HW9_Kim_Jaiyool

Jaiyool Kim

2019-11-15

Getting Started

Installation

First, install the keras R package from Github as follows:

```
devtools::install_github("rstudio/keras")
```

```
## Skipping install of 'keras' from a github remote, the SHA1 (ae249fb0) has not changed since last ins
##   Use `force = TRUE` to force installation
```

The Keras R interface uses the TensorFlow backend engine by default. To install both the core Keras library as well as the TensorFlow backend use the install_keras() function:

```
# install_keras()
library(keras)
```

This will provide you with default CPU-based installations of Keras and TensorFlow. If you want a more customized installation, e.g. if you want to take advantage of NVIDIA GPUs, see the documentation for install_keras().

Learning Keras

Below we walk through a simple example of using Keras to recognize handwritten digits from the MNIST dataset. After getting familiar with the basics, check out the tutorials and additional learning resources available on this website.

The Deep Learning with R book by François Chollet (the creator of Keras) provides a more comprehensive introduction to both Keras and the concepts and practice of deep learning.

You may also find it convenient to download the Deep Learning with Keras cheat sheet, a quick high-level reference to all of the capabilities of Keras.

MNIST Example

We can learn the basics of Keras by walking through a simple example: recognizing handwritten digits from the MNIST dataset. MNIST consists of 28 x 28 grayscale images of handwritten digits like these:

The dataset also includes labels for each image, telling us which digit it is. For example, the labels for the above images are 5, 0, 4, and 1.

Preparing the Data

The MNIST dataset is included with Keras and can be accessed using the dataset_mnist() function. Here we load the dataset then create variables for our test and training data:

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

The x data is a 3-d array (images,width,height) of grayscale values . To prepare the data for training we convert the 3-d arrays into matrices by reshaping width and height into a single dimension (28x28 images are flattened into length 784 vectors). Then, we convert the grayscale values from integers ranging between 0 to 255 into floating point values ranging between 0 and 1:

```
# 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
```

Note that we use the array_reshape() function rather than the dim<-() function to reshape the array. This is so that the data is re-interpreted using row-major semantics (as opposed to R's default column-major semantics), which is in turn compatible with the way that the numerical libraries called by Keras interpret array dimensions.

The y data is an integer vector with values ranging from 0 to 9. To prepare this data for training we one-hot encode the vectors into binary class matrices using the Keras to_categorical() function:

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

Defining the model

The core data structure of Keras is a model, a way to organize layers. The simplest type of model is the Sequential model, a linear stack of layers.

We begin by creating a sequential model and then adding layers using the pipe (%>%) operator:

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(784)) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 10, activation = 'softmax')
```

The input_shape argument to the first layer specifies the shape of the input data (a length 784 numeric vector representing a grayscale image). The final layer outputs a length 10 numeric vector (probabilities for each digit) using a softmax activation function.

Use the summary() function to print the details of the model:

```
summary(model)
```

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                    Output Shape                  Param #
## =========================================================================
## dense (Dense)                   (None, 256)                   200960
## _________________________________________________________________________
## dropout (Dropout)               (None, 256)                   0
## _________________________________________________________________________
## dense_1 (Dense)                 (None, 128)                   32896
## _________________________________________________________________________
## dropout_1 (Dropout)             (None, 128)                   0
## _________________________________________________________________________
## dense_2 (Dense)                 (None, 10)                    1290
## =========================================================================
## Total params: 235,146
## Trainable params: 235,146
## Non-trainable params: 0
## _________________________________________________________________________
```

Next, compile the model with appropriate loss function, optimizer, and metrics:

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

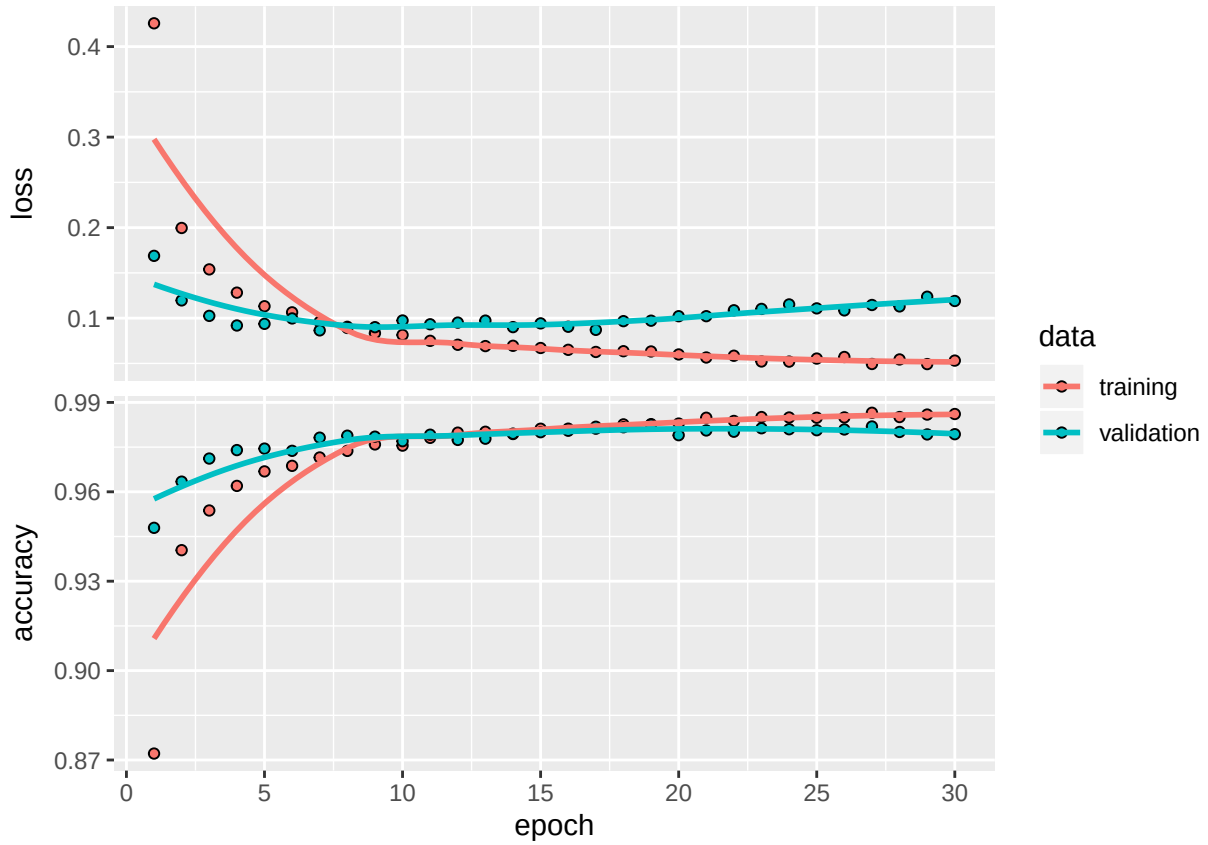
Training and Evaluation

Use the fit() function to train the model for 30 epochs using batches of 128 images:

```
history <- model %>% fit(
  x_train, y_train,
  epochs = 30, batch_size = 128,
  validation_split = 0.2
)
```

The history object returned by fit() includes loss and accuracy metrics which we can plot:

```
plot(history)
```



Evaluate the model's performance on the test data:

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

```
## $loss
## [1] 0.1078685
##
## $accuracy
## [1] 0.9798
```

Generate predictions on new data:

```
model %>% predict_classes(x_test)
```

```
##   [1] 7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 5 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4
##  [35] 7 2 7 1 2 1 1 7 4 2 3 5 1 2 4 4 6 3 5 5 6 0 4 1 9 5 7 8 9 3 7 4 6 4
##  [69] 3 0 7 0 2 9 1 7 3 2 9 7 7 6 2 7 8 4 7 3 6 1 3 6 9 3 1 4 1 7 6 9 6 0
## [103] 5 4 9 9 2 1 9 4 8 7 3 9 7 4 4 4 9 2 5 4 7 6 7 9 0 5 8 5 6 6 5 7 8 1
## [137] 0 1 6 4 6 7 3 1 7 1 8 2 0 2 9 9 5 5 1 5 6 0 3 4 4 6 5 4 6 5 4 5 1 4
## [171] 4 7 2 3 2 7 1 8 1 8 1 8 5 0 8 9 2 5 0 1 1 1 0 9 0 3 1 6 4 2 3 6 1 1
## [205] 1 3 9 5 2 9 4 5 9 3 9 0 3 6 5 5 7 2 2 7 1 2 8 4 1 7 3 3 8 8 7 9 2 2
## [239] 4 1 5 9 8 7 2 3 0 6 4 2 4 1 9 5 7 7 2 8 2 0 8 5 7 7 9 1 8 1 8 0 3 0
## [273] 1 9 9 4 1 8 2 1 2 9 7 5 9 2 6 4 1 5 8 2 9 2 0 4 0 0 2 8 4 7 1 2 4 0
## [307] 2 7 4 3 3 0 0 3 1 9 6 5 2 5 9 7 9 3 0 4 2 0 7 1 1 2 1 5 3 3 9 7 8 6
## [341] 3 6 1 3 8 1 0 5 1 3 1 5 5 6 1 8 5 1 7 9 4 6 2 2 5 0 6 5 6 3 7 2 0 8
## [375] 8 5 4 1 1 4 0 3 3 7 6 1 6 2 1 9 2 8 6 1 9 5 2 5 4 4 2 8 3 8 2 4 5 0
## [409] 3 1 7 7 5 7 9 7 1 9 2 1 4 2 9 2 0 4 9 1 4 8 1 8 4 5 9 8 8 3 7 6 0 0
## [443] 3 0 2 0 6 4 9 3 3 3 2 3 9 1 2 6 8 0 5 6 6 6 3 8 8 2 7 5 8 9 6 1 8 4
## [477] 1 2 5 9 1 9 7 5 4 0 8 9 9 1 0 5 2 3 7 8 9 4 0 6 3 9 5 2 1 3 1 3 6 5
```

```
##   [511] 7 4 2 2 6 3 2 6 5 4 8 9 7 1 3 0 3 8 3 1 9 3 4 4 6 4 2 1 8 2 5 4 8 8
##   [545] 4 0 0 2 3 2 7 1 0 8 7 4 4 7 9 6 9 0 9 8 0 4 6 0 6 3 5 4 8 3 3 9 3 3
##   [579] 3 7 8 0 2 2 1 7 0 6 5 4 3 8 0 9 6 3 8 0 9 9 6 8 6 8 5 7 8 6 0 2 4 0
##   [613] 2 8 3 1 9 7 5 8 0 8 4 6 2 6 7 9 3 2 9 8 2 2 9 2 7 3 5 9 1 8 0 2 0 5
##   [647] 2 1 3 7 6 7 1 2 5 8 0 3 7 2 4 0 9 1 8 6 7 7 4 3 4 9 1 9 5 1 7 3 9 7
##   [681] 6 9 1 3 3 8 3 3 6 7 2 4 5 8 5 1 1 4 4 3 1 0 7 7 0 7 9 4 4 8 5 5 4 0
##   [715] 8 2 1 6 8 4 8 0 4 0 6 1 7 3 2 6 7 2 6 9 3 1 4 6 2 5 4 2 0 6 2 1 7 3
##   [749] 4 1 0 5 4 3 1 1 7 4 9 9 4 8 4 0 2 4 5 1 1 6 4 7 1 9 4 2 4 1 5 5 3 8
##   [783] 3 1 4 5 6 8 9 4 1 5 3 8 0 3 2 5 1 2 8 3 4 4 0 8 8 3 3 1 7 3 5 9 6 3
##   [817] 2 6 1 3 6 0 7 2 1 7 1 4 2 4 2 1 7 9 6 1 1 2 4 8 1 7 7 4 8 0 7 3 1 3
##   [851] 1 0 7 7 0 3 5 5 2 7 6 6 9 2 8 3 5 2 2 5 6 0 8 2 9 2 8 8 8 8 7 4 9 5
##   [885] 0 6 6 3 2 1 3 2 2 9 3 0 0 5 7 8 1 4 4 6 0 2 9 1 4 7 4 7 3 9 8 8 4 7
##   [919] 1 2 1 2 2 3 2 3 2 3 9 1 7 4 0 3 5 5 8 6 3 2 6 7 6 6 3 2 7 9 1 1 7 4
##   [953] 6 4 9 5 6 3 3 4 7 8 9 1 1 0 9 1 4 4 5 4 0 6 2 2 3 1 5 1 2 0 3 8 1 2
##   [987] 6 7 1 6 2 3 9 0 1 2 2 0 8 9 9 0 2 5 1 9 7 8 1 0 4 1 7 9 5 4 2 6 8 1
##  [1021] 3 7 5 4 4 1 8 1 3 8 1 2 5 8 0 6 2 1 1 7 1 5 3 4 6 9 5 0 9 2 2 4 8 2
##  [1055] 1 7 2 4 9 4 4 0 3 9 2 2 3 3 8 3 5 7 3 5 8 1 2 4 4 6 4 9 5 1 0 6 9 5
##  [1089] 9 5 9 7 3 8 0 3 7 1 3 6 7 8 5 9 7 9 6 3 6 3 7 4 6 5 3 5 4 7 8 7 8 0
##  [1123] 7 6 8 8 7 3 3 1 9 5 2 7 3 5 1 1 2 1 4 7 4 7 5 4 5 4 0 8 3 6 9 6 0 2
##  [1157] 7 4 4 4 4 6 6 4 7 9 3 4 5 5 8 7 3 7 2 7 0 2 4 1 1 1 8 9 2 8 7 2 0 1
##  [1191] 5 0 4 1 7 0 6 0 8 6 8 1 8 0 3 3 7 2 3 6 2 1 6 1 1 3 7 9 0 8 0 5 4 0
##  [1225] 2 8 2 2 9 8 4 0 4 5 8 5 1 2 1 3 1 7 4 5 7 2 0 3 8 8 6 2 5 6 1 9 2 1
##  [1259] 5 8 1 0 2 4 4 3 6 8 8 2 4 0 5 0 4 4 7 9 3 4 1 5 9 7 3 5 8 8 0 5 3 3
##  [1293] 6 6 0 1 6 0 3 5 4 4 1 2 9 1 4 6 9 9 3 9 8 4 4 3 1 3 1 3 8 7 9 4 8 8
##  [1327] 7 9 9 1 4 5 6 0 5 2 2 2 1 5 5 2 4 9 6 2 7 7 2 2 1 1 2 8 3 7 2 4 1 7
##  [1361] 1 7 6 7 8 2 7 3 1 7 5 8 2 6 2 2 5 6 6 0 9 2 4 3 3 9 7 6 6 8 0 4 1 3
##  [1395] 8 2 9 1 8 0 6 7 2 1 0 5 5 2 0 2 2 0 2 4 9 8 0 9 9 4 6 5 4 9 1 8 3 4
##  [1429] 9 9 1 2 2 8 1 9 6 4 0 9 4 8 3 8 6 0 2 5 1 9 6 2 9 4 0 9 6 0 6 2 5 4
##  [1463] 2 3 8 4 5 5 0 3 8 5 3 5 8 6 5 7 6 3 3 9 6 1 1 2 9 0 4 3 3 6 9 5 7 3
##  [1497] 7 7 7 8 1 9 8 3 0 7 2 7 9 4 5 4 9 3 2 1 4 0 2 3 7 5 9 8 8 5 0 1 1 4
##  [1531] 7 3 9 0 0 0 6 6 2 3 7 8 4 7 7 9 2 4 1 6 5 2 4 9 9 1 8 4 0 9 8 4 8 7
##  [1565] 7 0 7 8 8 6 0 4 8 8 2 4 7 6 6 6 4 7 1 8 8 2 3 6 3 0 0 3 7 6 9 7 9 9
##  [1599] 5 4 3 3 6 1 2 3 7 3 3 2 0 8 3 8 4 3 6 3 5 0 2 0 9 0 7 4 6 9 3 5 1 9
##  [1633] 6 1 4 5 4 5 0 5 9 5 2 1 2 9 1 9 9 4 0 8 4 5 2 9 2 1 2 1 7 3 6 8 8 4
##  [1667] 9 1 9 8 5 7 5 1 1 8 6 5 2 4 4 7 2 3 5 6 8 8 6 2 3 1 0 5 8 9 2 9 6 7
##  [1701] 0 4 8 7 1 7 4 1 0 3 7 2 0 0 9 1 7 0 7 8 4 7 2 0 4 6 0 3 1 1 3 3 9 6
##  [1735] 7 4 1 5 3 0 8 7 3 9 6 9 3 5 0 2 7 4 5 1 7 5 8 0 8 8 1 5 0 3 0 3 1 4
##  [1769] 0 3 7 2 7 1 8 0 7 0 4 3 1 9 8 7 7 1 4 9 9 3 8 1 7 9 0 2 0 3 3 7 6 9
##  [1803] 2 3 3 7 7 0 0 7 5 2 9 8 7 4 4 2 6 6 1 9 6 8 2 9 0 8 5 1 1 6 3 5 1 1
##  [1837] 1 3 1 2 3 0 2 0 1 3 5 5 7 4 8 9 6 9 6 8 3 6 6 8 5 1 4 2 4 4 5 1 1 9
##  [1871] 0 2 4 9 5 7 1 8 2 5 6 9 8 7 1 1 6 7 6 3 2 2 0 8 9 2 5 1 0 8 1 4 5 7
##  [1905] 9 6 9 0 6 1 5 5 8 3 8 2 6 5 0 7 4 6 1 3 4 7 3 2 3 4 2 5 2 7 1 7 2 6
##  [1939] 4 1 5 7 8 6 0 1 8 2 5 7 7 6 9 3 5 8 4 2 4 0 8 8 3 4 9 2 7 5 8 6 5 6
##  [1973] 0 8 6 7 3 6 4 9 4 6 6 3 2 4 1 0 1 4 6 2 9 1 1 0 6 3 9 5 6 5 6 5 9 4
##  [2007] 6 4 3 9 1 3 4 1 9 1 7 1 1 9 3 5 4 0 9 3 6 1 7 5 5 3 3 0 1 3 7 5 8 6
##  [2041] 5 1 0 8 2 3 4 6 7 9 8 1 8 9 9 2 8 6 2 7 0 0 6 7 5 8 6 0 9 3 9 1 3 5
##  [2075] 4 3 3 5 5 6 3 0 2 3 4 2 3 0 9 9 4 7 2 1 4 7 0 6 2 8 5 2 8 5 7 3 0 8
##  [2109] 2 8 2 8 2 5 5 7 6 4 6 8 4 8 2 7 4 5 2 0 3 9 4 6 7 2 5 1 1 1 2 3 6 7
##  [2143] 8 7 6 4 8 9 4 8 6 3 8 3 1 0 6 2 2 5 6 9 8 8 1 4 1 7 8 4 6 1 8 4 3 1
##  [2177] 2 8 0 8 5 9 2 4 2 0 2 7 0 9 0 2 5 7 6 7 9 4 2 6 2 4 4 8 0 4 4 5 8 0
##  [2211] 6 8 9 8 5 6 9 0 4 8 7 1 3 4 6 8 0 9 1 3 3 6 9 8 7 1 0 6 7 1 7 5 2 7
##  [2245] 9 1 8 5 2 4 9 4 7 2 2 3 4 9 1 9 2 1 7 9 4 4 1 6 7 2 7 8 0 1 9 7 1 1
##  [2279] 7 5 3 3 5 1 3 7 6 1 3 8 7 5 9 6 0 0 2 8 8 2 3 7 1 3 0 3 4 4 3 8 9 2
##  [2313] 3 9 7 1 1 7 0 4 9 6 5 9 1 7 0 2 0 0 4 6 7 0 7 1 4 6 4 5 4 9 9 1 7 9
```

```
##  [2347] 5 3 3 8 2 3 6 2 2 1 1 1 1 1 6 9 8 4 3 7 1 6 4 8 0 4 7 4 2 4 0 7 0 1
##  [2381] 9 8 8 6 0 0 4 1 6 8 2 2 3 8 4 8 2 2 1 7 5 4 4 0 4 3 1 7 3 1 0 1 2 5
##  [2415] 4 2 1 0 1 8 9 1 4 8 3 8 9 3 6 2 8 3 2 2 1 0 4 2 9 2 4 3 7 9 1 5 2 9
##  [2449] 9 0 3 8 5 3 6 0 9 4 6 2 5 0 2 7 4 6 6 8 6 6 8 6 9 1 7 2 5 9 9 0 7 2
##  [2483] 7 6 7 0 6 5 4 4 7 2 0 9 9 2 2 9 4 4 2 3 3 2 1 7 0 7 6 4 1 3 8 7 4 5
##  [2517] 9 2 5 1 8 7 3 7 1 5 5 0 9 1 4 0 6 3 3 6 0 4 9 7 5 1 6 8 9 5 5 7 9 3
##  [2551] 8 3 8 1 5 3 5 0 5 5 3 8 6 7 7 7 3 7 0 5 9 0 2 5 5 3 1 7 7 8 6 5 5 3
##  [2585] 8 9 5 3 7 9 1 7 0 0 3 7 2 3 8 1 8 6 2 9 5 7 5 1 8 6 2 5 1 4 8 4 5 8
##  [2619] 3 0 6 2 7 3 3 2 1 0 7 3 4 0 3 9 3 2 8 9 0 3 8 0 7 6 5 4 7 3 0 0 8 6
##  [2653] 2 5 1 1 0 0 4 4 0 1 2 3 2 7 7 8 5 2 5 7 6 9 1 4 1 6 4 2 4 3 5 4 3 9
##  [2687] 5 0 1 5 3 8 9 1 9 7 9 5 5 2 7 4 6 0 1 1 1 0 4 4 7 6 3 0 0 4 3 0 6 1
##  [2721] 4 6 1 3 8 1 2 5 6 2 7 3 6 0 1 9 7 6 6 8 9 2 9 8 8 3 1 0 0 7 6 6 2 1
##  [2755] 6 9 3 1 8 6 4 0 6 0 0 0 6 3 5 9 3 4 5 5 8 5 3 0 4 0 2 9 6 8 2 3 1 2
##  [2789] 1 1 5 6 9 8 0 6 6 5 5 3 8 6 2 1 4 5 4 3 7 8 5 0 9 3 5 1 1 0 4 4 7 0
##  [2823] 1 7 0 1 6 1 4 5 6 6 5 7 8 4 4 7 2 5 3 7 0 7 7 9 6 4 2 8 5 7 8 3 9 5
##  [2857] 8 9 9 8 6 2 8 4 2 3 6 1 1 8 9 3 4 0 7 9 6 4 1 4 1 3 4 9 3 1 4 7 7 4
##  [2891] 7 2 9 3 0 8 0 8 4 0 4 4 1 5 2 8 3 4 9 5 2 8 1 5 3 7 9 4 2 5 6 2 5 9
##  [2925] 3 5 9 2 1 9 5 3 0 6 9 8 4 0 4 7 2 9 0 1 0 3 1 6 5 8 1 5 3 5 0 3 5 5
##  [2959] 9 2 8 7 0 4 9 1 9 7 7 5 3 2 0 9 1 8 6 2 3 9 6 2 1 9 1 3 5 5 0 3 8 3
##  [2993] 3 7 6 6 0 1 4 0 6 9 8 1 2 1 9 5 9 7 3 7 8 0 1 3 0 4 6 1 0 2 5 8 4 4
##  [3027] 1 1 5 4 6 6 0 6 9 2 6 2 7 1 7 9 4 0 0 3 8 2 2 3 1 6 0 5 7 7 9 2 6 7
##  [3061] 9 7 8 6 8 8 4 6 8 4 1 2 8 2 3 9 4 0 3 7 3 2 3 3 7 3 4 0 6 2 0 8 1 5
##  [3095] 3 5 4 1 7 1 5 7 5 7 3 2 2 7 3 7 3 7 8 5 4 5 2 9 6 5 3 6 7 4 1 7 1 5
##  [3129] 2 3 6 3 1 4 2 6 7 4 3 8 0 6 2 1 6 5 3 9 1 9 3 2 1 8 4 4 6 5 8 6 9 7
##  [3163] 7 8 6 9 7 3 9 4 0 5 4 6 4 1 2 3 0 0 2 6 6 5 7 0 8 6 4 7 9 0 7 3 4 2
##  [3197] 1 8 8 5 9 2 7 1 8 8 8 2 7 6 0 1 2 7 1 0 8 3 6 0 5 3 6 2 8 7 0 1 4 2
##  [3231] 1 1 4 4 4 4 7 1 6 2 9 9 0 0 1 8 8 4 3 4 2 0 6 1 6 1 2 2 2 1 2 3 7 8
##  [3265] 1 0 0 2 1 6 6 0 1 6 2 5 1 7 4 8 2 1 4 3 8 3 9 9 4 9 3 4 7 2 7 5 7 0
##  [3299] 4 3 3 2 6 7 6 0 0 6 7 7 0 5 5 8 1 0 7 0 2 8 1 5 0 8 8 0 3 2 7 7 8 6
##  [3333] 4 7 5 5 7 2 9 2 8 4 6 8 6 5 0 0 8 7 6 1 7 1 1 2 7 4 0 0 7 7 6 3 8 6
##  [3367] 4 2 0 9 4 0 5 7 8 2 7 4 7 1 1 3 6 6 2 9 1 9 4 8 3 6 9 5 9 6 2 4 6 7
##  [3401] 7 0 6 6 9 4 8 3 5 3 4 9 0 0 5 2 5 0 7 1 1 1 6 7 6 7 9 6 6 4 1 4 3 1
##  [3435] 1 2 2 4 1 0 8 7 6 3 4 0 0 6 3 3 0 7 1 7 1 1 3 1 0 9 9 7 5 4 1 4 8 9
##  [3469] 5 3 5 1 9 8 2 7 3 9 9 0 1 0 2 9 3 9 3 3 6 2 4 9 8 3 7 4 0 4 7 8 4 9
##  [3503] 8 1 9 7 5 9 2 8 2 2 0 2 2 3 8 4 6 8 4 8 2 4 6 7 9 3 3 9 4 3 1 9 4 7
##  [3537] 0 5 9 6 0 4 4 4 4 6 1 2 3 2 6 4 5 9 6 8 5 6 0 5 6 4 1 8 6 5 2 8 4 5
##  [3571] 5 4 7 7 0 7 8 2 2 3 7 0 1 8 0 7 1 9 8 7 5 5 9 1 7 5 4 3 1 2 2 1 6 6
##  [3605] 7 1 1 4 0 7 4 2 4 0 6 4 7 6 9 5 3 4 6 5 0 1 8 8 2 8 3 5 7 8 0 8 5 7
##  [3639] 1 1 0 1 3 7 8 5 0 7 1 1 0 1 1 4 5 2 7 6 2 3 0 2 8 5 9 6 9 7 2 1 3 6
##  [3673] 4 1 8 2 4 0 5 1 0 2 2 6 4 4 3 9 6 1 6 5 7 9 2 0 2 6 0 1 4 3 5 2 8 8
##  [3707] 0 8 8 9 0 9 6 7 6 3 9 3 4 7 7 7 4 9 0 6 4 4 4 2 7 2 8 1 0 0 7 8 3 3
##  [3741] 3 1 3 7 6 1 3 1 6 6 5 7 4 7 5 9 5 8 4 9 9 1 6 5 0 1 3 7 0 3 4 8 2 2
##  [3775] 0 2 8 1 5 1 6 8 8 9 1 2 1 3 5 1 0 9 4 4 8 3 2 5 9 7 6 6 2 0 0 0 5 8
##  [3809] 8 1 5 2 3 8 5 1 8 2 0 4 9 9 6 2 3 3 5 6 4 8 0 9 2 8 3 6 7 5 1 2 9 4
##  [3843] 9 1 2 8 6 0 7 0 9 1 1 5 7 5 9 9 1 9 5 9 2 5 0 4 1 0 8 9 0 8 9 8 9 4
##  [3877] 2 5 7 9 8 9 8 0 9 9 6 8 9 9 5 9 8 6 1 0 3 3 5 2 1 6 5 0 2 8 2 5 6 2
##  [3911] 3 0 2 2 6 4 3 5 5 1 7 2 1 6 9 1 9 9 5 5 1 6 2 2 8 6 7 1 4 6 0 6 0 5
##  [3945] 3 2 2 3 6 8 9 8 5 3 8 5 4 5 2 0 5 6 3 2 8 3 9 9 5 7 9 4 6 7 1 3 7 3
##  [3979] 6 6 0 9 0 1 9 4 2 8 8 0 1 6 9 7 5 3 4 7 4 9 4 4 3 6 3 1 1 7 6 9 1 8
##  [4013] 4 1 1 9 9 4 3 6 8 1 6 0 4 1 3 7 7 4 9 5 1 0 0 1 1 6 2 1 9 8 4 0 3 6
##  [4047] 4 9 0 7 1 6 5 7 5 2 5 1 8 5 4 7 0 6 7 3 2 5 8 1 0 4 5 7 1 8 5 1 7 0
##  [4081] 0 6 0 7 3 1 8 3 9 7 0 0 8 9 5 9 8 3 2 7 2 9 7 2 1 1 3 7 5 3 1 9 8 2
##  [4115] 2 2 8 8 5 7 3 8 9 8 8 6 8 2 3 9 7 5 6 2 9 2 8 8 1 6 2 8 7 9 1 8 0 1
##  [4149] 7 2 0 7 5 1 9 0 2 0 9 8 6 2 3 7 3 8 0 2 1 1 1 1 4 2 9 7 7 5 1 1 2 1
```

```
##  [4183] 9 9 9 1 0 2 0 2 1 1 4 6 4 1 5 4 9 7 7 7 5 6 2 6 2 8 0 6 9 6 1 9 7 7
##  [4217] 1 4 8 5 3 4 3 4 7 7 5 0 7 4 8 8 1 5 3 9 5 9 7 6 9 0 3 6 3 9 8 2 8 1
##  [4251] 2 8 6 8 5 5 3 9 4 9 2 5 1 5 1 1 4 1 4 4 3 5 9 1 2 2 3 3 0 2 9 0 0 9
##  [4285] 9 6 0 9 3 7 8 4 1 9 7 7 2 7 9 9 5 9 5 1 1 8 3 5 1 9 5 3 5 4 9 5 9 3
##  [4319] 1 9 0 9 7 5 4 9 2 0 1 0 5 1 4 9 3 3 6 1 5 2 5 2 2 0 9 2 6 6 0 1 2 0
##  [4353] 3 0 2 5 5 7 9 3 3 0 8 9 5 0 3 2 5 4 0 8 8 4 5 8 8 4 5 4 8 5 4 9 2 2
##  [4387] 1 2 6 8 8 7 0 3 6 6 4 3 8 8 7 2 2 0 0 9 3 9 9 1 9 8 6 6 4 2 6 9 2 8
##  [4421] 5 4 5 7 9 9 9 2 1 8 3 4 0 7 8 7 9 2 4 6 5 6 2 3 9 2 6 0 0 6 1 2 8 7
##  [4455] 9 8 2 0 4 7 7 5 0 5 6 4 6 7 4 3 0 7 5 0 7 4 2 0 8 9 9 4 2 4 6 7 8 7
##  [4489] 6 9 4 1 3 7 3 0 8 7 7 6 1 3 9 2 2 9 2 1 8 3 2 9 6 8 4 0 1 2 8 4 5 2
##  [4523] 7 8 1 1 3 0 3 5 7 0 3 1 9 3 5 3 1 7 7 3 0 8 4 8 2 6 6 2 9 7 3 9 0 9
##  [4557] 9 6 4 2 9 7 2 1 1 6 7 4 7 5 9 6 8 2 1 4 4 5 7 6 1 3 2 5 9 9 3 6 1 1
##  [4591] 4 6 9 7 2 1 5 1 4 6 3 8 1 1 0 3 1 6 8 4 9 0 7 3 0 4 9 0 6 6 6 3 6 7
##  [4625] 7 2 8 6 0 8 3 0 2 9 8 3 2 5 3 8 8 0 0 1 9 5 1 3 9 6 0 1 4 1 7 1 2 3
##  [4659] 7 9 7 4 9 9 3 9 2 8 2 7 1 8 0 9 1 0 1 7 7 9 6 9 9 9 2 1 6 1 3 5 7 1
##  [4693] 9 7 6 4 5 7 6 6 9 9 6 3 6 2 9 8 1 2 2 5 5 2 3 7 2 1 0 1 0 4 5 2 8 2
##  [4727] 8 3 5 1 7 7 1 1 2 9 7 8 4 0 5 0 7 8 8 4 7 7 8 5 8 4 9 8 1 3 8 0 3 1
##  [4761] 7 1 5 6 1 6 5 7 4 9 3 5 4 7 1 2 0 8 1 6 0 7 3 4 7 3 9 6 0 8 6 4 8 7
##  [4795] 7 9 3 8 6 9 7 2 3 4 0 2 1 3 3 5 5 7 2 4 6 7 2 8 3 0 8 7 8 4 0 8 4 4
##  [4829] 5 8 5 6 6 3 0 9 3 7 6 8 9 3 4 9 5 8 9 1 2 8 8 6 8 1 3 7 9 0 1 1 4 7
##  [4863] 0 8 1 7 4 5 7 1 2 1 1 3 9 6 2 1 2 6 8 7 6 6 9 3 7 0 5 2 8 0 5 4 3 8
##  [4897] 4 6 6 2 7 9 5 1 3 2 4 3 6 1 9 4 4 7 6 5 4 1 9 9 2 7 8 0 1 3 6 1 3 4
##  [4931] 1 1 1 5 6 0 7 0 7 2 3 2 5 2 2 9 4 9 8 1 2 1 6 1 2 7 8 0 0 0 8 2 2 9
##  [4965] 2 2 1 9 9 2 7 5 1 3 4 9 4 1 8 5 6 2 8 3 1 2 8 4 9 9 2 7 0 7 7 2 3 2
##  [4999] 4 0 3 9 9 8 4 1 0 6 0 9 6 8 6 1 1 9 8 9 2 3 5 5 9 4 2 1 9 4 3 9 6 0
##  [5033] 4 0 6 0 1 2 3 4 7 8 9 0 1 2 3 4 7 8 9 0 1 2 3 4 5 6 7 8 9 8 3 4 7 8
##  [5067] 6 3 1 0 9 7 1 9 3 8 4 7 3 0 9 1 4 5 4 6 2 0 6 2 1 1 1 1 7 2 4 7 5 2
##  [5101] 9 4 5 8 4 2 9 7 0 0 7 5 1 1 7 6 6 6 8 2 2 7 7 4 0 2 4 2 1 8 9 6 1 0
##  [5135] 5 9 6 9 8 0 3 0 8 3 9 6 3 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 8 9 0 1 2
##  [5169] 3 4 5 6 7 8 5 4 8 7 4 7 7 3 9 8 8 3 1 5 8 2 7 4 2 1 5 4 5 5 8 6 4 4
##  [5203] 4 1 8 7 5 5 1 8 9 1 3 6 3 3 2 2 6 9 9 6 5 5 3 3 8 1 6 5 6 8 1 9 7 6
##  [5237] 8 3 7 4 7 0 9 0 0 3 7 9 3 0 2 0 1 0 1 0 4 0 1 0 4 7 9 6 2 6 2 2 9 9
##  [5271] 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 8 0 5 6
##  [5305] 6 0 8 0 2 3 7 9 4 7 1 9 1 7 1 4 0 0 4 1 7 5 7 1 3 3 3 6 6 9 7 4 3 0
##  [5339] 2 5 2 6 0 8 9 4 3 5 4 8 1 5 9 0 6 4 3 6 3 3 8 1 4 7 5 7 2 2 0 0 1 7
##  [5373] 7 9 5 9 8 9 6 8 8 2 3 6 1 2 9 8 9 5 2 6 2 4 8 4 6 5 0 1 5 6 7 8 9 0
##  [5407] 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 7 4 2 0 9 0 1 5 8 8 0 2 7 8 4
##  [5441] 4 6 1 0 4 5 3 9 4 2 0 5 0 1 3 2 9 1 6 0 1 1 8 0 4 7 7 6 3 6 0 7 3 5
##  [5475] 4 2 4 1 8 3 5 6 7 0 6 7 1 2 5 8 1 9 3 8 2 8 7 6 7 1 4 6 2 9 3 0 1 2
##  [5509] 3 4 5 6 7 0 1 2 3 4 5 0 1 2 8 9 1 4 0 9 5 0 8 0 7 7 1 1 2 9 3 6 7 2
##  [5543] 3 8 1 2 9 8 8 7 1 7 1 1 0 3 4 2 6 4 7 4 2 7 4 9 1 0 6 8 5 5 5 3 5 9
##  [5577] 7 4 8 5 9 6 9 3 0 3 8 9 1 8 1 6 0 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
##  [5611] 7 8 9 0 1 2 3 4 5 6 7 8 9 3 5 3 2 9 3 2 1 4 5 5 2 3 2 1 3 9 7 2 8 2
##  [5645] 8 9 1 8 8 7 8 1 0 0 7 7 8 7 5 0 6 1 5 7 4 6 1 2 5 0 7 9 9 0 3 8 2 4
##  [5679] 8 1 8 6 5 9 0 0 0 3 7 1 6 4 2 6 6 0 4 5 4 1 3 8 6 3 9 9 5 9 3 7 8 5
##  [5713] 6 4 7 6 2 2 0 9 4 0 1 2 3 4 5 6 7 8 9 0 1 2 7 5 6 0 1 2 3 4 5 6 8 7
##  [5747] 1 3 2 8 0 7 5 9 9 6 0 7 4 1 3 2 1 2 3 8 3 2 6 5 6 8 2 7 4 8 1 8 0 5
##  [5781] 3 9 4 1 9 2 1 9 6 7 9 0 4 6 1 7 3 8 7 2 9 6 5 8 3 9 0 5 7 1 6 1 0 9
##  [5815] 3 3 4 4 0 6 2 5 4 2 3 4 6 0 0 2 0 1 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 0
##  [5849] 1 2 3 4 5 6 7 8 9 8 7 1 3 7 5 2 8 0 7 5 9 9 0 9 1 1 5 8 8 6 3 2 1 8
##  [5883] 3 2 6 5 6 0 4 1 0 5 3 1 9 2 1 9 6 0 4 6 1 7 3 8 7 2 9 6 5 8 3 5 7 1
##  [5917] 6 1 0 9 6 2 3 4 2 3 4 4 6 0 0 2 0 1 2 3 9 3 6 7 8 9 0 1 2 3 4 5 6 7
##  [5951] 8 9 0 1 2 8 4 5 6 7 8 9 8 6 5 0 6 8 9 4 1 9 5 8 0 4 8 9 1 4 0 5 5 2
##  [5985] 1 5 4 0 7 6 0 1 7 0 6 8 9 5 1 7 9 8 6 0 8 1 7 7 1 3 2 3 1 4 2 0 0 7
```

```
##  [6019] 8 4 6 4 9 5 8 4 7 2 5 6 3 6 9 6 3 2 2 4 6 9 0 2 5 5 1 3 3 9 7 8 7 2
##  [6053] 2 5 7 9 8 2 1 3 1 3 0 1 2 8 4 5 6 7 8 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3
##  [6087] 4 5 6 7 8 9 1 2 6 5 3 0 7 0 4 1 4 3 6 7 2 3 1 2 1 2 9 6 0 1 3 0 2 7
##  [6121] 5 7 6 2 9 1 9 0 6 0 6 0 2 0 6 1 5 8 4 3 0 1 5 4 4 8 5 7 5 7 8 3 4 8
##  [6155] 8 5 2 9 7 1 3 8 1 0 7 5 9 6 3 4 7 7 9 9 3 4 4 3 8 6 2 0 1 2 3 4 5 6
##  [6189] 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 8 3 9 5 5 2 6 8 4 9
##  [6223] 1 7 1 2 3 5 9 6 9 1 1 1 2 9 5 6 8 1 2 0 7 7 5 8 2 9 8 9 0 4 6 7 1 3
##  [6257] 4 5 6 0 3 6 8 7 0 4 2 7 4 7 5 4 3 4 2 8 1 5 1 2 0 2 5 6 4 3 0 0 0 3
##  [6291] 3 5 7 0 6 4 8 8 6 3 4 6 9 9 8 2 7 7 1 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
##  [6325] 5 6 7 8 0 1 2 3 4 5 6 7 8 2 1 7 2 5 0 8 0 2 7 8 8 3 6 0 2 7 6 6 1 2
##  [6359] 8 8 7 7 4 7 7 3 7 4 5 4 3 3 8 4 1 1 9 7 4 3 7 3 3 0 2 5 5 6 6 3 5 2
##  [6393] 5 9 9 8 4 1 0 6 0 9 6 8 8 5 6 1 1 9 8 9 2 3 5 5 9 4 2 1 9 2 9 2 0 6
##  [6427] 0 4 0 0 1 2 3 4 7 8 9 0 1 2 3 7 8 9 0 1 2 3 4 7 8 9 7 3 0 3 1 8 7 6
##  [6461] 4 0 2 6 8 3 2 8 1 2 0 7 1 0 4 4 5 8 0 6 2 3 1 5 1 8 5 9 4 0 7 5 8 8
##  [6495] 3 8 9 2 6 2 5 3 1 7 3 9 1 9 9 6 0 3 9 2 8 1 4 3 5 2 9 2 5 8 9 5 0 1
##  [6529] 2 4 5 6 0 1 2 3 4 5 6 7 1 2 3 4 5 1 0 4 5 6 6 3 4 4 2 9 1 0 6 4 9 7
##  [6563] 2 3 3 9 2 0 9 3 3 7 1 5 6 3 1 2 8 4 0 2 4 0 2 4 7 8 0 7 0 6 9 3 2 8
##  [6597] 6 7 6 7 5 1 0 8 1 6 7 2 9 7 9 5 8 6 2 6 2 8 1 7 5 0 1 1 3 4 4 9 1 8
##  [6631] 6 8 9 0 1 2 3 4 5 6 7 5 9 0 1 2 3 4 7 8 9 5 1 7 8 9 9 8 9 8 4 1 7 7
##  [6665] 3 3 7 6 6 6 1 9 0 1 7 6 3 2 1 7 1 3 9 1 7 6 8 4 1 4 3 6 9 6 1 4 4 7
##  [6699] 2 4 4 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 9 0 1 2 3 4 7 8 1 3 5 1 7 7
##  [6733] 2 1 4 8 3 4 4 3 9 7 4 1 2 3 5 9 1 6 0 1 0 0 2 7 7 1 1 4 0 4 7 3 6 8
##  [6767] 0 3 7 4 0 6 9 2 6 5 8 6 9 0 4 0 6 6 9 2 0 9 5 1 3 7 6 9 3 0 2 2 0 1
##  [6801] 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 2 1 7 2 5 0
##  [6835] 8 0 2 7 8 8 3 0 6 0 2 7 6 4 1 2 8 8 7 7 4 7 7 3 7 4 5 4 3 3 8 4 5 4
##  [6869] 1 1 9 7 4 3 7 3 3 0 2 5 5 6 3 1 5 2 5 9 9 8 4 1 0 6 0 9 6 8 8 5 6 1
##  [6903] 1 9 8 9 2 3 5 5 9 4 2 1 9 4 9 1 3 9 2 0 6 0 4 0 6 0 1 2 3 4 5 6 7 8
##  [6937] 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 3 8 0 7 1 0 7 5 5 6 9 0 1
##  [6971] 0 0 8 3 4 3 1 5 0 0 9 5 3 4 9 3 7 6 9 2 4 5 7 2 6 4 9 4 9 4 1 2 2 5
##  [7005] 8 1 3 2 9 4 3 8 2 2 1 2 8 6 5 1 6 7 2 1 3 9 3 8 7 5 7 0 7 4 8 8 5 0
##  [7039] 6 6 3 7 6 9 9 4 8 4 1 0 6 6 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
##  [7073] 0 1 2 3 4 5 6 7 8 9 7 4 0 4 0 1 7 9 5 1 4 2 8 9 4 3 7 8 2 4 4 3 3 6
##  [7107] 9 9 5 8 6 7 0 6 8 2 6 3 9 3 2 8 6 1 7 4 8 8 9 0 3 3 9 0 5 2 9 4 1 0
##  [7141] 3 7 5 8 7 7 8 2 9 7 1 2 6 4 2 5 2 3 6 6 5 0 0 2 8 1 6 1 0 4 3 1 6 1
##  [7175] 9 0 1 4 5 6 7 8 9 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 8 9 8 4 0 0 7 2 4 3
##  [7209] 8 6 6 3 2 6 3 3 6 1 4 7 8 0 3 1 9 0 1 9 1 2 7 0 1 3 8 2 9 2 7 6 5 5
##  [7243] 9 9 8 2 9 1 3 2 3 4 3 1 9 0 9 3 6 8 7 0 1 0 5 8 2 7 7 0 1 2 3 4 5 6
##  [7277] 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 1 7 4 8 1 5 6 5 7 2 8
##  [7311] 6 3 3 8 6 5 4 0 9 1 7 2 9 1 5 1 3 2 2 3 0 6 4 3 7 6 9 0 4 8 1 4 0 6
##  [7345] 1 2 6 9 2 2 3 5 5 1 0 7 7 9 6 2 9 4 7 0 2 3 4 0 0 8 8 8 5 1 3 7 4 9
##  [7379] 8 8 9 0 9 8 9 0 2 6 5 6 7 4 7 5 4 1 3 5 3 1 2 3 4 5 6 1 2 3 4 6 0 1
##  [7413] 2 4 5 6 7 8 1 7 2 4 1 4 1 4 9 6 8 4 5 3 7 8 8 3 3 5 6 7 0 6 1 6 8 7
##  [7447] 0 1 5 0 8 5 0 1 5 8 4 2 3 9 7 6 9 1 9 0 6 7 1 2 3 9 2 4 5 5 3 7 5 3
##  [7481] 1 8 2 2 3 0 2 9 4 9 7 0 2 7 4 9 9 2 5 9 8 3 8 6 7 0 0 1 2 3 4 5 6 7
##  [7515] 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 0 7 2 6 5 5 3 7 8 6 6
##  [7549] 6 6 4 3 8 8 3 0 1 9 0 5 4 1 9 1 2 7 0 1 3 8 2 9 2 7 4 2 6 5 5 9 9 1
##  [7583] 1 5 7 6 8 2 9 4 3 1 9 0 9 3 6 8 7 0 1 0 5 8 2 7 7 0 1 2 3 4 5 6 7 8
##  [7617] 9 0 1 2 3 4 5 8 9 0 1 2 3 4 5 6 7 8 9 2 1 2 1 3 9 9 8 5 3 7 0 7 7 5
##  [7651] 7 9 9 4 7 0 3 4 1 5 8 1 4 8 4 1 8 6 6 4 6 0 5 5 3 3 5 7 2 5 9 6 9 2
##  [7685] 6 2 1 2 0 8 3 8 3 0 8 7 4 9 5 0 9 7 0 0 4 6 0 9 1 6 2 7 6 8 3 5 2 1
##  [7719] 8 3 8 6 1 0 2 1 4 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4
##  [7753] 5 6 7 8 9 7 6 4 7 6 2 3 4 8 7 8 6 9 8 3 2 2 8 4 8 5 6 5 0 2 0 1 1 2
##  [7787] 9 6 8 2 1 0 6 5 2 9 7 5 3 9 2 7 1 8 3 8 1 9 5 5 0 1 1 9 8 2 6 0 4 5
##  [7821] 0 2 1 8 6 7 5 9 9 3 0 3 1 4 4 0 4 9 0 1 2 3 5 6 7 8 0 1 2 3 5 6 7 8
```

```
##  [7855] 9 0 1 2 3 5 6 7 8 9 9 7 0 9 0 1 5 8 8 0 9 3 2 7 8 4 6 1 0 4 9 4 2 0
##  [7889] 5 0 1 6 9 3 2 9 1 6 0 1 1 8 7 7 6 3 6 0 7 2 4 1 7 0 6 7 1 2 5 8 1 2
##  [7923] 2 8 7 6 8 7 1 6 2 9 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
##  [7957] 3 4 5 6 7 8 9 8 9 5 7 0 3 1 6 8 4 1 5 6 4 2 7 8 1 3 4 3 4 7 2 0 5 0
##  [7991] 1 9 2 3 2 3 5 5 7 8 4 9 9 7 1 1 9 0 7 8 3 4 8 6 3 8 0 9 6 2 1 0 1 0
##  [8025] 6 2 3 8 9 0 7 2 3 4 5 5 2 8 5 4 6 6 6 7 9 1 8 2 1 5 3 4 7 9 4 0 0 0
##  [8059] 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 9 0 1 3 1 5 1 2
##  [8093] 4 9 8 4 6 8 0 1 1 9 2 6 6 8 7 4 2 9 7 0 2 1 0 3 6 0 1 2 3 4 5 6 7 8
##  [8127] 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 8 6 5 9 7 0 2 3 4 3 8 5 1
##  [8161] 5 2 3 0 1 2 1 3 2 6 5 3 0 7 2 7 4 6 4 0 5 9 9 8 9 5 3 1 7 4 7 6 5 4
##  [8195] 0 0 6 6 2 0 6 3 7 7 4 4 3 9 2 8 9 6 0 9 5 3 8 8 7 1 4 0 4 8 5 2 3 9
##  [8229] 0 1 9 1 5 1 7 4 8 6 2 1 6 8 8 0 1 2 9 4 7 8 9 0 1 2 3 4 6 7 8 9 0 1
##  [8263] 2 3 4 7 8 9 1 4 5 3 3 0 9 5 4 9 0 8 4 6 7 0 7 7 1 6 9 1 3 6 2 3 8 2
##  [8297] 3 8 9 5 8 8 7 1 7 1 1 0 3 4 2 6 4 7 4 2 7 4 2 9 2 7 9 2 1 6 6 5 3 4
##  [8331] 8 5 9 6 9 0 6 3 0 8 1 6 0 0 1 2 3 4 5 6 7 0 1 2 3 4 7 8 9 0 1 2 3 4
##  [8365] 7 2 5 1 6 4 3 9 9 0 9 7 1 6 4 3 6 2 0 9 8 6 5 7 0 0 1 7 4 3 2 4 1 3
##  [8399] 7 6 4 7 7 7 9 8 4 3 6 2 8 3 5 8 0 5 4 7 1 3 1 7 9 6 2 0 9 1 7 3 3 9
##  [8433] 1 6 4 3 9 8 2 1 8 6 4 1 5 5 6 5 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
##  [8467] 8 9 0 1 2 3 4 5 6 7 8 9 6 9 7 0 2 3 4 3 8 5 1 3 0 1 2 1 3 2 0 7 2 6
##  [8501] 4 0 5 9 9 8 9 5 3 1 7 4 7 0 0 6 6 6 3 7 4 2 8 9 8 7 1 9 0 4 8 5 2 3
##  [8535] 9 0 1 9 1 5 1 7 6 1 2 1 6 8 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 0
##  [8569] 1 2 3 5 6 7 8 1 0 4 5 6 6 3 4 4 2 8 1 0 6 4 9 7 2 9 2 0 9 3 3 9 1 5
##  [8603] 2 3 1 6 7 3 7 8 4 0 2 4 0 2 4 7 8 0 7 0 6 9 3 2 4 8 6 0 5 7 5 1 0 8
##  [8637] 1 6 7 2 9 7 9 5 6 5 2 6 2 8 1 7 5 5 7 3 5 0 1 1 3 8 4 9 4 5 1 8 6 8
##  [8671] 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 3 5 3
##  [8705] 2 9 3 2 1 4 5 5 2 3 2 1 3 9 7 2 1 2 8 9 1 8 8 7 8 1 0 0 6 7 7 8 7 5
##  [8739] 0 6 1 5 7 4 6 1 2 5 0 7 9 9 0 3 4 4 8 4 1 8 6 5 9 0 0 0 3 7 1 6 4 6
##  [8773] 0 4 5 4 1 3 8 6 3 9 9 5 9 3 7 8 5 6 4 7 6 2 2 0 9 4 0 1 2 3 4 5 6 7
##  [8807] 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 6 4 2 6 4 7 5 5 4 7 2 9
##  [8841] 3 9 3 8 2 0 9 5 6 0 1 0 6 5 3 5 3 8 0 0 3 4 1 5 3 0 8 3 0 6 2 7 8 1
##  [8875] 7 1 3 8 5 4 2 0 9 7 6 7 4 1 6 2 6 7 1 9 8 0 6 9 4 9 9 6 2 3 7 1 9 2
##  [8909] 2 5 3 7 8 0 1 2 3 4 7 8 9 0 1 2 3 4 7 8 9 0 1 7 8 9 8 9 2 6 1 3 5 4
##  [8943] 8 2 6 4 3 4 5 9 2 0 3 9 4 9 7 3 8 7 4 4 9 8 5 8 2 6 6 2 3 1 3 2 7 3
##  [8977] 1 9 0 1 1 3 5 0 7 8 1 5 1 4 6 0 0 4 9 1 6 6 9 0 7 6 1 1 0 1 2 3 4 2
##  [9011] 2 3 4 5 6 2 0 1 2 2 8 6 3 9 2 1 9 3 9 6 1 7 2 4 4 5 7 0 0 1 6 6 8 2
##  [9045] 7 7 2 4 2 1 6 1 0 6 9 8 3 9 6 3 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7
##  [9079] 8 9 0 1 2 3 4 5 6 7 8 9 1 6 8 9 9 0 1 2 4 4 3 7 4 4 4 0 3 8 7 5 8 2
##  [9113] 1 7 5 3 8 5 2 5 1 1 6 2 1 3 8 6 4 2 6 2 5 5 0 2 8 0 6 8 1 7 9 1 9 2
##  [9147] 6 7 6 6 8 7 4 9 2 1 3 3 0 5 5 8 0 3 7 9 7 0 2 7 9 1 7 8 0 3 5 3 6 0
##  [9181] 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 7 8 9 6 4 2 6 4 7 8
##  [9215] 9 2 9 3 9 3 0 0 1 0 4 2 6 3 5 3 0 3 4 1 5 3 0 8 3 0 6 1 7 8 0 9 2 6
##  [9249] 7 1 9 6 9 4 9 9 6 7 1 2 5 3 7 8 0 1 2 4 5 6 7 8 9 0 1 3 4 5 6 7 8 0
##  [9283] 1 3 4 7 8 9 7 5 5 1 9 9 7 1 0 0 5 9 7 1 7 2 2 3 6 8 3 2 0 0 6 1 7 5
##  [9317] 8 6 2 9 4 8 8 7 1 0 8 7 7 5 8 5 3 4 6 1 1 5 5 0 7 2 3 6 4 1 2 4 1 5
##  [9351] 4 2 0 4 8 6 1 9 0 2 5 6 9 3 6 3 6 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
##  [9385] 7 8 9 0 1 2 3 5 6 7 8 1 0 9 5 7 5 1 8 6 9 0 4 1 9 3 8 4 4 7 0 1 9 2
##  [9419] 8 7 8 2 5 9 6 0 6 5 5 3 3 3 9 8 1 1 0 6 1 0 0 6 2 1 1 3 2 7 7 8 8 7
##  [9453] 8 4 6 0 2 0 7 0 3 6 8 7 1 5 9 9 3 7 2 4 9 4 3 6 2 2 5 3 2 5 5 9 4 1
##  [9487] 7 2 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 1 0
##  [9521] 1 2 7 5 3 4 4 0 0 6 9 6 6 5 7 2 3 4 4 9 1 4 0 7 9 5 7 2 3 1 4 4 0 9
##  [9555] 9 6 1 8 3 3 7 3 9 8 8 4 7 7 6 2 1 9 8 7 8 8 7 2 2 3 9 3 3 5 5 0 7 4
##  [9589] 5 6 5 1 4 1 1 2 8 2 6 1 5 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
##  [9623] 1 2 3 4 5 6 7 8 8 0 6 0 1 2 3 7 9 4 7 1 9 1 7 1 4 0 0 1 7 5 7 1 3 3
##  [9657] 3 1 6 9 7 1 3 0 7 6 0 8 9 4 3 5 4 8 1 5 9 0 6 3 3 8 1 4 7 5 2 0 0 1
```

```
##  [9691] 7 8 9 6 8 8 2 3 6 1 8 9 5 2 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
##  [9725] 0 1 2 3 4 6 6 7 8 9 7 4 6 1 4 0 9 9 3 7 8 0 7 5 8 5 3 2 2 0 5 5 6 0
##  [9759] 3 8 1 0 3 0 4 7 4 9 0 9 0 7 1 7 1 6 6 0 6 2 8 7 6 4 9 9 5 3 7 4 3 0
##  [9793] 1 6 6 1 1 3 2 1 0 0 1 2 3 4 7 8 4 0 1 2 3 4 5 6 7 8 0 1 2 3 4 7 8 9
##  [9827] 0 8 3 9 5 5 2 6 8 4 1 7 1 7 3 5 6 9 1 1 1 2 1 2 0 7 7 5 8 2 9 8 8 7
##  [9861] 3 4 6 8 7 0 4 2 7 7 5 4 3 4 2 8 1 5 1 0 2 3 3 5 7 0 6 8 6 3 9 9 8 2
##  [9895] 7 7 1 0 1 7 8 9 0 1 2 3 4 5 6 7 8 0 1 2 3 4 7 8 9 7 8 6 4 1 9 3 8 4
##  [9929] 4 7 0 1 9 2 8 7 8 2 6 0 6 5 3 3 3 9 1 4 0 6 1 0 0 6 2 1 1 7 7 8 4 6
##  [9963] 0 7 0 3 6 8 7 1 5 2 4 9 4 3 6 4 1 7 2 6 6 0 1 2 3 4 5 6 7 8 9 0 1 2
##  [9997] 3 4 5 6
```

Keras provides a vocabulary for building deep learning models that is simple, elegant, and intuitive. Building a question answering system, an image classification model, a neural Turing machine, or any other model is just as straightforward.

Tutorial: Basic Classification

In this guide, we will train a neural network model to classify images of clothing, like sneakers and shirts. It's fine if you don't understand all the details, this is a fast-paced overview of a complete Keras program with the details explained as we go.

```
library(keras)
```

Import the Fashion MNIST dataset

This guide uses the Fashion MNIST dataset which contains 70,000 grayscale images in 10 categories. The images show individual articles of clothing at low resolution (28 by 28 pixels), as seen here:

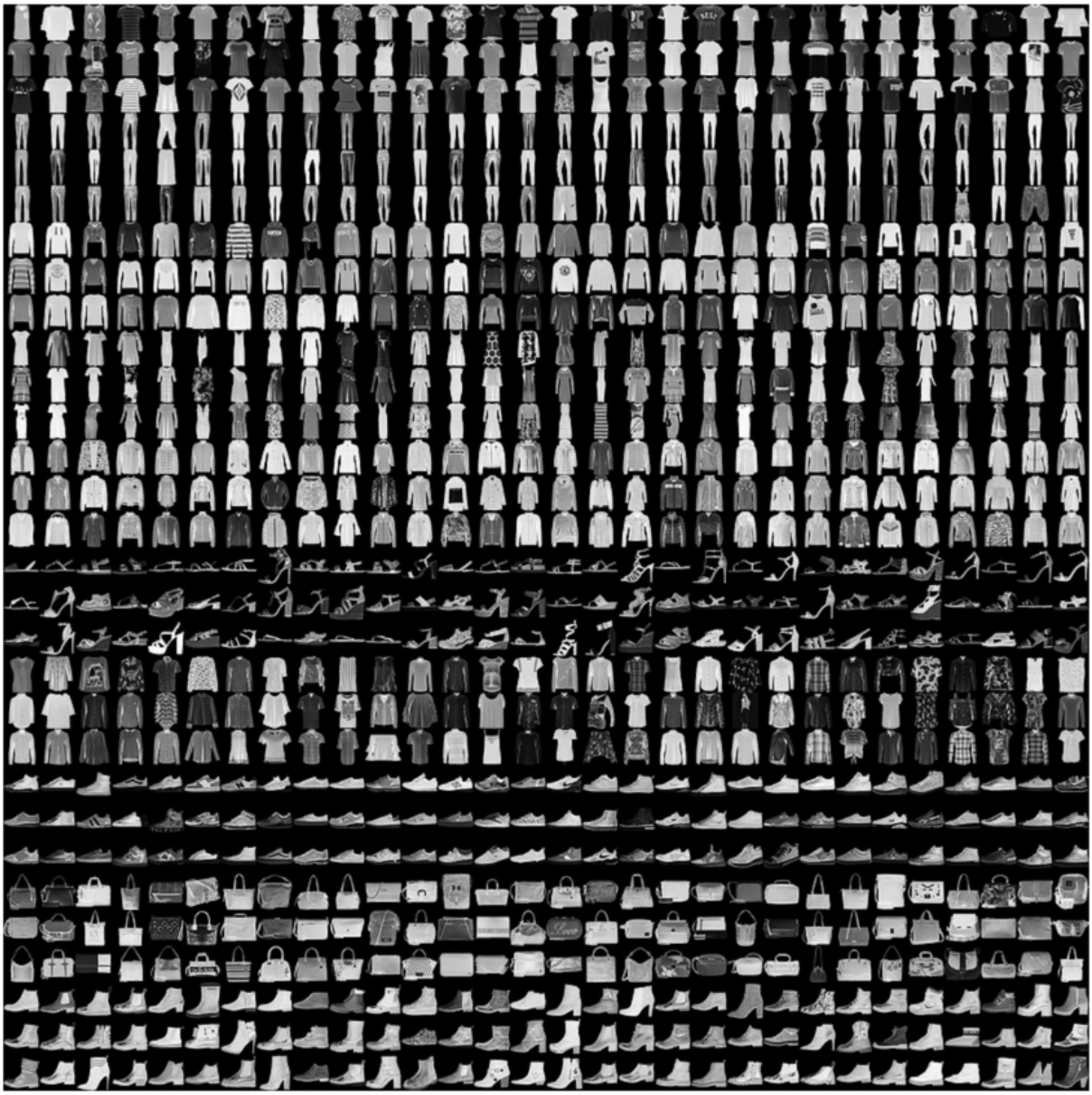


Figure 1. Fashion-MNIST samples (by Zalando, MIT License).

Fashion MNIST is intended as a drop-in replacement for the classic MNIST dataset—often used as the "Hello, World" of machine learning programs for computer vision. The MNIST dataset contains images of handwritten digits (0, 1, 2, etc) in an identical format to the articles of clothing we'll use here.

This guide uses Fashion MNIST for variety, and because it's a slightly more challenging problem than regular MNIST. Both datasets are relatively small and are used to verify that an algorithm works as expected. They're good starting points to test and debug code.

We will use 60,000 images to train the network and 10,000 images to evaluate how accurately the network learned to classify images. You can access the Fashion MNIST directly from Keras.

```
fashion_mnist <- dataset_fashion_mnist()

c(train_images, train_labels) %<-% fashion_mnist$train
c(test_images, test_labels) %<-% fashion_mnist$test
```

At this point we have four arrays: The train_images and train_labels arrays are the training set — the data the model uses to learn. The model is tested against the test set: the test_images, and test_labels arrays.

The images each are 28 x 28 arrays, with pixel values ranging between 0 and 255. The labels are arrays of integers, ranging from 0 to 9. These correspond to the class of clothing the image represents:

Each image is mapped to a single label. Since the class names are not included with the dataset, we'll store them in a vector to use later when plotting the images.

```
class_names = c('T-shirt/top',
                'Trouser',
                'Pullover',
                'Dress',
                'Coat',
                'Sandal',
                'Shirt',
                'Sneaker',
                'Bag',
                'Ankle boot')
```

Explore the data

Let's explore the format of the dataset before training the model. The following shows there are 60,000 images in the training set, with each image represented as 28 x 28 pixels:

```
dim(train_images)
```

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

Likewise, there are 60,000 labels in the training set:

```
dim(train_labels)
```

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

Each label is an integer between 0 and 9:

```
train_labels[1:20]
```

```
##  [1] 9 0 0 3 0 2 7 2 5 5 0 9 5 5 7 9 1 0 6 4
```

There are 10,000 images in the test set. Again, each image is represented as 28 x 28 pixels:

```
dim(test_images)
```

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

And the test set contains 10,000 images labels:

```
dim(test_labels)
```

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

Preprocess the data

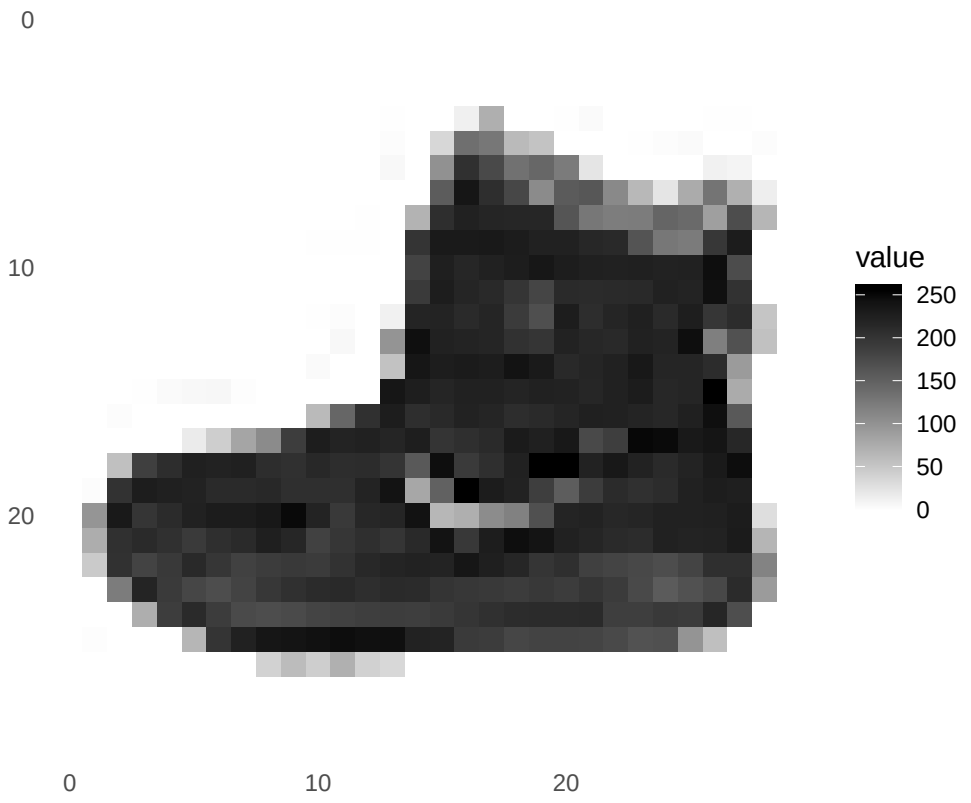
The data must be preprocessed before training the network. If you inspect the first image in the training set, you will see that the pixel values fall in the range of 0 to 255:

```
library(tidyr)
library(ggplot2)

image_1 <- as.data.frame(train_images[1, , ])
```

```
colnames(image_1) <- seq_len(ncol(image_1))
image_1$y <- seq_len(nrow(image_1))
image_1 <- gather(image_1, "x", "value", -y)
image_1$x <- as.integer(image_1$x)

ggplot(image_1, aes(x = x, y = y, fill = value)) +
  geom_tile() +
  scale_fill_gradient(low = "white", high = "black", na.value = NA) +
  scale_y_reverse() +
  theme_minimal() +
  theme(panel.grid = element_blank())   +
  theme(aspect.ratio = 1) +
  xlab("") +
  ylab("")
```



We scale these values to a range of 0 to 1 before feeding to the neural network model. For this, we simply divide by 255.

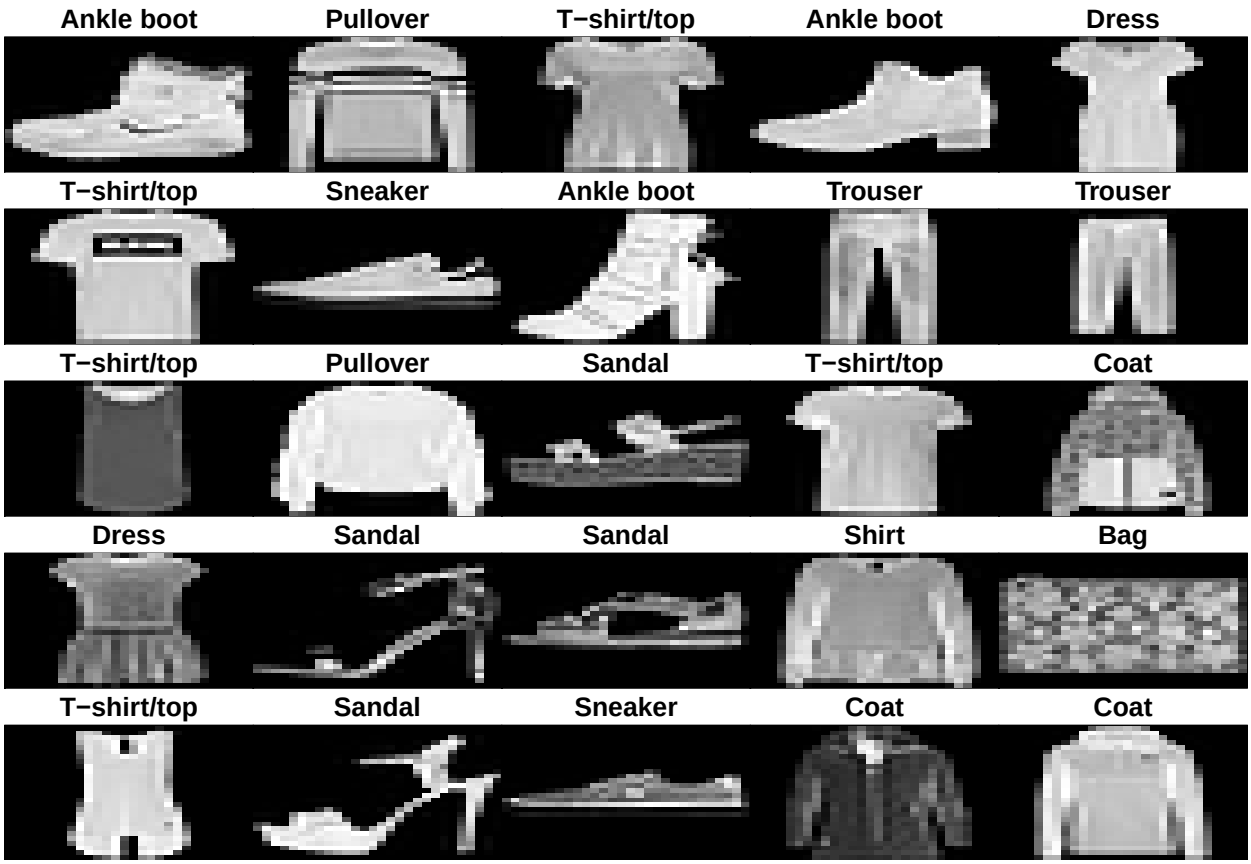
It's important that the training set and the testing set are preprocessed in the same way:

```
train_images <- train_images / 255
test_images <- test_images / 255
```

Display the first 25 images from the training set and display the class name below each image. Verify that the data is in the correct format and we're ready to build and train the network.

```
par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  img <- train_images[i, , ]
```

```
  img <- t(apply(img, 2, rev))
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste(class_names[train_labels[i] + 1]))
}
```



Build the model

Building the neural network requires configuring the layers of the model, then compiling the model.

Setup the layers

The basic building block of a neural network is the layer. Layers extract representations from the data fed into them. And, hopefully, these representations are more meaningful for the problem at hand.

Most of deep learning consists of chaining together simple layers. Most layers, like layer_dense, have parameters that are learned during training.

```
model <- keras_model_sequential()
model %>%
  layer_flatten(input_shape = c(28, 28)) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dense(units = 10, activation = 'softmax')
```

The first layer in this network, layer_flatten, transforms the format of the images from a 2d-array (of 28 by 28 pixels), to a 1d-array of 28 * 28 = 784 pixels. Think of this layer as unstacking rows of pixels in the image and lining them up. This layer has no parameters to learn; it only reformats the data.

After the pixels are flattened, the network consists of a sequence of two dense layers. These are densely-connected, or fully-connected, neural layers. The first dense layer has 128 nodes (or neurons). The second (and last) layer is a 10-node softmax layer —this returns an array of 10 probability scores that sum to 1. Each node contains a score that indicates the probability that the current image belongs to one of the 10 digit classes.

Compile the model

Before the model is ready for training, it needs a few more settings. These are added during the model's compile step:

Loss function — This measures how accurate the model is during training. We want to minimize this function to "steer" the model in the right direction. Optimizer — This is how the model is updated based on the data it sees and its loss function. Metrics —Used to monitor the training and testing steps. The following example uses accuracy, the fraction of the images that are correctly classified.

```
model %>% compile(
  optimizer = 'adam',
  loss = 'sparse_categorical_crossentropy',
  metrics = c('accuracy')
)
```

Train the model

Training the neural network model requires the following steps:

Feed the training data to the model — in this example, the train_images and train_labels arrays. The model learns to associate images and labels. We ask the model to make predictions about a test set — in this example, the test_images array. We verify that the predictions match the labels from the test_labels array. To start training, call the fit method — the model is "fit" to the training data:

```
model %>% fit(train_images, train_labels, epochs = 5)
```

As the model trains, the loss and accuracy metrics are displayed. This model reaches an accuracy of about 0.88 (or 88%) on the training data.

Evaluate accuracy

Next, compare how the model performs on the test dataset:

```
score <- model %>% evaluate(test_images, test_labels)

cat('Test loss:', score$loss, "\n")
```

```
## Test loss: 0.3485948
```

```
cat('Test accuracy:', score$acc, "\n")
```

```
## Test accuracy: 0.8745
```

It turns out, the accuracy on the test dataset is a little less than the accuracy on the training dataset. This gap between training accuracy and test accuracy is an example of overfitting. Overfitting is when a machine learning model performs worse on new data than on their training data.

Make predictions

With the model trained, we can use it to make predictions about some images.

```
predictions <- model %>% predict(test_images)
```

Here, the model has predicted the label for each image in the testing set. Let's take a look at the first prediction:

```
predictions[1, ]
```

```
##  [1] 2.747332e-08 8.823788e-11 9.931100e-10 7.578573e-11 1.527017e-08
##  [6] 2.992742e-02 1.287106e-08 6.675255e-03 1.280449e-07 9.633971e-01
```

A prediction is an array of 10 numbers. These describe the "confidence" of the model that the image corresponds to each of the 10 different articles of clothing. We can see which label has the highest confidence value:

```
which.max(predictions[1, ])
```

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

Alternatively, we can also directly get the class prediction:

```
class_pred <- model %>% predict_classes(test_images)
class_pred[1:20]
```

```
##  [1] 9 2 1 1 6 1 4 6 5 7 4 5 7 3 4 1 2 2 8 0
```

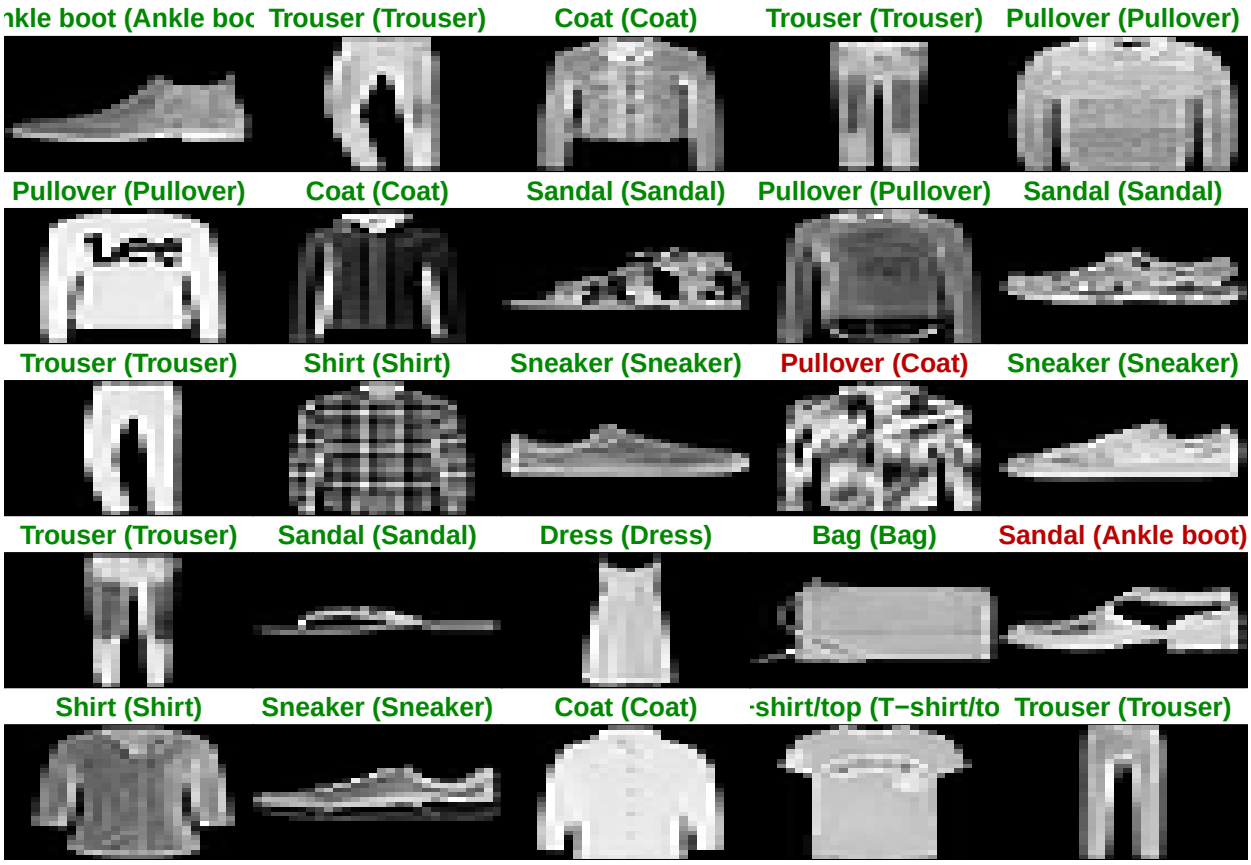
As the labels are 0-based, this actually means a predicted label of 9 (to be found in class_names[9]). So the model is most confident that this image is an ankle boot. And we can check the test label to see this is correct:

```
test_labels[1]
```

```
## [1] 9
```

Let's plot several images with their predictions. Correct prediction labels are green and incorrect prediction labels are red.

```
par(mfcol=c(5,5))
par(mar=c(0, 0, 1.5, 0), xaxs='i', yaxs='i')
for (i in 1:25) {
  img <- test_images[i, , ]
  img <- t(apply(img, 2, rev))
  # subtract 1 as labels go from 0 to 9
  predicted_label <- which.max(predictions[i, ]) - 1
  true_label <- test_labels[i]
  if (predicted_label == true_label) {
    color <- '#008800'
  } else {
    color <- '#bb0000'
  }
  image(1:28, 1:28, img, col = gray((0:255)/255), xaxt = 'n', yaxt = 'n',
        main = paste0(class_names[predicted_label + 1], " (",
                      class_names[true_label + 1], ")"),
        col.main = color)
}
```



Finally, use the trained model to make a prediction about a single image.

```
# Grab an image from the test dataset
# take care to keep the batch dimension, as this is expected by the model
img <- test_images[1, , , drop = FALSE]
dim(img)
```

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

Now predict the image:

```
predictions <- model %>% predict(img)
predictions
```

```
##              [,1]         [,2]         [,3]         [,4]         [,5]
## [1,] 2.747333e-08 8.823789e-11 9.931082e-10 7.578574e-11 1.527017e-08
##             [,6]         [,7]        [,8]         [,9]     [,10]
## [1,] 0.02992737 1.287109e-08 0.006675249 1.280447e-07 0.9633972
```

predict returns a list of lists, one for each image in the batch of data. Grab the predictions for our (only) image in the batch:

```
# subtract 1 as labels are 0-based
prediction <- predictions[1, ] - 1
which.max(prediction)
```

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

Or, directly getting the class prediction again:

```
class_pred <- model %>% predict_classes(img)
class_pred
```

```
## [1] 9
```

And, as before, the model predicts a label of 9.