STAT 5014 HW 9

David Edwards

11/18/2019

Example 1

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

```
# 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)
```

```
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')
```

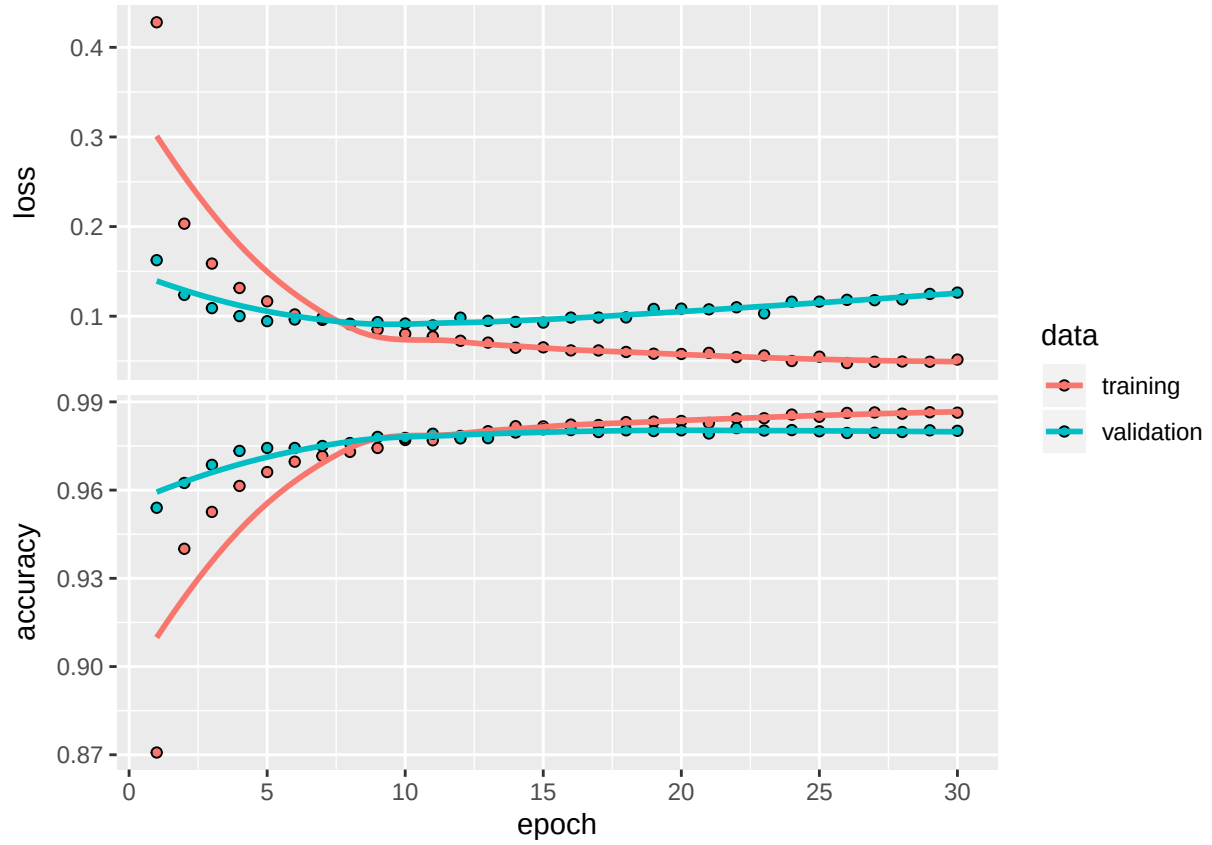
```
summary(model)
```

```
## 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
## _________________________________________________________________________
```

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

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

```
plot(history)
```




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

```
## $loss
## [1] 0.1094893
##
## $accuracy
## [1] 0.9808
```

Example 2

```
fashion_mnist <- dataset_fashion_mnist()
```

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

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

```
dim(train_images)
```

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

```
dim(train_labels)
```

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

```
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
```

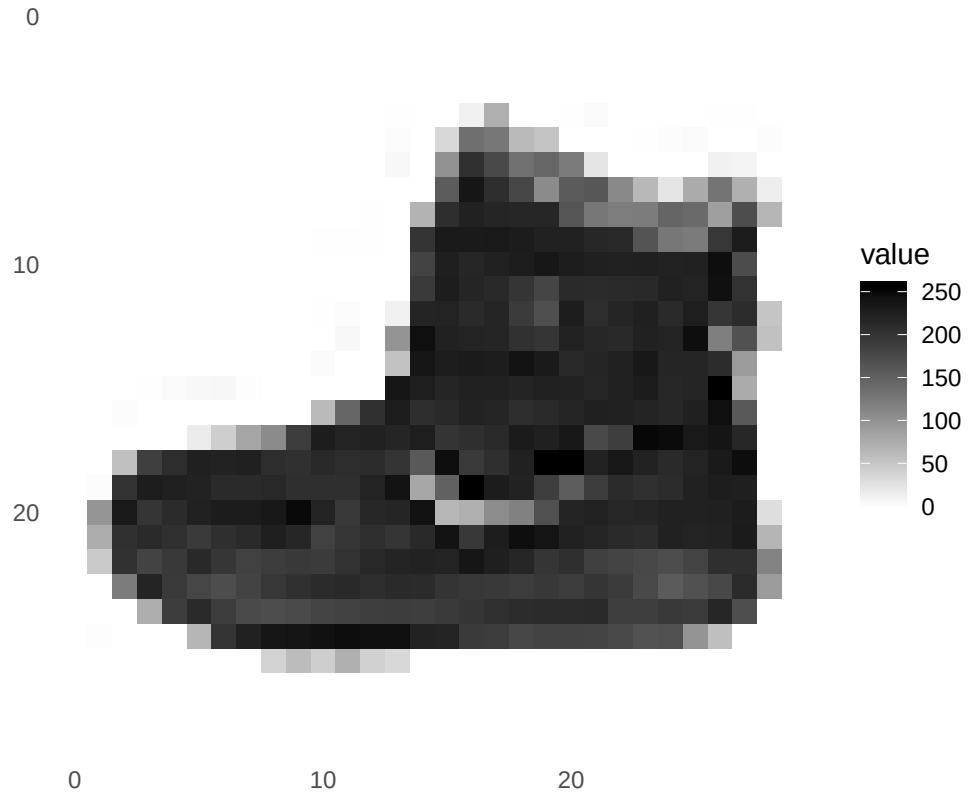
```
dim(test_images)
```

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

```
dim(test_labels)
```

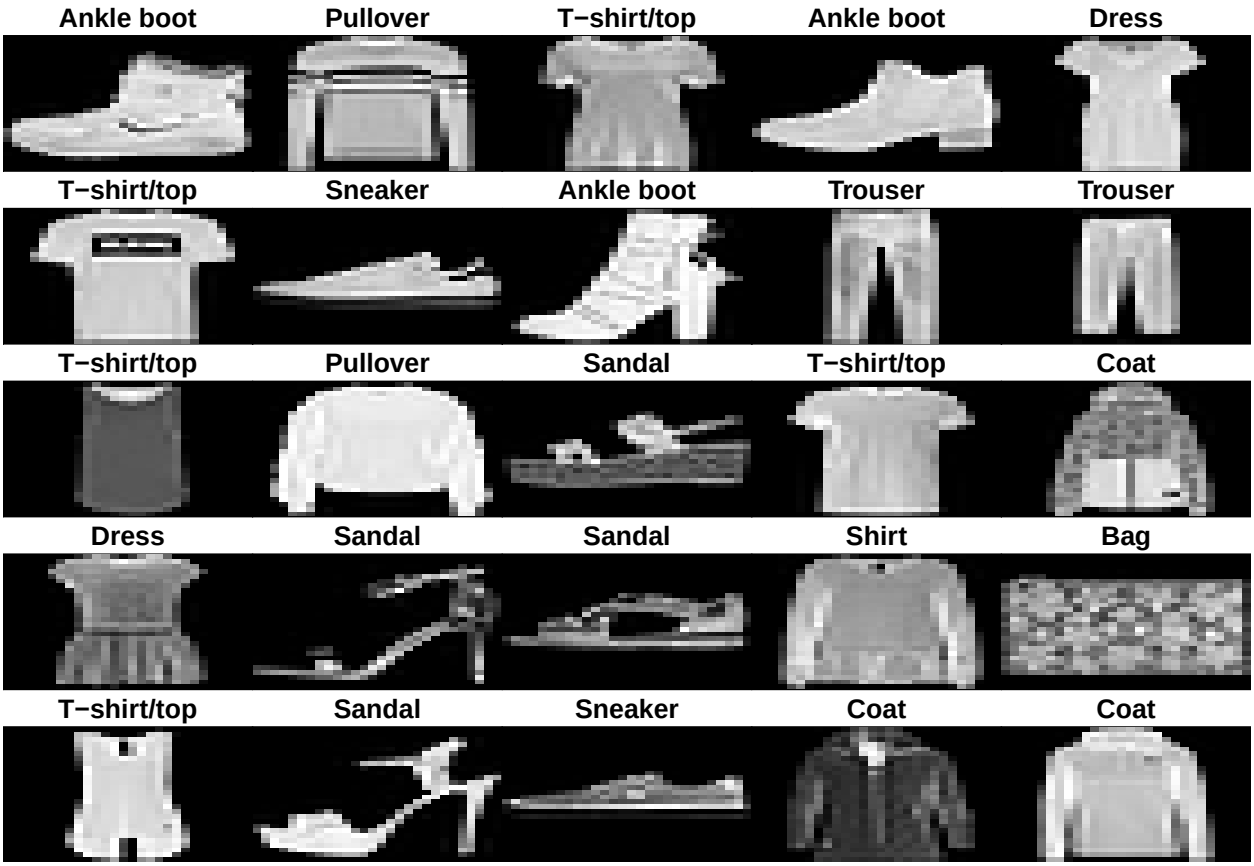
```
## [1] 10000
```

```
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("")
```




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

```
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]))
}
```




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

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

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

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

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

```
## Test loss: 0.3536332
```

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

```
## Test accuracy: 0.8724
```

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

```
predictions[1, ]
```

```
##  [1] 2.418602e-04 1.623269e-06 3.609604e-06 8.542939e-07 9.735770e-06
```

```
##  [6] 3.908670e-02 6.197199e-05 5.576788e-02 3.069744e-03 9.017560e-01
```

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

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

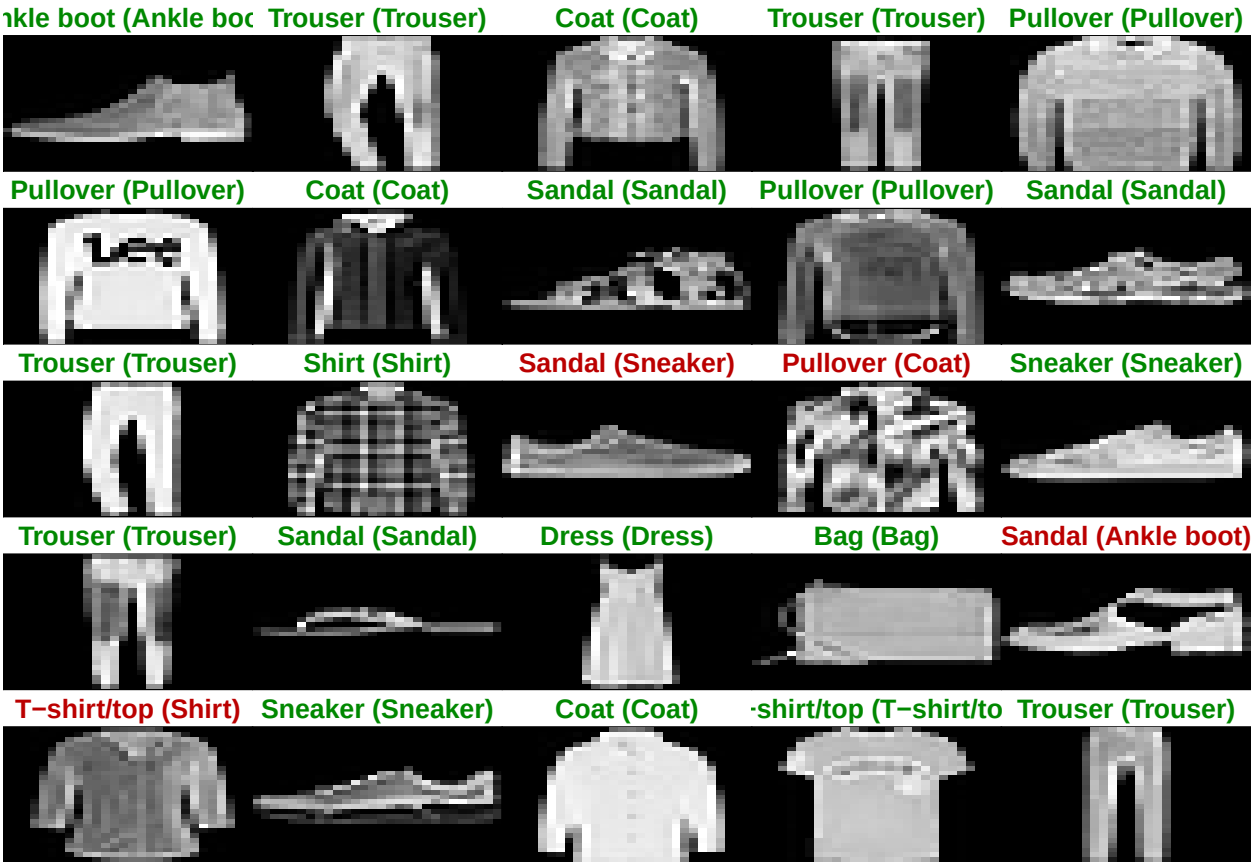
```
class_pred <- model %>% predict_classes(test_images)
class_pred[1:20]
```

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

```
test_labels[1]
```

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

```
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)
}
```




```
# 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
```

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

```
##              [,1]         [,2]         [,3]         [,4]        [,5]
## [1,] 0.0002418608 1.623272e-06 3.609615e-06 8.542955e-07 9.73577e-06
##             [,6]         [,7]       [,8]         [,9]     [,10]
## [1,] 0.03908666 6.197211e-05 0.05576788 0.003069747 0.901756
```

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

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

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

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