DSE6211 Preliminary Report

Joseph Annand

2024-02-19

Report

Initially, a feed-forward dense neural network is trained using three quarters of all observations from the ABC hotels data set. The model consists of two hidden layers and the output. The first hidden layer consists of 75 units and uses a rectified linear activation function (relu).The number of units was chosen to be approximately two-thirds of the number of features in the training set. The relu function is the default activation function for hidden layers in a multilayer perceptron, so the relu function is used for both hidden layers of the neural network. For the second hidden layer, the number of units was chosen to be approximately half the amount of the units in the previous layer. For the output layer, the number of units is equal to 1 because the aim of the neural network is to provide a single output for predicting the status of a given booking. For a binary classification problem, the sigmoidal activation function is used so that all outputs are between 0 and 1.

The optimizer for the model is RMSprop, which is an adaptive learning rate algorithm that helps reduce the amount of computational effort required to train the neural network compared to other optimizers. The loss function used for the neural network is binary cross-entropy; this loss function calculates the dissimilarity between true labels and predicted labels when the output is between 0 and 1. The metric used in training the model is accuracy to measure how well the model is performing in terms of predicting booking status.

In the first attempt to train the neural network, a batch size of 512 and 100 epochs were used. The training and validation set results are shown in Figure 1.

A graph of different colored lines

Description automatically generated

Figure 1: Initial NN with batch size 512 and 100 epochs

There are over 27,000 samples in the training data set, allowing for about 54 batches per epoch in the first attempt at training the model. The model overfits the data as seen by the difference between the validation and training loss curves: the training loss continues to decline while the validation loss holds steady after about 40 epochs. Though this is a good sign that the model is well optimized, there is still room for improvement. The initial goal in training is to achieve a model that has some generalization power, or ability to make predictions on never-before-seen observations, and that can overfit the training the data. Figure 1 shows that the initial neural network created here does not sufficiently overfit, as the validation loss holds steady rather than reaching a minimum and then rapidly increasing.

In attempt to force the desired overfitting, the batch size is increased to 1000. The results are shown in Figure 2. Increasing the batch size to 1000 did not help with optimizing the model. The model is still overfitting, but not to the point where the validation loss begins to trend upward. In order to achieve the best optimization and the most overfitting with the current optimizer, the representation power of the model must be expanded.

Two new models were created: one with three hidden layers and one with four hidden layers. Each additional hidden layer is the same as the first hidden layer of the initial neural network consisting of 75 units and using the relu activation function. Figures 3 and 4 show the learning curves for the models.

Increasing the size of the neural network does not appear to further optimize the model. The validation loss curves for the three-layer and four-layer neural network are similar to the curve seen in Figure 1. To further investigate the optimization of the initial neural network, the model using the initial network architecture of two hidden layers was fitted using a batch size equal to 512 and 200 epochs. The result curves are shown in Figure 5.

A graph of red and blue dots

Description automatically generated

Figure 2: Initial NN with batch size 1000

A graph of red and blue lines

Description automatically generated

Figure 3: Neural Network with three hidden layers

A graph of red and blue lines

Description automatically generated

Figure 4: Neural Network with four hidden layers

When epochs is set to 200, the validation loss noticeably begins to trend upward and drastically diverge from the training loss curve, indicating the overfitting in the model desired to maximize optimization. Iterating over the entire training set 200 times is not practical, however, so additional changes to the network will have to be made to optimize the model more efficiently. The optimizer could be changed to an Adam optimization algorithm, and the learning rate could be adjusted to speed up training.

Additional data processing will need to be performed before evaluating the model using the test set. In the initial processing steps, after one hot encoding the test and training sets, the test set had less features than the training set. This is likely due to the training set having observations with rare values that are not seen in the test set. Identifying and either manipulating or removing the observations with these rare values will be a priority before the final report.

A graph showing different colored lines

Description automatically generated

Figure 5: Two-layer neural network using 200 epochs

Appendix

## Load libraries

library(dplyr)  
library(lubridate)  
library(caret)

## Data Pre-processing

### Create training and test sets

project\_data <- read.csv("project\_data/project\_data.csv")  
  
rownames(project\_data) <- project\_data$Booking\_ID  
project\_data <- project\_data[, -1]  
  
training\_ind <- createDataPartition(project\_data$booking\_status,  
 p = 0.75,  
 list = F,  
 times = 1)  
  
training\_set <- project\_data[training\_ind, ]  
test\_set <- project\_data[-training\_ind, ]  
  
  
training\_set$booking\_status <- ifelse(training\_set$booking\_status=="canceled",1,0)  
test\_set$booking\_status <- ifelse(test\_set$booking\_status=="canceled",1,0)

### Training Set Categorical Features

training\_set$arrival\_date <- parse\_date\_time(training\_set$arrival\_date, "ymd")  
training\_set$booking\_date <- int\_start(interval(training\_set$arrival\_date - ddays(training\_set$lead\_time),   
 training\_set$arrival\_date))  
  
season\_months <- data.frame(winter = c(12,1,2), spring = c(3,4,5),  
 summer = c(6,7,8), fall = c(9,10,11))  
  
  
get\_season <- function(x) {  
 y <- month(x)  
 # print(y)  
 for (j in 1:length(colnames(season\_months))) {  
 # print(j)  
 if (y %in% season\_months[[j]]) {  
 # print(colnames(season\_months)[j])  
 return(colnames(season\_months)[j])  
 }  
 }  
}  
  
training\_set$arrival\_season <- sapply(training\_set$arrival\_date, get\_season)  
training\_set$arrival\_day <- wday(training\_set$arrival\_date)  
training\_set$arrival\_day <- ifelse(training\_set$arrival\_day %in% c(1,6,7),  
 "weekend", "weekday")  
  
  
categorical\_var <- c(1:7,10:13,15,18,19)  
for (i in categorical\_var) {  
 training\_set[[i]] <- factor(training\_set[[i]])  
}  
  
cat\_col <- colnames(training\_set[, c(1:7,10:13,15,18,19)])  
  
onehot\_encoder <- dummyVars(~ no\_of\_adults + no\_of\_children + no\_of\_weekend\_nights  
 + no\_of\_week\_nights + type\_of\_meal\_plan + required\_car\_parking\_space  
 + room\_type\_reserved + market\_segment\_type + repeated\_guest  
 + no\_of\_previous\_cancellations + no\_of\_previous\_bookings\_not\_canceled  
 + no\_of\_special\_requests + arrival\_season + arrival\_day,  
 training\_set[, c("no\_of\_adults","no\_of\_children","no\_of\_weekend\_nights",  
 "no\_of\_week\_nights","type\_of\_meal\_plan",  
 "required\_car\_parking\_space",  
 "room\_type\_reserved","market\_segment\_type",  
 "repeated\_guest","no\_of\_previous\_cancellations",  
 "no\_of\_previous\_bookings\_not\_canceled",  
 "no\_of\_special\_requests","arrival\_season",  
 "arrival\_day")],  
 levelsOnly = F,  
 fullRank = T)  
  
onehot\_enc\_training <- predict(onehot\_encoder, training\_set[, c("no\_of\_adults","no\_of\_children","no\_of\_weekend\_nights",  
 "no\_of\_week\_nights","type\_of\_meal\_plan",  
 "required\_car\_parking\_space",  
 "room\_type\_reserved","market\_segment\_type",  
 "repeated\_guest","no\_of\_previous\_cancellations",  
 "no\_of\_previous\_bookings\_not\_canceled",  
 "no\_of\_special\_requests","arrival\_season",  
 "arrival\_day")])  
  
training\_set <- cbind(training\_set, onehot\_enc\_training)

### Test set Categorical Features

test\_set$arrival\_date <- parse\_date\_time(test\_set$arrival\_date, "ymd")  
test\_set$booking\_date <- int\_start(interval(test\_set$arrival\_date - ddays(test\_set$lead\_time),   
 test\_set$arrival\_date))  
  
  
test\_set$arrival\_season <- sapply(test\_set$arrival\_date, get\_season)  
test\_set$arrival\_day <- wday(test\_set$arrival\_date)  
test\_set$arrival\_day <- ifelse(test\_set$arrival\_day %in% c(1,6,7),  
 "weekend", "weekday")  
  
  
categorical\_var <- c(1:7,10:13,15,18,19)  
for (i in categorical\_var) {  
 test\_set[[i]] <- factor(test\_set[[i]])  
}  
  
cat\_col <- colnames(test\_set[, c(1:7,10:13,15,18,19)])  
  
onehot\_encoder <- dummyVars(~ no\_of\_adults + no\_of\_children + no\_of\_weekend\_nights  
 + no\_of\_week\_nights + type\_of\_meal\_plan + required\_car\_parking\_space  
 + room\_type\_reserved + market\_segment\_type + repeated\_guest  
 + no\_of\_previous\_cancellations + no\_of\_previous\_bookings\_not\_canceled  
 + no\_of\_special\_requests + arrival\_season + arrival\_day,  
 test\_set[, c("no\_of\_adults","no\_of\_children","no\_of\_weekend\_nights",  
 "no\_of\_week\_nights","type\_of\_meal\_plan",  
 "required\_car\_parking\_space",  
 "room\_type\_reserved","market\_segment\_type",  
 "repeated\_guest","no\_of\_previous\_cancellations",  
 "no\_of\_previous\_bookings\_not\_canceled",  
 "no\_of\_special\_requests","arrival\_season",  
 "arrival\_day")],  
 levelsOnly = F,  
 fullRank = T)  
  
onehot\_enc\_test <- predict(onehot\_encoder, test\_set[, c("no\_of\_adults","no\_of\_children","no\_of\_weekend\_nights",  
 "no\_of\_week\_nights","type\_of\_meal\_plan",  
 "required\_car\_parking\_space",  
 "room\_type\_reserved","market\_segment\_type",  
 "repeated\_guest","no\_of\_previous\_cancellations",  
 "no\_of\_previous\_bookings\_not\_canceled",  
 "no\_of\_special\_requests","arrival\_season",  
 "arrival\_day")])  
  
test\_set <- cbind(test\_set, onehot\_enc\_test)

### Numerical features

test\_set[, c("lead\_time", "avg\_price\_per\_room")] <- scale(test\_set[, c("lead\_time", "avg\_price\_per\_room")],  
 center = apply(training\_set[, c("lead\_time", "avg\_price\_per\_room")], 2, mean),  
 scale = apply(training\_set[, c("lead\_time", "avg\_price\_per\_room")], 2, sd))  
training\_set[, c("lead\_time", "avg\_price\_per\_room")] <- scale(training\_set[, c("lead\_time", "avg\_price\_per\_room")])

### Create tensors

train\_col <- ncol(training\_set)  
test\_col <- ncol(test\_set)  
  
training\_features <- array(data = unlist(training\_set[, c(8,14,20:train\_col)]),  
 dim = c(nrow(training\_set), length(c(8,14,20:train\_col))))  
training\_labels <- array(data = unlist(training\_set[, "booking\_status"]),  
 dim = nrow(training\_set))  
  
test\_features <- array(data = unlist(test\_set[, c(8,14,20:test\_col)]),  
 dim = c(nrow(test\_set), length(c(8,14,20:test\_col))))  
test\_labels <- array(data = unlist(test\_set[, "booking\_status"]),  
 dim = nrow(test\_set))

## Feed-forward Dense Neural Network

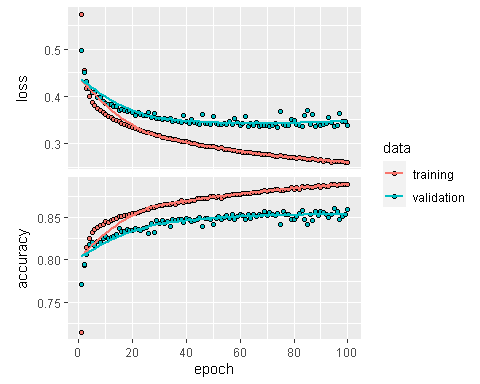
### Load Tensorflow workspace

library(reticulate)  
library(tensorflow)  
library(keras)  
  
use\_virtualenv("my\_tf\_workspace")

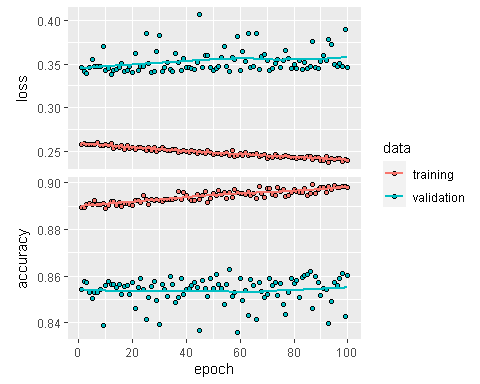
### First model

model <- keras\_model\_sequential(list(  
 layer\_dense(units = 75, activation = "relu"),  
 layer\_dense(units = 37, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")  
))  
  
compile(model,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")  
  
history <- fit(model, training\_features, training\_labels,  
 epochs = 100, batch\_size = 512, validation\_split = 0.33)  
  
history2 <- fit(model, training\_features, training\_labels,  
 epochs = 100, batch\_size = 1000, validation\_split = 0.33)  
  
history6 <- fit(model, training\_features, training\_labels,  
 epochs = 200, batch\_size = 512, validation\_split = 0.33)

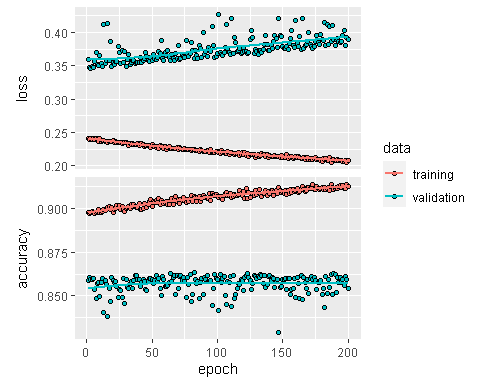
plot(history)



plot(history2)



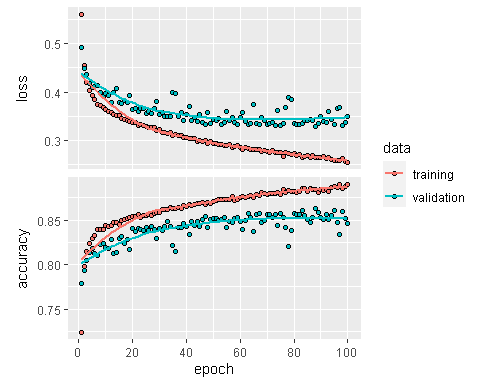
plot(history6)



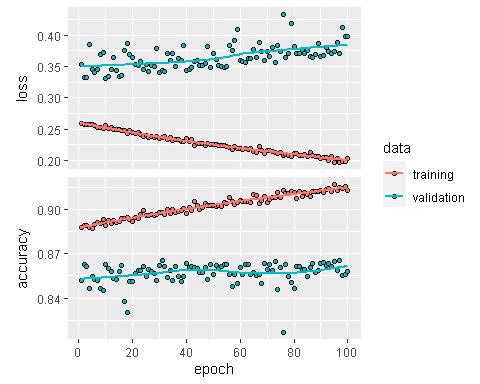
### Second model

model\_2 <- keras\_model\_sequential(list(  
 layer\_dense(units = 75, activation = "relu"),  
 layer\_dense(units = 75, activation = "relu"),  
 layer\_dense(units = 37, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")  
))  
  
compile(model\_2,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")  
  
history3 <- fit(model\_2, training\_features, training\_labels,  
 epochs = 100, batch\_size = 1000, validation\_split = 0.33)  
  
history4 <- fit(model\_2, training\_features, training\_labels,  
 epochs = 100, batch\_size = 512, validation\_split = 0.33)

plot(history3)



plot(history4)



### Thrid model

model\_3 <- keras\_model\_sequential(list(  
 layer\_dense(units = 75, activation = "relu"),  
 layer\_dense(units = 75, activation = "relu"),  
 layer\_dense(units = 75, activation = "relu"),  
 layer\_dense(units = 37, activation = "relu"),  
 layer\_dense(units = 1, activation = "sigmoid")  
))  
  
compile(model\_3,  
 optimizer = "rmsprop",  
 loss = "binary\_crossentropy",  
 metrics = "accuracy")  
  
history5 <- fit(model\_3, training\_features, training\_labels,  
 epochs = 100, batch\_size = 1000, validation\_split = 0.33)

## Epoch 1/100  
## 19/19 - 1s - loss: 0.5349 - accuracy: 0.7437 - val\_loss: 0.4814 - val\_accuracy: 0.7750 - 635ms/epoch - 33ms/step  
## Epoch 2/100  
## 19/19 - 0s - loss: 0.4425 - accuracy: 0.7981 - val\_loss: 0.4506 - val\_accuracy: 0.7901 - 86ms/epoch - 5ms/step  
## Epoch 3/100  
## 19/19 - 0s - loss: 0.4166 - accuracy: 0.8127 - val\_loss: 0.4233 - val\_accuracy: 0.8055 - 93ms/epoch - 5ms/step  
## Epoch 4/100  
## 19/19 - 0s - loss: 0.4020 - accuracy: 0.8206 - val\_loss: 0.4127 - val\_accuracy: 0.8140 - 95ms/epoch - 5ms/step  
## Epoch 5/100  
## 19/19 - 0s - loss: 0.3948 - accuracy: 0.8243 - val\_loss: 0.4378 - val\_accuracy: 0.7987 - 93ms/epoch - 5ms/step  
## Epoch 6/100  
## 19/19 - 0s - loss: 0.3833 - accuracy: 0.8329 - val\_loss: 0.3986 - val\_accuracy: 0.8275 - 92ms/epoch - 5ms/step  
## Epoch 7/100  
## 19/19 - 0s - loss: 0.3824 - accuracy: 0.8315 - val\_loss: 0.4324 - val\_accuracy: 0.8026 - 101ms/epoch - 5ms/step  
## Epoch 8/100  
## 19/19 - 0s - loss: 0.3701 - accuracy: 0.8398 - val\_loss: 0.3962 - val\_accuracy: 0.8233 - 93ms/epoch - 5ms/step  
## Epoch 9/100  
## 19/19 - 0s - loss: 0.3657 - accuracy: 0.8406 - val\_loss: 0.4883 - val\_accuracy: 0.7600 - 94ms/epoch - 5ms/step  
## Epoch 10/100  
## 19/19 - 0s - loss: 0.3664 - accuracy: 0.8414 - val\_loss: 0.4190 - val\_accuracy: 0.8144 - 94ms/epoch - 5ms/step  
## Epoch 11/100  
## 19/19 - 0s - loss: 0.3586 - accuracy: 0.8416 - val\_loss: 0.3972 - val\_accuracy: 0.8229 - 100ms/epoch - 5ms/step  
## Epoch 12/100  
## 19/19 - 0s - loss: 0.3573 - accuracy: 0.8460 - val\_loss: 0.4013 - val\_accuracy: 0.8200 - 95ms/epoch - 5ms/step  
## Epoch 13/100  
## 19/19 - 0s - loss: 0.3526 - accuracy: 0.8464 - val\_loss: 0.3788 - val\_accuracy: 0.8279 - 101ms/epoch - 5ms/step  
## Epoch 14/100  
## 19/19 - 0s - loss: 0.3529 - accuracy: 0.8445 - val\_loss: 0.4000 - val\_accuracy: 0.8136 - 104ms/epoch - 5ms/step  
## Epoch 15/100  
## 19/19 - 0s - loss: 0.3480 - accuracy: 0.8497 - val\_loss: 0.4087 - val\_accuracy: 0.8185 - 92ms/epoch - 5ms/step  
## Epoch 16/100  
## 19/19 - 0s - loss: 0.3476 - accuracy: 0.8470 - val\_loss: 0.3685 - val\_accuracy: 0.8355 - 93ms/epoch - 5ms/step  
## Epoch 17/100  
## 19/19 - 0s - loss: 0.3420 - accuracy: 0.8508 - val\_loss: 0.3676 - val\_accuracy: 0.8355 - 94ms/epoch - 5ms/step  
## Epoch 18/100  
## 19/19 - 0s - loss: 0.3439 - accuracy: 0.8512 - val\_loss: 0.3703 - val\_accuracy: 0.8340 - 99ms/epoch - 5ms/step  
## Epoch 19/100  
## 19/19 - 0s - loss: 0.3407 - accuracy: 0.8514 - val\_loss: 0.3753 - val\_accuracy: 0.8314 - 93ms/epoch - 5ms/step  
## Epoch 20/100  
## 19/19 - 0s - loss: 0.3345 - accuracy: 0.8552 - val\_loss: 0.3594 - val\_accuracy: 0.8373 - 89ms/epoch - 5ms/step  
## Epoch 21/100  
## 19/19 - 0s - loss: 0.3323 - accuracy: 0.8539 - val\_loss: 0.3611 - val\_accuracy: 0.8387 - 107ms/epoch - 6ms/step  
## Epoch 22/100  
## 19/19 - 0s - loss: 0.3305 - accuracy: 0.8552 - val\_loss: 0.3578 - val\_accuracy: 0.8428 - 94ms/epoch - 5ms/step  
## Epoch 23/100  
## 19/19 - 0s - loss: 0.3285 - accuracy: 0.8553 - val\_loss: 0.4035 - val\_accuracy: 0.8211 - 94ms/epoch - 5ms/step  
## Epoch 24/100  
## 19/19 - 0s - loss: 0.3278 - accuracy: 0.8529 - val\_loss: 0.3577 - val\_accuracy: 0.8433 - 100ms/epoch - 5ms/step  
## Epoch 25/100  
## 19/19 - 0s - loss: 0.3227 - accuracy: 0.8564 - val\_loss: 0.3940 - val\_accuracy: 0.8192 - 85ms/epoch - 4ms/step  
## Epoch 26/100  
## 19/19 - 0s - loss: 0.3219 - accuracy: 0.8589 - val\_loss: 0.3650 - val\_accuracy: 0.8369 - 99ms/epoch - 5ms/step  
## Epoch 27/100  
## 19/19 - 0s - loss: 0.3205 - accuracy: 0.8601 - val\_loss: 0.3499 - val\_accuracy: 0.8439 - 97ms/epoch - 5ms/step  
## Epoch 28/100  
## 19/19 - 0s - loss: 0.3178 - accuracy: 0.8607 - val\_loss: 0.3484 - val\_accuracy: 0.8444 - 100ms/epoch - 5ms/step  
## Epoch 29/100  
## 19/19 - 0s - loss: 0.3182 - accuracy: 0.8596 - val\_loss: 0.3553 - val\_accuracy: 0.8433 - 94ms/epoch - 5ms/step  
## Epoch 30/100  
## 19/19 - 0s - loss: 0.3143 - accuracy: 0.8603 - val\_loss: 0.3486 - val\_accuracy: 0.8454 - 92ms/epoch - 5ms/step  
## Epoch 31/100  
## 19/19 - 0s - loss: 0.3122 - accuracy: 0.8631 - val\_loss: 0.3500 - val\_accuracy: 0.8455 - 97ms/epoch - 5ms/step  
## Epoch 32/100  
## 19/19 - 0s - loss: 0.3096 - accuracy: 0.8652 - val\_loss: 0.3703 - val\_accuracy: 0.8318 - 100ms/epoch - 5ms/step  
## Epoch 33/100  
## 19/19 - 0s - loss: 0.3126 - accuracy: 0.8640 - val\_loss: 0.3649 - val\_accuracy: 0.8366 - 93ms/epoch - 5ms/step  
## Epoch 34/100  
## 19/19 - 0s - loss: 0.3069 - accuracy: 0.8651 - val\_loss: 0.3851 - val\_accuracy: 0.8220 - 88ms/epoch - 5ms/step  
## Epoch 35/100  
## 19/19 - 0s - loss: 0.3099 - accuracy: 0.8633 - val\_loss: 0.3489 - val\_accuracy: 0.8453 - 104ms/epoch - 5ms/step  
## Epoch 36/100  
## 19/19 - 0s - loss: 0.3069 - accuracy: 0.8665 - val\_loss: 0.3440 - val\_accuracy: 0.8469 - 89ms/epoch - 5ms/step  
## Epoch 37/100  
## 19/19 - 0s - loss: 0.3031 - accuracy: 0.8678 - val\_loss: 0.3466 - val\_accuracy: 0.8463 - 86ms/epoch - 5ms/step  
## Epoch 38/100  
## 19/19 - 0s - loss: 0.3030 - accuracy: 0.8686 - val\_loss: 0.3504 - val\_accuracy: 0.8405 - 97ms/epoch - 5ms/step  
## Epoch 39/100  
## 19/19 - 0s - loss: 0.3088 - accuracy: 0.8630 - val\_loss: 0.3549 - val\_accuracy: 0.8431 - 87ms/epoch - 5ms/step  
## Epoch 40/100  
## 19/19 - 0s - loss: 0.2965 - accuracy: 0.8711 - val\_loss: 0.3522 - val\_accuracy: 0.8433 - 90ms/epoch - 5ms/step  
## Epoch 41/100  
## 19/19 - 0s - loss: 0.2990 - accuracy: 0.8684 - val\_loss: 0.3459 - val\_accuracy: 0.8459 - 94ms/epoch - 5ms/step  
## Epoch 42/100  
## 19/19 - 0s - loss: 0.2977 - accuracy: 0.8672 - val\_loss: 0.3377 - val\_accuracy: 0.8518 - 92ms/epoch - 5ms/step  
## Epoch 43/100  
## 19/19 - 0s - loss: 0.2949 - accuracy: 0.8726 - val\_loss: 0.3395 - val\_accuracy: 0.8497 - 103ms/epoch - 5ms/step  
## Epoch 44/100  
## 19/19 - 0s - loss: 0.2960 - accuracy: 0.8710 - val\_loss: 0.3550 - val\_accuracy: 0.8435 - 95ms/epoch - 5ms/step  
## Epoch 45/100  
## 19/19 - 0s - loss: 0.2937 - accuracy: 0.8722 - val\_loss: 0.3604 - val\_accuracy: 0.8379 - 91ms/epoch - 5ms/step  
## Epoch 46/100  
## 19/19 - 0s - loss: 0.2904 - accuracy: 0.8743 - val\_loss: 0.3545 - val\_accuracy: 0.8444 - 89ms/epoch - 5ms/step  
## Epoch 47/100  
## 19/19 - 0s - loss: 0.2898 - accuracy: 0.8748 - val\_loss: 0.3554 - val\_accuracy: 0.8443 - 95ms/epoch - 5ms/step  
## Epoch 48/100  
## 19/19 - 0s - loss: 0.2879 - accuracy: 0.8762 - val\_loss: 0.3790 - val\_accuracy: 0.8347 - 93ms/epoch - 5ms/step  
## Epoch 49/100  
## 19/19 - 0s - loss: 0.2874 - accuracy: 0.8739 - val\_loss: 0.4004 - val\_accuracy: 0.8143 - 92ms/epoch - 5ms/step  
## Epoch 50/100  
## 19/19 - 0s - loss: 0.2888 - accuracy: 0.8758 - val\_loss: 0.3570 - val\_accuracy: 0.8434 - 93ms/epoch - 5ms/step  
## Epoch 51/100  
## 19/19 - 0s - loss: 0.2868 - accuracy: 0.8742 - val\_loss: 0.3532 - val\_accuracy: 0.8431 - 105ms/epoch - 6ms/step  
## Epoch 52/100  
## 19/19 - 0s - loss: 0.2851 - accuracy: 0.8762 - val\_loss: 0.3581 - val\_accuracy: 0.8427 - 94ms/epoch - 5ms/step  
## Epoch 53/100  
## 19/19 - 0s - loss: 0.2824 - accuracy: 0.8780 - val\_loss: 0.3447 - val\_accuracy: 0.8484 - 116ms/epoch - 6ms/step  
## Epoch 54/100  
## 19/19 - 0s - loss: 0.2816 - accuracy: 0.8782 - val\_loss: 0.3363 - val\_accuracy: 0.8527 - 94ms/epoch - 5ms/step  
## Epoch 55/100  
## 19/19 - 0s - loss: 0.2780 - accuracy: 0.8800 - val\_loss: 0.3488 - val\_accuracy: 0.8477 - 97ms/epoch - 5ms/step  
## Epoch 56/100  
## 19/19 - 0s - loss: 0.2810 - accuracy: 0.8785 - val\_loss: 0.3444 - val\_accuracy: 0.8525 - 100ms/epoch - 5ms/step  
## Epoch 57/100  
## 19/19 - 0s - loss: 0.2786 - accuracy: 0.8793 - val\_loss: 0.3407 - val\_accuracy: 0.8532 - 93ms/epoch - 5ms/step  
## Epoch 58/100  
## 19/19 - 0s - loss: 0.2771 - accuracy: 0.8809 - val\_loss: 0.3804 - val\_accuracy: 0.8340 - 93ms/epoch - 5ms/step  
## Epoch 59/100  
## 19/19 - 0s - loss: 0.2798 - accuracy: 0.8787 - val\_loss: 0.3367 - val\_accuracy: 0.8555 - 88ms/epoch - 5ms/step  
## Epoch 60/100  
## 19/19 - 0s - loss: 0.2775 - accuracy: 0.8804 - val\_loss: 0.3395 - val\_accuracy: 0.8544 - 103ms/epoch - 5ms/step  
## Epoch 61/100  
## 19/19 - 0s - loss: 0.2719 - accuracy: 0.8815 - val\_loss: 0.3375 - val\_accuracy: 0.8553 - 104ms/epoch - 5ms/step  
## Epoch 62/100  
## 19/19 - 0s - loss: 0.2720 - accuracy: 0.8847 - val\_loss: 0.3540 - val\_accuracy: 0.8504 - 89ms/epoch - 5ms/step  
## Epoch 63/100  
## 19/19 - 0s - loss: 0.2714 - accuracy: 0.8836 - val\_loss: 0.3373 - val\_accuracy: 0.8552 - 109ms/epoch - 6ms/step  
## Epoch 64/100  
## 19/19 - 0s - loss: 0.2711 - accuracy: 0.8828 - val\_loss: 0.3386 - val\_accuracy: 0.8560 - 97ms/epoch - 5ms/step  
## Epoch 65/100  
## 19/19 - 0s - loss: 0.2687 - accuracy: 0.8847 - val\_loss: 0.3368 - val\_accuracy: 0.8512 - 91ms/epoch - 5ms/step  
## Epoch 66/100  
## 19/19 - 0s - loss: 0.2720 - accuracy: 0.8812 - val\_loss: 0.3437 - val\_accuracy: 0.8527 - 104ms/epoch - 5ms/step  
## Epoch 67/100  
## 19/19 - 0s - loss: 0.2657 - accuracy: 0.8860 - val\_loss: 0.3744 - val\_accuracy: 0.8417 - 95ms/epoch - 5ms/step  
## Epoch 68/100  
## 19/19 - 0s - loss: 0.2672 - accuracy: 0.8834 - val\_loss: 0.3391 - val\_accuracy: 0.8573 - 95ms/epoch - 5ms/step  
## Epoch 69/100  
## 19/19 - 0s - loss: 0.2682 - accuracy: 0.8842 - val\_loss: 0.3380 - val\_accuracy: 0.8579 - 106ms/epoch - 6ms/step  
## Epoch 70/100  
## 19/19 - 0s - loss: 0.2611 - accuracy: 0.8886 - val\_loss: 0.3331 - val\_accuracy: 0.8569 - 97ms/epoch - 5ms/step  
## Epoch 71/100  
## 19/19 - 0s - loss: 0.2637 - accuracy: 0.8873 - val\_loss: 0.3322 - val\_accuracy: 0.8602 - 92ms/epoch - 5ms/step  
## Epoch 72/100  
## 19/19 - 0s - loss: 0.2599 - accuracy: 0.8896 - val\_loss: 0.3604 - val\_accuracy: 0.8412 - 101ms/epoch - 5ms/step  
## Epoch 73/100  
## 19/19 - 0s - loss: 0.2620 - accuracy: 0.8870 - val\_loss: 0.3481 - val\_accuracy: 0.8512 - 99ms/epoch - 5ms/step  
## Epoch 74/100  
## 19/19 - 0s - loss: 0.2644 - accuracy: 0.8849 - val\_loss: 0.3523 - val\_accuracy: 0.8454 - 95ms/epoch - 5ms/step  
## Epoch 75/100  
## 19/19 - 0s - loss: 0.2537 - accuracy: 0.8914 - val\_loss: 0.3605 - val\_accuracy: 0.8392 - 92ms/epoch - 5ms/step  
## Epoch 76/100  
## 19/19 - 0s - loss: 0.2588 - accuracy: 0.8888 - val\_loss: 0.3404 - val\_accuracy: 0.8562 - 104ms/epoch - 5ms/step  
## Epoch 77/100  
## 19/19 - 0s - loss: 0.2564 - accuracy: 0.8898 - val\_loss: 0.3429 - val\_accuracy: 0.8565 - 91ms/epoch - 5ms/step  
## Epoch 78/100  
## 19/19 - 0s - loss: 0.2572 - accuracy: 0.8909 - val\_loss: 0.3385 - val\_accuracy: 0.8574 - 99ms/epoch - 5ms/step  
## Epoch 79/100  
## 19/19 - 0s - loss: 0.2519 - accuracy: 0.8912 - val\_loss: 0.3348 - val\_accuracy: 0.8574 - 97ms/epoch - 5ms/step  
## Epoch 80/100  
## 19/19 - 0s - loss: 0.2562 - accuracy: 0.8910 - val\_loss: 0.3381 - val\_accuracy: 0.8582 - 94ms/epoch - 5ms/step  
## Epoch 81/100  
## 19/19 - 0s - loss: 0.2476 - accuracy: 0.8942 - val\_loss: 0.3908 - val\_accuracy: 0.8392 - 91ms/epoch - 5ms/step  
## Epoch 82/100  
## 19/19 - 0s - loss: 0.2552 - accuracy: 0.8894 - val\_loss: 0.3460 - val\_accuracy: 0.8485 - 101ms/epoch - 5ms/step  
## Epoch 83/100  
## 19/19 - 0s - loss: 0.2505 - accuracy: 0.8916 - val\_loss: 0.3624 - val\_accuracy: 0.8501 - 99ms/epoch - 5ms/step  
## Epoch 84/100  
## 19/19 - 0s - loss: 0.2497 - accuracy: 0.8945 - val\_loss: 0.3580 - val\_accuracy: 0.8443 - 99ms/epoch - 5ms/step  
## Epoch 85/100  
## 19/19 - 0s - loss: 0.2512 - accuracy: 0.8923 - val\_loss: 0.3341 - val\_accuracy: 0.8544 - 94ms/epoch - 5ms/step  
## Epoch 86/100  
## 19/19 - 0s - loss: 0.2482 - accuracy: 0.8929 - val\_loss: 0.3347 - val\_accuracy: 0.8586 - 102ms/epoch - 5ms/step  
## Epoch 87/100  
## 19/19 - 0s - loss: 0.2486 - accuracy: 0.8942 - val\_loss: 0.3396 - val\_accuracy: 0.8615 - 95ms/epoch - 5ms/step  
## Epoch 88/100  
## 19/19 - 0s - loss: 0.2427 - accuracy: 0.8969 - val\_loss: 0.3652 - val\_accuracy: 0.8416 - 96ms/epoch - 5ms/step  
## Epoch 89/100  
## 19/19 - 0s - loss: 0.2393 - accuracy: 0.8998 - val\_loss: 0.4947 - val\_accuracy: 0.8104 - 98ms/epoch - 5ms/step  
## Epoch 90/100  
## 19/19 - 0s - loss: 0.2456 - accuracy: 0.8954 - val\_loss: 0.3439 - val\_accuracy: 0.8608 - 88ms/epoch - 5ms/step  
## Epoch 91/100  
## 19/19 - 0s - loss: 0.2403 - accuracy: 0.8982 - val\_loss: 0.3520 - val\_accuracy: 0.8559 - 94ms/epoch - 5ms/step  
## Epoch 92/100  
## 19/19 - 0s - loss: 0.2442 - accuracy: 0.8969 - val\_loss: 0.3590 - val\_accuracy: 0.8418 - 102ms/epoch - 5ms/step  
## Epoch 93/100  
## 19/19 - 0s - loss: 0.2417 - accuracy: 0.8954 - val\_loss: 0.3469 - val\_accuracy: 0.8573 - 101ms/epoch - 5ms/step  
## Epoch 94/100  
## 19/19 - 0s - loss: 0.2376 - accuracy: 0.8983 - val\_loss: 0.3460 - val\_accuracy: 0.8589 - 95ms/epoch - 5ms/step  
## Epoch 95/100  
## 19/19 - 0s - loss: 0.2365 - accuracy: 0.9003 - val\_loss: 0.3768 - val\_accuracy: 0.8498 - 94ms/epoch - 5ms/step  
## Epoch 96/100  
## 19/19 - 0s - loss: 0.2385 - accuracy: 0.8970 - val\_loss: 0.4041 - val\_accuracy: 0.8253 - 90ms/epoch - 5ms/step  
## Epoch 97/100  
## 19/19 - 0s - loss: 0.2361 - accuracy: 0.8985 - val\_loss: 0.3684 - val\_accuracy: 0.8541 - 107ms/epoch - 6ms/step  
## Epoch 98/100  
## 19/19 - 0s - loss: 0.2377 - accuracy: 0.8969 - val\_loss: 0.4160 - val\_accuracy: 0.8352 - 94ms/epoch - 5ms/step  
## Epoch 99/100  
## 19/19 - 0s - loss: 0.2347 - accuracy: 0.8987 - val\_loss: 0.3473 - val\_accuracy: 0.8550 - 96ms/epoch - 5ms/step  
## Epoch 100/100  
## 19/19 - 0s - loss: 0.2321 - accuracy: 0.9014 - val\_loss: 0.3548 - val\_accuracy: 0.8457 - 97ms/epoch - 5ms/step

plot(history5)

