HW9_Brown_Nick

Nick Brown

11/19/2019

This weeks homework will be to simply get Keras and R working. To demonstrate this, you need to work through two of the examples in the Keras list of examples or tutorials. See here: https://keras.rstudio.com/

Install Keras

```
#Install keras and tensorflow
install.packages("keras")
```

```
## Installing package into 'C:/Users/Niko/Documents/R/win-library/3.6'
## (as 'lib' is unspecified)
```

```
## package 'keras' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Niko\AppData\Local\Temp\RtmpCObBd8\downloaded_packages
```

```
library(keras)
install_keras()
```

```
##
## Installation complete.
```

```
#devtools::install_github("rstudio/reticulate")
#devtools::install_github("rstudio/tensorflow")
#devtools::install_github("rstudio/keras")
#install.packages("reticulate")
library(reticulate)
use_virtualenv("r-tensorflow")
update.packages("reticulate")
reticulate::py_discover_config()
```

```
## python:         C:\Users\Niko\Anaconda3\envs\r-reticulate\python.exe
## libpython:      C:/Users/Niko/Anaconda3/envs/r-reticulate/python36.dll
## pythonhome:     C:\Users\Niko\ANACON~1\envs\R-RETI~1
## version:        3.6.9 |Anaconda, Inc.| (default, Jul 30 2019, 14:00:49) [MSC v.1915 64 bit (AMD64)]
## Architecture:   64bit
## numpy:          C:\Users\Niko\ANACON~1\envs\R-RETI~1\lib\site-packages\numpy
## numpy_version:  1.17.4
##
```

```
## python versions found:
##  C:\Users\Niko\Anaconda3\envs\r-reticulate\python.exe
##  C:\Users\Niko\ANACON~1\envs\R-RETI~1\python.exe
##  C:\Users\Niko\ANACON~1\python.exe
##  C:\Users\Niko\Anaconda3\python.exe
##  C:\Users\Niko\Anaconda3\envs\tensorflow\python.exe
##  C:\Users\Niko\Anaconda3\envs\tensorflow_cpu\python.exe
```

```
reticulate::use_condaenv("r-tensorflow")
reticulate::py_config()
```

```
## python:         C:\Users\Niko\Anaconda3\envs\r-reticulate\python.exe
## libpython:      C:/Users/Niko/Anaconda3/envs/r-reticulate/python36.dll
## pythonhome:     C:\Users\Niko\ANACON~1\envs\R-RETI~1
## version:        3.6.9 |Anaconda, Inc.| (default, Jul 30 2019, 14:00:49) [MSC v.1915 64 bit (AMD64)]
## Architecture:   64bit
## numpy:          C:\Users\Niko\ANACON~1\envs\R-RETI~1\lib\site-packages\numpy
## numpy_version:  1.17.4
## tensorflow:     C:\Users\Niko\ANACON~1\envs\R-RETI~1\lib\site-packages\tensorflow\__init__.p
##
## python versions found:
##  C:\Users\Niko\Anaconda3\envs\r-reticulate\python.exe
##  C:\Users\Niko\Anaconda3\envs\tensorflow\python.exe
##  C:\Users\Niko\ANACON~1\envs\R-RETI~1\python.exe
##  C:\Users\Niko\ANACON~1\python.exe
##  C:\Users\Niko\Anaconda3\python.exe
##  C:\Users\Niko\Anaconda3\envs\tensorflow_cpu\python.exe
```

```
#Install keras and tensorflow
#install.packages("tensorflow")
#library(tensorflow)
#install_tensorflow()
```

Keras Example 1 - Basic Classification (https://keras.rstudio.com/articles/tutorial_basic_classification.html)

```
# access the fashion mnist dataset
#tensorflow::install_tensorflow()
fashion_mnist <- dataset_fashion_mnist()
```

```
# Use keras to create arrays
c(train_images, train_labels) %<-% fashion_mnist$train
c(test_images, test_labels) %<-% fashion_mnist$test
```

```
class_names = c('T-shirt/top',
                'Trouser',
                'Pullover',
                'Dress',
                'Coat',
                'Sandal',
```

```r
                 'Shirt',
                 'Sneaker',
                 'Bag',
                 'Ankle boot')
```

```r
dim(train_images)
```

```
## [1] 60000    28    28
```

```r
dim(train_labels)
```

```
## [1] 60000
```

```r
train_labels[1:20]
```

```
##  [1] 9 0 0 3 0 2 7 2 5 5 0 9 5 5 7 9 1 0 6 4
```

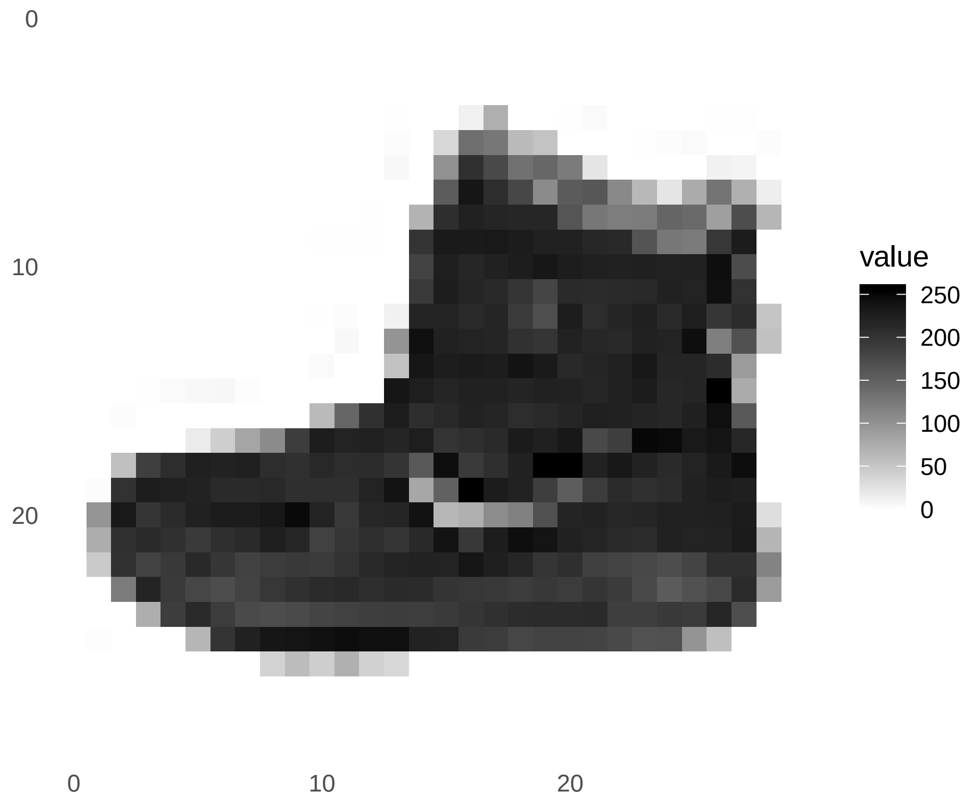
```r
dim(test_images)
```

```
## [1] 10000    28    28
```

```r
dim(test_labels)
```

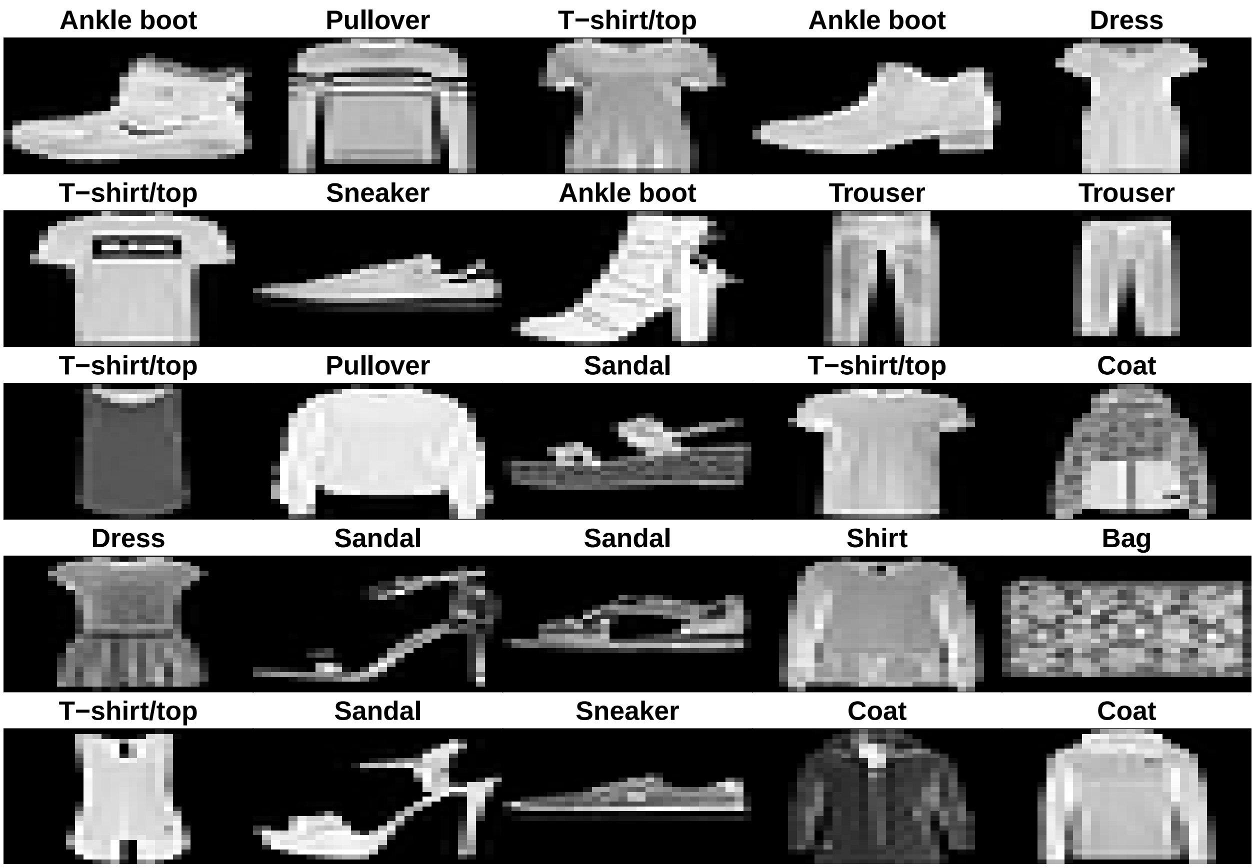
```
## [1] 10000
```

```r
library(tidyr)
library(ggplot2)

image_1 <- as.data.frame(train_images[1, , ])
colnames(image_1) <- seq_len(ncol(image_1))
image_1$y <- seq_len(nrow(image_1))
image_1 <- gather(image_1, "x", "value", -y)
image_1$x <- as.integer(image_1$x)

ggplot(image_1, aes(x = x, y = y, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black", na.value = NA) +
  scale_y_reverse() +
  theme_minimal() +
  theme(panel.grid = element_blank())   +
  theme(aspect.ratio = 1) +
  xlab("") +
  ylab("")
```




```
train_images <- train_images / 255
test_images <- test_images / 255
```

```
par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  img <- train_images[i, , ]
  img <- t(apply(img, 2, rev))
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste(class_names[train_labels[i] + 1]))
}
```




```r
model <- keras_model_sequential()
model %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')
```

```r
model %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)
```

```r
model %>% fit(train_images, train_labels, epochs = 5)
```

```r
score <- model %>% evaluate(test_images, test_labels)

cat('Test loss:', score$loss, "\n")
```

```
## Test loss: 0.3444997
```

```r
cat('Test accuracy:', score$acc, "\n")
```

```
## Test accuracy: 0.875
```

```r
predictions <- model %>% predict(test_images)
```

```r
predictions[1, ]
```

```
##  [1] 4.902042e-05 2.389006e-08 3.646624e-07 9.873634e-09 2.620097e-06
##  [6] 4.642367e-02 3.056894e-06 4.223496e-02 1.718430e-05 9.112691e-01
```

```r
which.max(predictions[1, ])
```

```
## [1] 10
```

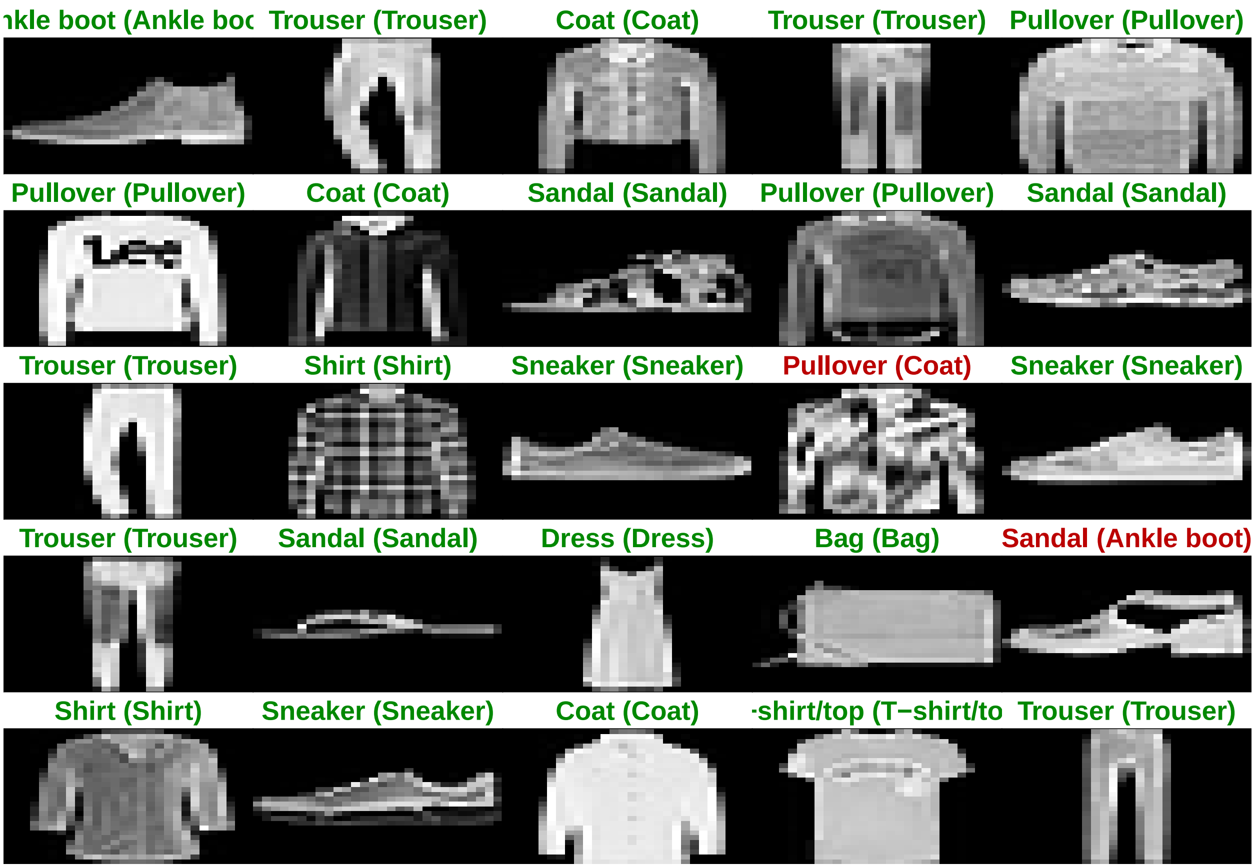
```r
class_pred <- model %>% predict_classes(test_images)
class_pred[1:20]
```

```
##  [1] 9 2 1 1 6 1 4 6 5 7 4 5 7 3 4 1 2 2 8 0
```

```r
test_labels[1]
```

```
## [1] 9
```

```r
par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  img <- test_images[i, , ]
  img <- t(apply(img, 2, rev))
  # subtract 1 as labels go from 0 to 9
  predicted_label <- which.max(predictions[i, ]) - 1
  true_label <- test_labels[i]
  if (predicted_label == true_label) {
    color <- '#008800'
  } else {
    color <- '#bb0000'
  }
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste0(class_names[predicted_label + 1], " (",
                      class_names[true_label + 1], ")"),
        col.main = color)
}
```




```r
# Grab an image from the test dataset
# take care to keep the batch dimension, as this is expected by the model
img <- test_images[1, , , drop = FALSE]
dim(img)
```

```
## [1]  1 28 28
```

```r
predictions <- model %>% predict(img)
predictions
```

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] 4.902042e-05 2.389006e-08 3.646618e-07 9.873634e-09 2.620092e-06
##            [,6]         [,7]       [,8]        [,9]     [,10]
## [1,] 0.04642368 3.056889e-06 0.04223494 1.71843e-05 0.9112691
```

```r
# subtract 1 as labels are 0-based
prediction <- predictions[1, ] - 1
which.max(prediction)
```

```
## [1] 10
```

```r
class_pred <- model %>% predict_classes(img)
class_pred
```

```
## [1] 9
```

Keras Example 2 - Basic Regression (https://keras.rstudio.com/articles/tutorial_basic_regression.html)

```
boston_housing <- dataset_boston_housing()

c(train_data, train_labels) %<-% boston_housing$train
c(test_data, test_labels) %<-% boston_housing$test
```

```
paste0("Training entries: ", length(train_data), ", labels: ", length(train_labels))
```

```
## [1] "Training entries: 5252, labels: 404"
```

```
train_data[1, ] # Display sample features, notice the different scales
```

```
##  [1]   1.23247   0.00000   8.14000   0.00000   0.53800   6.14200  91.70000
##  [8]   3.97690   4.00000 307.00000  21.00000 396.90000  18.72000
```

```
library(tibble)

column_names <- c('CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE',
                  'DIS', 'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT')
train_df <- as_tibble(train_data)
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Usin
## This warning is displayed once per session.
```

```
colnames(train_df) <- column_names

train_df
```

```
## # A tibble: 404 x 13
##      CRIM    ZN INDUS  CHAS   NOX    RM   AGE   DIS   RAD   TAX PTRATIO
##     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
##  1 1.23     0    8.14     0 0.538  6.14  91.7  3.98     4   307    21
##  2 0.0218  82.5  2.03     0 0.415  7.61  15.7  6.27     2   348    14.7
##  3 4.90     0   18.1      0 0.631  4.97 100    1.33    24   666    20.2
##  4 0.0396   0    5.19     0 0.515  6.04  34.5  5.99     5   224    20.2
##  5 3.69     0   18.1      0 0.713  6.38  88.4  2.57    24   666    20.2
##  6 0.284    0    7.38     0 0.493  5.71  74.3  4.72     5   287    19.6
##  7 9.19     0   18.1      0 0.7    5.54 100    1.58    24   666    20.2
##  8 4.10     0   19.6      0 0.871  5.47 100    1.41     5   403    14.7
##  9 2.16     0   19.6      0 0.871  5.63 100    1.52     5   403    14.7
## 10 1.63     0   21.9      0 0.624  5.02 100    1.44     4   437    21.2
## # ... with 394 more rows, and 2 more variables: B <dbl>, LSTAT <dbl>
```

```
train_labels[1:10] # Display first 10 entries
```

```
##  [1] 15.2 42.3 50.0 21.1 17.7 18.5 11.3 15.6 15.6 14.4
```

```r
# Test data is *not* used when calculating the mean and std.

# Normalize training data
train_data <- scale(train_data)

# Use means and standard deviations from training set to normalize test set
col_means_train <- attr(train_data, "scaled:center")
col_stddevs_train <- attr(train_data, "scaled:scale")
test_data <- scale(test_data, center = col_means_train, scale = col_stddevs_train)

train_data[1, ] # First training sample, normalized
```

```
##  [1] -0.2719092 -0.4830166 -0.4352220 -0.2565147 -0.1650220 -0.1762241
##  [7]  0.8120550  0.1165538 -0.6254735 -0.5944330  1.1470781  0.4475222
## [13]  0.8241983
```

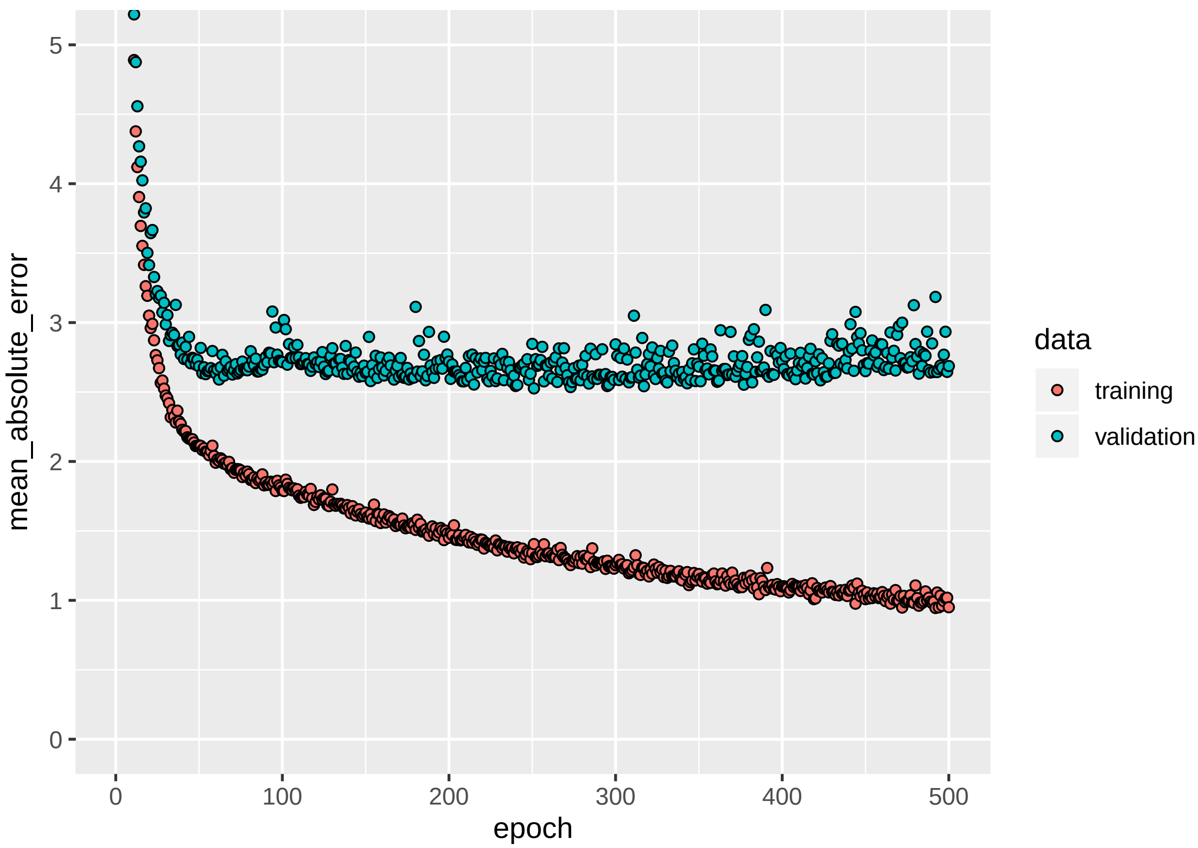
```r
build_model <- function() {

  model <- keras_model_sequential() %>%
    layer_dense(units = 64, activation = "relu",
                input_shape = dim(train_data)[2]) %>%
    layer_dense(units = 64, activation = "relu") %>%
    layer_dense(units = 1)

  model %>% compile(
    loss = "mse",
    optimizer = optimizer_rmsprop(),
    metrics = list("mean_absolute_error")
  )

  model
}

model <- build_model()
model %>% summary()
```

```
## Model: "sequential_1"
## ___________________________________________________________________________
## Layer (type)                     Output Shape                  Param #
## ===========================================================================
## dense_2 (Dense)                  (None, 64)                    896
## ___________________________________________________________________________
## dense_3 (Dense)                  (None, 64)                    4160
## ___________________________________________________________________________
## dense_4 (Dense)                  (None, 1)                     65
## ===========================================================================
## Total params: 5,121
## Trainable params: 5,121
## Non-trainable params: 0
## ___________________________________________________________________________
```

```r
# Display training progress by printing a single dot for each completed epoch.
print_dot_callback <- callback_lambda(
  on_epoch_end = function(epoch, logs) {
    if (epoch %% 80 == 0) cat("\n")
    cat(".")
  }
)

epochs <- 500

# Fit the model and store training stats
history <- model %>% fit(
  train_data,
  train_labels,
  epochs = epochs,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(print_dot_callback)
)
```

```
##
## ................................................................................
## ................................................................................
## ................................................................................
## ................................................................................
## ................................................................................
## ................................................................................
## ....................
```

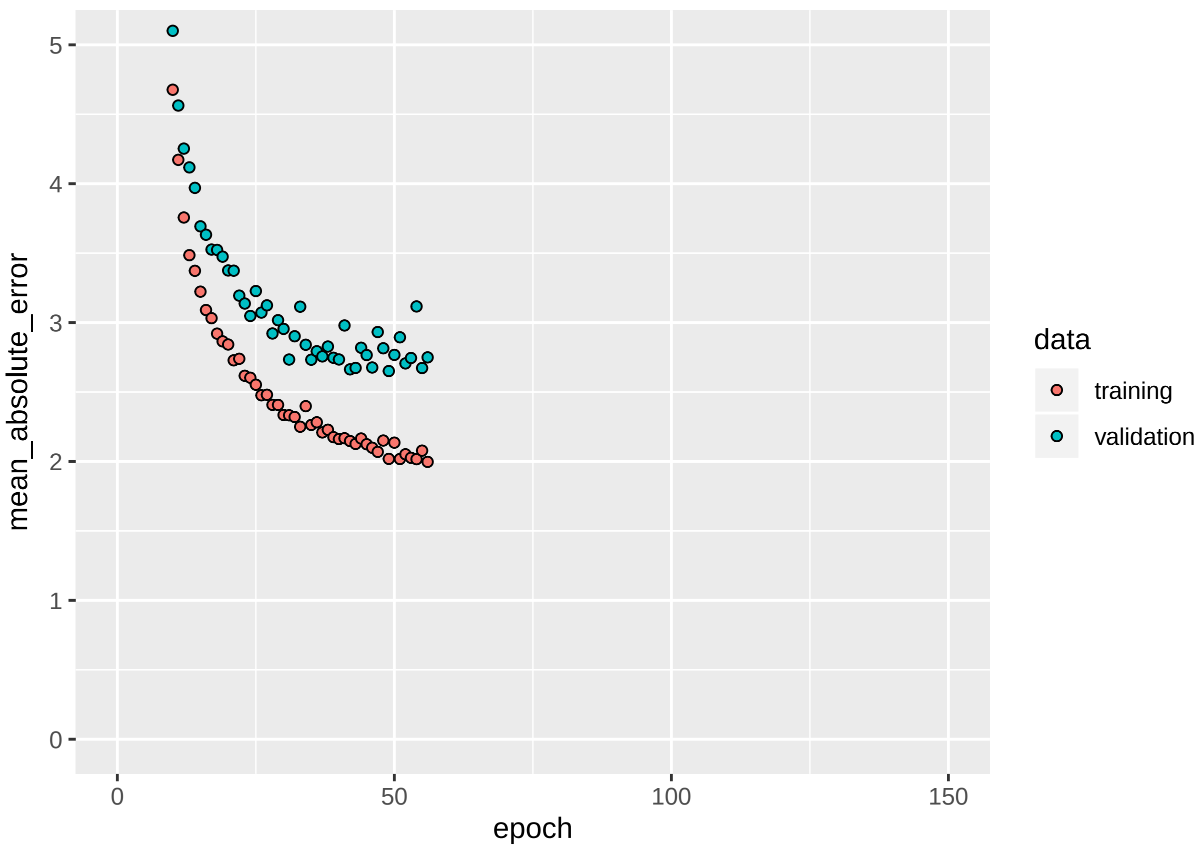
```r
library(ggplot2)

plot(history, metrics = "mean_absolute_error", smooth = FALSE) +
  coord_cartesian(ylim = c(0, 5))
```




```r
# The patience parameter is the amount of epochs to check for improvement.
early_stop <- callback_early_stopping(monitor = "val_loss", patience = 20)

model <- build_model()
history <- model %>% fit(
  train_data,
  train_labels,
  epochs = epochs,
  validation_split = 0.2,
  verbose = 0,
  callbacks = list(early_stop, print_dot_callback)
)
```

```
##
## ......................................................
```

```r
plot(history, metrics = "mean_absolute_error", smooth = FALSE) +
  coord_cartesian(xlim = c(0, 150), ylim = c(0, 5))
```




```
c(loss, mae) %<-% (model %>% evaluate(test_data, test_labels, verbose = 0))

paste0("Mean absolute error on test set: $", sprintf("%.2f", mae * 1000))
```

```
## [1] "Mean absolute error on test set: $3634.99"
```

```
test_predictions <- model %>% predict(test_data)
test_predictions[ , 1]
```

```
##   [1]  7.360172 17.727674 19.286283 31.492924 23.842319 19.129042 24.194000
##   [8] 20.140392 19.338114 22.041826 19.255390 17.030506 15.202568 38.707954
##  [15] 19.574522 17.384716 25.669197 20.266691 19.173162 37.192142 12.517274
##  [22] 15.525432 19.385483 14.118697 18.461596 24.786554 29.269190 25.014606
##  [29]  9.915028 19.493944 18.893757 15.001622 31.601305 23.904320 17.762680
##  [36]  8.609745 14.798226 18.181345 20.267059 22.934471 28.077101 26.308847
##  [43] 14.921614 38.271278 28.014555 23.185083 24.106813 15.198074 24.149023
##  [50] 20.572603 30.672798 17.020309 13.060958 15.674891 31.803551 25.862988
##  [57] 12.973571 44.519932 32.632397 21.959558 25.515806 17.642660 14.956965
##  [64] 17.580122 21.501554 20.046965 13.924839 20.115557 15.179756  7.241501
##  [71] 36.946842 27.025610 24.887987 14.793612 23.613636 15.887851 18.729872
##  [78] 22.028364 32.291267 11.311610 19.065870 35.943829 14.617715 14.709800
##  [85] 16.137909 15.546162 20.425226 20.440287 21.723217 31.225672 18.115891
##  [92] 16.806141 23.340033 38.593342 32.725361 19.377146 34.183556 54.535675
##  [99] 26.035700 46.069324 31.059116 20.249475
```