Package: petrographer (via r-universe)

May 21, 2026

Title Petrographic Thin Section Analysis with Deep Learning

Version 0.0.0.9000

Description Automated petrographic analysis of thin section images using deep learning models. Provides tools for training RF-DETR detection and segmentation models, running predictions with SAHI (Slicing Aided Hyper Inference), calculating morphological properties, and analyzing results. Supports both local and HPC training workflows with seamless R-Python integration via reticulate.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Depends R (>= 4.1)

Imports cli, dplyr, fs, glue, jsonlite, pins, processx, purrr, readr, reticulate, stringr, tibble, tidyr,

Suggests grid, ggplot2, hipergator, here, htmltools, knitr, patchwork, png, quarto, reactable, scico, testthat (>= 3.0.0), tools, withr

VignetteBuilder quarto

Config/testthat/edition 3

Remotes flmnh-ai/hipergator

URL https://flmnh-ai.github.io/petrographer/

BugReports https://github.com/flmnh-ai/petrographer/issues

SystemRequirements Python (>= 3.8), PyTorch, rfdetr, SAHI, supervision, opencv-python, scikit-image, pycocotools

Config/pak/sysreqs
cmake make libicu-dev libpng-dev libuv1-dev libopencv-dev libssl-dev python3 libx11-dev

Repository https://flmnh-ai.r-universe.dev

Date/Publication 2026-04-21 17:27:24 UTC

RemoteUrl https://github.com/flmnh-ai/petrographer

RemoteRef HEAD

RemoteSha cdaf5711967f2c2016b52ff728a364e441166f8e

Contents

`align_images` *Align thin section image pair*

Description

Aligns a PPL (plane-polarized light) image to an XPL (cross-polarized light) reference image using SIFT feature matching. Designed for images of the same slide taken after remounting (e.g., adding polarizing filters). Automatically validates that scale and rotation are reasonable and falls back to translation-only if needed.

Usage

```
align_images(ppl_path, xpl_path, output_path, method = "similarity")
```

Arguments

ppl_path	Path to PPL image
xpl_path	Path to XPL image (reference)
output_path	Path to save aligned PPL image. If a directory, generates filename from input.
method	Alignment method: "similarity" (rotation+scale+translation with automatic fallback to translation if checks fail) or "translation" (translation only, no rotation/scale)

Value

List with n_matches and method_used (invisibly)

---

analyze_segmentation_dir

Run a segmentation batch workflow on a directory of images

Description

This is the higher-level segmentation path: predict a folder, save overlays, write per-object measurements, and write per-image summaries in one call. It currently uses direct RF-DETR segmentation inference; revisit once SAHI supports Roboflow segmentation models cleanly.

Usage

```
analyze_segmentation_dir(
  input_dir,
  model,
  output_dir = "results/segmentation_batch",
  save_visualizations = TRUE,
  save_measurements = TRUE,
  save_summary = TRUE,
  save_population_stats = TRUE
)
```

Arguments

input_dir	Directory containing images
model	Segmentation PetrographyModel from from_pretrained()
output_dir	Output directory for overlays and tables
save_visualizations	Whether to save overlay images

save_measurements
: Whether to write per-object measurements CSV

save_summary
: Whether to write per-image summary CSV

save_population_stats
: Whether to write a JSON population summary

Value

A list with detections, per-image summary, population stats, and output directory

---

clean_temp_datasets *Delete auto-pinned temp datasets*

Description

When train_model() is called with data_dir rather than dataset_id, it auto-pins the dataset as _temp_<timestamp> (tagged temp = TRUE in metadata) so training is reproducible. Those pins persist in .petrographer/datasets/ and accumulate over time — each is a tar.gz of the full dataset. This helper removes them.

Usage

```
clean_temp_datasets(board = NULL, confirm = interactive())
```

Arguments

board
: Pins board (NULL/"local" = local dataset board).

confirm
: If TRUE, prompt before deleting. Defaults to TRUE in interactive sessions, FALSE otherwise (e.g. scripted cleanup).

Value

Character vector of deleted dataset ids (invisibly).

---

confusion_matrix *Compute a confusion matrix from SAHI predictions + COCO ground truth*

Description

For each image, greedily match predicted boxes to ground-truth boxes by IoU (highest-scoring prediction matches first). Predictions that match a GT box with IoU >= iou_threshold contribute a (gt_class, pred_class) cell. Unmatched predictions land in the "background" *row* (false positives); unmatched GT boxes land in the "background" *column* (false negatives).

Usage

```
confusion_matrix(
  predictions,
  annotation_json,
  iou_threshold = 0.5,
  score_threshold = 0
)
```

Arguments

predictions	Tibble of predictions with image_id, category_id, score, xmin, ymin, width, height (the predictions element of evaluate_model_sahi() output works directly).
annotation_json	Path to the COCO annotation JSON the predictions were produced against.
iou_threshold	Minimum IoU for a prediction to count as matching a GT box (default 0.5).
score_threshold	Filter predictions below this score before matching (default 0; keep all).

Details

Detection doesn't have a clean confusion-matrix definition (unmatched predictions and unmatched GT don't map onto classification confusion cleanly), so treat this as a diagnostic heatmap rather than a calibrated metric.

Value

A list with:

- long: long-format tibble (gt_label, pred_label, count)
- matrix: wide tibble, one row per GT label
- class_names: character vector of class labels (includes "background")
- iou_threshold: the threshold used

---

evaluate_model_sahi *Evaluate detections with SAHI and COCO metrics*

Description

Runs SAHI inference across the validation set described by a COCO-style annotation file and computes COCO metrics using pycocotools. The function returns a tidy summary of the standard 12 bbox metrics alongside the raw prediction table for further analysis.

Usage

```
evaluate_model_sahi(
  model,
  annotation_json,
  image_dir = NULL,
  use_slicing = TRUE,
  slice_size = NULL,
  overlap = 0.2,
  max_images = NULL,
  save_predictions = NULL,
  iou_type = "bbox",
  max_dets = 100
)
```

Arguments

model	A PetrographyModel from from_pretrained().
annotation_json	Path to COCO annotation JSON (e.g. valid/_annotations.coco.json).
image_dir	Directory containing the images referenced in the annotation file. If NULL, image paths are resolved relative to the annotation file.
use_slicing	Whether to use SAHI sliced inference (default TRUE).
slice_size	Slice size for SAHI inference (pixels, default: use model's resolution)
overlap	Overlap ratio between slices (default 0.2).
max_images	Optional maximum number of images to evaluate (useful for smoke tests).
save_predictions	Optional path to write COCO-format predictions JSON.
iou_type	IoU type to evaluate ("bbox" by default).
max_dets	Maximum detections per image for evaluation. For dense detection (100+ objects), set to 300 or higher (default: 100).

Value

A list with elements summary (tibble of COCO metrics), predictions (tibble of detections), and coco_eval (pycocotools object).

---

evaluate_training *Evaluate model training*

Description

Parses RF-DETR training metrics and exports:

- training_metrics.csv: per-epoch losses, learning rate, class error
- validation_metrics.csv: per-epoch validation losses + AP/AR + summary map
- validation_classwise.csv: per-class AP/precision/recall (when logged)

Usage

```
evaluate_training(
  model_id = NULL,
  model_dir = NULL,
  board = NULL,
  output_dir = "results/evaluation"
)
```

Arguments

model_id	Model ID (will be resolved from local board)
model_dir	Directory containing trained model (alternative to model_id)
board	Pins board (NULL = local board, only used if model_id provided)
output_dir	Output directory for results (default: 'results/evaluation')

Details

Source-file preference (first one found wins):

1. `metrics.csv` — RF-DETR >= 1.6.0 (PyTorch Lightning CSVLogger)
2. `log.txt` — RF-DETR < 1.6.0 (native training loop, JSONL)

Value

List with parsed tibbles and summary statistics

`from_pretrained` *Load a pretrained RF-DETR model*

Description

Loads RF-DETR models from local training board, hub, or custom board. By default, checks local models first, then falls back to the public hub.

Usage

```
from_pretrained(
  model_id,
  version = NULL,
  board = NULL,
  device = "cpu",
  confidence = 0.5,
  resolution = NULL
)
```

Arguments

model_id	Model name (e.g., "shell_v3")
version	Specific version (NULL for latest)
board	Board to load from: • NULL (default): check local first (.petrographer/models/), then hub • "local": only check locally trained models (.petrographer/models/) • Custom board object
device	Device: "cpu", "cuda", or "mps"
confidence	Detection threshold
resolution	Input image resolution for RF-DETR (default: auto-detect from variant). Defaults by variant: • Detection (patch_size=16, divisible by 32): nano=384, small=512, medium=576, large=704 • Segmentation (patch_size=12, divisible by 12): seg_nano=312, seg_small=384, seg_medium=432, seg_large=504, seg_xlarge=624, seg_2xlarge=768, seg_preview=432 Override only if you know your dataset wants a different input size.

Value

PetrographyModel object containing RF-DETR model

Examples

```
## Not run:
# Smart loading (checks local first, then hub)
model <- from_pretrained("my_model")

# Force local only
model <- from_pretrained("my_model", board = "local")

# Force hub only
hub_board <- pins::board_url("https://flmnh-ai.s3.us-east-1.amazonaws.com/.petrographer/models/")
model <- from_pretrained("public_model", board = hub_board)

## End(Not run)
```

---

get_dataset_path *Get path to pinned dataset*

Description

Returns filesystem path to a pinned dataset tar.gz file. The tar.gz should be extracted at training time.

Usage

```
get_dataset_path(dataset_id, board = "local", version = NULL)
```

Arguments

dataset_id	Dataset name
board	Pins board (or board object)
version	Specific version to retrieve (NULL = latest)

Value

Path to dataset tar.gz file

---

get_population_stats *Get overall population statistics*

Description

Get overall population statistics

Usage

```
get_population_stats(.data)
```

Arguments

.data	Data frame with detections

Value

Named list of population-level statistics

---

get_training_dataset *Get the dataset used to train a model*

Description

Retrieves the exact dataset version that was used to train a model. This ensures reproducibility by downloading the specific version from pins.

Usage

```
get_training_dataset(model_id, board = NULL)
```

Arguments

model_id	Model name
board	Pins board to load model from (NULL = local training board)

Value

Path to the dataset tar.gz file

Examples

```
## Not run:
# Retrieve the exact dataset version used to train a model. Returns a
# .tar.gz path; `train_model()` and `validate_dataset()` accept this directly
# and extract transparently.
dataset_path <- get_training_dataset("my_model")

# Retrain on the same data
train_model(data_dir = dataset_path, model_id = "my_model_v2", ...)

# Or inspect without retraining
validate_dataset(dataset_path)

## End(Not run)
```

---

list_datasets *List pinned datasets*

Description

Lists all pinned datasets on a board.

Usage

```
list_datasets(board = "local")
```

Arguments

board	Pins board (NULL = local board, "local" = local board)

list_models *List available models*

Description

List available models

Usage

```
list_models(board = NULL)
```

Arguments

board	Pins board (NULL = public hub, "local" = local training board)

list_trained_models *List all locally trained models*

Description

List all locally trained models

Usage

```
list_trained_models()
```

Value

Character vector of pinned model names

model_info *Get model info*

Description

Get model info

Usage

```
model_info(model_id, board = NULL)
```

Arguments

model_id	Model name
board	Pins board (NULL = public hub, "local" = local training board)

`pg_plot_annotations` *Plot COCO annotations on an image*

Description

Draws ground-truth bounding boxes (and segmentation masks when present) from a COCO-style annotation file on top of an image. Rendering is delegated to Roboflow's `supervision` library via reticulate, so GT overlays stay visually consistent with prediction overlays produced by `predict_image()`.

Usage

```
pg_plot_annotations(
  image_path,
  annotation_json,
  categories = NULL,
  output = NULL,
  display = NULL,
  draw_labels = TRUE,
  mask_opacity = 0.4
)
```

Arguments

`image_path`	Path to the image file.
`annotation_json`	Path to the COCO annotations JSON containing the image.
`categories`	Optional vector of category names or ids to keep. `NULL` keeps all.
`output`	Destination PNG path. Defaults to a tempfile.
`display`	Whether to render the annotated image on the current graphics device. Defaults to `TRUE` in interactive sessions and inside knitr (so the image appears under the chunk).
`draw_labels`	Whether to draw category labels. Default `TRUE`.
`mask_opacity`	Opacity for mask fills (segmentation datasets). Default `0.4`.

Value

The path to the written PNG, invisibly.

pg_plot_dataset_image *Plot a sample image from a dataset directory*

Description

Plot a sample image from a dataset directory

Usage

```
pg_plot_dataset_image(
  dataset_dir,
  split = c("train", "valid", "test"),
  image_name = NULL,
  annotation_json = NULL,
  ...
)
```

Arguments

dataset_dir	Directory containing dataset splits (train, valid, or optionally test).
split	Which split to draw from ("train", "valid", or "test").
image_name	Optional specific image file name; otherwise a random one.
annotation_json	Optional explicit path to annotation JSON.
...	Additional arguments passed to pg_plot_annotations().

Value

The path to the written PNG, invisibly.

pg_save_dataset_preview *Save a preview image for a dataset*

Description

Convenience helper that writes a preview.png alongside the dataset so the pkgdown catalog can embed a thumbnail.

Usage

```
pg_save_dataset_preview(
  dataset_dir,
  split = "valid",
  output = fs::path(dataset_dir, "preview.png"),
  overwrite = TRUE,
  ...
)
```

Arguments

dataset_dir	Dataset directory (containing split subfolders).
split	Which split to sample (default "valid").
output	Path for the preview image (default <dataset_dir>/preview.png).
overwrite	Whether to overwrite an existing preview.
...	Additional arguments passed to pg_plot_dataset_image().

Value

The path to the written preview image (invisibly).

---

pin_dataset *Pin a dataset to a board*

Description

Pins a COCO-format dataset directory to a pins board for versioning and reuse. The dataset is compressed as tar.gz before pinning.

Usage

```
pin_dataset(data_dir, dataset_id, board = NULL, metadata = list())
```

Arguments

data_dir	Path to dataset directory, or a .tar.gz / .tgz archive (transparently extracted before re-pinning).
dataset_id	Name for the pinned dataset
board	Pins board (NULL = local board at .petrographer/)
metadata	Optional metadata list

pin_model *Pin a trained RF-DETR model to a board*

Description

Uploads model files to a pins board for versioning and sharing. Maintainers should call `pins::write_board_manifest()` after pinning to update the board manifest for board_url() consumers.

Usage

```
pin_model(model_dir, model_id, board, metadata = list())
```

Arguments

`model_dir`	Directory with model files
`model_id`	Name for the model
`board`	Pins board to pin to
`metadata`	Optional metadata list

predict.PetrographyModel *Predict objects in an image*

Description

S3 method for `stats::predict()` that runs inference on an image with a `PetrographyModel`. Delegates to `predict_image()`.

Usage

```
## S3 method for class 'PetrographyModel'
predict(
  object,
  image_path,
  use_slicing = TRUE,
  slice_size = NULL,
  overlap = 0.2,
  save_visualizations = FALSE,
  output_dir = NULL,
  ...
)
```

Arguments

object	A PetrographyModel from from_pretrained(). (Named object rather than model to match the stats::predict generic.)
image_path	Path to image file.
use_slicing	Whether to use SAHI sliced inference (default TRUE).
slice_size	Slice size in pixels (default: model's resolution).
overlap	Overlap ratio between slices (default 0.2).
save_visualizations	Whether to save prediction visualization.
output_dir	Output directory (auto-generated if NULL).
...	Unused; present for generic compatibility.

Value

Tibble with detection results.

---

predict_image *Predict objects in a single image*

Description

Predict objects in a single image

Usage

```
predict_image(
  image_path,
  model,
  use_slicing = TRUE,
  slice_size = NULL,
  overlap = 0.2,
  output_dir = NULL,
  save_visualizations = TRUE
)
```

Arguments

image_path	Path to image file
model	PetrographyModel object from from_pretrained()
use_slicing	Whether to use SAHI sliced inference (default: TRUE)
slice_size	Size of slices for SAHI in pixels (default: use model's resolution). Must be divisible by 56 for RF-DETR.
overlap	Overlap ratio between slices (default: 0.2)
output_dir	Output directory (auto-generated if NULL)
save_visualizations	Whether to save prediction visualization (default: TRUE)

Value

Tibble with detection results and morphological properties

`predict_images` *Predict objects in multiple images (directory)*

Description

Predict objects in multiple images (directory)

Usage

```
predict_images(
  input_dir,
  model,
  use_slicing = TRUE,
  slice_size = NULL,
  overlap = 0.2,
  output_dir = "results/batch",
  save_visualizations = TRUE
)
```

Arguments

`input_dir`	Directory containing images
`model`	PetrographyModel object from from_pretrained()
`use_slicing`	Whether to use SAHI sliced inference (default: TRUE)
`slice_size`	Size of slices for SAHI in pixels (default: use model's resolution)
`overlap`	Overlap ratio between slices (default: 0.2)
`output_dir`	Output directory (default: 'results/batch')
`save_visualizations`	Whether to save prediction visualizations (default: TRUE)

Value

Tibble with detection results for all images

slice_dataset *Slice COCO dataset for varying image sizes*

Description

Uses SAHI to slice images and annotations into tiles. Images smaller than `slice_size` are treated as single slices (no fragmentation). Larger images are split into overlapping tiles, which increases training samples and improves detection of small objects in large images.

Usage

```
slice_dataset(
  input_dir,
  output_dir,
  slice_size = 1024,
  overlap = 0.2,
  min_area_ratio = 0.1,
  output_format = ".jpg"
)
```

Arguments

`input_dir`	Input directory with `train/` and `valid/` subdirectories containing COCO annotations. If a `test/` directory exists, it will also be sliced.
`output_dir`	Output directory for sliced dataset (will be created)
`slice_size`	Slice size in pixels (default: 1024). Images smaller than this will not be fragmented.
`overlap`	Overlap ratio between adjacent slices (default: 0.2, or 20%)
`min_area_ratio`	Minimum area ratio to keep object fragments (default: 0.1). Objects cut by slice boundaries with less than 10% visible area are dropped.
`output_format`	Output image format: ".jpg" or ".png" (default: ".jpg"). JPG minimizes storage (~10x smaller) with minimal quality loss. Use PNG for lossless slicing.

Details

This is particularly useful for dense detection datasets with varying image sizes. For the inclusions dataset, slicing with `slice_size = 1024` will:

- Keep small images (<1024px) intact as single slices
- Split large images (>1024px) into 2-4 overlapping tiles
- Result: ~2x more training images with better small object coverage

Value

Path to sliced dataset directory (invisibly)

Examples

```
## Not run:
# Slice dataset for training
sliced_dir <- slice_dataset(
  input_dir = "data/processed/inclusions",
  output_dir = "data/processed/inclusions_sliced",
  slice_size = 1024,
  overlap = 0.2
)

# Train on sliced dataset
train_model(data_dir = sliced_dir, ...)

## End(Not run)
```

`summarize_by_image` *Summarize detections by image*

Description

Summarize detections by image

Usage

```
summarize_by_image(.data)
```

Arguments

`.data`	Data frame with detections

Value

Summary tibble with per-image statistics

`train_model` *Train a new petrography detection model*

Description

Orchestrates local or HPC training using RF-DETR. Models are automatically pinned to the local board (.petrographer/) for versioning.

Usage

```
train_model(
  dataset_id = NULL,
  data_dir = NULL,
  model_id = NULL,
  model_variant = "nano",
  resolution = NULL,
  epochs = 10,
  batch_size = NA,
  grad_accum_steps = NA,
  learning_rate = NULL,
  device = "cuda",
  use_amp = NULL,
  amp_dtype = "bf16",
  gradient_checkpointing = NULL,
  num_workers = NULL,
  time_hours = 4,
  validate_every = 2L,
  early_stopping_patience = NULL
)
```

Arguments

dataset_id	Name of pinned dataset to use for training (preferred).
data_dir	Path to dataset directory (alternative to dataset_id; will be auto-pinned with temp ID).
model_id	Name for the trained model (used for pins). Defaults to dataset_id if not provided.
model_variant	RF-DETR model variant. Detection: "nano" (default), "small", "medium", "large". Segmentation: "seg_nano", "seg_small", "seg_medium", "seg_large", "seg_xlarge", "seg_2xlarge". Legacy: "seg_preview".
resolution	Image resolution for training. Auto-detected from variant if not specified.
epochs	Number of training epochs. Default: 10.
batch_size	Batch size for training. If NA (default), uses 2.
grad_accum_steps	Gradient accumulation steps. If NA (default), auto-calculated as 16 / batch_size for effective batch size of 16.
learning_rate	Learning rate. If NULL (default), uses model default.
device	Device for local training: 'cpu', 'cuda', or 'mps' (default: 'cuda').
use_amp	Use automatic mixed precision training (default: TRUE for CUDA, FALSE otherwise). Reduces memory usage by ~40%.
amp_dtype	AMP dtype: 'bf16' (recommended for modern GPUs) or 'fp16' (default: 'bf16').
gradient_checkpointing	Enable gradient checkpointing (default: FALSE). Reduces memory usage by ~30%.

num_workers	Number of data loading workers (default: 8).
time_hours	Time limit for HPC training in hours (default: 3). Examples: 4 = 4 hours, 0.5 = 30 minutes, 1.5 = 1.5 hours. Ignored for local training.
validate_every	Validate every N epochs (default: 1). Set to NULL to use model default.
early_stopping_patience	Stop training if validation loss doesn't improve for N epochs (default: 10). Set to NULL to disable early stopping.

Details

Training mode (local vs HPC) is auto-detected based on `hipergator` configuration. For HPC training, call `hipergator::hpg_configure()` before `train_model()` to set connection details (host, user, base_dir).

Value

Model ID (can be loaded with `from_pretrained(model_id)`).

---

validate_dataset *Validate a COCO-style dataset directory*

Description

Performs existence checks for expected splits and annotations, then runs annotation diagnostics (counts, size distribution, potential issues) for each split.

Usage

```
validate_dataset(data_dir, quiet = FALSE)
```

Arguments

data_dir	Directory containing 'train' and 'valid' subdirectories, or a `.tar.gz` / `.tgz` archive thereof (e.g. the path returned by `get_training_dataset()`). Archives are transparently extracted into a session-scoped cache.
quiet	If TRUE, suppress CLI output while still returning diagnostics

Value

A list with validation flags, counts, size metrics, and diagnostics

Index