

# Package ‘ONAM’

January 26, 2026

**Type** Package

**Title** Fitting Interpretable Neural Additive Models Using  
Orthogonalization

**Version** 1.0.1

**Description** An algorithm for fitting interpretable additive neural  
networks for identifiable and visualizable feature effects using post  
hoc orthogonalization. Fit custom neural networks intuitively using  
established 'R' 'formula' notation, including interaction effects of  
arbitrary order while preserving identifiability to enable a  
functional decomposition of the prediction function. For more details see  
Koehler et al. (2025) <[doi:10.1038/s44387-025-00033-7](https://doi.org/10.1038/s44387-025-00033-7)>.

**License** MIT + file LICENSE

**BugReports** [https://github.com/Koehlibert/ONAM\\_R/issues](https://github.com/Koehlibert/ONAM_R/issues)

**Depends** keras3, reticulate

**Imports** dplyr, scales, rlang, ggplot2, pROC

**Suggests** akima, RColorBrewer, testthat (>= 3.0.0)

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** David Köhler [aut, cre] (ORCID:  
<<https://orcid.org/0009-0006-0027-4046>>)

**Maintainer** David Köhler <[koehler@imbie.uni-bonn.de](mailto:koehler@imbie.uni-bonn.de)>

**Repository** CRAN

**Date/Publication** 2026-01-26 11:50:02 UTC

## Contents

decompose . . . . .	2
install_conda_env . . . . .	3

onam . . . . .	4
plot_inter_effect . . . . .	6
plot_main_effect . . . . .	7
predict.onam . . . . .	8
summary.onam . . . . .	9
<b>Index</b>	<b>10</b>

---

decompose	<i>Get variance decomposition of orthogonal neural additive model</i>
-----------	---

---

**Description**

Get variance decomposition of orthogonal neural additive model

**Usage**

decompose(object, data = NULL)

**Arguments**

- object Either model of class onam as returned from [onam](#) or model evaluation outcome as returned from [predict.onam](#)
- data Data for which the model is to be evaluated. If NULL (DEFAULT), the data from model fitting is used. with which model was fitted.

**Value**

Returns a named vector of percentage of variance explained by each interaction order.

**Examples**

```
# Basic example for a simple ONAM-model
# Create training data
n <- 1000
x1 <- runif(n, -2, 2)
x2 <- runif(n, -2, 2)
y <- sin(x1) + ifelse(x2 > 0, pweibull(x2, shape = 3),
  pweibull(-x2, shape = 0.5)) +
  x1 * x2
data_train <- cbind(x1, x2, y)
# Define model
model_formula <- y ~ mod1(x1) + mod1(x2) +
  mod1(x1, x2)
mod1 <- function(inputs) {
  outputs <- inputs %>%
    layer_dense(units = 16, activation = "relu") %>%
    layer_dense(units = 8, activation = "linear",
      use_bias = TRUE) %>%
```

```
      layer_dense(units = 1, activation = "linear",
                  use_bias = TRUE)
    keras_model(inputs, outputs)
  }
  list_of_deep_models <- list(mod1 = mod1)
  # Fit model
  mod <- onam(model_formula, list_of_deep_models,
             data_train, n_ensemble = 1, epochs = 10)
  decompose(mod)
```

---

install_conda_env	<i>Set up conda environment for keras functionality</i>
-------------------	---

---

## Description

Helper function to install Keras and packages necessary for package functionality into a conda environment. Use this function if `keras3::install_keras()` does not work, esp. on windows machines.

## Usage

```
install_conda_env(
  envname = "r-keras",
  python_version = "python=3.10",
  overwrite = FALSE
)
```

## Arguments

envname	Name for the conda environment to be created.
python_version	Python version to be installed in the conda environment.
overwrite	Should an existing conda environment of name envname be overwritten if present?

## Value

No return value, called for side effects

## See Also

[keras3::install\\_keras\(\)](#)

---

onam

*Fit orthogonal neural additive model*


---

## Description

Fits an interpretable neural additive model with post hoc orthogonalization for a given network architecture and user-specified feature sets.

## Usage

```
onam(
  formula,
  list_of_deep_models,
  data,
  model = NULL,
  prediction_function = NULL,
  model_data = NULL,
  categorical_features = NULL,
  target = "continuous",
  n_ensemble = 10,
  epochs = 500,
  callback = NULL,
  progresstext = FALSE,
  verbose = 0
)
```

## Arguments

formula	Formula for model fitting. Specify deep parts with the same name as <code>list_of_deep_models</code> .
list_of_deep_models	List of named models used in <code>model_formula</code> .
data	Data to be fitted
model	Prediction model that is to be explained. Output of the model as returned from <code>prediction_function(model)</code> will be used as model output. If <code>NULL</code> (default), the outcome has to be present in <code>data</code> .
prediction_function	Prediction function to be used to generate the outcome. Only used if <code>model</code> is specified. If <code>NULL</code> (default), S3-method based on the <code>model</code> argument is used.
model_data	Data used for generating predictions of <code>model</code> . Necessary for some models that require specific data formats, i.e. <code>xgboost</code> . If <code>NULL</code> (default), <code>data</code> is used. Only used if <code>model</code> is specified.
categorical_features	Vector of feature names of categorical features.

target	Target of prediction task. Can be either "continuous" or "binary". For "continuous"(default), an additive model for the prediction of a continuous outcome is fitted. For "binary", a binary classification with sigmoid activation in the last layer is fitted.
n_ensemble	Number of orthogonal neural additive model ensembles
epochs	Number of epochs to train the model. See <a href="#">fit</a> for details.
callback	Callback to be called during training. See <a href="#">fit</a> for details.
progresstext	Show model fitting progress. If TRUE, shows current number of ensemble being fitted
verbose	Verbose argument for internal model fitting. used for debugging. See <a href="#">fit</a> for details.

### Value

Returns a model object of class `onam`, containing all ensemble members, ensemble weights, and main and interaction effect outputs.

### Examples

```
# Basic example for a simple ONAM-model
# Create training data
n <- 1000
x1 <- runif(n, -2, 2)
x2 <- runif(n, -2, 2)
y <- sin(x1) + ifelse(x2 > 0, pweibull(x2, shape = 3),
  pweibull(-x2, shape = 0.5)) +
  x1 * x2
data_train <- cbind(x1, x2, y)
# Define model
model_formula <- y ~ mod1(x1) + mod1(x2) +
  mod1(x1, x2)
mod1 <- function(inputs) {
  outputs <- inputs %>%
    layer_dense(units = 16, activation = "relu") %>%
    layer_dense(units = 8, activation = "linear",
      use_bias = TRUE) %>%
    layer_dense(units = 1, activation = "linear",
      use_bias = TRUE)
  keras_model(inputs, outputs)
}
list_of_deep_models <- list(mod1 = mod1)
# Fit model
mod <- onam(model_formula, list_of_deep_models,
  data_train, n_ensemble = 1, epochs = 10)
summary(mod)
```

---

plot_inter_effect	<i>Plot Interaction Effect</i>
-------------------	--------------------------------

---

## Description

Plot Interaction Effect

## Usage

```
plot_inter_effect(
  object,
  feature1,
  feature2,
  interpolate = FALSE,
  custom_colors = "spectral",
  n_interpolate = 200
)
```

## Arguments

object	Either model of class <code>onam</code> as returned from <code>onam</code> or model evaluation outcome as returned from <code>predict.onam</code>
feature1, feature2	Effects to be plotted.
interpolate	If TRUE, values will be interpolated for a smooth plot. If FALSE (default), only observations in the data will be plotted.
custom_colors	color palette object for the interaction plot. Default is "spectral", returning a color palette based on the spectral theme.
n_interpolate	number of values per coordinate axis to interpolate. Ignored if 'interpolate = FALSE'.

## Value

Returns a 'ggplot2' object of the specified effect interaction

## Examples

```
# Basic example for a simple ONAM-model
# Create training data
n <- 1000
x1 <- runif(n, -2, 2)
x2 <- runif(n, -2, 2)
y <- sin(x1) + ifelse(x2 > 0, pweibull(x2, shape = 3),
  pweibull(-x2, shape = 0.5)) +
  x1 * x2
data_train <- cbind(x1, x2, y)
# Define model
```

```

model_formula <- y ~ mod1(x1) + mod1(x2) +
  mod1(x1, x2)
mod1 <- function(inputs) {
  outputs <- inputs %>%
    layer_dense(units = 16, activation = "relu") %>%
    layer_dense(units = 8, activation = "linear",
      use_bias = TRUE) %>%
    layer_dense(units = 1, activation = "linear",
      use_bias = TRUE)
  keras_model(inputs, outputs)
}
list_of_deep_models <- list(mod1 = mod1)
# Fit model
mod <- onam(model_formula, list_of_deep_models,
  data_train, n_ensemble = 1, epochs = 10)
plot_inter_effect(mod, "x1", "x2")

```

---

plot_main_effect	<i>Plot Main Effect</i>
------------------	-------------------------

---

## Description

Plot Main Effect

## Usage

```
plot_main_effect(object, feature)
```

## Arguments

object	Either model of class <code>onam</code> as returned from <a href="#">onam</a> or model evaluation outcome as returned from <a href="#">predict.onam</a>
feature	Feature for which the effect is to be plotted, must be present in the model formula. For interaction terms, use <code>plotInteractionEffect</code>

## Value

Returns a `ggplot2` object of the specified effect

## Examples

```

# Basic example for a simple ONAM-model
# Create training data
n <- 1000
x1 <- runif(n, -2, 2)
x2 <- runif(n, -2, 2)
y <- sin(x1) + ifelse(x2 > 0, pweibull(x2, shape = 3),
  pweibull(-x2, shape = 0.5)) +

```

```

    x1 * x2
data_train <- cbind(x1, x2, y)
# Define model
model_formula <- y ~ mod1(x1) + mod1(x2) +
  mod1(x1, x2)
mod1 <- function(inputs) {
  outputs <- inputs %>%
    layer_dense(units = 16, activation = "relu") %>%
    layer_dense(units = 8, activation = "linear",
      use_bias = TRUE) %>%
    layer_dense(units = 1, activation = "linear",
      use_bias = TRUE)
  keras_model(inputs, outputs)
}
list_of_deep_models <- list(mod1 = mod1)
# Fit model
mod <- onam(model_formula, list_of_deep_models,
  data_train, n_ensemble = 1, epochs = 10)
plot_main_effect(mod, "x1")

```

---

predict.onam

---

*Evaluate orthogonal neural additive model*


---

## Description

Evaluate orthogonal neural additive model

## Usage

```

## S3 method for class 'onam'
predict(object, newdata = NULL, ...)

```

## Arguments

object	model of class onam as returned from <a href="#">onam</a> to be evaluated
newdata	Data for which the model is to be evaluated. If NULL (default), data with which model was fitted is used.
...	some methods for this generic require additional arguments. None are used in this method.

## Value

Returns a list containing data, model output for each observation in newdata and main and interaction effects obtained by the model



---

summary.onam	<i>Get summary of an onam object</i>
--------------	--------------------------------------

---

**Description**

generates a summary of a fitted onam object including information on ensembling strategy and performance metrics such as correlation and degree of interpretability

**Usage**

```
## S3 method for class 'onam'  
summary(object, ...)  
  
## S3 method for class 'summary.onam'  
print(x, ...)
```

**Arguments**

object	onam object of class onam as returned from <a href="#">onam</a> to be summarized
...	further arguments passed to or from other methods.
x	object of class <a href="#">summary.onam</a> .

**Details**

For examples see [example\(onam\)](#)

**Value**

Gives summary of the onam object, including model inputs, number of ensembles, correlation of model output and original outcome variable, and interpretability metrics `i_1` and `i_2`

# Index

`decompose`, [2](#)

`example(onam)`, [9](#)

`fit`, [5](#)

`install_conda_env`, [3](#)

`keras3::install_keras()`, [3](#)

`onam`, [2](#), [4](#), [6–9](#)

`plot_inter_effect`, [6](#)

`plot_main_effect`, [7](#)

`predict(predict.onam)`, [8](#)

`predict.onam`, [2](#), [6](#), [7](#), [8](#)

`print.summary.onam(summary.onam)`, [9](#)

`summary.onam`, [9](#), [9](#)