Deep Learning

Indrajith Wasala Mudiyanselage

Section 01

Question 01

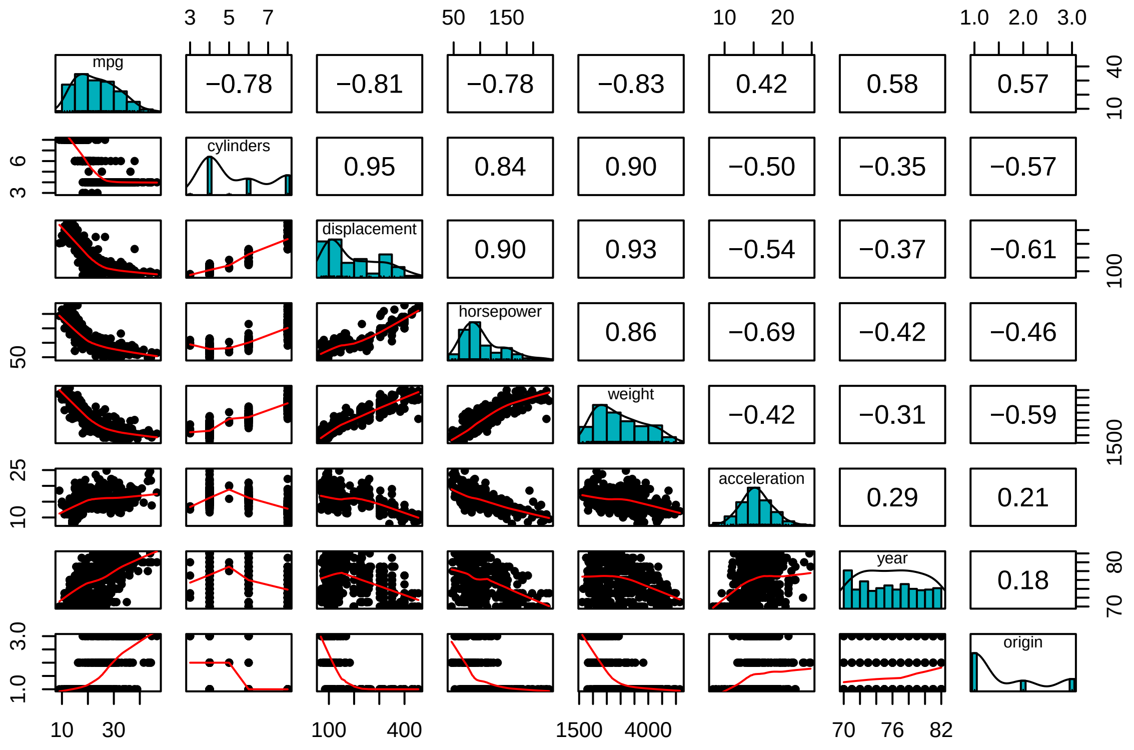

Figure 1: Pairwise scatterplots

a) The pairwise scatter plot visually reveals that the suggested response variable (mpg) has a strong negative linear relationship with the variables cylinders, displacement, horsepower and weight and moderately positive linear relationship between acceleration, year and origin. Also, note that predictor variables cylinders, displacement, horsepower and weight have a strong positive linear relationship between each other while most of the other predictors have weak negative linear relationships. The correlation matrix confirms the above information.The Histograms visually suggest that the response variable mpg have a right skewed distribution. Also visually, all the predictors does not have symmetric distributions except acceleration.

b) MSE via LOOCV = 11.37113. Regression coefficients,

```
##                Estimate Std. Error    t value     Pr(>|t|)
## (Intercept)  23.4459184  0.1680733 139.498139 0.000000e+00
## cylinders    -0.8415931  0.5514496  -1.526147 1.277965e-01
## displacement  2.0819598  0.7864080   2.647430 8.444649e-03
## horsepower   -0.6524692  0.5306734  -1.229512 2.196328e-01
## weight       -5.4990690  0.5538510  -9.928787 7.874953e-21
## acceleration  0.2222978  0.2726998   0.815174 4.154780e-01
## year          2.7656487  0.1877716  14.728795 3.055983e-39
## origin        1.1487821  0.2240437   5.127492 4.665681e-07
```

c) MSE via LOOCV = 11.37113.

d) Shallow learning model MSE via LOOCV 488.5207.

	Coefficients
(Intercept)	0.208653
cylinders	-0.49084339
displacement	0.39686212
horsepower	0.15018430
weight	-1.02059686
acceleration	0.60515058
year	-0.05495063
origin	-0.22021827

Table 1: Regression coefficints by shallow learning model

e) The estimates reported in above parts are not identical. If we run more epochs we may get more closer estimates (The nural network needs more training).

Question 02

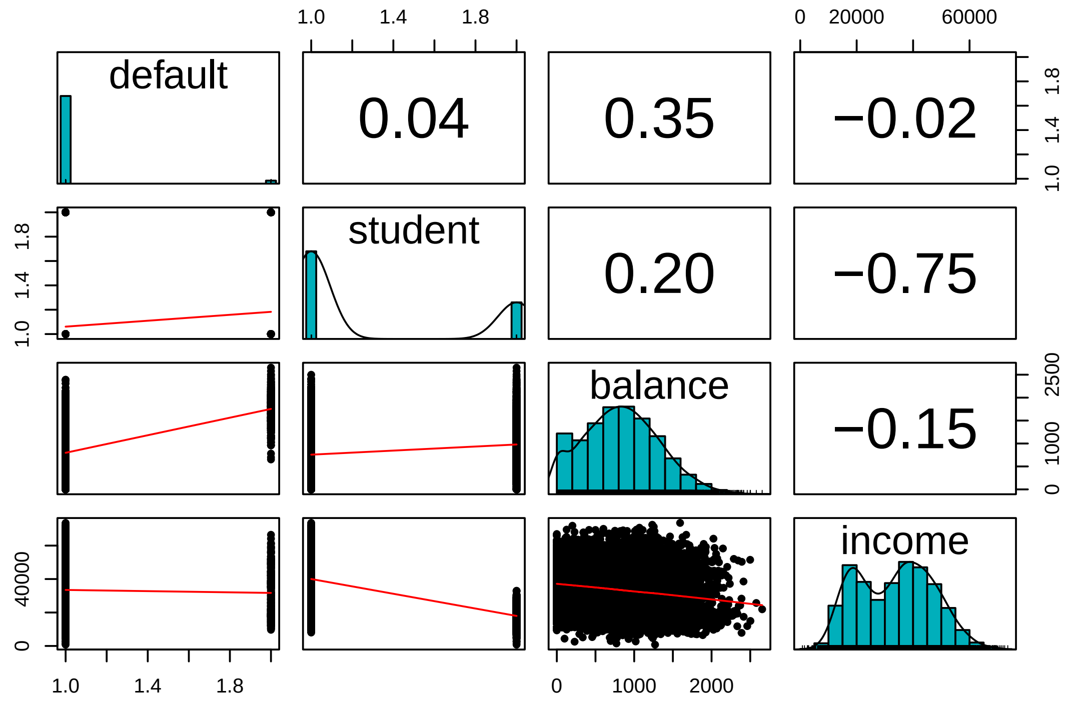

Figure 2: Pairwise scatterplots

a) The pairwise scatter plot visually reveals that the suggested response variable (default) has a weak positive linear relationship with the variables student and balance while income has weak negative linear relationship. The correlation matrix confirms the above information.The Histograms visually suggest that non of the variables have symetric distribution.

b) 5-fold CV estimate of test error is 0.02680055. Logistic regression coefficients (Note that this is without standardizing training data),

```
##                  Estimate   Std. Error    z value      Pr(>|z|)
## (Intercept) -1.086905e+01 4.922555e-01 -22.080088 4.911280e-108
## studentYes  -6.467758e-01 2.362525e-01  -2.737646  6.188063e-03
## balance      5.736505e-03 2.318945e-04  24.737563 4.219578e-135
## income       3.033450e-06 8.202615e-06   0.369815  7.115203e-01
```

c) 5-fold CV estimate of test error is 0.0268.

d) 5-fold CV test error of the shallow learning model is 0.03545891.

	Coefficients
(Intercept)	-2.537609
studentYes	-5.6365714073
balance	0.0167175867
income	-0.0008408058

Table 2: Regression coefficints by shallow learning model

e) The estimates reported in above parts are not identical. If we run more epochs we may get more closer estimates (The shallow learning model needs more training).

Question 03

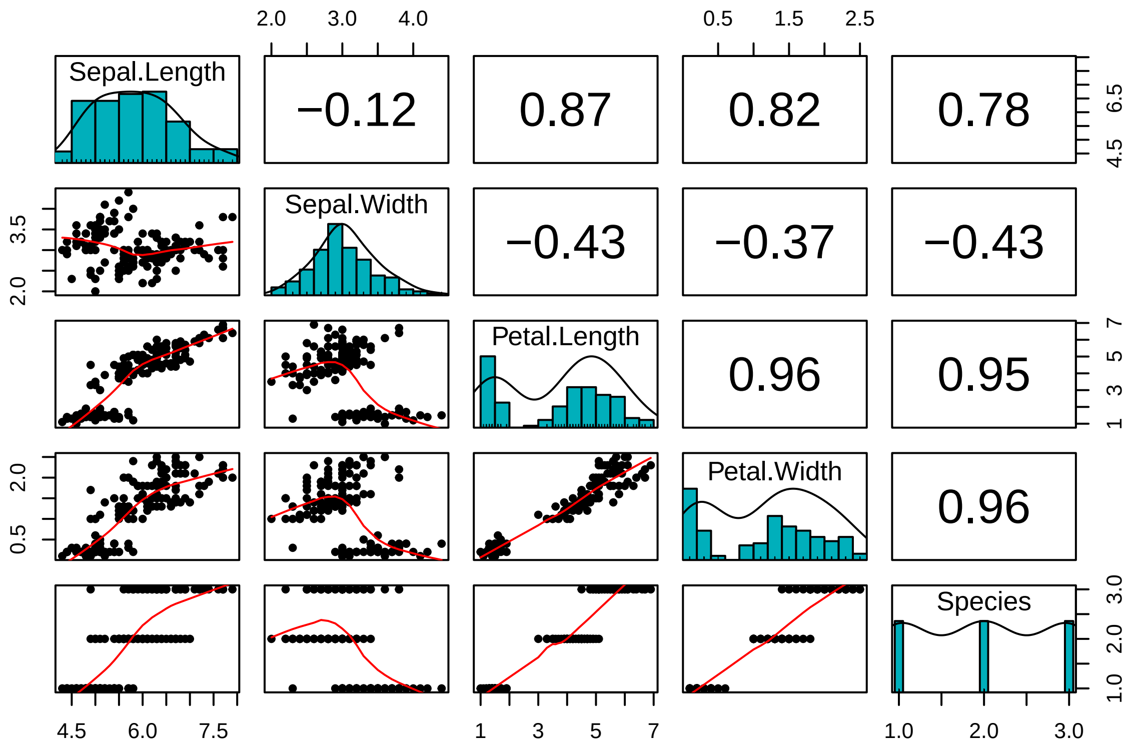

Figure 3: Pairwise scatterplots

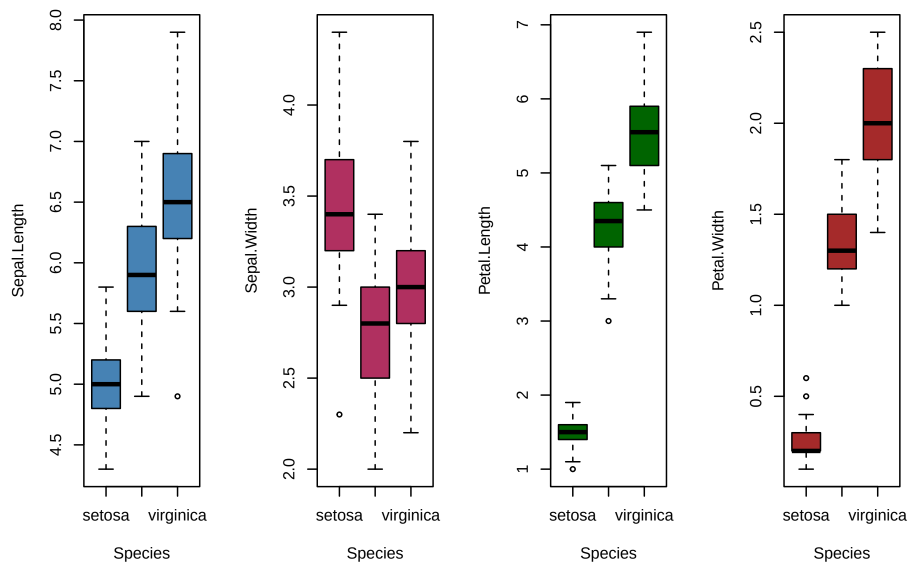

Figure 4: Boxplots of responce variable with predictor variables

a) The pairwise scatter plot visually reveals that the suggested response variable (Species) has a strong positive linear relationship with all the variables except Sepal.Width. The correlation matrix confirms the above information.The Histograms visually suggest that non of the variables have symmetric distribution except Sepal.Width. Moreover the side by side boxplots reveal the distribution of the predictors according to the levels of the response variable.

b) 5-fold CV estimate of test error is 0.02666667. The regression coefficients,

```
##            (Intercept) Sepal.Length Sepal.Width Petal.Length Petal.Width
## versicolor    9.023056    -3.345346   -3.039437     13.69517    8.090794
## virginica   -11.021851    -5.380110   -5.945487     30.29748   21.980253
```

c) 5-fold CV test error of the shallow learning model is $1 - 0.6933334 = 0.3066666$.

		Coefficients	
(Intercept)	-0.267862678	0.199472770	-0.007312237
Sepal.Length	-0.6619897	0.3302517	0.6933144
Sepal.Width	-0.5235697	0.1453871	-0.1324961
Petal.Length	-0.7744051	1.1369798	1.2270440
Petal.Width	-0.2853383	-0.6473362	-0.2898701

Table 3: Regression coefficints by shallow learning model

d) The estimates reported in above parts are not identical. If we run more epochs we may get more closer estimates (The shallow learning model needs more training).

Section 2 (R codes)

```r
knitr::opts_chunk$set(echo = TRUE}

## ----echo=FALSE,warning=FALSE,fig.align="center",fig.cap="Pairwise scatterplots",out.width = "100%"----

### Question 01

## a)

library(ISLR)
library(psych)
pairs.panels(Auto[,-9],
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = FALSE # show correlation ellipses
             )


## ----echo=FALSE-------------------------------------------------------------

## b)

mean <- apply(Auto[,-c(1,9)], 2, mean)
std <- apply(Auto[,-c(1,9)], 2, sd)

# Standardize the training and test features
Auto_data <- as.data.frame(scale(Auto[,-c(1,9)], center = mean, scale = std))
Auto_data1<-cbind(Auto[,1],Auto_data)
colnames(Auto_data1)[1]<-c("mpg")
reg<-glm(mpg~., data=Auto_data1) # Fitting regression
summary(reg)$coefficient

## ----include=FALSE----------------------------------------------------------
library(boot)
cv.est <- cv.glm(Auto_data1, reg)$delta[1]


## ----include=FALSE----------------------------------------------------------

## c)

LOOCV<-function(dataset){
  n<-length(dataset[,1])
  RSS_i<-c()
  for (i in 1:n) {
    newdata<-dataset[-i,] # Data without ith observation (Training data)
    testdata<-dataset[i,] # ith observation
    fit <- glm(mpg ~ .,  data = newdata) # Fit linear regression for Training data
    lr.fit <- predict(fit, testdata, type = "response")
    RSS_i[i] <- (testdata$mpg-lr.fit)^2 # Get RSS for test
  }
MSE<- mean(RSS_i)
return(list(LOOCV_MSE=MSE))
}
LOOCV(dataset=Auto_data1)


## ---- include=FALSE---------------------------------------------------------

## d)
```

```r
mean <- apply(Auto[,-c(1,9)], 2, mean)
std <- apply(Auto[,-c(1,9)], 2, sd)
# Standardize the training and test features
Auto_data <- as.data.frame(scale(Auto[,-c(1,9)], center = mean, scale = std))
Auto_data1<-cbind(Auto[,1],Auto_data)
colnames(Auto_data1)[1]<-c("mpg")

train_data<-as.matrix(Auto_data1[,-1])
train_targets<-as.matrix(Auto_data1[,1],ncol=1)

# Specify and Pre compile the model
build_model <- function(){
  # specify the model
model <- keras_model_sequential() %>%
layer_dense(units = 1, input_shape = dim(train_data)[2],activation = 'linear')
  #compile the model
 model %>% compile(
   optimizer = "rmsprop",
  loss = "mse")
}

MSE<-c()

for (i in 1:nrow(train_data)) {
  val_data <- matrix(train_data[i,],nrow=1)
  val_targets <- train_targets[i]
  partial_train_data <- train_data[-i,]
  partial_train_targets <- train_targets[-i]


  model <- build_model() # use precompiled model function

  model %>% fit(partial_train_data, partial_train_targets,
                epochs = 50,batch_size = 128, verbose = 0) # trains the model in silent mode (verbose=0)
  pred_targets <-predict(model, val_data)
  MSE[i] <- (val_targets - pred_targets)^2
  #weights <- model %>% get_layer(index = 1) %>% get_weights()
}

weights <- model %>% get_layer(index = 1) %>% get_weights()


################################################################################
## ----echo=FALSE,warning=FALSE,fig.align="center",fig.cap="Pairwise scatterplots",out.width = "80%"----

### Question 02

## a)

library(ISLR)
library(psych)
pairs.panels(Default,
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = FALSE # show correlation ellipses
             )


## ----echo=FALSE-------------------------------------------------------------------

## b)
```

```r
logi_reg <- glm(default ~ ., data = Default, family = "binomial")
summary(logi_reg)$coefficient

## ----include=FALSE--------------------------------------------------------------
# Using caret package

library(caret)

glm.fit2 <- train(
    form = default ~ .,
    data = Default,
    trControl = trainControl(method = "cv", number = 5,
        seeds = set.seed(1)),
    method = "glm",
    family = "binomial")

Miss_clas<-1-glm.fit2$results[[2]] # Misclassification rate = 1 - Accuracy,


## ----include=FALSE--------------------------------------------------------------

## c)

MY_CV<-function(dataset,fold=5){

  k<-fold
  set.seed(1)
  indices <- sample(1:nrow(dataset),replace = FALSE)
  folds <- cut(indices, breaks = k, labels = FALSE)
  pred<-c()
  lr.pred.fit<-c()

  for (i in 1:k) {
    val_indices <- which(folds == i, arr.ind = TRUE) # prepares the validation data: data from partition #k
    val_data <- Default[val_indices,]
    partial_train_data <- Default[-val_indices,]
    fit <- glm(default ~ ., family = binomial, data = partial_train_data)
    lr.prob.fit <- predict(fit, val_data, type = "response")
    pred <- ifelse(lr.prob.fit >= 0.5, "Yes", "No")
    lr.pred.fit[i]<-mean(pred==val_data[,"default"])
  }

MSE<- 1 - mean(lr.pred.fit)

return(list(MSE=MSE))
}
Error<-MY_CV(dataset=Default)



## ---- include=FALSE-------------------------------------------------------------

## d)

train_data<-Default[,-1] # Without default
train_data[,1]<-as.numeric(unclass(train_data[,1])-1) # Convert student (No=0,Yes=1)
train_data<-as.matrix(train_data)
train_targets<-as.matrix((unclass(Default[,1])-1),ncol=1) # Converted default (No=0,Yes=1)


build_model <- function(){
  # specify the model
 model <- keras_model_sequential() %>%
 layer_dense(units = 1, input_shape = dim(train_data)[2],activation = 'sigmoid')
 # compile the model
 model %>% compile(
   optimizer = "rmsprop",
   loss = "binary_crossentropy",
   metrics = c("mae") # mean absolute error
 )
}

# K-fold CV
k <- 5
indices <- sample(1:nrow(train_data))
folds <- cut(indices, breaks = k, labels = FALSE)
num_epochs <- 50 # number of epochs
all_scores <- c()
for (i in 1:k){
  cat("Processing fold #", i, "\n")

  val_indices <- which(folds == i, arr.ind = TRUE) # prepares the validation data: data from partition #k
  val_data <- train_data[val_indices,]
  val_targets <- train_targets[val_indices]

  partial_train_data <- train_data[-val_indices,] # prepares the training data: data from all other partitions
  partial_train_targets <- train_targets[-val_indices]

  model <- build_model() # use precompiled model function

  model %>% fit(partial_train_data, partial_train_targets,
                epochs = num_epochs, batch_size = 16,
                verbose = 0) # trains the model in silent mode (verbose = 0)

  results <- model %>% evaluate(val_data, val_targets, verbose = 0) # evaluate model on the validation data
  all_scores <- c(all_scores, results["mae"])
}
all_scores
mean(all_scores)


## ---- include=FALSE--------------------------------------------------------------
weights2 <- model %>% get_layer(index = 1) %>% get_weights()


################################################################################

## ----echo=FALSE,warning=FALSE,fig.align="center",fig.cap="Pairwise scatterplots",out.width = "80%"----

### Question 03

## a)

pairs.panels(iris,
             method = "pearson", # correlation method
             hist.col = "#00AFBB",
             density = TRUE,  # show density plots
             ellipses = FALSE # show correlation ellipses
             )


## ---- echo=FALSE,warning=FALSE,fig.align="center",fig.cap="Boxplots of responce variable with predictor variabl
par(mfrow = c(1, 4))
boxplot(iris$Sepal.Length ~ iris$Species,
        col='steelblue',
        xlab='Species',
```

```r
        ylab='Sepal.Length')
boxplot(iris$Sepal.Width ~ iris$Species,
        col='maroon',
        xlab='Species',
        ylab='Sepal.Width')
boxplot(iris$Petal.Length ~ iris$Species,
        col='darkgreen',
        xlab='Species',
        ylab='Petal.Length')
boxplot(iris$Petal.Width ~ iris$Species,
        col='brown',
        xlab='Species',
        ylab='Petal.Width')


## ----include=FALSE----------------------------------------------------------------

## b)

# Standardize the training and test features
mean <- apply(iris[,-c(5)], 2, mean)
std <- apply(iris[,-c(5)], 2, sd)

iris_data <- as.data.frame(scale(iris[,-c(5)], center = mean, scale = std))
iris_data1<-cbind(iris_data,iris[,5])
colnames(iris_data1)[5]<-c("Species")


## ----include=FALSE----------------------------------------------------------------
library(nnet)
mul_reg <- multinom(Species ~ ., data = iris_data1)


## ----echo=FALSE-------------------------------------------------------------------
summary(mul_reg)$coefficien


## ----include=FALSE----------------------------------------------------------------
multi.fit <- train(
    form = Species ~ .,
    data = iris_data1,
    trControl = trainControl(method = "cv", number = 5,
        seeds = set.seed(1)),
    method = "multinom")

Miss_clas<-1-multi.fit$results[[2]] # Misclassification rate = 1 - Accuracy,
Miss_clas


## ---- include=FALSE---------------------------------------------------------------

## c)

train_data<-iris_data1[,-5]
train_data<-as.matrix(train_data)
train_targets<-as.matrix((unclass(iris[,5])-1),ncol=1)

train_labels <- to_categorical(train_targets)

build_model <- function(){
  # specify the model
 model <- keras_model_sequential() %>%
 layer_dense(units = 3, input_shape = dim(train_data)[2],activation = 'softmax')
 # compile the model
 model %>% compile(
   optimizer = "rmsprop",
   loss = "categorical_crossentropy",
   metrics = c("accuracy") #  monitor classification accuracy
 )
}

# K-fold CV
k <- 5
indices <- sample(1:nrow(train_data))
folds <- cut(indices, breaks = k, labels = FALSE)
num_epochs <- 100 # number of epochs
all_scores <- c()
for (i in 1:k){
  cat("Processing fold #", i, "\n")

  val_indices <- which(folds == i, arr.ind = TRUE) # prepares the validation data: data from partition #k
  val_data <- train_data[val_indices,]
  val_targets <- train_labels[val_indices,]

  partial_train_data <- train_data[-val_indices,] # prepares the training data: data from all other partitions
  partial_train_targets <- train_labels[-val_indices,]

  model <- build_model() # use precompiled model function

  model %>% fit(partial_train_data, partial_train_targets,
                epochs = num_epochs, batch_size = 32,
                verbose = 0) # trains the model in silent mode (verbose = 0)

  results <- model %>% evaluate(val_data, val_targets, verbose = 0) # evaluate model on the validation data
  all_scores <- c(all_scores, results[2])
}
all_scores
mean(all_scores)


## ---- include=FALSE-------------------------------------------------------------
weights3 <- model %>% get_layer(index = 1) %>% get_weights()
```