SPRAWOZDANIE

Zajęcia: Analiza Procesów Uczenia
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium 6
Data 12.05.2023r.
Temat: "Ucznie głębokie w R.
Klasyfikator obrazów za pomocą Keras"
Wariant 5

Jarosław Waliczek
Informatyka
II stopień,
stacjonarne
1 semestr

1. Polecenie:

Zadanie dotyczy konstruowania sieci głębokiej w celu klasyfikacji obrazów pobranych ze zbioru danych. Warianty zadania są określone zbiorem danych obrazów, który może być pobrany na stronie https://keras.io/api/datasets/

5. CIFAR-10

2. Wprowadzane dane:

Dane są pobrane ze strony https://keras.io/api/datasets/

3. Wykorzystane komendy:

Konfiguracja bibliotek i ładowanie danych:

```
setwd("C:/Users/jaro9/OneDrive/Desktop/apu/zad6")

install.packages("keras")
library("keras")
install.packages("tensorflow")
library("tensorflow")
tensorflow::install_tensorflow()
library(reticulate)
library(keras)
virtualenv_create("myenv")
install_keras(method="virtualenv", envname="myenv")
#load data cifar 10
cifar <- dataset_cifar10()

x_train <- cifar$train$x
x_test <- cifar$test$x
y_train <- cifar$train$y
y_test <- cifar$test$y
```

Operacje na danych (wersja liniowa):

```
#------------------------------------------------- wersja liniowa

#set up data
#change matrix shape
x_train <- array_reshape(x_train, c(nrow(x_train), 3072))
#normalize
x_train <- x_train / 255

x_test <- array_reshape(x_test, c(nrow(x_test), 3072))
x_test <- x_test / 255

#set classes
y_train <- to_categorical(y_train, num_classes = 10)
y_test <- to_categorical(y_test, num_classes = 10)

#256 neurons, dropout rate 0.25
model <- keras_model_sequential() %>%
  layer_dense(units = 256, activation = "relu", input_shape = c(3072)) %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 64, activation = "relu") %>%
  layer_dropout(rate = 0.25) %>%
  layer_dense(units = 10, activation = "relu")

summary(model)

#set model parameters
model %>% compile(
  loss = "categorical_crossentropy",     #calculate loss
  optimizer = optimizer_adam(),          #optimization
  metrics = c("accuracy")                #accuracy
)

#train model
history <- model %>%
  fit(
    x_train, y_train,          #input
    epochs = 50,
    batch_size = 128,          #128 pictures
    validation_split = 0.15
  )

#check quality
model %>% evaluate(x_test, y_test)
```

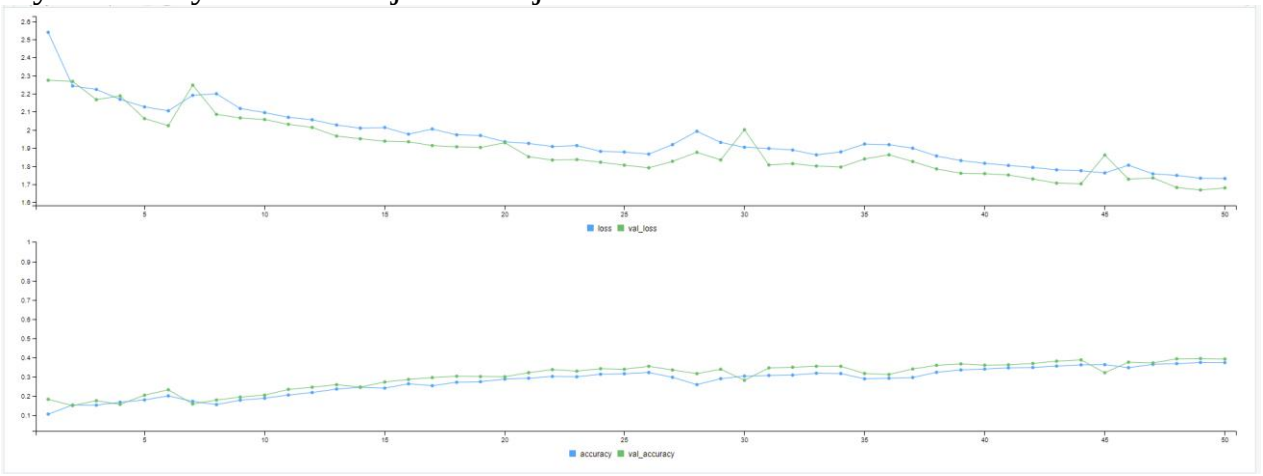
Operacje na danych (wersja spłaszczona):

```
#---------------------------------------- wersja sp³aszczona
cifar <- dataset_cifar10()

x_train <- cifar$train$x
x_test <- cifar$test$x
y_train <- cifar$train$y
y_test <- cifar$test$y

#set up data
#normalize
x_train <- x_train / 255

x_test <- x_test / 255

#set classes
y_train <- to_categorical(y_train, num_classes = 10)
y_test <- to_categorical(y_test, num_classes = 10)

#create model
model <- keras_model_sequential() %>%
  layer_flatten(input_shape = c(32, 32, 3)) %>%
  layer_dense(units = 128, activation = "relu") %>%
  layer_dense(units = 10, activation = "softmax")

#print model
summary(model)

#set model parameters
model %>% compile(
  loss = "categorical_crossentropy",      #calculate loss
  optimizer = optimizer_adam(),           #optimization
  metrics = c("accuracy")                 #accuracy
)

#train model
history <- model %>%
  fit(
    x_train, y_train,
    epochs = 50,
    batch_size = 128,
    validation_split = 0.15
  )

#check model quality
model %>% evaluate(x_test, y_test)

#predict
model %>% predict(x_test) %>% k_argmin()
```

4. Wynik działania:

Kod programu dostępny w repozytorium: https://github.com/Jaro233/APU.git

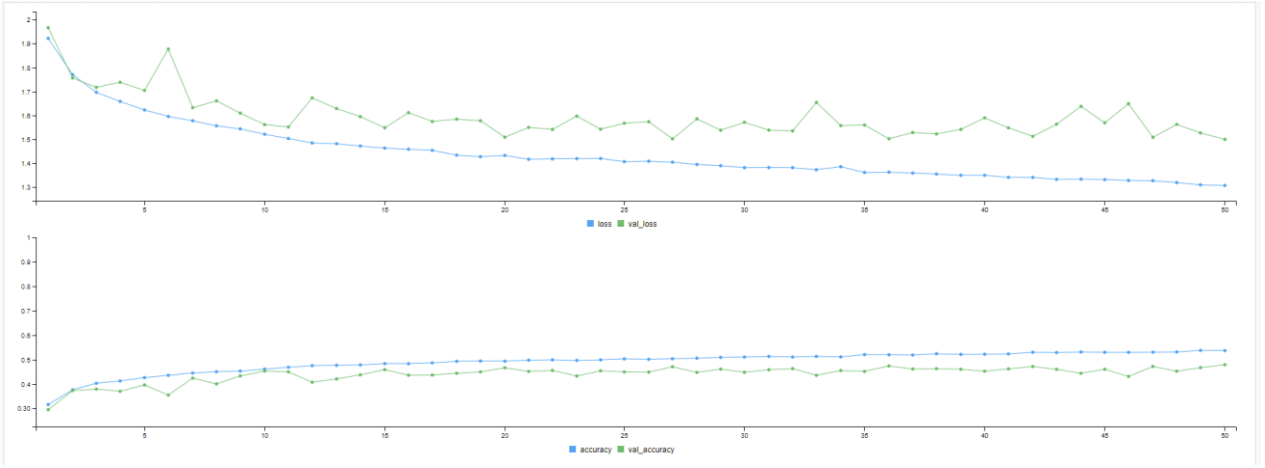
Wytrenowany model wersji liniowej:



Jakość modelu wersji liniowej:

```
> model %>% evaluate(x_test, y_test)

  1/313 [..............................] - ETA: 6s - loss: 1.4592 - accuracy: 0.4375
 31/313 [=>............................] - ETA: 0s - loss: 1.5896 - accuracy: 0.4204
 62/313 [====>.........................] - ETA: 0s - loss: 1.5896 - accuracy: 0.4385
 94/313 [========>.....................] - ETA: 0s - loss: 1.6118 - accuracy: 0.4192
124/313 [==========>...................] - ETA: 0s - loss: 1.6032 - accuracy: 0.4226
152/313 [=============>................] - ETA: 0s - loss: 1.6032 - accuracy: 0.4196
182/313 [================>.............] - ETA: 0s - loss: 1.6096 - accuracy: 0.4181
211/313 [===================>..........] - ETA: 0s - loss: 1.6180 - accuracy: 0.4156
240/313 [======================>.......] - ETA: 0s - loss: 1.6160 - accuracy: 0.4173
267/313 [========================>.....] - ETA: 0s - loss: 1.6188 - accuracy: 0.4161
284/313 [==========================>...] - ETA: 0s - loss: 1.6185 - accuracy: 0.4164
311/313 [============================>.] - ETA: 0s - loss: 1.6182 - accuracy: 0.4161
313/313 [==============================] - 1s 2ms/step - loss: 1.6176 - accuracy: 0.4157
    loss accuracy
1.617603 0.415700
```

Wytrenowany model wersji spłaszczonej:



Jakość modelu wersji spłaszczonej:

```
> model %>% evaluate(x_test, y_test)

  1/313 [..............................] - ETA: 5s - loss: 1.1480 - accuracy: 0.5625
 45/313 [===>..........................] - ETA: 0s - loss: 1.4609 - accuracy: 0.4875
 89/313 [=======>......................] - ETA: 0s - loss: 1.4804 - accuracy: 0.4807
131/313 [===========>..................] - ETA: 0s - loss: 1.4678 - accuracy: 0.4907
176/313 [===============>..............] - ETA: 0s - loss: 1.4699 - accuracy: 0.4883
221/313 [====================>.........] - ETA: 0s - loss: 1.4717 - accuracy: 0.4900
266/313 [========================>.....] - ETA: 0s - loss: 1.4747 - accuracy: 0.4884
311/313 [============================>.] - ETA: 0s - loss: 1.4738 - accuracy: 0.4888
313/313 [==============================] - 0s 1ms/step - loss: 1.4738 - accuracy: 0.4888
    loss accuracy
1.473762 0.488800
```

5. Wnioski:

Dzięki bibliotece tensorflow i keras możliwe było trenowanie modelu oraz ocenienie jego jakości.