HW #1

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_CODE\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

library(keras)

library(DataExplorer)

library(tidyverse)

# Load data

mnist\_fashion <- dataset\_fashion\_mnist()

train\_images <- mnist\_fashion$train$x

train\_labels <- mnist\_fashion$train$y

test\_images <- mnist\_fashion$test$x

test\_labels <- mnist\_fashion$test$y

# View the data

glimpse(train\_images)

glimpse(train\_labels)

# View an image

pic <- train\_images[1,,]

plot(as.raster(pic, max = 255))

# Show dimision

dim(train\_images)

dim(test\_images)

# Reshape the data

train\_images <- array\_reshape(train\_images, c(60000, 28\*28))

test\_images <- array\_reshape(test\_images, c(10000, 28\*28))

# Scale the data

train\_images <- train\_images / 255

test\_images <- test\_images / 255

# Categorize labels & add extra category

train\_labels <- to\_categorical(train\_labels, num\_classes = 11)#ONE-Hot encoding

test\_labels <- to\_categorical(test\_labels, num\_classes = 11)

# Build the model

network <- keras\_model\_sequential() %>%

layer\_dense(units = 16, activation = "relu", initializer\_he\_normal(), input\_shape = c(28 \* 28)) %>%

layer\_dense(units = 16, activation = "relu", initializer\_he\_normal()) %>%

layer\_dense(units = 11, activation = "softmax")

network %>% compile(

optimizer = "rmsprop", # optimizer\_adam(lr=.0001, decay = 1E-6),

loss = "categorical\_crossentropy",

metrics = c("accuracy"))

# View the network

network

# Train the model

network %>% fit(train\_images, train\_labels, epochs = 10, batch\_size = 256, validation\_split = 0.2)

­­­­



**Changes**

**Base:**

**$loss**

**[1] 0.4844345**

**$acc**

**[1] 0.8337**

1. # layers = 4
   1. Slight rise
2. # layers = 2
   1. Slight rise in accuracy
3. # neurons = 16-20-11
   1. Slight drop
4. # neurons = 20-16-11
   1. Increase
5. Activation functions = relu, selu, softmax
   1. Increase
6. Activation functions = selu, relu, softmax
   1. Very slight rise
7. Optimizer = adam
   1. Drop
8. Optimizer = Nadam
   1. Drop
9. Batch size = 128
   1. Rise
10. Batch size = 512
    1. Drop
11. Epochs = 20
    1. Drop
12. Epochs = 2
    1. Drop

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Load libraries

library(keras)

library(DataExplorer)

library(tidyverse)

# Load data

mnist\_fashion <- dataset\_fashion\_mnist()

train\_images <- mnist\_fashion$train$x

train\_labels <- mnist\_fashion$train$y

test\_images <- mnist\_fashion$test$x

test\_labels <- mnist\_fashion$test$y

# View the data

glimpse(train\_images)

glimpse(train\_labels)

# View an image

pic <- train\_images[1,,]

plot(as.raster(pic, max = 255))

# Show dimision

dim(train\_images)

dim(test\_images)

# Reshape the data

train\_images <- array\_reshape(train\_images, c(60000, 28\*28))

test\_images <- array\_reshape(test\_images, c(10000, 28\*28))

# Scale the data (0-1)

train\_images <- train\_images / 255

test\_images <- test\_images / 255

# Categorize labels & add extra category

train\_labels <- to\_categorical(train\_labels, num\_classes = 10)#ONE-Hot encoding

test\_labels <- to\_categorical(test\_labels, num\_classes = 10)

# Build the model

network <- keras\_model\_sequential() %>%

layer\_dense(units = 512, activation = "relu", initializer\_he\_normal(), input\_shape = c(28 \* 28)) %>%

layer\_dense(units = 256, activation = "relu", initializer\_he\_normal()) %>%

layer\_dense(units = 256, activation = "relu", initializer\_he\_normal()) %>%

layer\_dense(units = 10, activation = "softmax")

network %>% compile(

optimizer = "adam", # optimizer\_adam(lr=.0001, decay = 1E-6),

loss = "categorical\_crossentropy",

metrics = c("accuracy"))

# View the network

#network

# Train the model

network %>% fit(train\_images, train\_labels, epochs = 15, batch\_size = 256, validation\_split = 0.2)

metrics <- network %>% evaluate(test\_images, test\_labels)

metrics

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



Best Loss/Acc

$loss

[1] 0.3750155

$acc

[1] 0.8883