Package 'leakaudit'

July 29, 2026

Title Detect and Audit Train/Test Leakage from Near-Duplicate Images

Version 0.1.0

Description Detects near-duplicate images across dataset splits using perceptual hashing, reports the resulting train/validation/test contamination, and produces a corrected, leak-free split assignment. Intended for machine learning researchers who need to verify that image classification splits do not share near-duplicate samples across partitions before reporting model metrics.

License MIT + file LICENSE

Encoding UTF-8

Imports magick

Suggests testthat (>= 3.0.0), knitr, rmarkdown

Config/testthat/edition 3

URL https://github.com/anakincodex/leakaudit

BugReports https://github.com/anakincodex/leakaudit/issues

Depends R (>= 4.1)

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Kartik Patel [aut, cre],
Samruddhi Amol Shah [aut],
BitandByte [cph]

Maintainer Kartik Patel <kartikpatel.id@gmail.com>

Repository CRAN

Date/Publication 2026-07-29 16:40:13 UTC

Contents

`clean_splits` *Produce a leak-free split assignment*

Description

Resolves duplicate groups that span multiple splits by assigning every member of a leaked group to a single split, chosen by `priority`. This mirrors the common convention of keeping duplicates in `train` (so no information is discarded, only potential evaluation leakage) but any priority order can be supplied.

Usage

```
clean_splits(grouped_df, priority = c("train", "val", "test"))
```

Arguments

`grouped_df`	A data frame with columns `path`, `group_id`, and `split`, as produced by `find_duplicate_groups()`.
`priority`	Character vector giving split precedence, highest priority first. Defaults to `c("train", "val", "test")`: a leaked group touching train and test is fully reassigned to train.

Value

`grouped_df` with `split` replaced by the corrected, leak-free assignment, plus a logical column `reassigned` flagging which rows were changed.

Examples

```
groups <- data.frame(
  path = c("a.jpg", "b.jpg", "c.jpg"),
  split = c("train", "test", "val"),
  group_id = c(1, 1, 2)
)
clean_splits(groups)
```

`compute_hashes` *Compute perceptual hashes for a set of images*

Description

Computes a 64-bit difference hash (dHash) for each image path. dHash is robust to resizing, mild compression, and small crops, which makes it well suited to catching near-duplicate images that differ only in resolution, format, or minor edits – the kind of duplication that commonly leaks across train/validation/test splits.

Usage

```
compute_hashes(paths, split = NULL)
```

Arguments

`paths`	Character vector of file paths to image files. Any format readable by `magick::image_read()` is supported (jpg, png, tif, ...).
`split`	A character vector the same length as `paths` giving the split each image currently belongs to (e.g. `"train"`, `"val"`, `"test"`). Optional; if omitted, all images are treated as unsplit.

Value

A data frame with columns `path`, `split`, and `hash` (a 16 character hex string encoding the 64-bit dHash). Any path that fails to load produces `NA` in `hash` with a warning, rather than stopping the whole run.

Examples

```
## Not run:
hashes <- compute_hashes(
  paths = list.files("images/", full.names = TRUE),
  split = rep(c("train", "test"), length.out = 10)
)

## End(Not run)
```

dhash_audit *Audit train/val/test leakage from duplicate groups*

Description

Given duplicate groups (from `find_duplicate_groups()`) and each image's split assignment, reports how many groups span more than one split – i.e. how many near-duplicate images "leak" across the train/validation/test boundary – and what fraction of each split is affected.

Usage

```
dhash_audit(grouped_df)
```

Arguments

`grouped_df`	A data frame with columns `path`, `group_id`, and `split`, as produced by `find_duplicate_groups()` (with `split` supplied to `compute_hashes()`).

Value

An object of class `leakage_report` (a list) with:

leaked_groups data frame of group_ids that span >1 split, with the splits involved and member count

n_leaked_images total images sitting in a leaked group

pct_leaked_by_split named numeric vector, % of each split's images that belong to a leaked group

total_images total images audited

Examples

```
groups <- data.frame(
  path = c("a.jpg", "b.jpg", "c.jpg", "d.jpg"),
  split = c("train", "test", "train", "val"),
  group_id = c(1, 1, 2, 3)
)
dhash_audit(groups)
```

find_duplicate_groups *Group near-duplicate images by hash distance*

Description

Given a table of image hashes (as produced by `compute_hashes()`), clusters images into duplicate groups: any two images within `threshold` Hamming distance of each other are placed in the same group, and group membership propagates transitively (A~B~C all end up in one group even if A and C aren't directly close).

Usage

```
find_duplicate_groups(hash_df, threshold = 5)
```

Arguments

`hash_df`	A data frame with at least columns `path` and `hash`, as returned by `compute_hashes()`.
`threshold`	Maximum Hamming distance (in bits, out of 64) for two images to be considered near-duplicates. Defaults to 5, a reasonably conservative value for dHash; lower it for stricter (exact-duplicate-only) matching, raise it to catch more aggressive near-duplicates at the cost of more false positives.

Value

`hash_df` with an added integer column `group_id`. Images with no near-duplicates get their own unique `group_id`. Row order is preserved.

Examples

```
hashes <- data.frame(
  path = c("a.jpg", "b.jpg", "c.jpg", "d.jpg"),
  hash = c("0000000000000000", "0000000000000001",
           "ffffffffffffffff", "1234567890abcdef")
)
find_duplicate_groups(hashes, threshold = 5)
```

hamming_distance *Hamming distance between two hex-encoded perceptual hashes*

Description

Computes the number of differing bits between two perceptual hashes encoded as hexadecimal strings (as produced by `compute_hashes()`).

Usage

```
hamming_distance(hash1, hash2)
```

Arguments

hash1	A character scalar or vector of hex-encoded hashes.
hash2	A character scalar or vector of hex-encoded hashes, the same length as hash1 (or length 1, recycled).

Value

An integer vector of Hamming distances (number of differing bits).

Examples

```
hamming_distance("ff00", "ff01")
hamming_distance(c("ffff", "0000"), "0000")
```

Index