---

title: "Build a Deep Learning Based Image Classifier with R"

output: html\_notebook

---

```{r}

## Task 1: Project Overview & Import Libraries ##

library(keras)

```

```{r}

## Task 2: Import the Fashion MNIST Dataset ##

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')

```

```{r}

## Task 3: Data Exploration ##

dim(train\_images)

dim(train\_labels)

train\_labels[1:20]

dim(test\_images)

dim(test\_labels)

```

```{r}

## Task 4: Preprocess the Data ##

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]))

}

```

```{r}

## Task 5: Build the Model ##

model <- keras\_model\_sequential()

model %>%

layer\_flatten(input\_shape = c(28, 28)) %>%

layer\_dense(units = 128, activation = 'relu') %>%

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

```

```{r}

## Task 6: Compile the Model ##

model %>% compile(

optimizer = 'adam',

loss = 'sparse\_categorical\_crossentropy',

metrics = c('accuracy')

)

summary(model)

```

```{r}

## Task 7: Train and Evaluate the Model ##

model %>% fit(

train\_images, train\_labels,

epochs = 10, validation\_split=0.2)

score <- model %>% evaluate(test\_images, test\_labels)

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

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

```

```{r}

## Task 8: Make Predictions on Test Data ##

predictions <- model %>% predict(test\_images)

predictions[1, ]

which.max(predictions[1, ])

class\_pred <- model %>% predict\_classes(test\_images)

class\_pred[1:20]

test\_labels[1]

# 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)

predictions <- model %>% predict(img)

predictions

# subtract 1 as labels are 0-based

prediction <- predictions[1, ] - 1

which.max(prediction)

class\_pred <- model %>% predict\_classes(img)

class\_pred

```

```{r}

## Plot Images with Predictions ##

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)

}

```