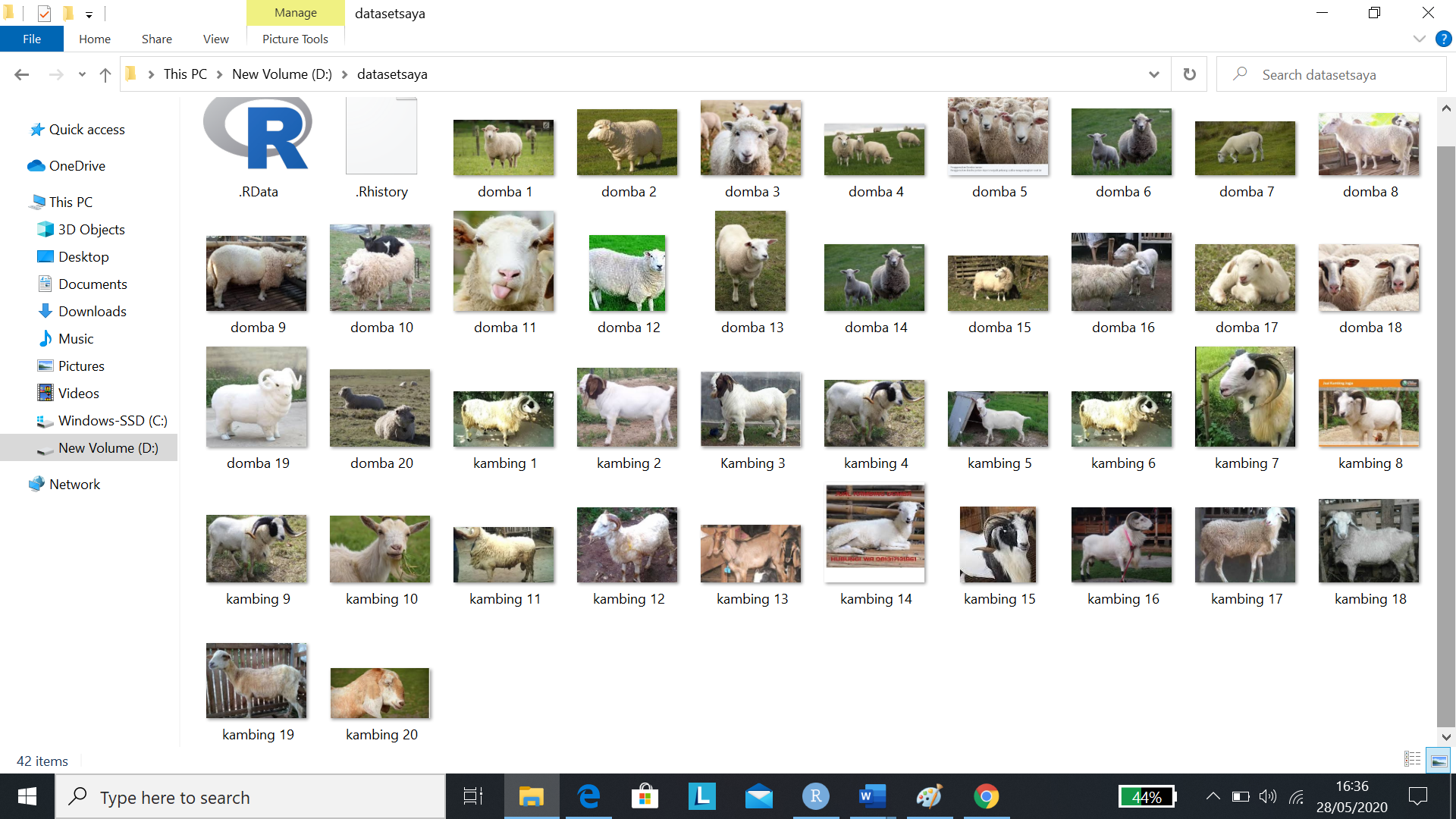
**LAMPIRAN**

**Lampiran 1. Dataset kambing dan domba**

**Lampiran 2. Kode program CNN untuk klasifikasi kambing dan domba**

library (devtools)

library(keras)

library (BiocManager)

library(EBImage)

library(reticulate)

library(tensorflow)

#input data

setwd('D:/datasetsaya')

# Read Images

images <- list.files()

images

summary(images)

list\_of\_images = lapply(images,readImage)

list\_of\_images

display(list\_of\_images[[5]])

#createtest

test <- list\_of\_images [c(17:20,37:40)]

test

display(test[[1]])

par(mfrow=c(2,4))

for (i in 1:8) plot(test[[i]])

#createtrain

train<- list\_of\_images [c(1:16,21:36)]

train

display (train[[1]])

par(mfrow=c(4,8))

for (i in 1:32) plot(train[[i]])

# Resize & combine

for (i in 1:32) {train[[i]] <- resize(train[[i]], 32, 32)}

for (i in 1:8) {test[[i]] <- resize(test[[i]], 32, 32)}

train <- combine(train)

str(train)

x <- tile(train, 8)

display(x, title='Pictures')

test <- combine(test)

y <- tile(test, 4)

display(y, title = 'Pics')

# Reorder dimension

dim(test)

dim(train)

train <- aperm(train, c(4,1,2,3))

test <- aperm(test, c(4,1,2,3))

str(train)

str(test)

# Response

trainy <- c(rep(0,16),rep(1,16))

testy <- c(rep(0,4),rep(1,4))

#install.packages

#devtools::install\_github('rstudio/keras')

library(reticulate)

# One hot encoding

trainLabels <- to\_categorical(trainy)

testLabels <- to\_categorical(testy)

trainLabels

testLabels

# Model

model <- keras\_model\_sequential()

model %>%

layer\_conv\_2d(filters = 32,

kernel\_size = c(3,3),

activation = 'relu',

input\_shape = c(32, 32, 3)) %>%

layer\_conv\_2d(filters = 32,

kernel\_size = c(3,3),

activation = 'relu') %>%

layer\_max\_pooling\_2d(pool\_size = c(2,2)) %>%

layer\_dropout(rate = 0.01) %>%

layer\_conv\_2d(filters = 64,

kernel\_size = c(3,3),

activation = 'relu') %>%

layer\_conv\_2d(filters = 64,

kernel\_size = c(3,3),

activation = 'relu') %>%

layer\_max\_pooling\_2d(pool\_size = c(2,2)) %>%

layer\_dropout(rate = 0.01) %>%

layer\_flatten() %>%

layer\_dense(units = 256, activation = 'relu') %>%

layer\_dropout(rate=0.01) %>%

layer\_dense(units = 2, activation = 'softmax') %>%

compile(loss = 'categorical\_crossentropy',

optimizer = optimizer\_sgd(lr = 0.01,

decay = 1e-6,

momentum = 0.9,

nesterov = T),

metrics = c('accuracy'))

summary(model)

# Fit model

history <- model %>%

fit(train,

trainLabels,

epochs =70,

batch\_size = 32,

validation\_split = 0.2,

validation\_data = list(test, testLabels)

)

plot(history)

model %>% evaluate(train, trainLabels)

pred <- model %>% predict\_classes(train)

table(Predicted = pred, Actual = trainy)

pred

trainy

model %>% evaluate(test, testLabels)

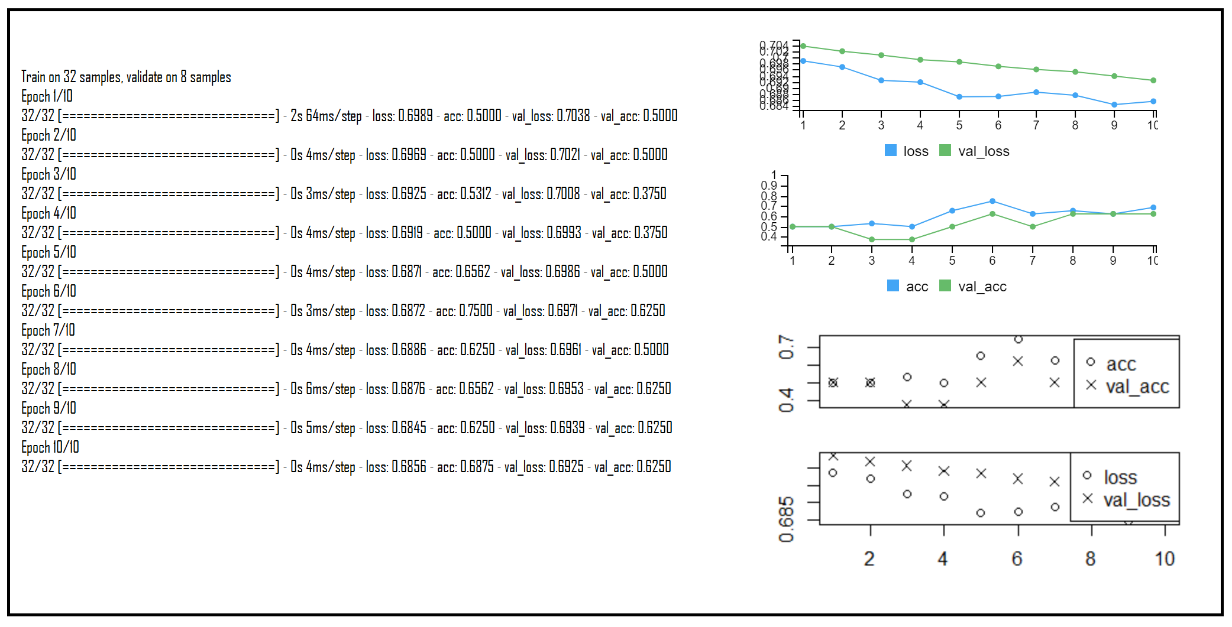
pred <- model %>% predict\_classes(test)

table(Predicted = pred, Actual = testy)

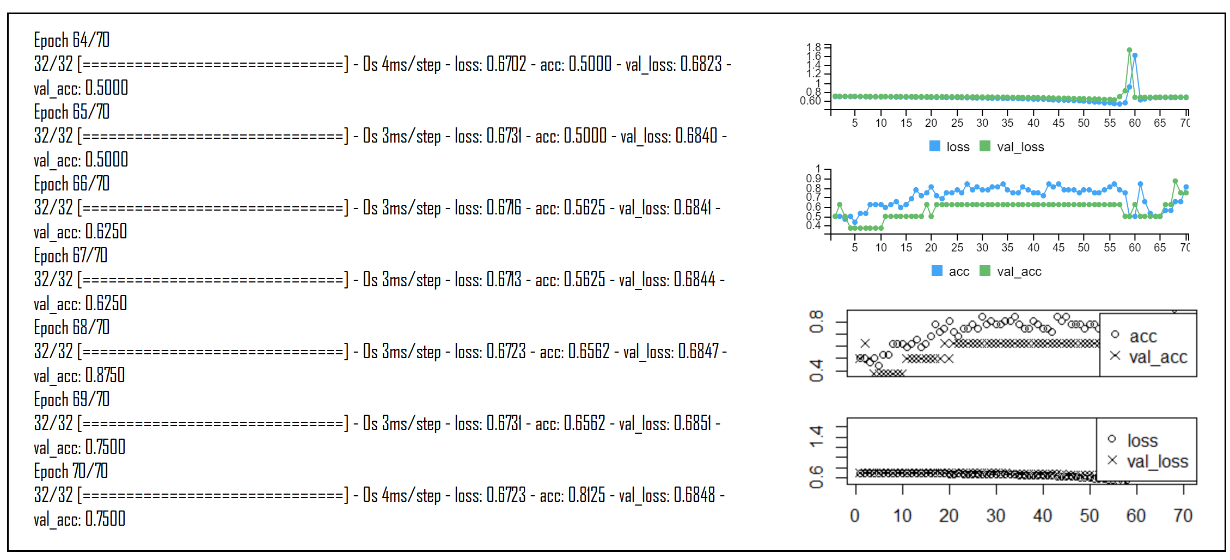
pred

testy

**Lampiran 3. Contoh hasil CNN dengan epoch = 10**



**Lampiran 4. Contoh hasil CNN dengan epoch 70**



**Lampiran 5. Contoh hasil CNN dengan epoch 100**

