Package 'soilFlux'

March 26, 2026

Title Physics-Informed Neural Networks for Soil Water Retention Curves

Version 0.1.5

Description Implements a physics-informed one-dimensional convolutional neural network (CNN1D-PINN) for estimating the complete soil water retention curve (SWRC) as a continuous function of matric potential, from soil texture, organic carbon, bulk density, and depth. The network architecture ensures strict monotonic decrease of volumetric water content with increasing suction by construction, through cumulative integration of non-negative slope outputs (monotone integral architecture). Four physics-based residual constraints adapted from Norouzi et al. (2025) <doi:10.1029/2024WR038149> are embedded in the loss function: (S1) linearity at the dry end (pF in [5, 7.6]); (S2) non-negativity at pF = 6.2; (S3) non-positivity at pF = 7.6; and (S4) a near-zero derivative in the saturated plateau region (pF in [-2, -0.3]). Includes tools for data preparation, model training, dense prediction, performance metrics, texture classification, and publication-quality visualisation.

License MIT + file LICENSE

Encoding UTF-8

SystemRequirements Python (>= 3.8), TensorFlow (>= 2.14), Keras (>= 3.0)

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr (>= 1.1.0), tidyr (>= 1.3.0), ggplot2 (>= 3.4.0), purrr (>= 1.0.0), tibble (>= 3.2.0), stringr (>= 1.5.0), reticulate (>= 1.34.0), tensorflow (>= 2.14.0), rlang (>= 1.1.0)

Suggests keras3 (>= 1.0.0), ggtern, readxl (>= 1.4.0), scales (>= 1.2.0), knitr, rmarkdown, testthat (>= 3.0.0), withr

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/HugoMachadoRodrigues/soilFlux,
https://hugomachadorodrigues.github.io/soilFlux/

BugReports https://github.com/HugoMachadoRodrigues/soilFlux/issues

NeedsCompilation no

Author Hugo Rodrigues [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8070-8126>)

Maintainer Hugo Rodrigues <rodrigues.machado.hugo@gmail.com>

Repository CRAN

Date/Publication 2026-03-26 09:30:02 UTC

Contents

`add_texture` *Add texture classification column to a data frame*

Description

Add texture classification column to a data frame

Usage

```
add_texture(
  df,
  sand_col = "sand_total",
  silt_col = "silt",
  clay_col = "clay",
  out_col = "Texture"
)
```

Arguments

`df`	A data frame.
`sand_col`	Column name for sand (default `"sand_total"`).
`silt_col`	Column name for silt (default `"silt"`).
`clay_col`	Column name for clay (default `"clay"`).
`out_col`	Name of the output column (default `"Texture"`).

Value

The input data frame with an additional `out_col` column.

Examples

```
df <- data.frame(sand_total = c(70, 20), silt = c(15, 50), clay = c(15, 30))
add_texture(df)
```

`apply_minmax` *Apply a fitted min-max scaler to a data frame*

Description

Scales `df[, scaler$cols]` using the precomputed min/range. Returns a numeric matrix with the same column order as `scaler$cols`.

Usage

```
apply_minmax(df, scaler)
```

Arguments

`df`	A data frame or tibble.
`scaler`	A scaler object returned by `fit_minmax()`.

Value

A numeric matrix scaled to approximately [0, 1] per column. Columns correspond to `scaler$cols`.

Examples

```
df_train <- data.frame(sand = c(20, 40, 60), clay = c(30, 20, 10))
sc       <- fit_minmax(df_train, c("sand", "clay"))
df_new   <- data.frame(sand = c(50, 25), clay = c(15, 28))
apply_minmax(df_new, sc)
```

`build_residual_sets` *Build physics residual point sets (S1 – S4)*

Description

Generates four sets of collocation points (random soil-property vectors and associated pF values) used to evaluate the physics constraints during training.

Usage

```
build_residual_sets(
  df_raw,
  x_inputs,
  S1 = 1500L,
  S2 = 500L,
  S3 = 500L,
  S4 = 1500L,
  pF_lin_min = 5,
  pF_lin_max = 7.6,
  pF0_pos = 6.2,
  pF1_neg = 7.6,
  pF_sat_min = -2,
  pF_sat_max = -0.3,
  seed = 123L
)
```

Arguments

df_raw	Data frame with covariate columns (training split).
x_inputs	Character vector of covariate column names.
S1	Number of S1 points — dry-end linearity (default 1500).
S2	Number of S2 points — non-negativity at pF0 (default 500).
S3	Number of S3 points — non-positivity at pF1 (default 500).
S4	Number of S4 points — saturated plateau (default 1500).
pF_lin_min	Lower pF for the S1 linearity constraint (default 5.0).
pF_lin_max	Upper pF for the S1 linearity constraint (default 7.6).
pF0_pos	pF at which theta must be >= 0 — S2 (default 6.2).
pF1_neg	pF at which theta must be <= 0 — S3 (default 7.6).
pF_sat_min	Lower pF for the S4 plateau constraint (default -2.0).
pF_sat_max	Upper pF for the S4 plateau constraint (default -0.3).
seed	Integer random seed (default 123).

Value

A named list with four data frames: `set1`, `set2`, `set3`, `set4`. Each data frame has one row per collocation point, with columns corresponding to `x_inputs` (sampled uniformly within training range) and a `pF` column.

Examples

```
df <- data.frame(
  clay       = c(20, 30, 10),
  silt       = c(30, 40, 20),
  sand_total = c(50, 30, 70),
```

```
  Depth_num  = c(15, 30, 60)
)
sets <- build_residual_sets(df, c("clay", "silt", "sand_total", "Depth_num"),
                            S1 = 50L, S2 = 20L, S3 = 20L, S4 = 50L)
```

`build_swrc_model` *Build the CNN1D monotone-integral SWRC model*

Description

Constructs a Keras model implementing the monotone-integral architecture of Norouzi et al. (2025). The returned list contains two models that share weights:

- `theta_model` — full prediction model (pF query + covariates → theta).
- `param_model` — extracts the saturated water content (theta_s) from covariates only.

Usage

```
build_swrc_model(n_covariates, hidden = c(128L, 64L), dropout = 0.1, K = 64L)
```

Arguments

`n_covariates`	Integer. Number of soil-property covariates (`p`).
`hidden`	Integer vector of length 2. Number of filters in the first and second Conv1D layers (default `c(128L, 64L)`).
`dropout`	Numeric dropout rate after each Conv1D layer (default `0.10`).
`K`	Integer. Number of knot points for the cumulative integration grid (default `64L`).

Value

A named list:

`theta_model` Keras model: inputs `[Xseq_knots, pf_norm]`, output shape `[N, 2]` (theta_hat, theta_s).

`param_model` Keras model: input `Xseq_knots`, output shape `[N, 1]` (theta_s only).

`K` The `K` value used.

`dk` The knot spacing `1 / (K - 1)`.

`knot_grid` Numeric vector of knot positions.

Examples

```
mod <- build_swrc_model(n_covariates = 9L)
```

classify_texture *Classify soil texture according to the USDA system*

Description

Returns the USDA texture class name for each row based on the sand, silt, and clay fractions. Inputs are expected in per-cent (0–100) and must sum to approximately 100.

Usage

```
classify_texture(sand, silt, clay, tol = 1)
```

Arguments

sand	Numeric vector: sand content (%).
silt	Numeric vector: silt content (%).
clay	Numeric vector: clay content (%).
tol	Tolerance for the 100 % sum check (default 1.0).

Value

Character vector of USDA texture class names. Returns NA for rows where values are missing or do not sum to approximately 100.

Examples

```
classify_texture(sand = c(70, 20, 10, 40),
                 silt = c(15, 50, 30, 40),
                 clay = c(15, 30, 60, 20))
```

data_prep *Data preparation for CNN1D SWRC modelling*

Description

Functions to prepare soil data for input to the CNN1D monotone-integral model, including 3-D sequence array construction and observation matrix creation.

evaluate_swrc *Compute metrics from a swrc_fit on new data*

Description

A convenience wrapper that calls `predict_swrc()` and `swrc_metrics()`.

Usage

```
evaluate_swrc(object, newdata, obs_col = "theta_n")
```

Arguments

`object`	A `swrc_fit` object.
`newdata`	Data frame with covariate columns and `matric_head` and `theta_n` columns.
`obs_col`	Name of the observed theta column in `newdata` (default `"theta_n"`).

Value

A tibble with columns `R2`, `RMSE`, `MAE`, `n`.

fit_minmax *Fit a min-max scaler from a training data frame*

Description

Computes per-column minimum and range from `df[, cols]`. Constant columns (range == 0) are assigned a range of 1 to avoid division by zero.

Usage

```
fit_minmax(df, cols)
```

Arguments

`df`	A data frame or tibble.
`cols`	Character vector of column names to include.

Value

A list with elements:

`min` Named numeric vector of per-column minima.

`rng` Named numeric vector of per-column ranges.

`cols` The character vector `cols` (stored for later use).

Examples

```
df <- data.frame(sand = c(20, 40, 60), clay = c(30, 20, 10))
sc <- fit_minmax(df, c("sand", "clay"))
sc
```

`fit_swrc` *Fit a physics-informed CNN1D SWRC model*

Description

The main user-facing function for training. Given prepared training and (optionally) validation data, it builds the model, creates physics residual sets, runs the training loop with early stopping, and returns a fitted object for prediction and evaluation.

Usage

```
fit_swrc(
  train_df,
  x_inputs,
  val_df = NULL,
  hidden = c(128L, 64L),
  dropout = 0.1,
  lr = 0.001,
  epochs = 80L,
  batch_size = 256L,
  patience = 5L,
  K = 64L,
  lambdas = norouzi_lambdas("norouzi"),
  S1 = 1500L,
  S2 = 500L,
  S3 = 500L,
  S4 = 1500L,
  pF_lin_min = 5,
  pF_lin_max = 7.6,
  pF0_pos = 6.2,
  pF1_neg = 7.6,
  pF_sat_min = -2,
  pF_sat_max = -0.3,
  wet_split_cm = 4.2,
  w_wet = 1,
  w_dry = 1,
  pf_left = -2,
  pf_right = 7.6,
  seed = 123L,
  verbose = TRUE
)
```

Arguments

train_df	Data frame for training (output of prepare_swrc_data()).
x_inputs	Character vector of covariate column names.
val_df	Optional validation data frame (same structure as train_df). If NULL, early stopping is skipped.
hidden	Integer vector of length 2: Conv1D filter counts (default c(128L, 64L)).
dropout	Dropout rate (default 0.10).
lr	Learning rate for the Adam optimizer (default 1e-3).
epochs	Maximum number of epochs (default 80).
batch_size	Mini-batch size (default 256).
patience	Early-stopping patience in multiples of 5 epochs (default 5).
K	Number of knot points (default 64L).
lambdas	Named list of loss weights; use norouzi_lambdas() to generate (default: norouzi_lambdas("norouzi"
S1, S2, S3, S4	Residual set sizes (defaults: 1500, 500, 500, 1500).
pF_lin_min	Lower pF for S1 linearity constraint (default 5.0).
pF_lin_max	Upper pF for S1 linearity constraint (default 7.6).
pF0_pos	pF threshold for S2 (default 6.2).
pF1_neg	pF threshold for S3 (default 7.6).
pF_sat_min	Lower pF for S4 (default -2.0).
pF_sat_max	Upper pF for S4 (default -0.3).
wet_split_cm	Matric head (cm) separating wet/dry end (default 4.2).
w_wet	Sample weight for wet observations (default 1.0).
w_dry	Sample weight for dry observations (default 1.0).
pf_left	Left pF domain boundary (default -2.0).
pf_right	Right pF domain boundary (default 7.6).
seed	Random seed (default 123).
verbose	Logical; print progress (default TRUE).

Value

An S3 object of class swrc_fit, a named list containing:

theta_model The fitted Keras model.

param_model The theta_s extractor model.

x_inputs Covariate names used.

scaler Fitted min-max scaler.

K Number of knot points.

dk Knot spacing.

knot_grid Knot positions in [0, 1].

`pf_left`,`pf_right` pF domain boundaries.

`theta_factor` Unit multiplier for theta.

`best_epoch` Epoch at which validation loss was minimised.

`lambdas` Loss weights used during training.

`history` Data frame of per-epoch training/validation losses.

Examples

```
if (reticulate::py_module_available("tensorflow")) {
  df  <- prepare_swrc_data(swrc_example, depth_col = "depth")
  fit <- fit_swrc(df,
                  x_inputs = c("clay", "silt", "bd_gcm3", "soc", "Depth_num"),
                  epochs = 2L, verbose = FALSE)
}
```

---

fix_bd_units *Detect and correct bulk-density units*

Description

If the median raw value is > 10 it is assumed to be in kg/m3 and is divided by 100 to convert to g/cm3.

Usage

```
fix_bd_units(bd_raw)
```

Arguments

`bd_raw`	Numeric vector of raw bulk-density values.

Value

Numeric vector in g/cm3.

Examples

```
fix_bd_units(c(1.2, 1.45, 1.3))   # already g/cm3
fix_bd_units(c(120, 145, 130))    # kg/m3 -> g/cm3
```

head_from_pf *Convert pF to matric head (cm)*

Description

Convert pF to matric head (cm)

Usage

```
head_from_pf(pf)
```

Arguments

pf	Numeric vector of pF values.

Value

Numeric vector of matric head values in cm.

Examples

```
head_from_pf(c(0, 1, 2, 3, 4.2))
```

head_normalize *Normalise matric head (cm) to the pF domain*

Description

Convenience wrapper: converts head to pF then normalises.

Usage

```
head_normalize(h_cm, pf_left = -2, pf_right = 7.6)
```

Arguments

h_cm	Numeric vector of matric heads in cm.
pf_left	Left boundary (default -2).
pf_right	Right boundary (default 7.6).

Value

Numeric vector in [0, 1].

Examples

```
head_normalize(c(1, 10, 100, 15850))
```

---

invert_minmax *Invert a min-max scaling transformation*

Description

Converts scaled values back to original units.

Usage

```
invert_minmax(X_scaled, scaler)
```

Arguments

X_scaled	Numeric matrix (or vector) of scaled values.
scaler	A scaler object returned by fit_minmax().

Value

Numeric matrix in the original (unscaled) units.

Examples

```
df <- data.frame(sand = c(20, 40, 60), clay = c(30, 20, 10))
sc <- fit_minmax(df, c("sand", "clay"))
Xs <- apply_minmax(df, sc)
invert_minmax(Xs, sc)
```

---

io *Save and load fitted SWRC models*

Description

Functions to persist a fitted swrc_fit object to disk and reload it for later use (prediction, spatial mapping, etc.).

load_swrc_model *Load a previously saved SWRC model from disk*

Description

Reconstructs the CNN1D Keras model from the saved weights and metadata and returns a `swrc_fit`-compatible list that can be passed to `predict_swrc()`, `predict_swrc_dense()`, etc.

Usage

```
load_swrc_model(dir = "./models/swrc", name = "swrc_model")
```

Arguments

`dir`	Directory containing the saved files (default `"./models/swrc"`).
`name`	Stem name used when saving (default `"swrc_model"`).

Value

A `swrc_fit` object (without `history` or `param_model`).

Examples

```
if (reticulate::py_module_available("tensorflow")) {
  df  <- prepare_swrc_data(swrc_example, depth_col = "depth")
  fit <- fit_swrc(df,
                  x_inputs = c("clay", "silt", "bd_gcm3", "soc", "Depth_num"),
                  epochs = 2L, verbose = FALSE)
  save_swrc_model(fit, dir = tempdir(), name = "model_test")
  fit2 <- load_swrc_model(tempdir(), "model_test")
}
```

metrics *Performance metrics for SWRC models*

Description

Functions to compute R^2, RMSE, and MAE for soil water content predictions.

`model` *CNN1D monotone-integral model architecture*

Description

Build the physics-informed 1-D CNN with a monotone integral output layer, as described in Norouzi et al. (2025).

Details

Architecture:

The model takes two inputs:

1. `Xseq_knots`: a 3-D tensor of shape `[N, K, p+1]` — for each observation, `p` scaled covariates are broadcast across `K` knot positions, and the knot positions themselves form the last channel.
2. `pf_norm`: a 2-D tensor of shape `[N, 1]` — the query pF value normalised to `[0, 1]`.

The output satisfies:

$$\hat{\theta}(pF) = \theta_s - \int_0^{pF} \text{softplus}(s(t))\, dt$$

where $s(t)$ is a 1-channel convolutional output. Monotone decrease is guaranteed **by construction** because the integrand is always positive.

References

Norouzi, A. M., et al. (2025). Physics-Informed Neural Networks for Estimating a Continuous Form of the Soil Water Retention Curve. *Journal of Hydrology*.

`norouzi_lambdas` *Return default Norouzi et al. (2025) loss weights (lambdas)*

Description

Table 1 of Norouzi et al. (2025) defines six loss-weight hyperparameters. This function returns them as a named list that can be passed to `fit_swrc()` and `compute_physics_loss()`.

Usage

```
norouzi_lambdas(config = c("norouzi", "smooth"))
```

Arguments

`config`	Character string; either `"norouzi"` (exact replication, default) or `"smooth"` (lambda3 = 10 for a smoother dry-end).

Value

A named list:

`lambda_wet` Weight for wet-end data loss (lambda1 = 1).

`lambda_dry` Weight for dry-end data loss (lambda2 = 10).

`lambda3` S1 dry-end linearity (lambda3 = 1 or 10).

`lambda4` S2 non-negativity at pF0 (lambda4 = 1000).

`lambda5` S3 non-positivity at pF1 (lambda5 = 1000).

`lambda6` S4 saturated-plateau flatness (lambda6 = 1).

Examples

```
norouzi_lambdas()
norouzi_lambdas("smooth")
```

---

parse_depth *Parse a soil depth string into midpoint and label*

Description

Accepts strings of the form `"0-5"`, `"5-15"`, `"100"`, etc. and returns the numeric midpoint and a human-readable label (e.g. `"0-5 cm"`).

Usage

```
parse_depth(s)
```

Arguments

`s`	A character string describing a depth interval or single depth.

Value

A named list with elements:

`mid` Numeric midpoint in cm.

`label` Character label, e.g. `"0-5 cm"`.

Examples

```
parse_depth("0-5")
parse_depth("100-200")
parse_depth("30")
```

parse_depth_column *Parse depth column in a data frame*

Description

Applies `parse_depth()` row-wise and appends `Depth_num` and `Depth_label` columns.

Usage

```
parse_depth_column(df, depth_col = "depth")
```

Arguments

`df`	A data frame.
`depth_col`	Name of the depth column (character string).

Value

The input data frame with two extra columns: `Depth_num` (numeric midpoint) and `Depth_label` (factor, ordered by depth).

Examples

```
df <- data.frame(depth = c("0-5", "5-15", "15-30"), x = 1:3)
parse_depth_column(df, "depth")
```

pf_from_head *Convert matric head (cm) to pF*

Description

Convert matric head (cm) to pF

Usage

```
pf_from_head(h_cm)
```

Arguments

`h_cm`	Numeric vector of matric head values in cm (positive).

Value

Numeric vector of pF values ($\log_{10}(h)$).

Examples

```
pf_from_head(c(1, 10, 100, 1000, 15850))
```

`pf_normalize` *Normalise pF values to [0, 1]*

Description

Maps the pF domain `[pf_left, pf_right]` linearly to `[0, 1]`. Values outside the domain are clipped.

Usage

```
pf_normalize(pf, pf_left = -2, pf_right = 7.6)
```

Arguments

`pf`	Numeric vector of pF values.
`pf_left`	Left boundary of the pF domain (default `-2`).
`pf_right`	Right boundary of the pF domain (default `7.6`).

Value

Numeric vector in `[0, 1]`.

Examples

```
pf_normalize(c(-2, 0, 4, 7.6))
```

`physics` *Physics-informed constraints for SWRC modelling*

Description

Functions for defining, building, and evaluating the four physics-based residual constraints of Norouzi et al. (2025).

References

Norouzi, A. M., et al. (2025). Physics-Informed Neural Networks for Estimating a Continuous Form of the Soil Water Retention Curve. *Journal of Hydrology*.

plots *Publication-quality plots for SWRC analysis*

Description

Functions for visualising soil water retention curves, predicted vs. observed scatter plots, and model performance metrics.

plot_pred_obs *Plot predicted vs. observed water content*

Description

Creates a scatter plot of predicted vs. observed theta with a 1:1 line and optional regression line, optionally faceted by a grouping variable.

Usage

```
plot_pred_obs(
  df,
  obs_col = "theta_n",
  pred_col = "theta_predicted",
  group_col = NULL,
  show_lm = TRUE,
  show_stats = TRUE,
  ncol = 5L,
  base_size = 12,
  point_alpha = 0.25,
  title = NULL
)
```

Arguments

df	Data frame containing observed and predicted columns.
obs_col	Column name for observed theta (default "theta_n").
pred_col	Column name for predicted theta (default "theta_predicted").
group_col	Column name for facet grouping (default NULL).
show_lm	Logical; add a linear regression line (default TRUE).
show_stats	Logical; add R^2, RMSE, MAE text annotations (default TRUE).
ncol	Number of facet columns when group_col is supplied (default 5).
base_size	Base font size (default 12).
point_alpha	Point transparency (default 0.25).
title	Plot title (default NULL).

Value

A ggplot object.

Examples

```
df_plot <- data.frame(
  theta_n         = c(0.30, 0.25, 0.20, 0.15, 0.10),
  theta_predicted = c(0.28, 0.26, 0.22, 0.14, 0.11)
)
plot_pred_obs(df_plot)
```

---

plot_swrc *Plot soil water retention curves (SWRC)*

Description

Creates a ggplot2 figure showing continuous SWRC predictions (lines) optionally overlaid with observed data points.

Usage

```
plot_swrc(
  pred_curves,
  obs_points = NULL,
  curve_col = "theta",
  obs_col = "theta_n",
  group_col = "PEDON_ID",
  facet_row = NULL,
  facet_col = NULL,
  x_limits = NULL,
  y_limits = c(-0.25, 7.75),
  line_colour = "steelblue4",
  point_colour = "black",
  base_size = 12,
  title = NULL
)
```

Arguments

pred_curves	A data frame (or tibble) of dense curve predictions, typically returned by predict_swrc_dense(). Must contain columns pF and theta.
obs_points	Optional data frame of observed data. Must contain pF and theta columns (or matric_head and a theta column).
curve_col	Column in pred_curves for the predicted theta (default "theta").
obs_col	Column in obs_points for observed theta (default "theta_n").

group_col	Column name used to distinguish individual profiles in pred_curves (default "PEDON_ID").
facet_row	Column for facet rows (default NULL).
facet_col	Column for facet columns (default NULL).
x_limits	Numeric vector of length 2 for the x-axis (theta) range (default NULL, auto).
y_limits	Numeric vector of length 2 for the y-axis (pF) range (default c(-0.25, 7.75)).
line_colour	Colour of the predicted curve lines (default "steelblue4").
point_colour	Colour of the observed data points (default "black").
base_size	Base font size for theme_bw (default 12).
title	Plot title (default NULL).

Value

A ggplot object.

Examples

```
pred_curves <- data.frame(
  PEDON_ID = rep(c("P1", "P2"), each = 5),
  pF       = rep(c(0, 1, 2, 3, 4), 2),
  theta    = c(0.42, 0.36, 0.28, 0.18, 0.08,
               0.38, 0.32, 0.25, 0.16, 0.07)
)
plot_swrc(pred_curves, group_col = "PEDON_ID")
```

---

plot_swrc_metrics *Plot model performance metric comparison*

Description

Creates a bar chart comparing R^2, RMSE, and MAE across multiple models or configurations.

Usage

```
plot_swrc_metrics(
  metrics_df,
  model_col = "model",
  palette = "Blues",
  base_size = 12,
  title = NULL
)
```

Arguments

metrics_df	A data frame with columns model (character/factor), R2, RMSE, MAE. Typically produced by stacking the output of swrc_metrics().
model_col	Column name for the model identifier (default "model").
palette	RColorBrewer palette (default "Blues").
base_size	Base font size (default 12).
title	Plot title (default NULL).

Value

A ggplot object.

Examples

```
m1 <- swrc_metrics(c(0.30, 0.25, 0.20), c(0.28, 0.26, 0.22)) |>
  dplyr::mutate(model = "Model 1")
m2 <- swrc_metrics(c(0.30, 0.25, 0.20), c(0.29, 0.24, 0.21)) |>
  dplyr::mutate(model = "Model 2")
plot_swrc_metrics(dplyr::bind_rows(m1, m2))
```

---

plot_training_history *Plot training loss history*

Description

Plots the per-epoch training and (optionally) validation loss from the history slot of a swrc_fit object.

Usage

```
plot_training_history(
  fit,
  loss_col = "loss",
  val_col = "val_mse",
  base_size = 12
)
```

Arguments

fit	A swrc_fit object.
loss_col	Column in fit$history to display (default "loss").
val_col	Validation loss column (default "val_mse"). Pass NULL to omit.
base_size	Base font size (default 12).

Value

A ggplot object.

`predict` *Prediction from fitted SWRC models*

Description

Functions for generating soil water content predictions from a fitted `swrc_fit` object, both at specific pF points and as dense continuous curves.

Value

See `predict_swrc()` and `predict_swrc_dense()` for return value details.

`predict.swrc_fit` *Predict method for swrc_fit*

Description

Dispatches to `predict_swrc()`.

Usage

```
## S3 method for class 'swrc_fit'
predict(object, newdata, pf = NULL, heads = NULL, ...)
```

Arguments

`object`	A `swrc_fit` object returned by `fit_swrc()`.
`newdata`	A data frame with the same covariate columns used during training (i.e. `object$x_inputs`). Must have a `matric_head` column **or** supply `pf` directly.
`pf`	Optional numeric vector of pF values (overrides `matric_head` in `newdata`).
`heads`	Optional numeric vector of matric heads in cm (overrides `matric_head` in `newdata`).
`...`	Ignored.

Value

A numeric vector of predicted volumetric water content values (m3/m3), one per row in `newdata`.

predict_swrc *Predict water content at specific pF or matric-head values*

Description

Given a fitted `swrc_fit` object and a new data frame of soil properties, returns predicted volumetric water content at each supplied pF (or matric head) value.

Usage

```
predict_swrc(object, newdata, pf = NULL, heads = NULL, ...)
```

Arguments

`object`	A `swrc_fit` object returned by `fit_swrc()`.
`newdata`	A data frame with the same covariate columns used during training (i.e. `object$x_inputs`). Must have a `matric_head` column **or** supply `pf` directly.
`pf`	Optional numeric vector of pF values (overrides `matric_head` in `newdata`).
`heads`	Optional numeric vector of matric heads in cm (overrides `matric_head` in `newdata`).
`...`	Ignored.

Value

A numeric vector of predicted theta values (m3/m3), one per row in `newdata`.

Examples

```
if (reticulate::py_module_available("tensorflow")) {
  df   <- prepare_swrc_data(swrc_example, depth_col = "depth")
  fit  <- fit_swrc(df,
                   x_inputs = c("clay", "silt", "bd_gcm3", "soc", "Depth_num"),
                   epochs = 2L, verbose = FALSE)
  pred <- predict_swrc(fit, newdata = df)
}
```

predict_swrc_dense *Predict dense SWRC curves for a set of soil profiles*

Description

For each unique (PEDON_ID × depth) profile in newdata, predicts theta across a dense grid of pF values and returns a tidy long-format tibble.

Usage

```
predict_swrc_dense(
  object,
  newdata,
  n_points = 1000L,
  pf_range = NULL,
  id_cols = c("PEDON_ID", "Depth_num", "Depth_label", "Texture")
)
```

Arguments

object	A swrc_fit object.
newdata	A data frame with covariate columns plus (optionally) PEDON_ID, Depth_num, Depth_label, and Texture.
n_points	Number of equally spaced pF points (default 1000).
pf_range	Numeric vector of length 2: min and max pF values for the output grid (default c(-2, 7.6)).
id_cols	Character vector of columns used to identify profiles (default c("PEDON_ID","Depth_num","Depth_labe

Value

A tibble with columns: all id_cols present in newdata, pF, matric_head, and theta (predicted volumetric water content in m3/m3).

Examples

```
if (reticulate::py_module_available("tensorflow")) {
  df    <- prepare_swrc_data(swrc_example, depth_col = "depth")
  fit   <- fit_swrc(df,
                    x_inputs = c("clay", "silt", "bd_gcm3", "soc", "Depth_num"),
                    epochs = 2L, verbose = FALSE)
  dense <- predict_swrc_dense(fit, newdata = df, n_points = 50)
}
```

predict_theta_s *Extract saturated water content (theta_s) from covariates*

Description

Uses the `param_model` (which maps covariate inputs to theta_s) to extract the modelled saturated water content for each row of `newdata`.

Usage

```
predict_theta_s(object, newdata)
```

Arguments

`object`	A `swrc_fit` object.
`newdata`	Data frame with covariate columns.

Value

Numeric vector of theta_s values (m3/m3).

prepare_swrc_data *Prepare a soil data frame for SWRC modelling*

Description

A convenience wrapper that:

1. Renames columns to standard names.
2. Parses the depth column.
3. Fixes bulk-density units.
4. Detects and normalises volumetric water content units.
5. Computes per-profile maximum theta.
6. Drops rows with missing key variables.

Usage

```
prepare_swrc_data(
  df,
  x_cols = NULL,
  depth_col = "depth",
  fix_bd = TRUE,
  fix_theta = TRUE
)
```

Arguments

df	A data frame with soil characterization data.
x_cols	Named character vector mapping standard names to actual column names. Standard names: "PEDON_ID", "sand", "silt", "clay", "soc", "bd", "matric_head", "water_content", "depth". Unneeded variables may be omitted.
depth_col	Column name for depth (character string, default "depth"). If already parsed, set to NULL.
fix_bd	Logical; apply fix_bd_units() to bulk density (default TRUE).
fix_theta	Logical; scale theta to m3/m3 if needed (default TRUE).

Value

A tibble with standardised columns plus Depth_num, Depth_label, bd_gcm3, theta_n (normalised WC), and theta_max_n (per-profile maximum theta).

Examples

```
df <- data.frame(
  ID           = rep("P01", 3),
  sand_pct     = c(50, 50, 50),
  silt         = c(30, 30, 30),
  clay         = c(20, 20, 20),
  soc          = c(1.2, 1.2, 1.2),
  bd_gcc       = c(1.3, 1.3, 1.3),
  matric_head  = c(10, 100, 1000),
  theta        = c(0.40, 0.30, 0.20),
  depth        = c("0-30", "0-30", "0-30")
)
df_prep <- prepare_swrc_data(df,
  x_cols = c(PEDON_ID = "ID", sand = "sand_pct", bd = "bd_gcc",
             water_content = "theta"))
```

---

print.swrc_fit *Print method for swrc_fit*

Description

Print method for swrc_fit

Usage

```
## S3 method for class 'swrc_fit'
print(x, ...)
```

Arguments

x	An swrc_fit object.
...	Ignored.

Value

Invisibly returns x (called for its side effect of printing a summary of the fitted model to the console).

---

save_swrc_model *Save a fitted SWRC model to disk*

Description

Saves the Keras model weights as an HDF5 file and the R metadata (scalers, hyperparameters, etc.) as an .rds file inside dir.

Usage

```
save_swrc_model(fit, dir, name = "swrc_model")
```

Arguments

fit	A swrc_fit object returned by fit_swrc().
dir	Directory where the model will be saved. Created if it does not exist.
name	Stem name for the output files (default "swrc_model").

Value

Invisibly returns a named list with paths to the two files:

weights_path Path to the .weights.h5 file.

meta_path Path to the .rds metadata file.

Examples

```
if (reticulate::py_module_available("tensorflow")) {
  df  <- prepare_swrc_data(swrc_example, depth_col = "depth")
  fit <- fit_swrc(df,
                  x_inputs = c("clay", "silt", "bd_gcm3", "soc", "Depth_num"),
                  epochs = 2L, verbose = FALSE)
  save_swrc_model(fit, dir = tempdir(), name = "model_test")
}
```

scale *Min-max feature scaling*

Description

Fit and apply a column-wise min-max scaler to a data frame or matrix, mapping each feature to [0, 1].

summary.swrc_fit *Summary method for swrc_fit*

Description

Summary method for swrc_fit

Usage

```
## S3 method for class 'swrc_fit'
summary(object, ...)
```

Arguments

object	An swrc_fit object.
...	Ignored.

Value

Invisibly returns object (called for its side effect of printing a detailed summary including covariates, training parameters, and loss weights to the console).

swrc_example *Example soil water retention dataset*

Description

A synthetic but physically realistic soil characterisation dataset generated to illustrate the functions in **soilFlux**. The data mimics the structure of the Florida Soil Characterization Database (FSCD) used in Rodrigues et al. / Norouzi et al. (2025), with van Genuchten curves used to produce internally consistent water-content observations.

Usage

```
swrc_example
```

Format

A data frame with 4 800 rows and 14 columns:

PEDON_ID Character. Unique soil profile identifier (e.g. `"P0001"`).

sand_total Numeric. Total sand content (%).

silt Numeric. Silt content (%).

clay Numeric. Clay content (%).

soc Numeric. Soil organic carbon (%).

bd Numeric. Bulk density (g/cm3).

sand_vf Numeric. Very fine sand fraction (%).

sand_f Numeric. Fine sand fraction (%).

sand_m Numeric. Medium sand fraction (%).

sand_c Numeric. Coarse sand fraction (%).

matric_head Numeric. Matric head (cm H2O, positive).

water_content Numeric. Volumetric water content (m3/m3). Approximately 1% of values are `NA` to simulate missing measurements.

depth Character. Depth interval (e.g. `"0-5"`, `"5-15"`, etc.).

Texture Character. USDA texture class (one of: Sand, Loamy Sand, Sandy Loam, Loam, Silt Loam, Clay Loam, Clay).

Details

The dataset contains 120 unique profiles (`PEDON_ID`) across five depth intervals (0–5, 5–15, 15–30, 30–60, 60–100 cm) and eight matric-head points per depth (pF approximately 0, 1, 1.5, 2, 2.5, 3, 4.2, 7). Profiles are evenly distributed across seven USDA texture classes.

Water-content observations were generated with the van Genuchten (1980) equation using parameters that vary realistically with texture and depth, then small Gaussian noise was added.

Source

Synthetic dataset generated by `data-raw/create_example_data.R`.

References

van Genuchten, M. T. (1980). A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, 44(5), 892–898.

Examples

```
data("swrc_example")
head(swrc_example)
table(swrc_example$Texture[!duplicated(swrc_example$PEDON_ID)])

# Prepare for modelling
df <- prepare_swrc_data(swrc_example, depth_col = "depth")
```

swrc_metrics *Compute regression metrics for SWRC predictions*

Description

Returns R^2, RMSE, and MAE between observed and predicted volumetric water content (or any continuous response).

Usage

```
swrc_metrics(observed, predicted, na.rm = TRUE)
```

Arguments

observed	Numeric vector of observed values.
predicted	Numeric vector of predicted values (same length).
na.rm	Logical; remove NA pairs before computing (default TRUE).

Value

A tibble::tibble() with one row and columns R2, RMSE, MAE, n (number of non-missing pairs).

Examples

```
obs  <- c(0.30, 0.25, 0.20, 0.15, 0.10)
pred <- c(0.28, 0.26, 0.22, 0.14, 0.11)
swrc_metrics(obs, pred)
```

swrc_metrics_by_group *Compute regression metrics by group*

Description

Applies swrc_metrics() within each level of one or more grouping variables, returning a tidy data frame.

Usage

```
swrc_metrics_by_group(df, obs_col, pred_col, group_col, na.rm = TRUE)
```

Arguments

df	A data frame containing observed and predicted columns.
obs_col	Name of the observed-values column (character string).
pred_col	Name of the predicted-values column (character string).
group_col	Character vector of grouping column names.
na.rm	Logical; passed to swrc_metrics() (default TRUE).

Value

A tibble with one row per group and columns: grouping variables, R2, RMSE, MAE, n.

Examples

```
df <- data.frame(
  obs  = c(0.30, 0.25, 0.20, 0.15, 0.10, 0.35, 0.28, 0.18),
  pred = c(0.28, 0.26, 0.22, 0.14, 0.11, 0.33, 0.27, 0.19),
  texture = c("Clay","Clay","Clay","Clay","Clay","Sand","Sand","Sand")
)
swrc_metrics_by_group(df, "obs", "pred", "texture")
```

---

swrc_model_exists *Check whether a model directory contains a valid saved model*

Description

Check whether a model directory contains a valid saved model

Usage

```
swrc_model_exists(dir = "./models/swrc", name = "swrc_model")
```

Arguments

dir	Directory path.
name	Model stem name.

Value

Logical scalar.

texture *USDA soil texture classification*

Description

Classify soil samples into USDA texture classes from sand, silt, and clay percentages, and create texture triangle plots.

texture_triangle *Plot a soil texture triangle (ternary diagram)*

Description

Creates a ternary diagram coloured by a grouping variable using ggplot2. Requires the ggtern package (not a hard dependency).

Usage

```
texture_triangle(
  df,
  sand_col = "sand_total",
  silt_col = "silt",
  clay_col = "clay",
  color_col = NULL,
  title = "Soil Texture Triangle",
  point_size = 1.5,
  alpha = 0.6
)
```

Arguments

df	A data frame.
sand_col	Column name for sand (default "sand_total").
silt_col	Column name for silt (default "silt").
clay_col	Column name for clay (default "clay").
color_col	Column name for colouring points (default NULL for a single colour).
title	Plot title.
point_size	Point size (default 1.5).
alpha	Point transparency (default 0.6).

Value

A ggplot object (or ggtern object if ggtern is available).

Examples

```
if (requireNamespace("ggtern", quietly = TRUE)) {
  df <- data.frame(sand_total = c(70, 20, 10),
                   silt = c(15, 50, 30),
                   clay = c(15, 30, 60),
                   Texture = c("Sand", "Silt Loam", "Clay"))
  p <- texture_triangle(df, color_col = "Texture")
}
```

`theta_unit_factor` *Detect theta unit scale factor*

Description

Returns 100 if the maximum value suggests percentage units (> 1.5), otherwise returns 1 (m3/m3 assumed).

Usage

```
theta_unit_factor(theta_vec)
```

Arguments

`theta_vec`	Numeric vector of volumetric water content values.

Value

Numeric scalar: 100 (percentage) or 1 (m3/m3).

Examples

```
theta_unit_factor(c(0.1, 0.35, 0.5))  # returns 1
theta_unit_factor(c(10, 35, 50))       # returns 100
```

`train` *Training the CNN1D SWRC model*

Description

High-level and low-level functions for training the physics-informed CNN1D model, including the eager-mode train step and the main fitting loop with early stopping.

`utils` *Utility functions for soilFlux*

Description

Internal and exported helpers for pF conversion, depth parsing, and unit detection.

Index