MACS 30200 HW1

Ling Dai

```
#import libraries
library(keras)
library(tensorflow)
use_python("/Users/lingdai/anaconda3/bin/python3")

#import data
fmnist <- dataset_fashion_mnist()

#set seed
set.seed(1234)
```

```
train_images <- fmnist$train$x
train_labels <- fmnist$train$y
test_images <- fmnist$test$x
test_labels <- fmnist$test$y

train_labels <- to_categorical(train_labels)
test_labels <- to_categorical(test_labels)

#Preprocess the data by converting the data to a 2D tensor with individual values between 0 and 1
img_rows <- img_cols <- 28
train_images <- array_reshape(train_images, c(60000, 28*28))
train_images <- train_images / 255
str(train_images)
```

```
##  num [1:60000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
```

```
test_images <- array_reshape(test_images, c(10000, 28*28))
test_images <- test_images / 255
str(test_images)
```

```
##  num [1:10000, 1:784] 0 0 0 0 0 0 0 0 0 0 ...
```

```
#Randomly split the training data into 50,000 training observations and 10,000 validation observations
training_ind <- sample(60000, size = 50000)
training_images <- train_images[training_ind, ]
training_labels <- train_labels[training_ind, ]
validation_images <- train_images[-training_ind, ]
validation_labels <- train_labels[-training_ind, ]
```

Initial Model

For the initial model, the validation loss reached a minimum at the 9th epoch.

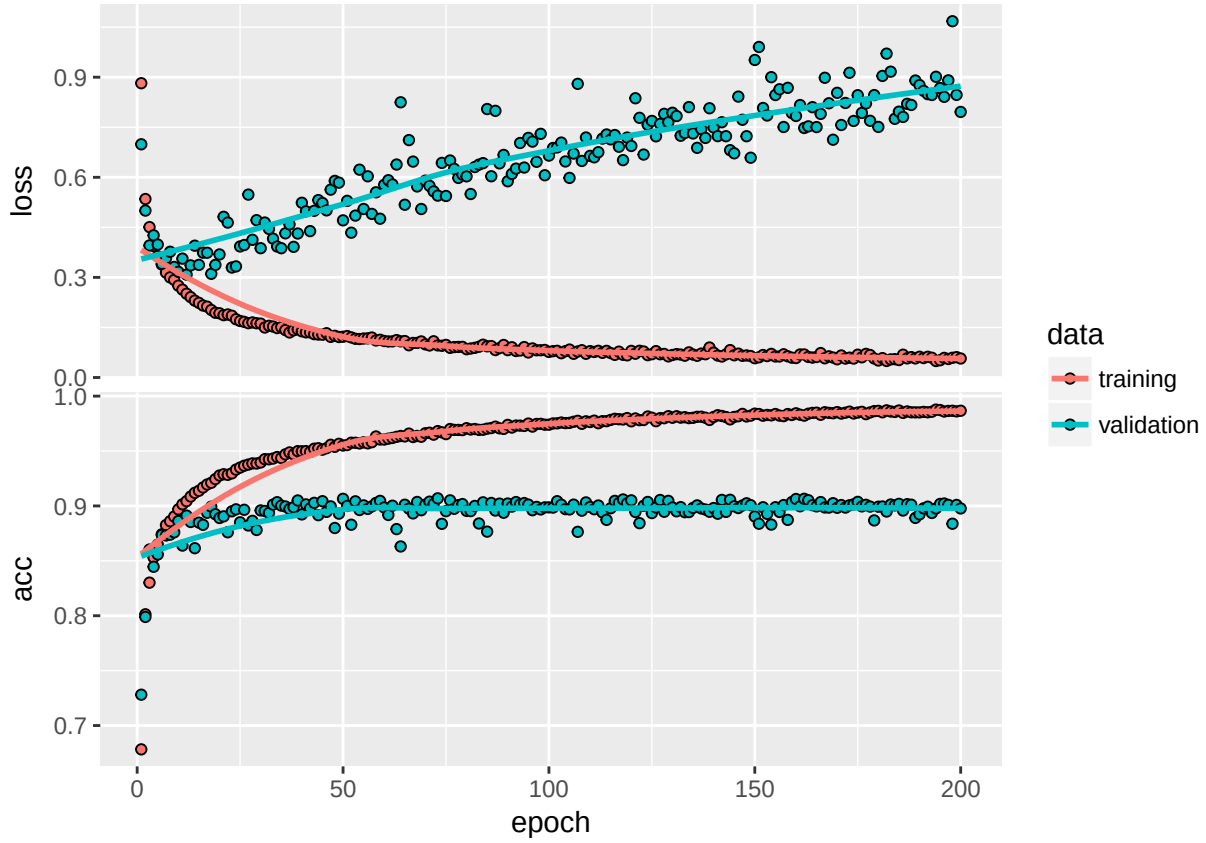
```
#initial model
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
```

```
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history1 <- network %>% fit(training_images, training_labels,
                            epochs = 200, batch_size = 512,
                            validation_data = list(validation_images, validation_labels))

plot(history1)
```




```
min(history1$metrics$val_loss)
```

```
## [1] 0.3082998
```

```
which(history1$metrics$val_loss==min(history1$metrics$val_loss))
```

```
## [1] 12
```

Drop-out Model

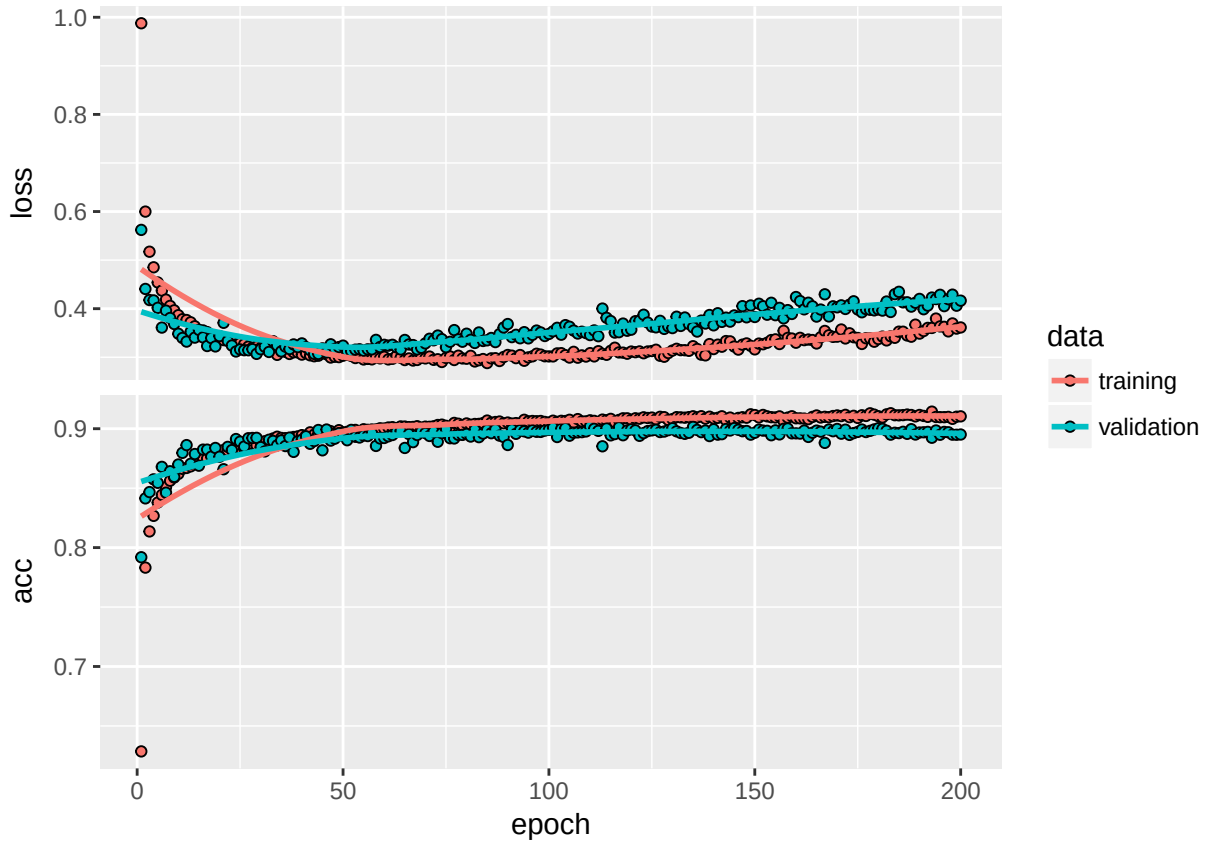
The drop-out model attained a minimum validation loss at the 39nd epoch. Graphically comparing the validation loss curves of these two models, we can see that the drop-out model performs better than the initial model, especially as the number of epochs increases.

```
network <- keras_model_sequential() %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 512, activation = "relu", input_shape = c(28 * 28)) %>%
  layer_dropout(rate = 0.5) %>%
  layer_dense(units = 10, activation = "softmax")

network %>% compile(
  optimizer = "rmsprop",
  loss = "categorical_crossentropy",
  metrics = c("accuracy")
)

history2 <- network %>% fit(training_images, training_labels,
                            epochs = 200, batch_size = 512,
                            validation_data = list(validation_images, validation_labels))

plot(history2)
```



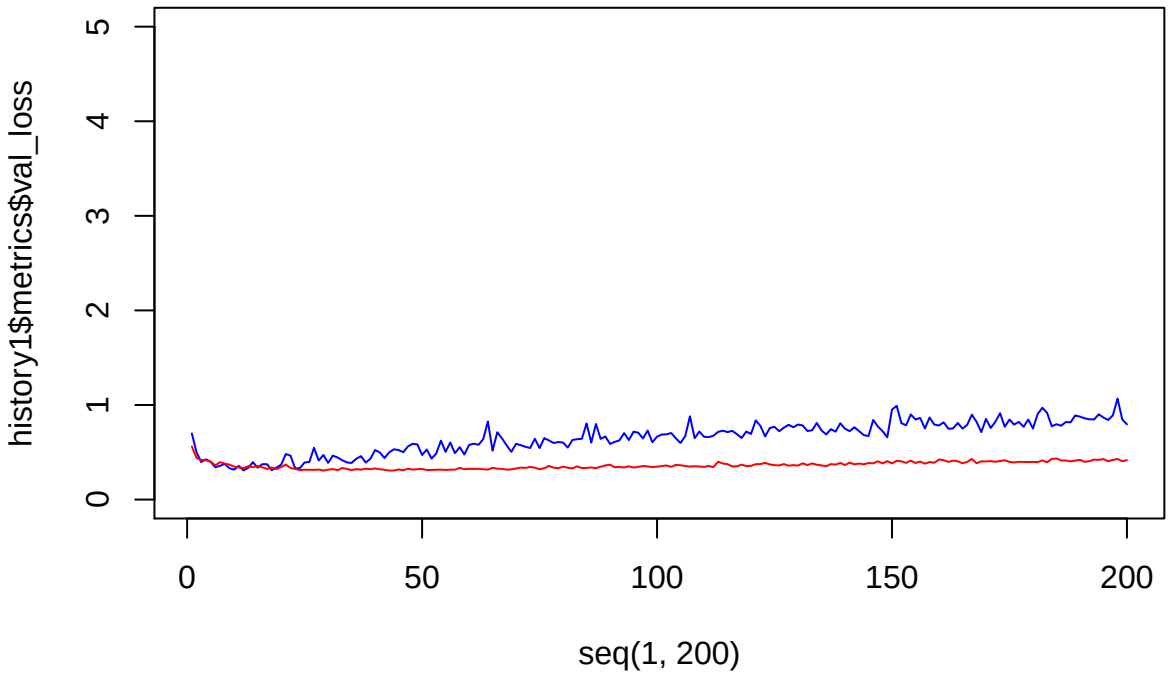

```
min(history2$metrics$val_loss)
```

```
## [1] 0.306814
```

```
which(history2$metrics$val_loss==min(history2$metrics$val_loss))
```

```
## [1] 44
```

```
plot(seq(1,200), history1$metrics$val_loss, type='l', col="blue", ylim=c(0,5))
lines(seq(1,200), history2$metrics$val_loss, col="red", ylim=c(0,5))
```


history1$metrics$val_loss
0
1
2
3
4
5
0
50
100
150
200
seq(1, 200)