> head(max(average\_acc\_history$validation\_acc))

[1] 0.910905

> # Evaluate on Testset

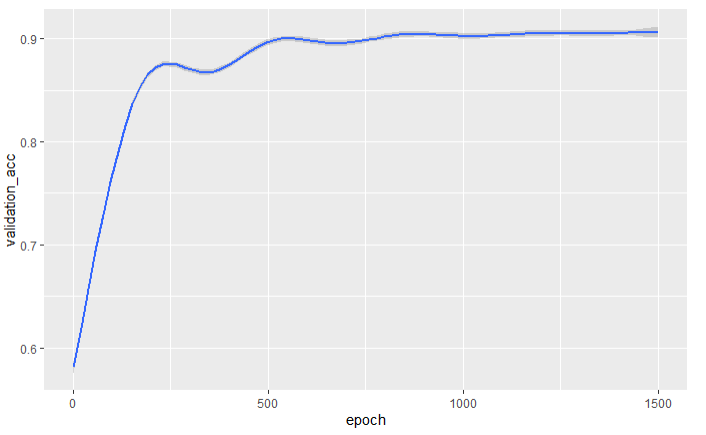
> eval <- evaluate(model, test\_data, test\_targets, verbose = 1)

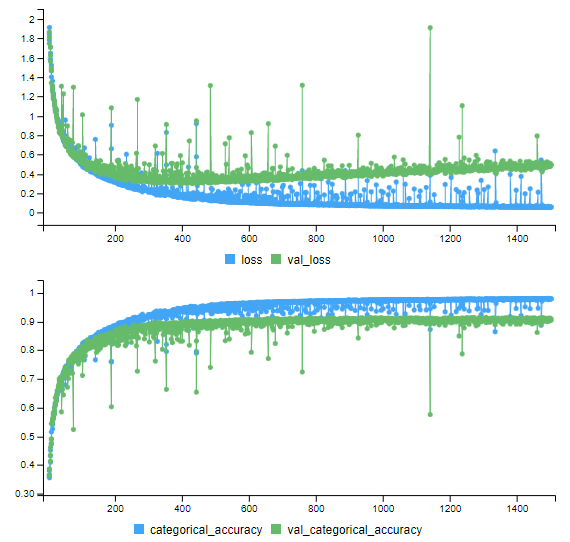
423/423 [==============================] - 2s 4ms/step - loss: 2.1197 - categorical\_accuracy: 0.7426

> head(eval)

loss categorical\_accuracy

2.1197028 0.7425867





#train\_data <- matrix\_data

train\_data <- data.matrix(matrix\_data)

test\_data <- data.matrix(matrix\_data\_test)

train\_targets <- data.matrix(matrix\_targets)

test\_targets <- data.matrix(matrix\_targets\_test)

# Function to build the model

build\_model <- function() {

model <- keras\_model\_sequential() %>%

#layer\_batch\_normalization(axis = -1L, input\_shape = dim(train\_data)[[2]]) %>%

layer\_dense(units = 64, activation = "relu", input\_shape = dim(train\_data)[[2]]) %>%

layer\_dense(units = 64, activation = "relu") %>%

#layer\_dropout(0.3) %>%

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

layer\_dense(units = 256, activation = "relu") %>%

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

layer\_dense(units = 64, activation = "relu") %>%

layer\_dense(units = 64, activation = "relu") %>%

layer\_dense(units = 8, activation = "softmax")

model %>% compile(

optimizer = optimizer\_sgd(learning\_rate = 0.1),

loss = "categorical\_crossentropy",

metrics = "categorical\_accuracy"

)

}