ML and AI project

Shubhang Periwal 19201104

4/22/2020

```
data <- load("data_activity_recognition.RData")
tensorflow::tf$random$set_seed(0)
set.seed(19201104)
```

```
library(keras)
library(tfruns)
library(reticulate)
library(jsonlite)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(Rtsne)
```

```
dim(x_test)#checking the dimensions of the variables
```

```
## [1] 1520  125   45
```

```
x_org_train <- x_train#storing the original data for CNN
x_org_test <- x_test #storing the original data for CNN
range(x_test)
```

```
## [1] -116.08  120.53
```

```
x_train <-array_reshape(x_train,c(nrow(x_train),125*45))
#converting x data to 2 dimensions
x_test <-array_reshape(x_test,c(nrow(x_test),125*45))

dim(x_test)
```

```
## [1] 1520 5625
```

```
#checking for new dimensions
```

```
x_train <- scale(x_train)#scaling data
x_test <- scale(x_test)
x_test <- as.matrix(x_test) #converting data to matrix to perform computaions such
# as PCA and correlation
x_train <- as.matrix(x_train)
xval <- sample(1:nrow(x_train),600)
#separating validation set from original set
xntrain <- setdiff(1:nrow(x_train),xval)
x_val <- x_train[xval,]
x_train <- x_train[xntrain,]
```

```
corr <- cor(x_train)
#performing correlation, to remove similar variables
# this is done to reduce the dimensions and decrease the
# number of parameters of the model, which might lead to a higher accuracy
corr[upper.tri(corr)] <- 0
#making diagnal as 0, so I don't end up removing
# all the variables and upper triangular part as 0
diag(corr)<- 0
```

```
x_test <- x_test[,!apply(corr,2,function(x) any(x > .70))]
x_train <- x_train[,!apply(corr,2,function(x) any(x > .70))]
x_val <- x_val[,!apply(corr,2,function(x) any(x > .70))]
#removing highly related data with correlation of more than 0.7
dim(x_train)
```

```
## [1] 7000 1729
```

```
#computing pca to reduce dimensions, to improve algorithm accuracy
pca <- prcomp(x_train)
prop <- cumsum(pca$sdev^2)/sum(pca$sdev^2)# compute cumulative proportion of variance
Q <-length( prop[prop<0.95] ) #maintaining atleast 95% of information
Q
```

```
## [1] 902
```

```
#checking for Q, if large then we need to further decrease the number of dimensions
```

```
prop <- cumsum(pca$sdev^2)/sum(pca$sdev^2)# compute cumulative proportion of variance
Q <-length( prop[prop<0.90] ) #maintaining atleast 90% of information
Q
```

```
## [1] 656
```

```
# only a handful is retained
#more than 90 % data can be explained using  dimensions, so
```

Using 656 dimensions and retaining 90% information

```r
#using pca to to finally convert the main data to reduced number of dimensions
x1_train <- predict(pca,x_train)[,1:Q]
x1_test <- predict(pca,x_test)[,1:Q]
x1_val <- predict(pca,x_val)[,1:Q]
```

```r
#converting data to factor, followed by one-hot encoding for analysis
#separating y into training testing and validation
y1_train <- as.numeric(factor(y_train))
y1_test <- as.numeric(factor(y_test))
yc_train <- to_categorical(y1_train)
yc_test <- to_categorical(y1_test)
yc_train <- yc_train
yc_test <- yc_test
yf_train <- as.factor(y_train)
yf_test <- as.factor(y_test)

yc_test <- yc_test[,-1]
yc_train <- yc_train[,-1]
yc_val <- yc_train[xval,]
yc_train <- yc_train[xntrain,]
V <- ncol(x1_train)
```

```r
#trying out a basic neural network with 4 layers to check out how neural network performs
model2 <-  keras_model_sequential() %>%
layer_dense(units = 128, activation = "relu", input_shape = V) %>%# first hidden
layer_dense(units = 128, activation = "relu") %>%     #second layer
layer_dense(units = 64, activation = "relu") %>%      #third layer
layer_dense(units = 64, activation = "relu") %>%    #fourth layer
layer_dense(units = 19, activation = "softmax") %>%  #outputlayer
compile(
loss = "categorical_crossentropy", metrics = "accuracy",
optimizer = optimizer_sgd(),
)
# count parameters
count_params(model2)
```

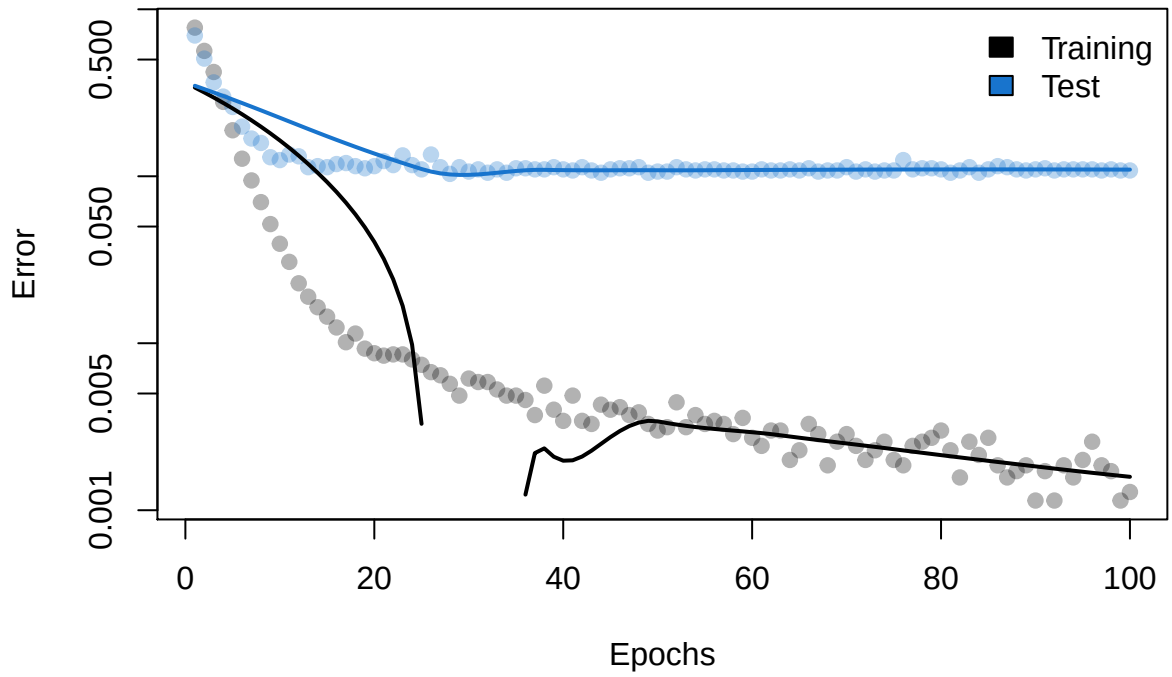
```
## [1] 114259
```

```r
fit <- model2 %>% fit(
x = x1_train, y = yc_train,
validation_data = list(x1_val, yc_val),
epochs = 100,
verbose = 0,

)
# store accuracy on test set for each run
score <- model2 %>% evaluate(
  x1_test, yc_test,
  verbose = 0
)
score
```

```
## $loss
```

```
## [1] 0.6068159
##
## $accuracy
## [1] 0.8855263
```

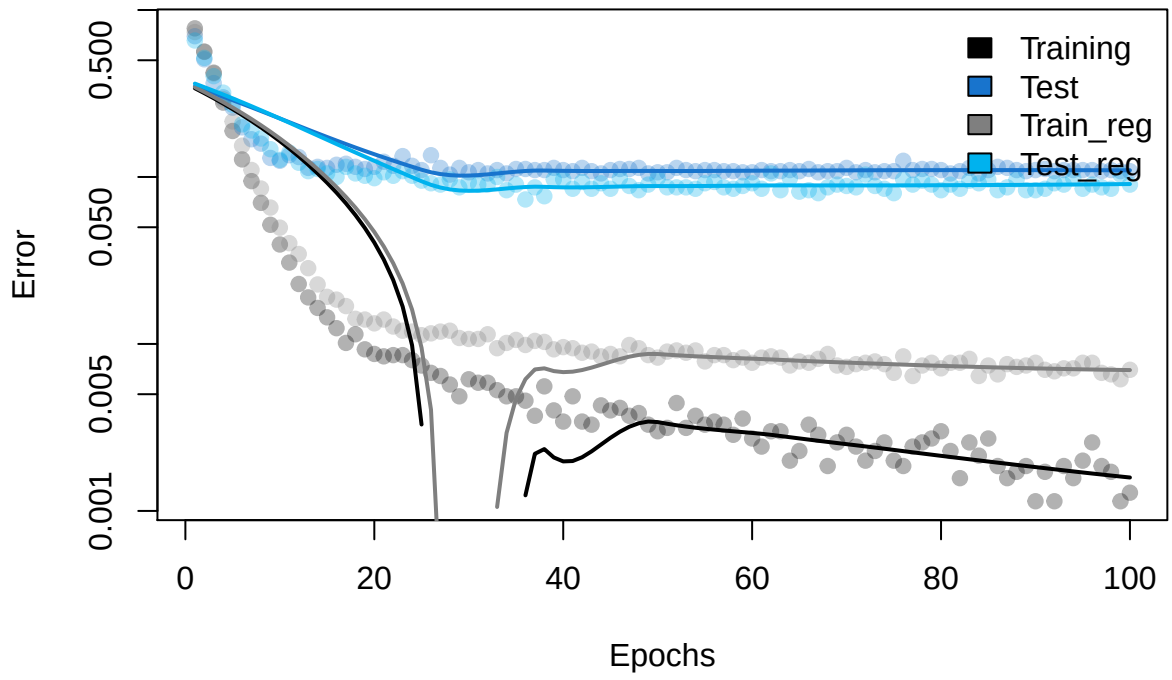
```
# to add a smooth line to points
smooth_line <- function(y) {
x <- 1:length(y)
out <- predict( loess(y ~ x) )
return(out)
}
# some colors will be used later
cols <- c("black", "dodgerblue3", "gray50", "deepskyblue2")
# check performance ---> error
out <- 1 - cbind(fit$metrics$accuracy,fit$metrics$val_accuracy)
matplot(out, pch = 19, ylab = "Error", xlab = "Epochs",
col = adjustcolor(cols[1:2], 0.3),log = "y")
# on log scale to visualize better differences


matlines(apply(out, 2, smooth_line), lty = 1, col = cols[1:2], lwd = 2)
legend("topright", legend = c("Training", "Test"),
fill = cols[1:2], bty = "n")
```



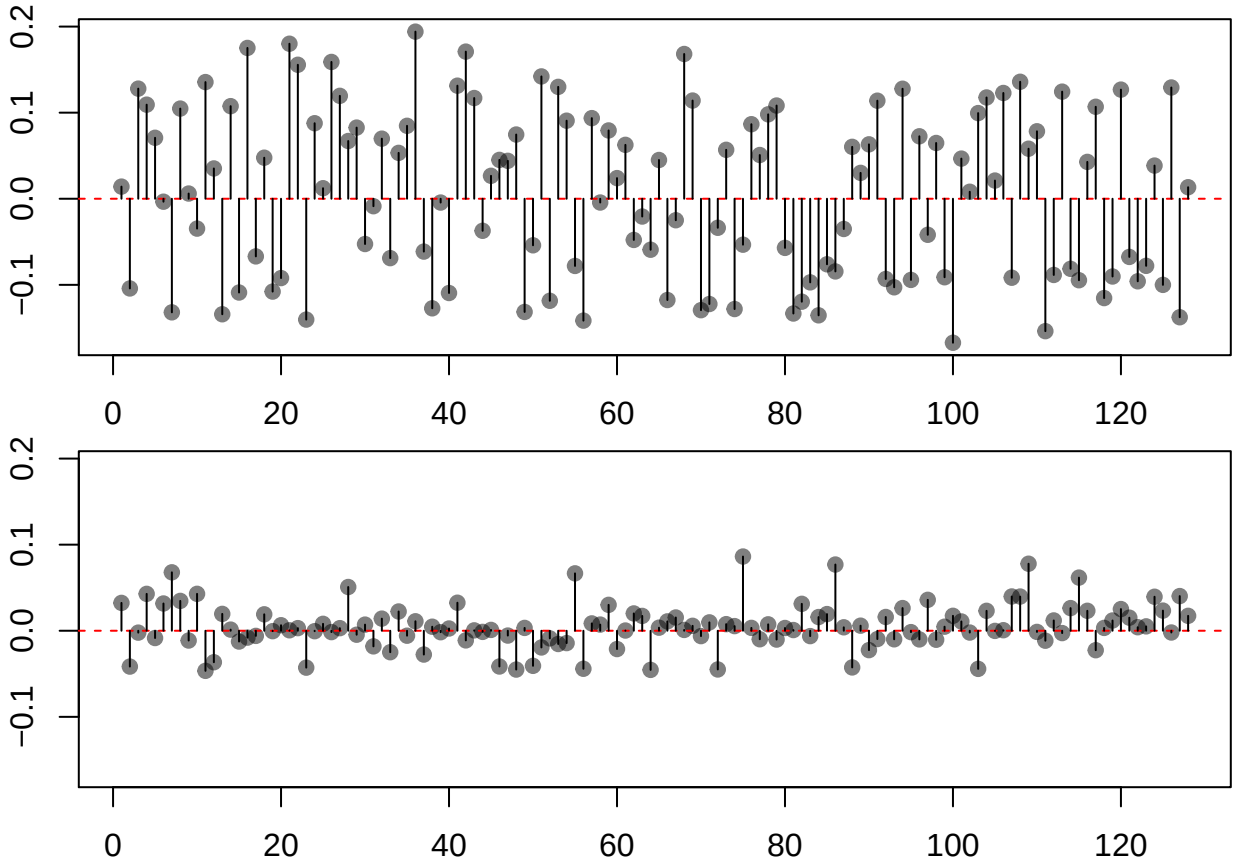
```r
#trying the same model as above, but with regularization to prevent overfitting
model_reg <- keras_model_sequential() %>%
layer_dense(units = 128, activation = "relu", input_shape = V,
kernel_regularizer = regularizer_l2(l = 0.009)) %>%
layer_dense(units = 128, activation = "relu",
kernel_regularizer = regularizer_l2(l = 0.009)) %>%
layer_dense(units = 64, activation = "relu",
kernel_regularizer = regularizer_l2(l = 0.009)) %>%
layer_dense(units = 32, activation = "relu",
kernel_regularizer = regularizer_l2(l = 0.009)) %>%
layer_dense(units = 19, activation = "softmax") %>%
compile(
loss = "categorical_crossentropy",
optimizer = optimizer_sgd(),
metrics = "accuracy"
)
# train and evaluate on test data at each epoch
fit_reg <- model_reg %>% fit(
x = x1_train, y = yc_train,
validation_data = list(x1_val, yc_val),
epochs = 100,
verbose = 0
)

# store accuracy on test set for each run
score <- model_reg %>% evaluate(
  x1_test, yc_test,
  verbose = 0
)
score
```

```
## $loss
## [1] 0.6161871
##
## $accuracy
## [1] 0.9131579
```

```r
out <- 1 - cbind(fit$metrics$accuracy,
fit$metrics$val_accuracy,
fit_reg$metrics$accuracy,
fit_reg$metrics$val_accuracy)
# check performance
matplot(out, pch = 19, ylab = "Error", xlab = "Epochs",
col = adjustcolor(cols, 0.3),
log = "y")

matlines(apply(out, 2, smooth_line), lty = 1, col = cols, lwd = 2)
legend("topright", legend = c("Training", "Test", "Train_reg", "Test_reg"),
fill = cols, bty = "n")
```




```
# get all weights
w_all <- get_weights(model2)
w_all_reg <- get_weights(model_reg)
# weights of first hidden layer
# one input --> 64 units
w <- w_all[[3]][1,]
w_reg <- w_all_reg[[3]][1,]
# compare visually the magnitudes
par(mfrow = c(2,1), mar = c(2,2,0.5,0.5))
r <- range(w)
n <- length(w)
plot(w, ylim = r, pch = 19, col = adjustcolor(1, 0.5))
abline(h = 0, lty = 2, col = "red")
segments(1:n, 0, 1:n, w)
#
plot(w_reg, ylim = r, pch = 19, col = adjustcolor(1, 0.5))
abline(h = 0, lty = 2, col = "red")
segments(1:n, 0, 1:n, w_reg)
```




```
#CNN data preprocessing
# we need to convert this data to 3 dimensions, so that we can use convolution neural network
dim(x_org_train)
```

```
## [1] 7600  125   45
```

```
dim(x_org_test)
```

```
## [1] 1520  125   45
```

```
xcnn_train <-array_reshape(x_org_train,c(nrow(x_org_train),125,45,1))
xcnn_test <-array_reshape(x_org_test,c(nrow(x_org_test),125,45,1))
xcnn_val <- xcnn_train[xval,1:125,1:45,1]

xcnn_train <- xcnn_train[xntrain,1:125,1:45,1]
xcnn_train <- array_reshape(xcnn_train,c(nrow(xcnn_train),125,45,1))
xcnn_val <- array_reshape(xcnn_val,c(nrow(xcnn_val),125,45,1))
```

```
#cnn model
model <- keras_model_sequential() %>%
#
# convolutional layers

layer_conv_2d(filters = 32, kernel_size = c(2,2), activation = "relu",input_shape = c(125,45,1)) %>%
```

```
layer_max_pooling_2d(pool_size = c(2,2)) %>%

layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = "relu") %>%

layer_max_pooling_2d(pool_size = c(2,2)) %>%

layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = "relu") %>%

layer_max_pooling_2d(pool_size = c(2,2)) %>%
#
# fully connected layers
layer_flatten() %>%
layer_dense(units = 64, activation = "relu", kernel_regularizer = regularizer_l2(0.1)) %>%
layer_dropout(0.1) %>%
layer_dense(units = 19, activation = "softmax") %>%
#
# compile
compile(
loss = "categorical_crossentropy",
metrics = "accuracy",
optimizer = optimizer_adam()
)
# model training
# NOTE : this will require some time
fit <- model %>% fit(
x = xcnn_train, y = yc_train,
validation_data = list(xcnn_val, yc_val),
epochs = 50,
batch_size = 40, # roughly 0.5% of training observations
verbose = 0
)

score <- model %>% evaluate(
  xcnn_test, yc_test,
  verbose = 0
)
score
```

```
## $loss
## [1] 0.1585414
##
## $accuracy
## [1] 0.9868421
```

```
# to add a smooth line to points
smooth_line <- function(y) {
x <- 1:length(y)
out <- predict( loess(y ~ x) )
return(out)
}
# check performance
cols <- c("black", "dodgerblue3")
out <- cbind(fit$metrics$accuracy,
```

```
fit$metrics$val_accuracy)
matplot(out, pch = 19, ylab = "Accuracy", xlab = "Epochs",
col = adjustcolor(cols[1:2], 0.3),
log = "y")
matlines(apply(out, 2, smooth_line), lty = 1, col = cols[1:2], lwd = 2)
legend("bottomright", legend = c("Training", "Test"),
fill = cols[1:2], bty = "n")
```

```
model_reg <- keras_model_sequential() %>%
#
# convolutional layers

layer_conv_2d(filters = 32, kernel_size = c(2,2), activation = "relu",input_shape = c(125,45,1),kernel_

layer_max_pooling_2d(pool_size = c(2,2)) %>%

layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = "relu",kernel_regularizer = regularizer_

layer_max_pooling_2d(pool_size = c(2,2)) %>%

layer_conv_2d(filters = 64, kernel_size = c(3,3), activation = "relu",kernel_regularizer = regularizer_

layer_max_pooling_2d(pool_size = c(2,2)) %>%
#
# fully connected layers
layer_flatten() %>%
layer_dense(units = 64, activation = "relu", kernel_regularizer = regularizer_l2(0.1)) %>%
layer_dropout(0.1) %>%
layer_dense(units = 19, activation = "softmax") %>%
#
# compile
compile(
loss = "categorical_crossentropy",
metrics = "accuracy",
optimizer = optimizer_adam()
)
# model training
# NOTE : this will require some time
fit <- model %>% fit(
x = xcnn_train, y = yc_train,
validation_data = list(xcnn_val, yc_val),
epochs = 50,
batch_size = 40, # roughly 0.5% of training observations
verbose = 0
)

score <- model %>% evaluate(
  xcnn_test, yc_test,
  verbose = 0
)
score
```

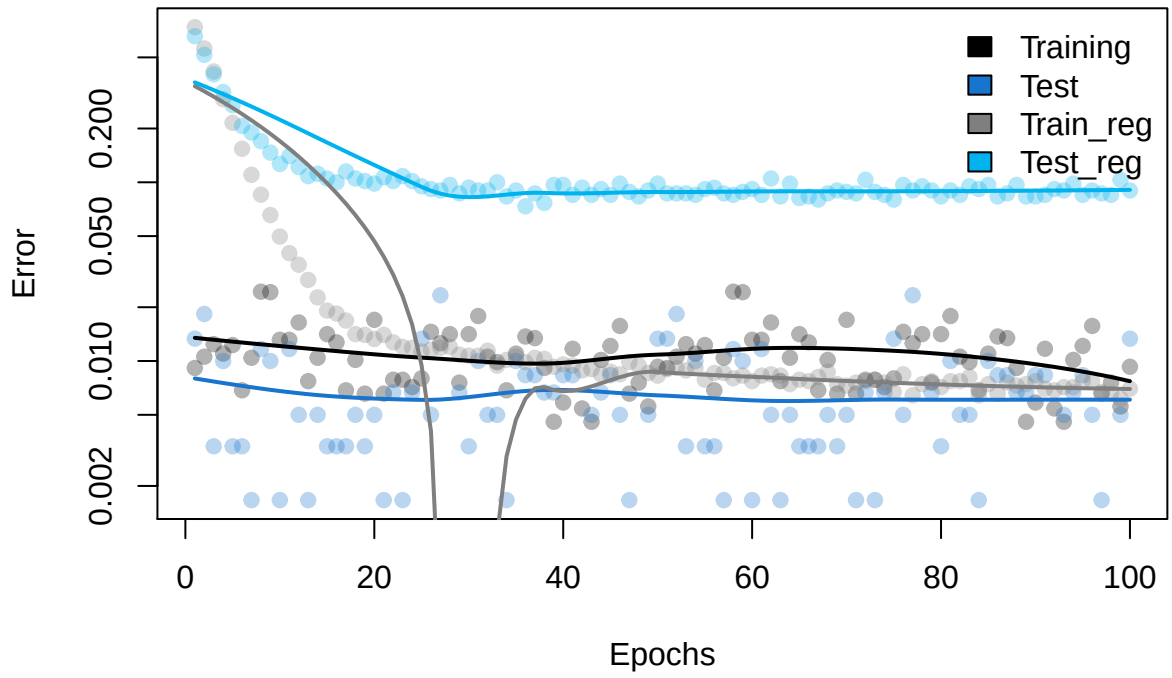
```
## $loss
```

```
## [1] 0.1581469
##
## $accuracy
## [1] 0.9868421
```

```
out <- 1 - cbind(fit$metrics$accuracy,
fit$metrics$val_accuracy,
fit_reg$metrics$accuracy,
fit_reg$metrics$val_accuracy)
# check performance
matplot(out, pch = 19, ylab = "Error", xlab = "Epochs",
col = adjustcolor(cols, 0.3),
log = "y")
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log = log): 6 y values <= 0
## omitted from logarithmic plot
```

```
matlines(apply(out, 2, smooth_line), lty = 1, col = cols, lwd = 2)
legend("topright", legend = c("Training", "Test", "Train_reg", "Test_reg"),
fill = cols, bty = "n")
```




```
#setting a grid of values for the flags/hyperparameters of interest:
hdlayer1 <- c(128,64,32)
dropout1 <- c(0,0.1,0.3)
```

```r
hdlayer2 <- c(64,16)
dropout2 <- c(0,0.2)
hdlayer3 <- c(64,32,16)
dropout3 <- c(0,0.1)
# total combinations 3 x 3 x 2 x 2 x 3 x 2 = 216
```

```r
# run ------------------------------------------------------------
runs <- tuning_run("mlai_1.R", #creating runs to simulate output
                    runs_dir = "nn",
                    flags = list(
                   hdlayer_1 = hdlayer1,
                   dropout_1 = dropout1,
                   hdlayer_2 = hdlayer2,
                   dropout_2 = dropout2,
                   hdlayer_3 = hdlayer3,
                   dropout_3 = dropout3
                      ),
                  sample = 0.1)
```

```
## 216 total combinations of flags (sampled to 21 combinations)

## Training run 1/21 (flags = list(32, 0.3, 16, 0.2, 16, 0.1))

## Using run directory nn/2020-04-28T12-06-00Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn/2020-04-28T12-06-00Z

## Training run 2/21 (flags = list(32, 0, 16, 0.2, 32, 0.1))

## Using run directory nn/2020-04-28T12-06-55Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-06-55Z

## Training run 3/21 (flags = list(32, 0.1, 64, 0, 16, 0.1))

## Using run directory nn/2020-04-28T12-07-52Z


##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-07-52Z

## Training run 4/21 (flags = list(64, 0, 16, 0, 64, 0))

## Using run directory nn/2020-04-28T12-08-33Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-08-33Z

## Training run 5/21 (flags = list(32, 0, 16, 0, 16, 0.1))

## Using run directory nn/2020-04-28T12-09-04Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-09-04Z

## Training run 6/21 (flags = list(64, 0, 64, 0, 16, 0))

## Using run directory nn/2020-04-28T12-09-54Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-09-54Z

## Training run 7/21 (flags = list(32, 0, 64, 0, 32, 0.1))

## Using run directory nn/2020-04-28T12-10-20Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-10-20Z

## Training run 8/21 (flags = list(32, 0.3, 16, 0, 64, 0))

## Using run directory nn/2020-04-28T12-11-07Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-11-07Z

## Training run 9/21 (flags = list(32, 0.1, 16, 0.2, 64, 0.1))

## Using run directory nn/2020-04-28T12-11-46Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-11-46Z

## Training run 10/21 (flags = list(64, 0.1, 64, 0, 16, 0))

## Using run directory nn/2020-04-28T12-12-39Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-12-39Z

## Training run 11/21 (flags = list(128, 0.3, 64, 0, 16, 0.1))

## Using run directory nn/2020-04-28T12-13-20Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-13-20Z

## Training run 12/21 (flags = list(128, 0, 64, 0, 16, 0.1))

## Using run directory nn/2020-04-28T12-14-02Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-14-02Z

## Training run 13/21 (flags = list(64, 0.1, 64, 0.2, 64, 0))

## Using run directory nn/2020-04-28T12-14-51Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-14-51Z

## Training run 14/21 (flags = list(64, 0.3, 16, 0, 16, 0))

## Using run directory nn/2020-04-28T12-15-24Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-15-24Z

## Training run 15/21 (flags = list(32, 0, 16, 0.2, 16, 0.1))

## Using run directory nn/2020-04-28T12-16-15Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-16-15Z

## Training run 16/21 (flags = list(64, 0.1, 16, 0.2, 16, 0.1))

## Using run directory nn/2020-04-28T12-16-58Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-16-58Z

## Training run 17/21 (flags = list(128, 0.3, 64, 0, 32, 0.1))

## Using run directory nn/2020-04-28T12-17-54Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-17-54Z

## Training run 18/21 (flags = list(128, 0.1, 16, 0, 16, 0.1))

## Using run directory nn/2020-04-28T12-18-36Z
```

```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-18-36Z

## Training run 19/21 (flags = list(32, 0.1, 16, 0, 16, 0.1))

## Using run directory nn/2020-04-28T12-19-15Z


##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +    .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-19-15Z

## Training run 20/21 (flags = list(32, 0, 16, 0, 64, 0))

## Using run directory nn/2020-04-28T12-19-50Z
```

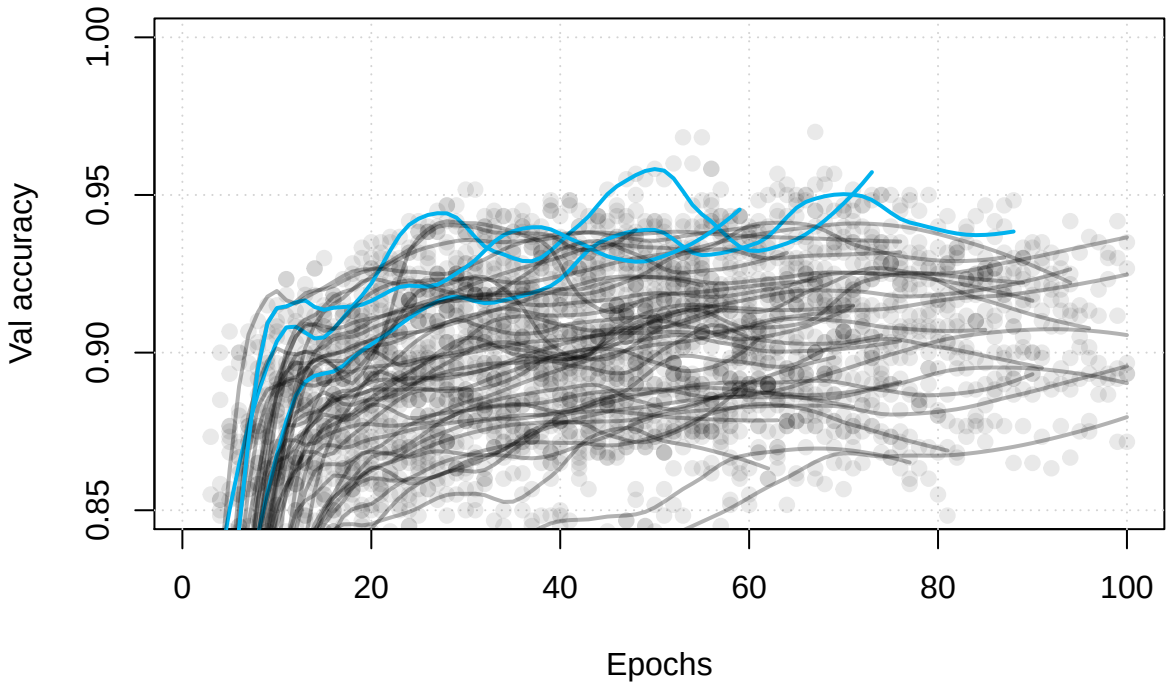
```
##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +     .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-19-50Z

## Training run 21/21 (flags = list(128, 0.1, 64, 0, 64, 0.1))

## Using run directory nn/2020-04-28T12-20-29Z

##
## > library(keras)
##
## > FLAGS <- flags(flag_integer("hdlayer_1", 256), flag_numeric("dropout_1",
## +     0.2), flag_integer("hdlayer_2", 64), flag_numeric("dropout_2",
## +     .... [TRUNCATED]
##
## > model <- keras_model_sequential() %>% layer_dense(units = FLAGS$hdlayer_1,
## +     input_shape = ncol(x1_train), activation = "relu", name = "layer_1 ..." ... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = 64, verbose = 0, callback .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn/2020-04-28T12-20-29Z
```

```
#sampling 21 models
```

```
#Determing the optimal configuration for the data
#Extracting values from the stored runs
```

```
read_metrics <- function(path, files =NULL)
{
path <- paste0(path, "/")
if(is.null(files)) files <- list.files(path)
n <- length(files)
out <- vector("list", n)
for(i in 1:n) {
dir <- paste0(path, files[i], "/tfruns.d/")
out[[i]] <- jsonlite::fromJSON(paste0(dir, "metrics.json"))
out[[i]]$flags <- jsonlite::fromJSON(paste0(dir, "flags.json"))
out[[i]]$evaluation <- jsonlite::fromJSON(paste0(dir,"evaluation.json"))
}
return(out)
}
#Plotting the corresponding validation learning curves
plot_learning_curve <- function(x, ylab = NULL, cols = NULL, top = 3,
span = 0.4, ...)
{
smooth_line <- function(y) {
x <- 1:length(y)
out <- predict(loess(y~x, span = span))
return(out)
}
matplot(x, ylab = ylab, xlab = "Epochs", type = "n", ...)
grid()
matplot(x, pch = 19, col = adjustcolor(cols, 0.3), add = TRUE)
tmp <- apply(x, 2, smooth_line)
tmp <- sapply(tmp, "length<-", max(lengths(tmp)))
set <- order(apply(tmp, 2, max, na.rm = TRUE), decreasing = TRUE)[1:top]
cl <- rep(cols, ncol(tmp))
cl[set] <- "deepskyblue2"
matlines(tmp, lty = 1, col = cl, lwd = 2)
}
```

```
# extract results
out <- read_metrics("nn")
# extract validation accuracy and plot learning curve
acc <- sapply(out, "[[", "val_accuracy")
plot_learning_curve(acc, col = adjustcolor("black", 0.3), ylim = c(0.85, 1),ylab = "Val accuracy", top =
```




```
res1<- ls_runs(metric_val_accuracy > 0.8, runs_dir = "nn", order = metric_val_accuracy)
res1
```

```
## Data frame: 42 x 30
##                    run_dir metric_val_accuracy eval_loss eval_accuracy
## 1  nn/2020-04-28T09-01-45Z              0.9500    0.2112        0.9566
## 2  nn/2020-04-28T09-03-43Z              0.9483    0.2681        0.9480
## 3  nn/2020-04-28T12-20-29Z              0.9450    0.3519        0.9362
## 4  nn/2020-04-28T12-17-54Z              0.9433    0.3944        0.9447
## 5  nn/2020-04-28T09-03-05Z              0.9383    0.3586        0.9388
## 6  nn/2020-04-28T12-16-58Z              0.9350    0.5153        0.9092
## 7  nn/2020-04-28T12-13-20Z              0.9350    0.3743        0.9447
## 8  nn/2020-04-28T12-10-20Z              0.9300    0.3194        0.9276
## 9  nn/2020-04-28T12-09-54Z              0.9283    0.3819        0.9296
## 10 nn/2020-04-28T12-18-36Z              0.9267    0.4442        0.9237
##    metric_loss metric_accuracy metric_val_loss
## 1       0.0787          0.9977          0.2378
## 2       0.1171          0.9987          0.2840
## 3       0.1673          0.9923          0.3488
## 4       0.2484          0.9809          0.3871
## 5       0.1844          0.9884          0.3809
## 6       0.3509          0.9444          0.4131
## 7       0.2567          0.9756          0.3664
## 8       0.0926          0.9953          0.3363
## 9       0.1453          0.9970          0.4122
## 10      0.2276          0.9801          0.4434
```

```
## # ... with 32 more rows
## # ... with 23 more columns:
## #   flag_hdlayer_1, flag_dropout_1, flag_hdlayer_2, flag_dropout_2,
## #   flag_hdlayer_3, flag_dropout_3, samples, batch_size, epochs,
## #   epochs_completed, metrics, model, loss_function, optimizer,
## #   learning_rate, script, start, end, completed, output, source_code,
## #   context, type
```

```
res1 <- res1[,c(2,4,8:13)]
res1[1:10,]
```

```
## Data frame: 10 x 8
##    metric_val_accuracy eval_accuracy flag_hdlayer_1 flag_dropout_1
## 1               0.9500        0.9566            128            0.0
## 2               0.9483        0.9480            128            0.0
## 3               0.9450        0.9362            128            0.1
## 4               0.9433        0.9447            128            0.3
## 5               0.9383        0.9388            128            0.1
## 6               0.9350        0.9092             64            0.1
## 7               0.9350        0.9447            128            0.3
## 8               0.9300        0.9276             32            0.0
## 9               0.9283        0.9296             64            0.0
## 10              0.9267        0.9237            128            0.1
##    flag_hdlayer_2 flag_dropout_2 flag_hdlayer_3 flag_dropout_3
## 1              64            0.0             32            0.0
## 2              64            0.0             16            0.0
## 3              64            0.0             64            0.1
## 4              64            0.0             32            0.1
## 5              64            0.2             32            0.0
## 6              16            0.2             16            0.1
## 7              64            0.0             16            0.1
## 8              64            0.0             32            0.1
## 9              64            0.0             16            0.0
## 10             16            0.0             16            0.1
```

```
# run ------------------------------------------------------------------
dropout_set <- c(0, 0.3, 0.4, 0.5)
lambda_set <- c(0, exp( seq(-6, -4, length = 9) ))
lr_set <- c(0.001, 0.002, 0.005, 0.01)
bs_set <- c(0.005, 0.01, 0.02, 0.03)*nrow(x1_train)

runs <- tuning_run("mlai_2.R",
                   runs_dir = "nn2",
                   flags = list(
                     dropout = dropout_set,
                     lambda = lambda_set,
                     lr = lr_set,
                     bs = bs_set
                   ),
                   sample = 0.02)
```

```
## 640 total combinations of flags (sampled to 12 combinations)
```

```
## Training run 1/12 (flags = list(0.4, 0.00247875217666636, 0.01, 140))

## Using run directory nn2/2020-04-28T12-20-58Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-20-58Z

## Training run 2/12 (flags = list(0.3, 0.0111089965382423, 0.005, 35))

## Using run directory nn2/2020-04-28T12-21-21Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-21-21Z

## Training run 3/12 (flags = list(0.5, 0.00408677143846407, 0.002, 35))

## Using run directory nn2/2020-04-28T12-22-05Z
```

```
##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn2/2020-04-28T12-22-05Z

## Training run 4/12 (flags = list(0.5, 0.00865169520312063, 0.002, 70))

## Using run directory nn2/2020-04-28T12-23-05Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'


##
## Run completed: nn2/2020-04-28T12-23-05Z

## Training run 5/12 (flags = list(0.3, 0.0183156388887342, 0.002, 140))

## Using run directory nn2/2020-04-28T12-23-41Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
```

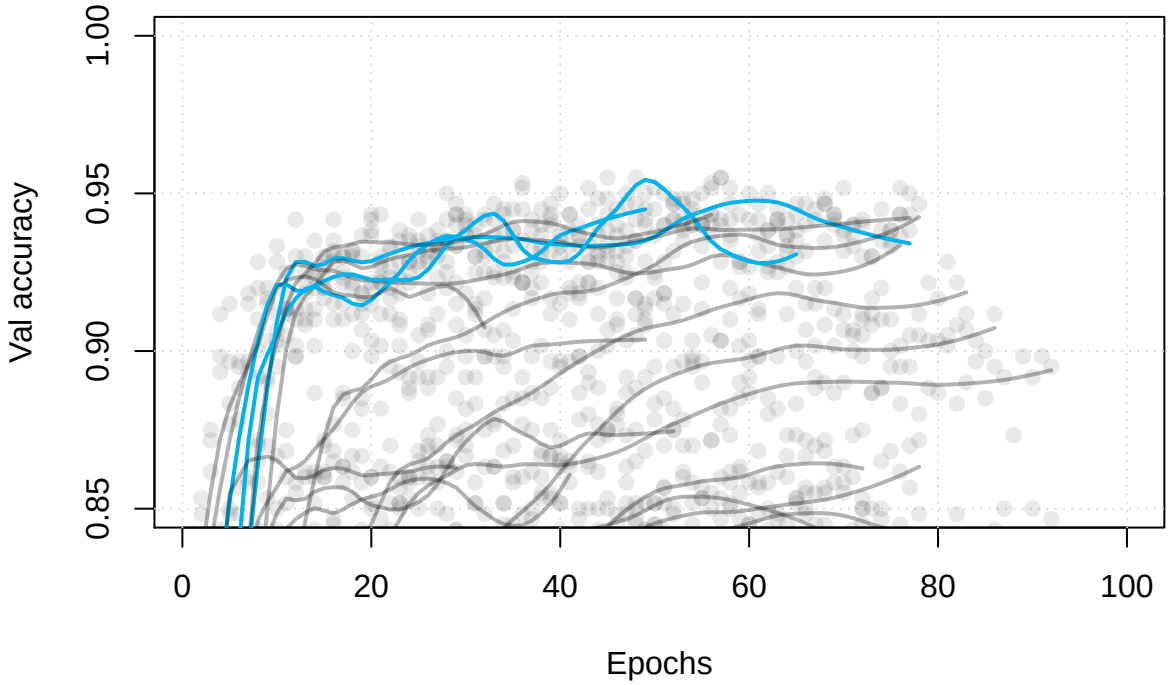
```
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-23-41Z

## Training run 6/12 (flags = list(0.3, 0.00408677143846407, 0.005, 35))

## Using run directory nn2/2020-04-28T12-23-58Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-23-58Z

## Training run 7/12 (flags = list(0.5, 0.0142642339089993, 0.005, 35))

## Using run directory nn2/2020-04-28T12-24-35Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)
```

```
## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-24-35Z

## Training run 8/12 (flags = list(0.4, 0.00408677143846407, 0.002, 210))

## Using run directory nn2/2020-04-28T12-25-23Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-25-23Z

## Training run 9/12 (flags = list(0.5, 0.00247875217666636, 0.001, 70))

## Using run directory nn2/2020-04-28T12-25-43Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-25-43Z
```

```
## Training run 10/12 (flags = list(0.5, 0.0142642339089993, 0.005, 70))

## Using run directory nn2/2020-04-28T12-26-23Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-26-23Z

## Training run 11/12 (flags = list(0.4, 0.00865169520312063, 0.001, 210))

## Using run directory nn2/2020-04-28T12-26-51Z

##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-26-51Z

## Training run 12/12 (flags = list(0.3, 0.00247875217666636, 0.002, 140))

## Using run directory nn2/2020-04-28T12-27-12Z
```

```
##
## > FLAGS <- flags(flag_numeric("dropout", 0.4), flag_numeric("lambda",
## +     0.01), flag_numeric("lr", 0.01), flag_numeric("bs", 100))
##
## > model <- keras_model_sequential() %>% layer_dense(units = 128,
## +     input_shape = V, activation = "relu", name = "layer_1", kernel_regularizer = r .... [TRUNCATED]
##
## > fit <- model %>% fit(x = x1_train, y = yc_train, validation_data = list(x1_val,
## +     yc_val), epochs = 100, batch_size = FLAGS$bs, verbose = 0,
## + .... [TRUNCATED]
##
## > score <- model %>% evaluate(x1_test, yc_test, verbose = 0)

## Warning in value[[3L]](cond): Error occurred resetting tf graph:
## AttributeError: module 'tensorflow' has no attribute 'reset_default_graph'

##
## Run completed: nn2/2020-04-28T12-27-12Z
```

```
# extract results
out <- read_metrics("nn2")
# extract validation accuracy and plot learning curve
acc <- sapply(out, "[[", "val_accuracy")
plot_learning_curve(acc, col = adjustcolor("black", 0.3), ylim = c(0.85, 1),ylab = "Val accuracy", top =
```



```
res2<- ls_runs(metric_val_accuracy > 0.8, runs_dir = "nn2", order = metric_val_accuracy)
res2
```

```
## Data frame: 20 x 28
##                     run_dir metric_val_accuracy eval_loss eval_accuracy
## 1  nn2/2020-04-28T10-44-20Z              0.9500    0.4423        0.9467
## 2  nn2/2020-04-28T12-25-23Z              0.9467    0.6642        0.9211
## 3  nn2/2020-04-28T10-48-08Z              0.9467    0.3771        0.9395
## 4  nn2/2020-04-28T10-47-28Z              0.9450    0.7194        0.9112
## 5  nn2/2020-04-28T12-27-12Z              0.9433    0.5596        0.9329
## 6  nn2/2020-04-28T10-45-43Z              0.9433    0.4117        0.9368
## 7  nn2/2020-04-28T10-44-39Z              0.9383    0.6286        0.9322
## 8  nn2/2020-04-28T12-26-51Z              0.9117    0.7487        0.9046
## 9  nn2/2020-04-28T12-25-43Z              0.9117    0.6977        0.8934
## 10 nn2/2020-04-28T12-23-41Z              0.9083    1.0036        0.8980
##    metric_loss metric_accuracy metric_val_loss
## 1       0.3020          0.9889          0.4346
## 2       0.7645          0.8974          0.6423
## 3       0.2359          0.9943          0.3668
## 4       0.6889          0.9326          0.6639
## 5       0.5571          0.9383          0.5441
## 6       0.2766          0.9927          0.4302
## 7       0.6164          0.9404          0.6079
## 8       0.8320          0.8814          0.7140
## 9       0.8503          0.8397          0.6558
## 10      1.1218          0.8631          1.0004
## # ... with 10 more rows
## # ... with 21 more columns:
## #   flag_dropout, flag_lambda, flag_lr, flag_bs, samples, batch_size,
## #   epochs, epochs_completed, metrics, model, loss_function,
## #   optimizer, learning_rate, script, start, end, completed, output,
## #   source_code, context, type
```

```
res2 <- res2[,c(2,4,8:13)]
res2[1:10,]
```

```
## Data frame: 10 x 8
##    metric_val_accuracy eval_accuracy flag_dropout flag_lambda flag_lr
## 1               0.9500        0.9467          0.0      0.0052   0.001
## 2               0.9467        0.9211          0.4      0.0041   0.002
## 3               0.9467        0.9395          0.0      0.0032   0.002
## 4               0.9450        0.9112          0.3      0.0032   0.002
## 5               0.9433        0.9329          0.3      0.0025   0.002
## 6               0.9433        0.9368          0.0      0.0111   0.001
## 7               0.9383        0.9322          0.3      0.0052   0.002
## 8               0.9117        0.9046          0.4      0.0087   0.001
## 9               0.9117        0.8934          0.5      0.0025   0.001
## 10              0.9083        0.8980          0.3      0.0183   0.002
##    flag_bs samples batch_size
## 1      210    7000        210
## 2      210    7000        210
## 3      210    7000        210
## 4       70    7000         70
```

```
## 5       140    7000         140
## 6        35    7000          35
## 7       210    7000         210
## 8       210    7000         210
## 9        70    7000          70
## 10      140    7000         140
```