrast is amplified.
10. **Apply Preprocessing**

- Executes all chosen transformations in the specified order.
- The result is shown in the **Processed** preview.

11. **Save Processed Image**

- Exports the single processed image to disk, with any extension (`.png`, `.jpg`, `.tiff`, etc.).

1.2 Bulk Preprocessing

1. **Input/Output Folders**
    - **Input**: Folder containing images to preprocess.
    - **Output**: Destination folder to save processed images.
2. **Output Format & Suffix**
    - **Convert to Format**: `jpg`, `png`, or `tiff`.
    - **Output Suffix**: e.g., `_processed`. Final filenames might look like `myimage_processed.jpg`.
3. **Process Bulk Images**
    - Runs a **QThread** so the UI remains responsive.
    - Applies the same pipeline configured in single-image mode (i.e., the same thresholding, morphological ops, etc.) to all images under `Input` recursively.

1.3 Overlay (ch01 -> Red, ch02 -> Green)

For combining multiple monochrome channel images into a single color-coded overlay:

1. **Input/Output**: Similar to bulk preprocessing, specify in/out folders.
2. **ch01 substring** / **ch02 substring**: Identifiers in the filenames that separate the "red channel" images (`-ch01`) from the "green channel" images (`-ch02`).
3. **Normalize Channels**: Each channel is scaled from min=0 to max=255.
4. **Hist Equal**: Global histogram equalization before merging.
5. **CLAHE**: Local histogram enhancement before merging.
6. **Overlay Suffix**: e.g., `_overlay`, added to the final merged file.

1.4 Filter by Substring

1. **Input/Output**: Folders for the filter operation.
2. **Substring to Keep**: e.g., "ch01". Only files containing this substring in the name are copied.

3. **Filter and Copy Images**: Useful for quickly splitting channels or partial subsets from a large data folder.

---

2. Data Exploration Plugin

Purpose

To **explore dataset structure** before training. Generates bar plots of class distributions, logs data counts, and displays sample images per class.

Major Features

1. **Data Folder**
    - Choose the parent folder containing subfolders named after each class (e.g., `data/ClassA`, `data/ClassB`, etc.).
2. **Max Images/Class**
    - Limits the number of sample images displayed in the gallery for each class (default: 3).
3. **Analyze Dataset**
    - **Scans each subfolder** to count images (`jpg`, `png`, `tif`, etc.).
    - **Displays**:
        - **Bar Chart**: Number of images for each class.
        - **Gallery**: A scrollable preview of sample images.
    - **Logs**: If any classes have zero images or if the folder structure is malformed.
4. **Progress Bar & Log**
    - The scanning runs in a background thread. The log displays real-time updates, such as “Class 'A': 120 images found.”

Useful for verifying you have enough images per class and quickly spotting data imbalance issues.

---

3. Training Plugin

Purpose

To **train convolutional neural networks** with a wide variety of parameters and advanced features. Uses **PyTorch Lightning** under the hood.

Key Configuration Areas

3.1 Dataset Splits

1. **Dataset Folder**: Root directory containing subfolders for each class.
2. **Val Split** & **Test Split** (percentages): The plugin will split your dataset accordingly.
    - E.g., 15% for validation, 15% for test → 70% training.

3.2 Network Architecture

Choose from:

- `resnet18`, `resnet50`, `resnet101`
- `densenet`
- `vgg`
- `inception`
- `mobilenet`
- `efficientnet_b0`
- `convnext_tiny`
- `convnext_large`

Toggle **Use 299 for Inception** if using the Inception architecture, to handle 299×299 input.

3.3 Augmentations

- **Brightness/Contrast**: Randomly adjusts image brightness and contrast.
- **Hue/Saturation**: Perturbs color balance.
- **Gauss Noise**: Injects random Gaussian noise.
- **Rotation**: Up to ±30° (configurable) rotation.
- **Flipping**: Horizontal, vertical, or both. Also specify flip probability.
- **Random Crop**: Randomly crops and resizes images.
- **Elastic**: Elastic transformations for warping images.
- **Grid Distortion**: Distorts image via a grid warping.
- **Optical Distortion**: Simulated lens distortion.
- **Normalize Pixel**: Typical ImageNet mean/std normalization.
- **MixUp / CutMix**:
    - Blends or cuts-and-pastes pairs of images with a Beta(α, α) sampling to augment training.

3.4 Hyperparameters

- **Learning Rate**: Initial LR, e.g., 1e-4.
- **Momentum**: For SGD-based optimizers.
- **Weight Decay**: L2 regularization.
- **Optimizer**: `adam`, `sgd`, `adamw`, or `lamb` (if installed).
- **Scheduler**: `none`, `steplr`, `reducelronplateau`, `cosineannealing`, `cycliclr`.
- **Scheduler Params**: Additional parameters, e.g., `step_size=10,gamma=0.1`.

3.5 Training Controls

- **Epochs**: Maximum number of epochs to train.
- **Batch Size**: e.g., 8, 16, 32…
- **Early Stopping**: Halts training if validation metric does not improve for N epochs.
    - Additional fields: **Monitor Metric** (`val_loss` or `val_acc`), **Patience**, **Min Delta**, **Mode** (`min`/`max`).

3.6 Regularization and Advanced Settings

- **Dropout Rate**: Probability of dropout before the final classifier.
- **Label Smoothing**: Softens hard labels in cross-entropy.
- **Freeze Entire Backbone**: Disables gradient updates in the backbone (fine-tuning only the final layer).
- **Loss Function**:
    - `cross_entropy`, `focal`, `bce` (for single-class but still multi-output?), `bce_single_logit` (strictly 1-class).
- **Gradient Clip Val**: Prevent exploding gradients.
- **LR Finder**: Automatic learning-rate suggestion.
    - **Accept LR Suggestion**: If checked, applies the found LR to training.
- **TensorBoard Logger**: Logs to `tb_logs/` for visualization in TensorBoard.
- **Mixed Precision**: If checked, uses `precision=16` (FP16) for faster GPU training.
- **Warmup Epochs**: Allows a warmup schedule before the main LR policy.
- **Workers**: Number of multiprocessing workers to load data.
- **Gradient Checkpointing**: Saves memory by trading off some compute overhead.
- **Gradient Accumulation**: Accumulate gradients over N mini-batches before stepping.
- **Check Val Every N Epoch**: Frequency of validation.

3.7 Partial Freezing Options (ResNet/ConvNeXt)

- For ResNet-like architectures: `conv1_bn1`, `layer1`, `layer2`, `layer3`, `layer4`.
- For ConvNeXt: `convnext_block0`..`convnext_block3`.
  Allows selective freezing of certain stages.

3.8 Training Execution

- **Start Training**: Kicks off a background thread. Real-time logs and progress updates appear in the text field.
- After training completes, a “best checkpoint” (`.ckpt`) is saved and test set metrics (confusion matrix, classification report) are displayed.

3.9 Optuna Hyperparameter Tuning (Optional)

- **Tune with Optuna**: Automates search for best LR, dropout, etc.
- **Trials**: Number of runs.
- **Timeout**: Maximum seconds to spend searching (0 = no limit).
- **Use Test Loss as Optuna Objective**: If checked, will optimize on the test metric (not recommended in typical practice).

3.10 Export Results

- After training, export confusion matrix, classification report, and optionally zip up TensorBoard logs.

---

4. Evaluation/Inference Plugin

Purpose

To **deploy or test** a trained model on images (single or batch). Allows Grad-CAM overlays, misclassification analysis, and advanced interpretability (SHAP).

Main Features

4.1 Checkpoint Selection

- **Browse Checkpoint...**: Load a `.ckpt` from the training stage.
- Automatically parses model architecture, class names, etc.

4.2 Transform Customization

- Similar to training transforms: e.g., resize, normalization means, standard deviations.

4.3 Single Image Inference

1. **Test Image**: Choose one image.
2. **Run Inference (Single)**: Predicts classes for that image.
    - **Top-k**: Show top-k predictions and their confidence scores.
    - **Min Confidence**: Filter out predictions below this percentage.
3. **Grad-CAM++**: If checked, calculates and overlays a heatmap (Grad-CAM or Grad-CAM++) highlighting regions most significant for the predicted class.

4.4 Evaluation & Batch Inference

1. **Inference Folder**: A directory of images to predict in bulk.
2. **GT CSV (optional)**: If you have ground-truth labels (filename, label) in a CSV, the plugin calculates metrics: confusion matrix, classification report, etc.

3. **Batch Grad-CAM**: For each image, generate a Grad-CAM overlay.
4. **Export CSV**: Save a `predictions.csv` with top-k predictions for each file. Also, `misclassified.csv` if ground truth is provided.
5. **SHAP Analysis** (if `shap` is installed):
    - **Enable SHAP**: Produces a SHAP summary plot for a subset of images.
    - **SHAP Samples**: Number of images to visualize via SHAP.
    - **SHAP BG Samples**: Size of the background set.

4.5 Final Outputs:

- **Confusion Matrix**: For classification results, saved to text and PNG.
- **ROC/PR Curves**: If binary or multi-class.
- **Calibration Curve**: For binary.
- **Classification Report**: Precision, recall, F1.
- **Metrics JSON**: Summaries stored in `metrics.json`.