Lab Report 3

By: Abhishek Shah, Ravi Seth, Tanya Ralliaram

```
# Insert necessary packages
library(keras)
```

```
## Warning: package 'keras' was built under R version 4.0.4
```

```
library(tidyverse)
library(neuralnet)
```

```
## Warning: package 'neuralnet' was built under R version 4.0.4
```

Question 1: Classification using NNets

```
mnist <- dataset_fashion_mnist()
```

1.1: Get Data

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

dim(x_train)
```

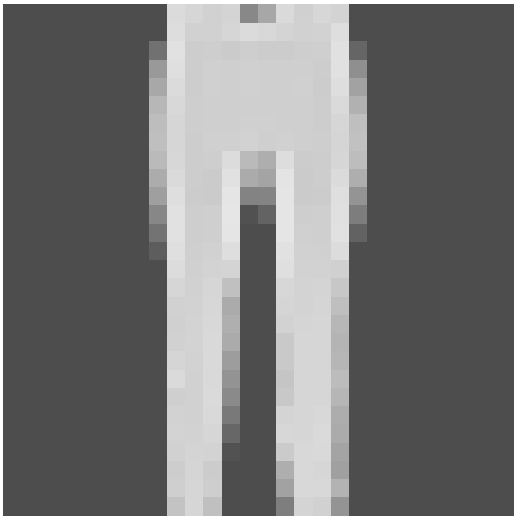
```
## [1] 60000    28    28
```

```
dim(x_test)
```

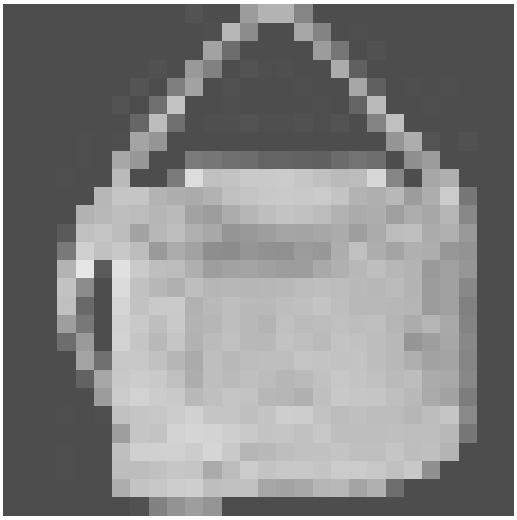
```
## [1] 10000    28    28
```

1.2: Plot

```
par(pty="s") # for keeping the aspect ratio 1:1
trouser <- x_train[107,28:1,1:28]
image(t(trouser), col = gray.colors(256), axes = FALSE)
```



```r
bag <- x_train[213,28:1,1:28]
image(t(bag), col = gray.colors(256), axes = FALSE)
```



```r
boot <- x_train[1,28:1,1:28]
image(t(boot), col = gray.colors(256), axes = FALSE)
```

1.3: Process the dataset

```
# reshape
x_train <- array_reshape(x_train, c(nrow(x_train), 784))
x_test <- array_reshape(x_test, c(nrow(x_test), 784))

# rescale
x_train <- x_train / 255
x_test <- x_test / 255

y_train <- to_categorical(y_train, 10)
y_test  <- to_categorical(y_test, 10)
```

1.4: Fit a Shallow Network

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dense(units = 10, activation = 'softmax')
#summary(model)

model %>% compile(
```

```
  loss = 'categorical_crossentropy',
  optimizer = optimizer_adam(),
  metrics = c('accuracy')
)

history <- model %>% fit(
  x_train, y_train,
  epochs = 10,
  batch_size = 128,
  validation_split = 0.2
)

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

```
##      loss  accuracy
## 0.3370254 0.8819000
```

The settings that performed the best were using 256 neurons for the hidden layer and using 'relu' as the activation function.

1.5: Fit a Deep Neural Network

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_adam(),
  metrics = c('accuracy')
)

history <- model %>% fit(
  x_train, y_train,
  epochs = 10,
  batch_size = 128,
  validation_split = 0.2
)
model %>% evaluate(x_test, y_test)
```

```
##      loss  accuracy
## 0.3342643 0.8848000
```

The model that seems to generate the best test accuracy is using 256 neurons for both hidden layers and using 'relu' as the activation function.