HW 4 MATH 868

James DG

Fall 2022

Visualized the hand-written digits from train.csv

The original data are from http://yann.lecun.com/exdb/mnist/

The images contain grey levels.

```
mnist <- read.csv('train.csv', header=T)
y_t=mnist[,1] # True label
y_train=y_t[1:10000] # train set
y_test=y_t[10001:42000] # test set
```

Homework problems:

1. Load the train.csv file and convert it into proper format that can be processed by convolutional neural networks.

```
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.1.2
```

```
x_train = mnist[1:10000,-1]
x_test  = mnist[10001:42000,-1]

# Use reticulate to call array_reshape function from Python (numpy?) to make each
# digit sample a 28x28 matrix
x_train_2d = reticulate::array_reshape(as.matrix(x_train), dim=c(dim(x_train)[1], 28, 28), )
x_test_2d = reticulate::array_reshape(as.matrix(x_test), dim=c(dim(x_test)[1], 28, 28), )

train_images <- array_reshape(x_train_2d, c(dim(x_train_2d)[1], 28, 28, 1))
train_images <- train_images / 255                                    # normalize data

test_images <- array_reshape(x_test_2d, c(dim(x_test_2d)[1], 28, 28, 1))
test_images <- test_images / 255                                      # normalize data
train_labels <- to_categorical(y_train)
```

```
## Loaded Tensorflow version 2.9.2
```

```
test_labels <- to_categorical(y_test)
```

2. Use optimizer adam to train the model, with training rate equal to 0.0001, batch size equal to 32, validation_split equal 0.3, 100 epochs without early stopping. Plot training history.

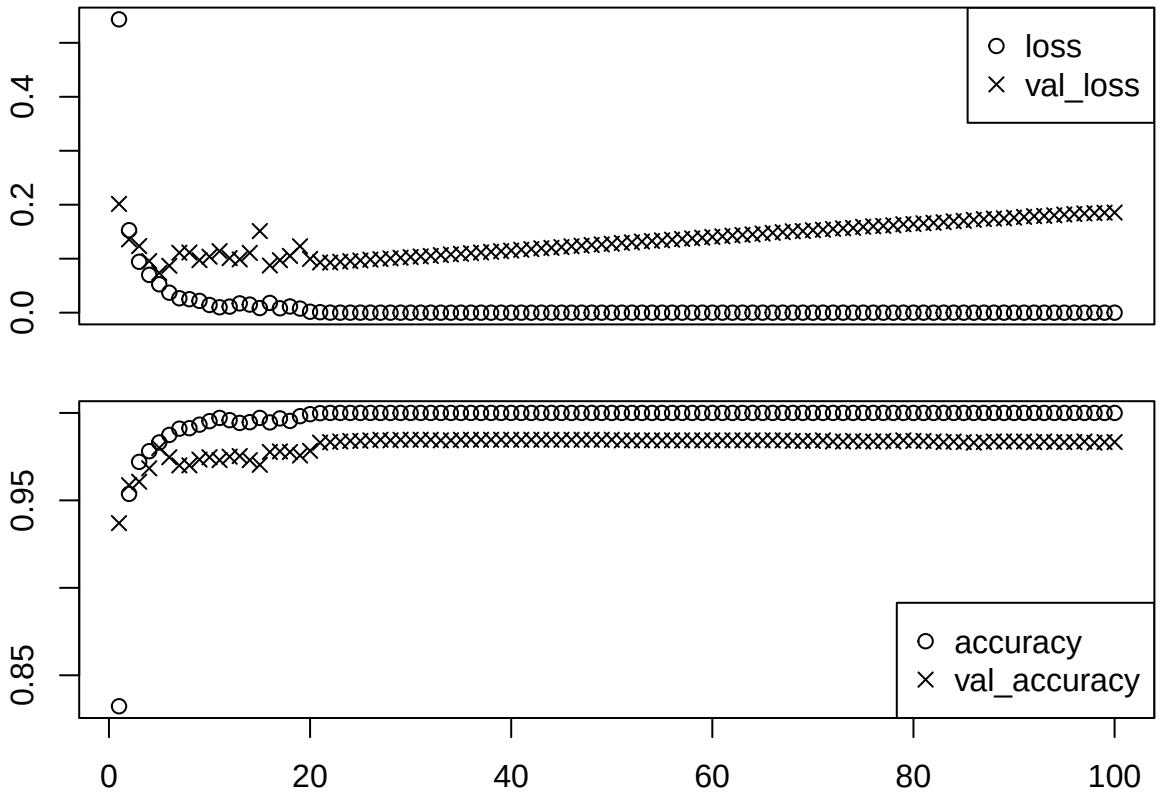
```
model <- keras_model_sequential() %>%
         layer_conv_2d(filters = 32, kernel_size = c(3, 3), activation = "relu",
             input_shape = c(28, 28, 1)) %>%
             layer_max_pooling_2d(pool_size = c(2, 2)) %>%
             layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu") %>%
             layer_max_pooling_2d(pool_size = c(2, 2)) %>%
             layer_conv_2d(filters = 64, kernel_size = c(3, 3), activation = "relu")


model <- model %>%
     layer_flatten() %>%
         layer_dense(units = 64, activation = "relu") %>%
             layer_dense(units = 10, activation = "softmax")


model %>% compile(
               optimizer = "adam",
               loss = "categorical_crossentropy",
               metrics = c("accuracy") )

history <- model %>% fit(
                train_images, train_labels,
                epochs = 100, batch_size=32,
                learning_rate=0.0001,
                validation_split = 0.3 )

plot(history)
```



3. Apply the trained model to testing set and make prediction for the last 10 graphs. Generate the prediced digits for the last 10 graphs.

```
pre_10 = model %>% predict(test_images[1:10, , ,1 , drop=F])
dim(pre_10)
```

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

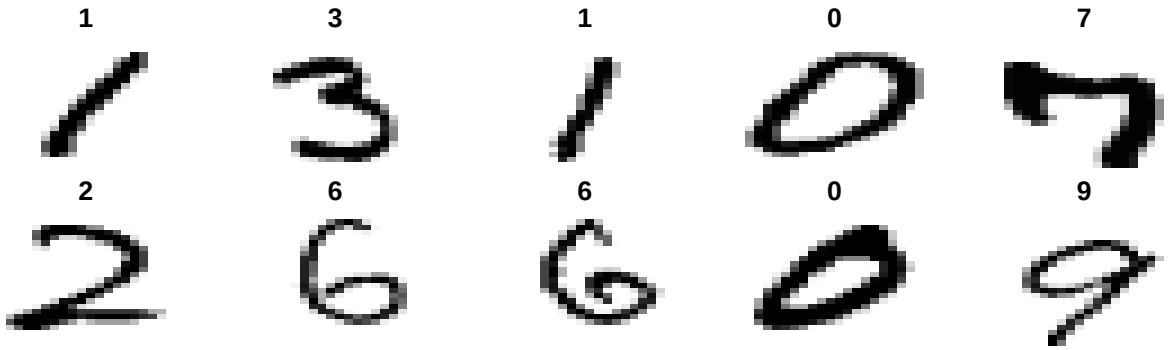
```
y_pre = apply(pre_10, MARGIN=1, which.max) - 1 # need subtract one, since the labels starting from 0

#_pre == test_labels[1:10]

cat('The prediction accuracy of model is ', mean(y_pre == y_test[1:10])*100, '%', '.', '\n', sep='')
```

```
## The prediction accuracy of model is 100%.
```

```
par(mfrow=c(5,5), mar=c(0,0,1,0))
for(i in sample(1:32000,10,replace=FALSE)){
    dat <- matrix(as.numeric(x_test[i, 1:784]), nrow=28, byrow =F)[,28:1]
    image(dat, axes=FALSE,col=grey(seq(1,0,length=256)), ylab='', xlab='', main=y_test[i])
}
```


1
3
1
0
7
2
6
6
0
9

‘‘