```
##### Server.R script

packages <- c('imager', "shiny", "jpeg", "png", "reticulate", "devtools")


if (length(setdiff(packages, rownames(installed.packages()))) > 0) {

   install.packages(setdiff(packages, rownames(installed.packages())))

}


if (length(setdiff("keras", rownames(installed.packages()))) > 0) {

   devtools::install_github("rstudio/keras")

}


require(imager)

require(shiny)

require(jpeg)

require(png)

library(reticulate)

library(keras)


#setwd(tempfile())

#setwd("/Users/aiden/Desktop/data/cifar10_densenet")


load("envir.RData")

model<<-load_model_hdf5("densenet.h5")


synsets <<- readLines("synset.txt")
```

```
shinyServer(function(input, output) {
  ntext <- eventReactive(input$goButton, {
    print(input$url)
    if (input$url == "http://") {
      NULL
    } else {
      tmp_file <- tempfile()
      download.file(input$url, destfile = tmp_file, mode = 'wb')
      tmp_file
    }
  })

  output$originImage = renderImage({
    list(src = if (input$tabs == "Upload Image") {
      if (is.null(input$file1)) {
        if (input$goButton == 0 || is.null(ntext())) {
          'dog.jpg'
        } else {
          ntext()
        }
      } else {
        input$file1$datapath
      }
    } else {
      if (input$goButton == 0 || is.null(ntext())) {
        if (is.null(input$file1)) {
```

```
            'dog.jpg'
        } else {
            input$file1$datapath
        }
    } else {
        ntext()
    }
  },
  title = "Original Image")
}, deleteFile = FALSE)


output$res <- renderText({
  src = if (input$tabs == "Upload Image") {
    if (is.null(input$file1)) {
      if (input$goButton == 0 || is.null(ntext())) {
        'dog.jpg'
      } else {
        ntext()
      }
    } else {
      input$file1$datapath
    }
  } else {
    if (input$goButton == 0 || is.null(ntext())) {
      if (is.null(input$file1)) {
        'dog.jpg'
```

```
    } else {
      input$file1$datapath
    }
  } else {
    ntext()
  }
}

img <- load.image(src)
plot(img)
img <- image_load(src, target_size = c(32,32))
img
x <- image_to_array(img)
# ensure we have a 4d tensor with single element in the batch dimension,
x <- array_reshape(x, c(1, dim(x)))

# normalize
x[,,,1] <- (x[,,,1] - mea1) / sds1
x[,,,2] <- (x[,,,2] - mea2) / sds2
x[,,,3] <- (x[,,,3] - mea3) / sds3

# predcit
preds <- model %>% predict(x)

# output result as string
max.idx <- order(preds[1,], decreasing = TRUE)[1]
```

```
      result <- synsets[max.idx]
      res_str <- ""
      tmp <- strsplit(result[1], " ")[[1]]
      res_str <- paste0(res_str, tmp[2])
      res_str
    })
})

##### ui.R script
require(imager)
require(shiny)
require(jpeg)
require(png)

shinyUI(
  fluidPage(
    includeCSS("bootstrap.css"),

    pageWithSidebar(
      headerPanel(title = 'Image Classification(cifar10) using DenseNet',
                  windowTitle = 'Image Classification(cifar10) using DenseNet'),

      fluidRow(
        column(1),
        column(9,
               tabsetPanel(
```

```
            id = "tabs",
            tabPanel("Upload Image",
                     fileInput('file1', 'Upload a PNG / JPEG File:')),
            tabPanel(
              "Use the URL",
              textInput("url", "Image URL:", "http://"),
              actionButton("goButton", "Go!")
            )
          ),
          h3(titlePanel("DESCRIPTION - CIFAR-10 분류")),
          h3(titlePanel("비행기, 자동차, 새, 사슴, 개, 말, 트럭, 고양이, 개구리, 배"))

          ),
    column(2)
),


mainPanel(
    h3("Image"),
    tags$hr(),
    imageOutput("originImage", height = "auto"),
    tags$hr(),
    h3("What is this?"),
    tags$hr(),
    verbatimTextOutput("res")
)
```

```
      )))


##### densenet_model.R

# Libraries ---------------------------------------------------------------

library(keras)

library(densenet)


# Parameters --------------------------------------------------------------


batch_size <- 64

epochs <- 300


# Data Preparation --------------------------------------------------------


# see ?dataset_cifar10 for more info

cifar10 <- dataset_cifar10()


# Normalisation

for(i in 1:3){

  mea <- mean(cifar10$train$x[,,,i])

  sds <- sd(cifar10$train$x[,,,i])


  cifar10$train$x[,,,i] <- (cifar10$train$x[,,,i] - mea) / sds

  cifar10$test$x[,,,i] <- (cifar10$test$x[,,,i] - mea) / sds
```

```
}

x_train <- cifar10$train$x

x_test <- cifar10$test$x


y_train <- to_categorical(cifar10$train$y, num_classes = 10)

y_test <- to_categorical(cifar10$test$y, num_classes = 10)


# Model Definition ----------------------------------------------------


input_img <- layer_input(shape = c(32, 32, 3))

model <- application_densenet(include_top = TRUE, input_tensor = input_img, dropout_rate = 0.2)


opt <- optimizer_sgd(lr = 0.1, momentum = 0.9, nesterov = TRUE)


model %>% compile(

  optimizer = opt,

  loss = "categorical_crossentropy",

  metrics = "accuracy"

)


# Model fitting -------------------------------------------------------


# callbacks for weights and learning rate

lr_schedule <- function(epoch) {


  if(epoch <= 150) {
```

```
    0.1
  } else if(epoch > 150 && epoch <= 225){
    0.01
  } else {
    0.001
  }

}

lr_reducer <- callback_learning_rate_scheduler(lr_schedule)

history <- model %>% fit(
  x_train, y_train,
  batch_size = batch_size,
  epochs = epochs,
  validation_data = list(x_test, y_test),
  callbacks = list(
    lr_reducer
  )
)
save_model_hdf5(model, "densenet.h5")
save(history, "densenet_history.RData")
# 변수 환경 저장 -> envir.Rdata
plot(history)
evaluate(model, x_test, y_test)
```

```
##### moving chart

library(plotly)

library(dplyr)


# densenet_history 로드는 그냥 rstudio 로함

# 명령어 무엇????


# frame 만드는 펑션


accumulate_by <- function(dat, var) {

  var <- lazyeval::f_eval(var, dat)

  lvls <- plotly:::getLevels(var)

  dats <- lapply(seq_along(lvls), function(x) {

    cbind(dat[var %in% lvls[seq(1, x)], ], frame = lvls[[x]])

  })

  dplyr::bind_rows(dats)

}


# 원하는 데이터 셋만들기


df2 <- as.data.frame(densenet_history$metrics)

df2$epoch <- c(1:300)




dd <- df2 %>% select(epoch, val_acc, acc) %>%
```

```
  mutate(val_acc = round(val_acc*100,0), acc = round(acc*100,0)) %>%
  accumulate_by(~epoch)

dd

# plot

p <- dd %>%
  plot_ly(
    x = ~epoch,
    y = ~val_acc,
    frame = ~frame,
    type = 'scatter',
    mode = 'lines',
    line = list(simplyfy = F),name = 'val_acc'
  ) %>%
  add_trace(y=~acc, line = list(shape = 'spline',dash ='dot'),name='acc') %>%
  layout(
    xaxis = list(
      title = "epoch",
      zeroline = F
    ),
    yaxis = list(
      title = "",
      zeroline = F
    )
```

```
  ) %>%
  animation_opts(
    frame = 100,
    transition = 0,
    redraw = FALSE
  ) %>%
  animation_slider(
    hide = T
  ) %>%
  animation_button(
    x = 1, xanchor = "right", y = 0, yanchor = "bottom"
  )

p
```