AML Neural Networks

Sushma

2024-07-07

```r
# Load Keras, Tensorflow, ggplot for deep Learning, Machine learning computation and Data visualization
library(keras)
library(tensorflow)
library(ggplot2)
```

```r
# Load the IMDb dataset with a vocabulary size limited to 10,000 words.
imdb_data <- dataset_imdb(num_words = 10000)

# Extract training data and labels from the IMDb dataset.
train_data <- imdb_data$train$x
train_labels <- imdb_data$train$y

# Extract test data and labels from the IMDb dataset.
test_data <- imdb_data$test$x
test_labels <- imdb_data$test$y
```

```r
# Prepare the data
vectorize_sequences <- function(sequences, dimension = 10000) {
  results <- matrix(0, nrow = length(sequences), ncol = dimension)
  for (i in seq_along(sequences)) {
    results[i, sequences[[i]]] <- 1
  }
  results
}

# Vectorize the training and test data sequences
x_train <- vectorize_sequences(train_data)
x_test <- vectorize_sequences(test_data)

# Convert train and test labels to matrix format
y_train <- as.matrix(train_labels)
y_test <- as.matrix(test_labels)
```

```r
# 1. Number of Hidden Layers

# One hidden layer model
model_one_hidden <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = 'relu', input_shape = c(10000)) %>%  # First hidden layer with 1
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an

# Three hidden layers model
model_three_hidden <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = 'relu', input_shape = c(10000)) %>%  # First hidden layer with 1
  layer_dense(units = 16, activation = 'relu') %>%  # Second hidden layer with 16 units and Relu activa
  layer_dense(units = 16, activation = 'relu') %>%  # Third hidden layer with 16 units and Relu activat
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an
```

```
# 2. Number of Hidden Units

# Model with 32 hidden units
model_32_units <- keras_model_sequential() %>%
  layer_dense(units = 32, activation = 'relu', input_shape = c(10000)) %>%  # Hidden layer with 32 unit
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an

# Model with 64 hidden units
model_64_units <- keras_model_sequential() %>%
  layer_dense(units = 64, activation = 'relu', input_shape = c(10000)) %>%  # Hidden layer with 64 unit
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an
```

```
# 3. Loss Function
# Mean Squared Error (MSE) loss function
# Define a sequential Keras model
model_mse_loss <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = 'relu', input_shape = c(10000)) %>%  # Hidden layer with 16 unit
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an

# Compile the model
model_mse_loss %>% compile(optimizer = 'rmsprop',   # RMSprop optimizer
                           loss = 'mse',            # Mean Squared Error (MSE) loss function
                           metrics = c('accuracy'))  # Accuracy metric for evaluation
```

```
# 4. Activation Function
# tanh activation function
# Define a sequential Keras model with tanh activation
model_tanh_activation <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = 'tanh', input_shape = c(10000)) %>%  # Hidden layer with 16 unit
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an
```

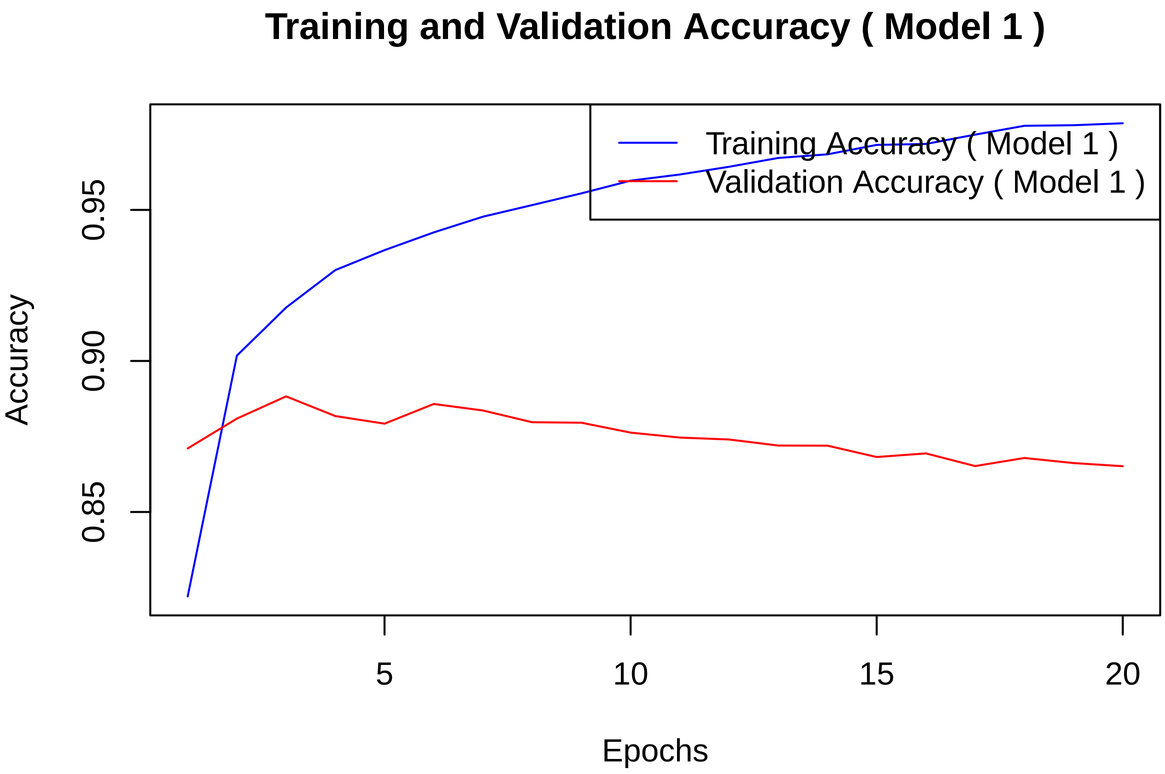
```
# 5. Regularization and Dropout
# L2 Regularization
# Load the Keras library for deep learning
library(keras)

# Regularization with L2 (ridge) penalty
model_l2_regularization <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = 'relu', input_shape = c(10000), kernel_regularizer = regularizer
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an

# Dropout regularization
model_dropout <- keras_model_sequential() %>%
  layer_dense(units = 16, activation = 'relu', input_shape = c(10000)) %>%  # Hidden layer with 16 unit
  layer_dropout(rate = 0.5) %>%  # Dropout layer with dropout rate of 0.5
  layer_dense(units = 1, activation = 'sigmoid')  # Output layer with 1 unit (binary classification) an
```

```
# Function to create, compile, and fit the model# Function to compile and fit a Keras model
compile_and_fit_model <- function(model, x_train, y_train, x_val, y_val, epochs = 20, batch_size = 512)
  # Compile the model with optimizer, loss function, and metrics
  model %>% compile(optimizer = 'rmsprop',                 # Optimizer: RMSprop
                    loss = 'binary_crossentropy',          # Loss function: Binary cross-entropy
                    metrics = c('accuracy'))               # Metrics for evaluation: Accuracy

  # Fit the model on training data, validating on validation data
  history <- model %>% fit(x_train, y_train,               # Training data and labels
                           epochs = epochs,                # Number of epochs
                           batch_size = batch_size,        # Batch size
                           validation_data = list(x_val, y_val),  # Validation data and labels
                           verbose = 0)                    # Verbosity level (0: silent, 1: progress ba

  # Return training history
  history
}
```

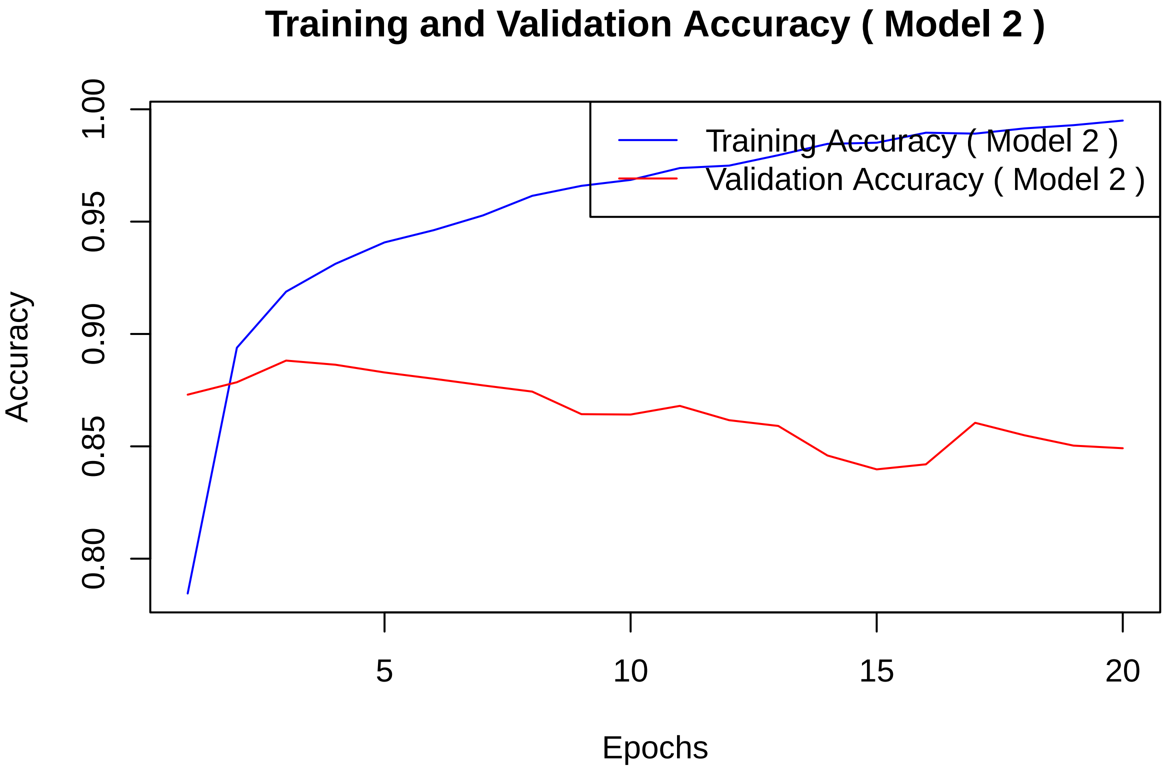
```
# Function to plot the training and validation accuracy
# Function to plot training and validation accuracy from model history
plot_history <- function(history, label) {
  # Extract training and validation accuracy from history object
  acc <- history$metrics$accuracy
  val_acc <- history$metrics$val_accuracy

  # Create sequence of epochs for x-axis
  epochs <- seq_along(acc)

  # Plot training accuracy in blue
  plot(epochs, acc, type = 'l', col = 'blue', xlab = 'Epochs', ylab = 'Accuracy',
       main = paste('Training and Validation Accuracy (', label, ')'))

  # Add validation accuracy to the plot in red
  lines(epochs, val_acc, type = 'l', col = 'red')

  # Add legend to differentiate between training and validation accuracy
  legend('topright',
         legend = c(paste('Training Accuracy (', label, ')'), paste('Validation Accuracy (', label, ')'
         col = c('blue', 'red'),
         lty = 1:1)
}
```

```
# Create a list of models
models_list <- list(model_one_hidden, model_three_hidden, model_32_units, model_64_units, model_mse_los

# Loop through each model, print summary, compile, fit, and plot
for (i in seq_along(models_list)) {
  label <- paste('Model', i)

  # Print model summary
  cat('\n\n', label, 'Summary:')
  summary(models_list[[i]])

  # Compile and fit the model
  history <- compile_and_fit_model(models_list[[i]], x_train, y_train, x_test, y_test)

  # Plot the training and validation accuracy
  plot_history(history, label)
  # Clear session to release memory
  gc()
}
```

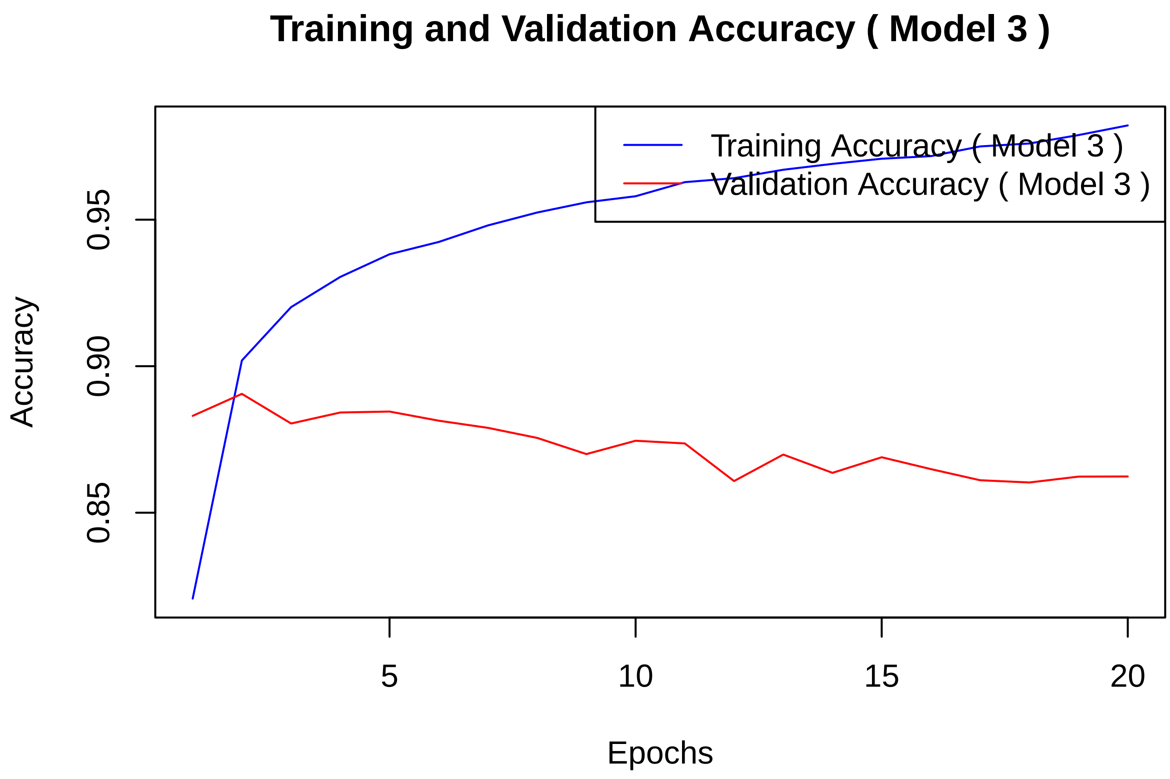
```
##
##
##  Model 1 Summary:Model: "sequential"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_1 (Dense)                    (None, 16)                      160016
##  dense (Dense)                      (None, 1)                       17
## ================================================================================
## Total params: 160033 (625.13 KB)
## Trainable params: 160033 (625.13 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```


Training and Validation Accuracy (Model 1)

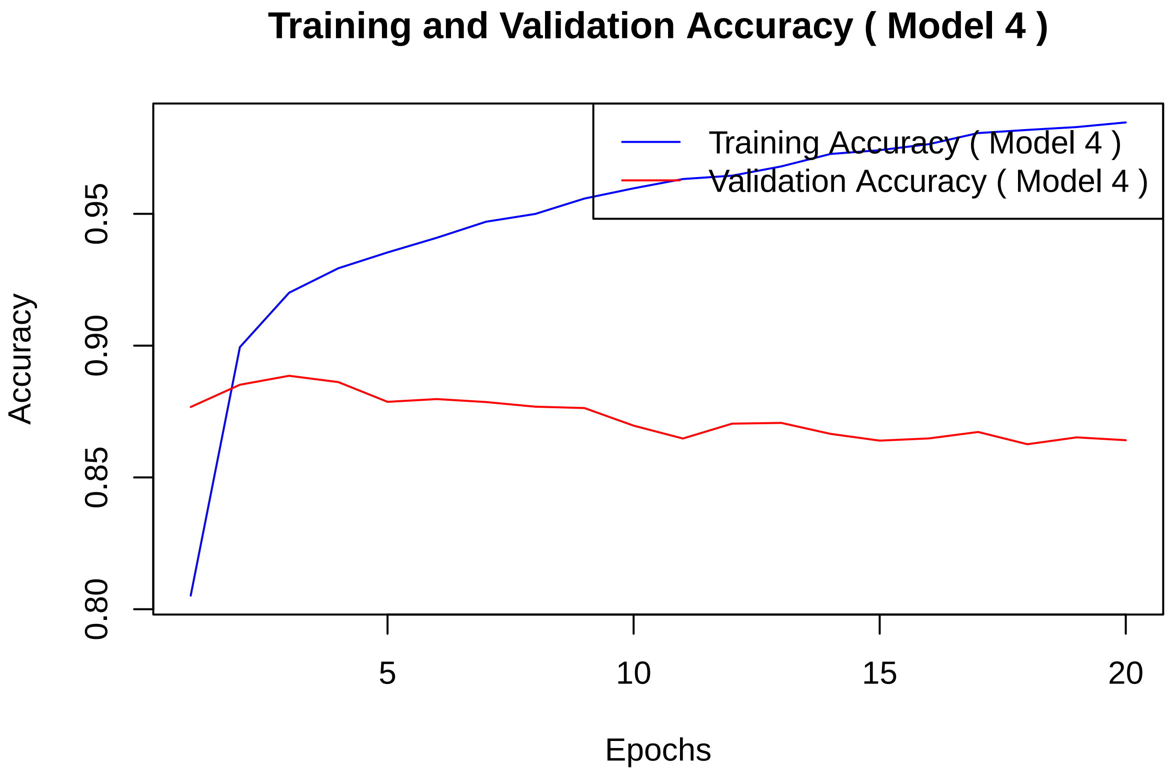
```
##
##
##  Model 2 Summary:Model: "sequential_1"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_5 (Dense)                    (None, 16)                      160016
##  dense_4 (Dense)                    (None, 16)                      272
##  dense_3 (Dense)                    (None, 16)                      272
##  dense_2 (Dense)                    (None, 1)                       17
## ================================================================================
## Total params: 160577 (627.25 KB)
## Trainable params: 160577 (627.25 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```



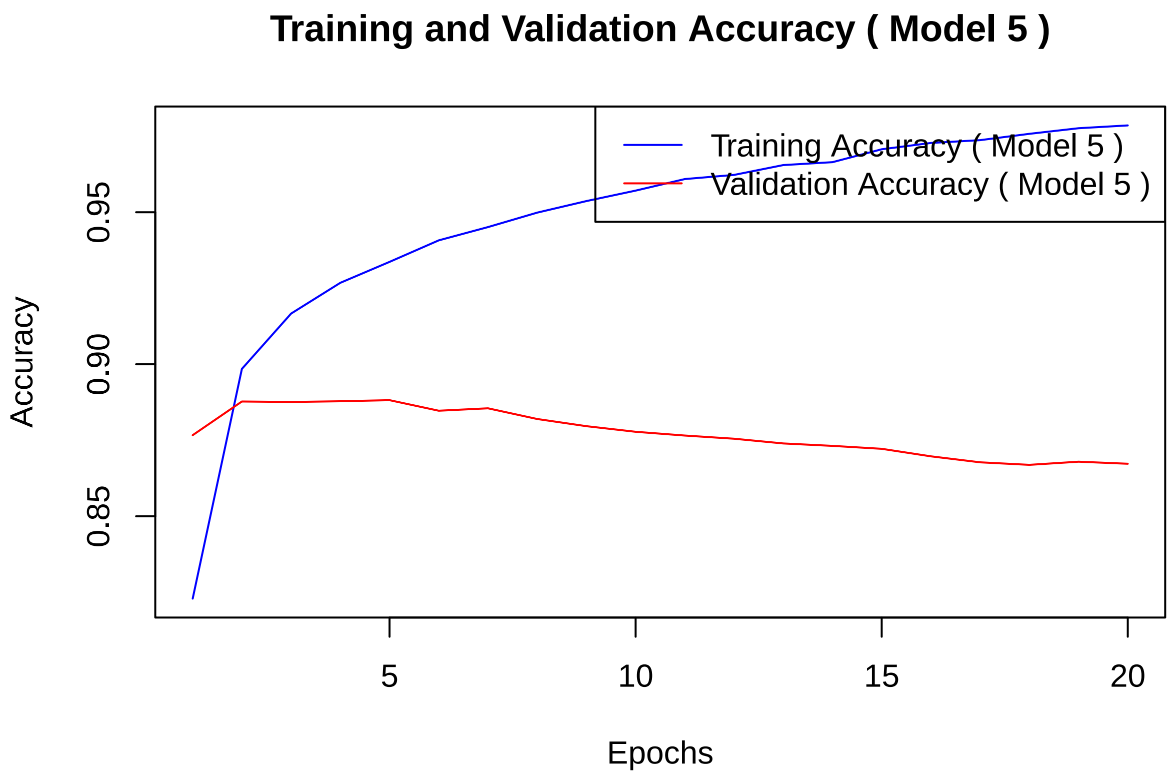
```
##
##
##  Model 3 Summary:Model: "sequential_2"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                     Param #
## ================================================================================
##  dense_7 (Dense)                    (None, 32)                       320032
##  dense_6 (Dense)                    (None, 1)                        33
## ================================================================================
## Total params: 320065 (1.22 MB)
## Trainable params: 320065 (1.22 MB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```



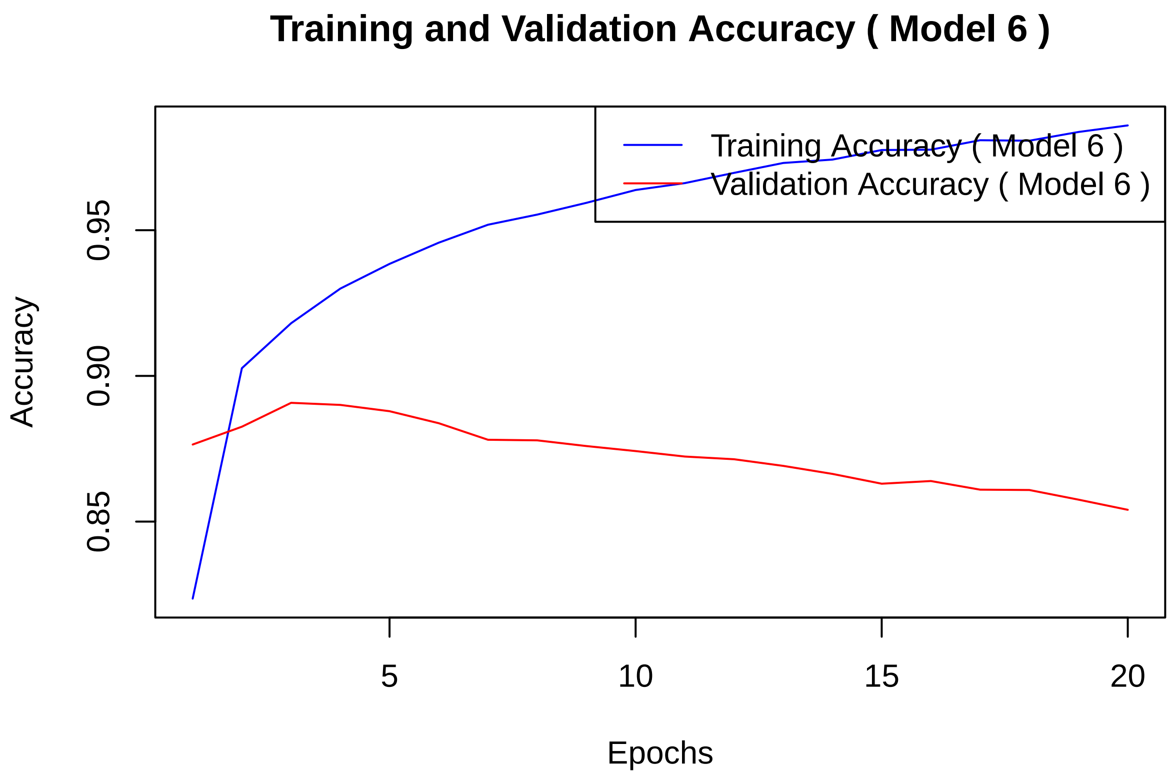
```
##
##
##  Model 4 Summary:Model: "sequential_3"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_9 (Dense)                    (None, 64)                      640064
##  dense_8 (Dense)                    (None, 1)                       65
## ================================================================================
## Total params: 640129 (2.44 MB)
## Trainable params: 640129 (2.44 MB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```



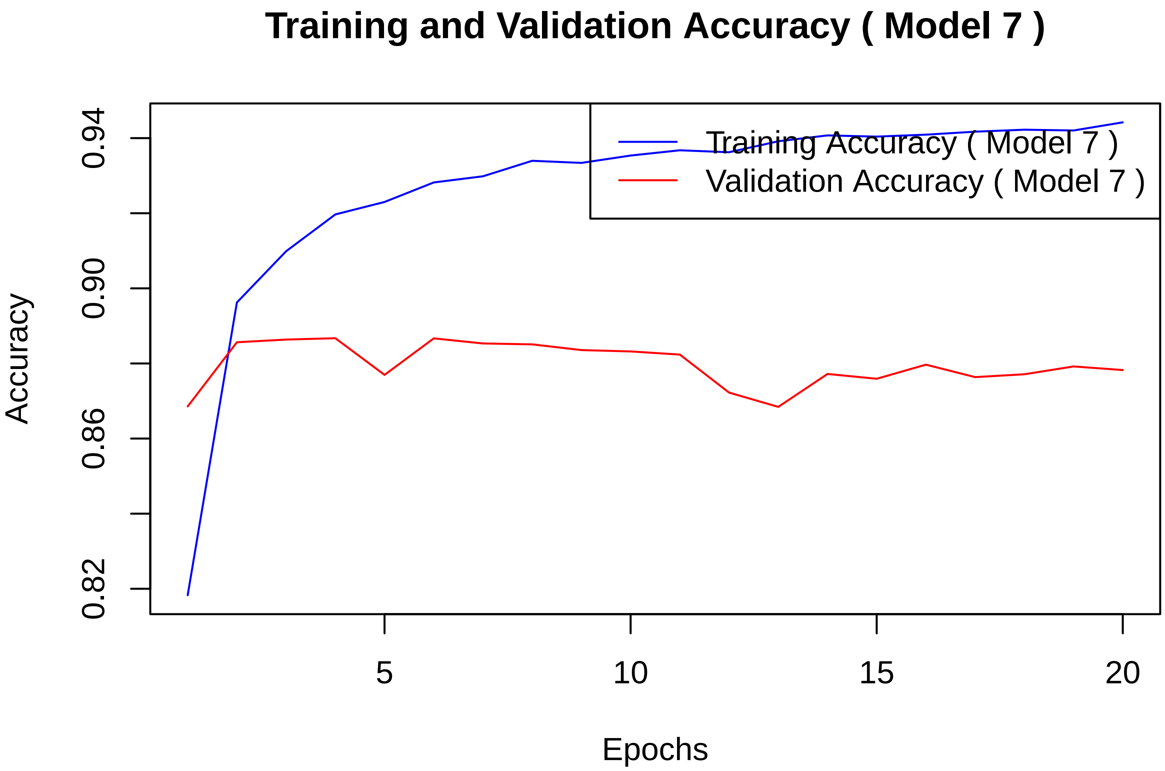
```
##
##
##  Model 5 Summary:Model: "sequential_4"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_11 (Dense)                   (None, 16)                      160016
##  dense_10 (Dense)                   (None, 1)                       17
## ================================================================================
## Total params: 160033 (625.13 KB)
## Trainable params: 160033 (625.13 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```



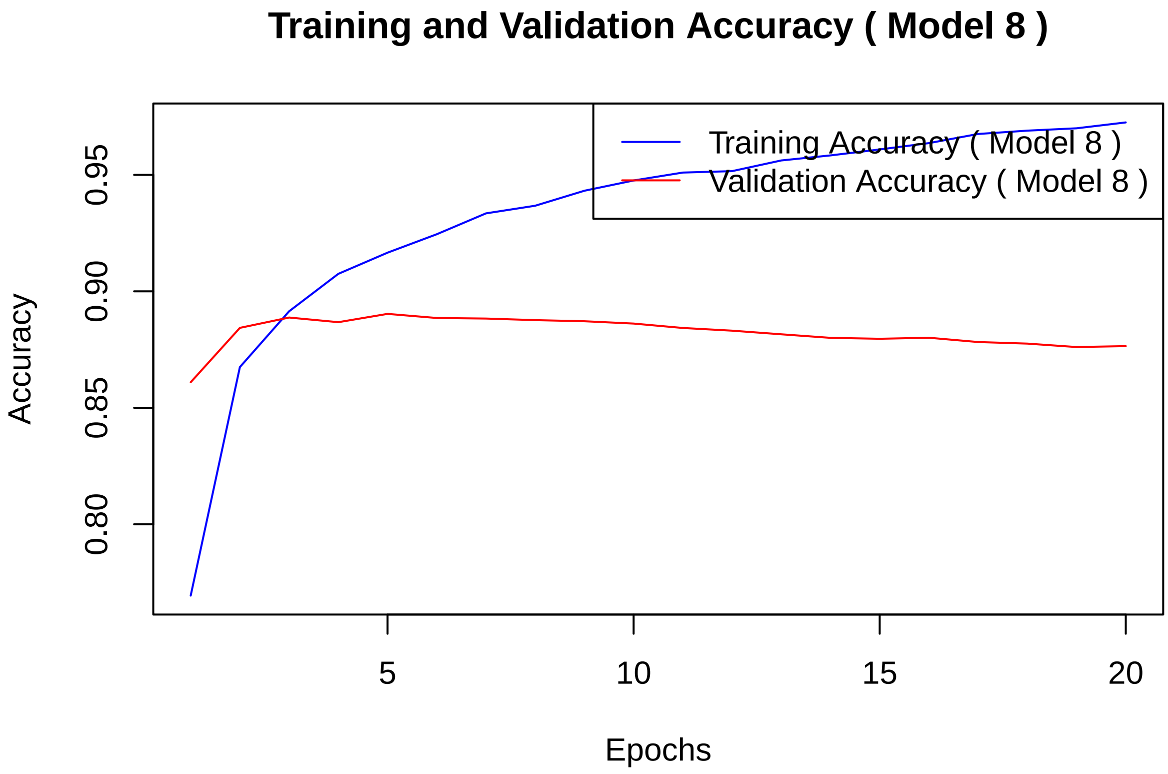
```
##
##
##  Model 6 Summary:Model: "sequential_5"
## ________________________________________________________________________________
##  Layer (type)                        Output Shape                    Param #
## ================================================================================
##  dense_13 (Dense)                    (None, 16)                      160016
##  dense_12 (Dense)                    (None, 1)                       17
## ================================================================================
## Total params: 160033 (625.13 KB)
## Trainable params: 160033 (625.13 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```



```
##
##
##  Model 7 Summary:Model: "sequential_6"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_15 (Dense)                   (None, 16)                      160016
##  dense_14 (Dense)                   (None, 1)                       17
## ================================================================================
## Total params: 160033 (625.13 KB)
## Trainable params: 160033 (625.13 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```



```
##
##
##  Model 8 Summary:Model: "sequential_7"
## ________________________________________________________________________________
##  Layer (type)                       Output Shape                    Param #
## ================================================================================
##  dense_17 (Dense)                   (None, 16)                      160016
##  dropout (Dropout)                  (None, 16)                      0
##  dense_16 (Dense)                   (None, 1)                       17
## ================================================================================
## Total params: 160033 (625.13 KB)
## Trainable params: 160033 (625.13 KB)
## Non-trainable params: 0 (0.00 Byte)
## ________________________________________________________________________________
```


Training and Validation Accuracy (Model 8)
Training Accuracy (Model 8)
Validation Accuracy (Model 8)
Accuracy
0.80
0.85
0.90
0.95
Epochs
5
10
15
20