Part 1 Option B - Handwritten Digit Recognition with Neural Networks

UnderGrad Team

```
library(ggthemes)
library(keras)
library(R.matlab)
library(tidyverse)

set.seed(42)
```

Part 2

A function computing the output of the neural net given an 784 x 1 vector representing a digit

```
softmax <- function(y){
    return(exp(y)/sum(exp(y)))
}

forward <- function(X, W, b){
  L1 <- W%*%X + b
  output <- softmax(L1)
  return(output)
}
```

Part 3

Part 4

Part 5

Part 6

Part 7

See https://tensorflow.rstudio.com/guide/keras/ for documentation. Here we define the neural network.

```r
# Load data
mnist <- readMat('mnist_all.mat')
data_train <- data.frame()
data_test <- data.frame()

for (i in 0:9) {
  train_digit <- mnist[paste0('train', i)][[1]] %>% data.frame
  train_digit['Y'] <- i
  data_train <- rbind(train_digit, data_train)

  test_digit <- mnist[paste0('test', i)][[1]] %>% data.frame
  test_digit['Y'] <- i
  data_test <- rbind(test_digit, data_test)
}

# Shuffle training dataset
data_train <- data_train[sample(nrow(data_train)), ]
data_test <- data_test[sample(nrow(data_test)), ]

# Split into X and Y
X_train <- data_train %>% select(-Y) %>% as.matrix()
Y_train <- data_train$Y
X_test <- data_test %>% select(-Y) %>% as.matrix()
Y_test <- data_test$Y

# Scale by 255
X_train <- X_train / 255.0
X_test <- X_test / 255.0

# Convert Y to categorical
Y_train_cat <- to_categorical(Y_train)
Y_test_cat <- to_categorical(Y_test)
```

```r
# Create model
model <- keras_model_sequential()
model %>%
  layer_dense(units = 300, activation = 'tanh', input_shape = c(ncol(X_train))) %>%
  layer_dense(units = 10, activation = 'softmax')

model %>% compile(
  loss = 'categorical_crossentropy',
  optimizer = optimizer_sgd(lr = 0.01),
  metrics = c('accuracy')
)

summary(model)
```

```
## Model: "sequential"
## ________________________________________________________________________________
## Layer (type)                        Output Shape                    Param #
## ================================================================================
## dense_1 (Dense)                     (None, 300)                     235500
## ________________________________________________________________________________
```

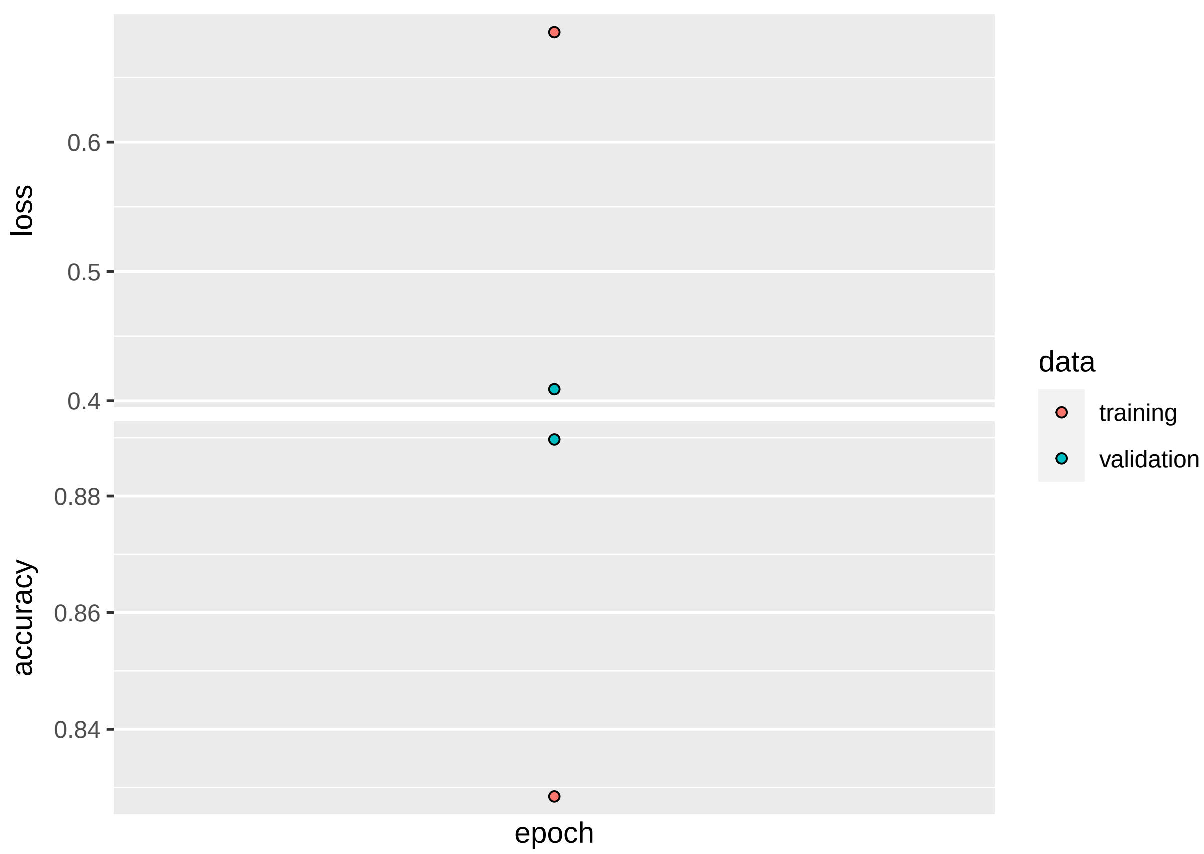
```
## dense (Dense)                         (None, 10)                        3010
## ================================================================================
## Total params: 238,510
## Trainable params: 238,510
## Non-trainable params: 0
## ________________________________________________________________________________
```

Part 8

Training neural network with mini-batch gradient descent.

```r
# Train model
history <- model %>% fit(
  X_train, Y_train_cat,
  epochs = 1, batch_size = 50,
  validation_data = list(X_test, Y_test_cat))

plot(history)
```



Displaying 20 digits which were classified correctly

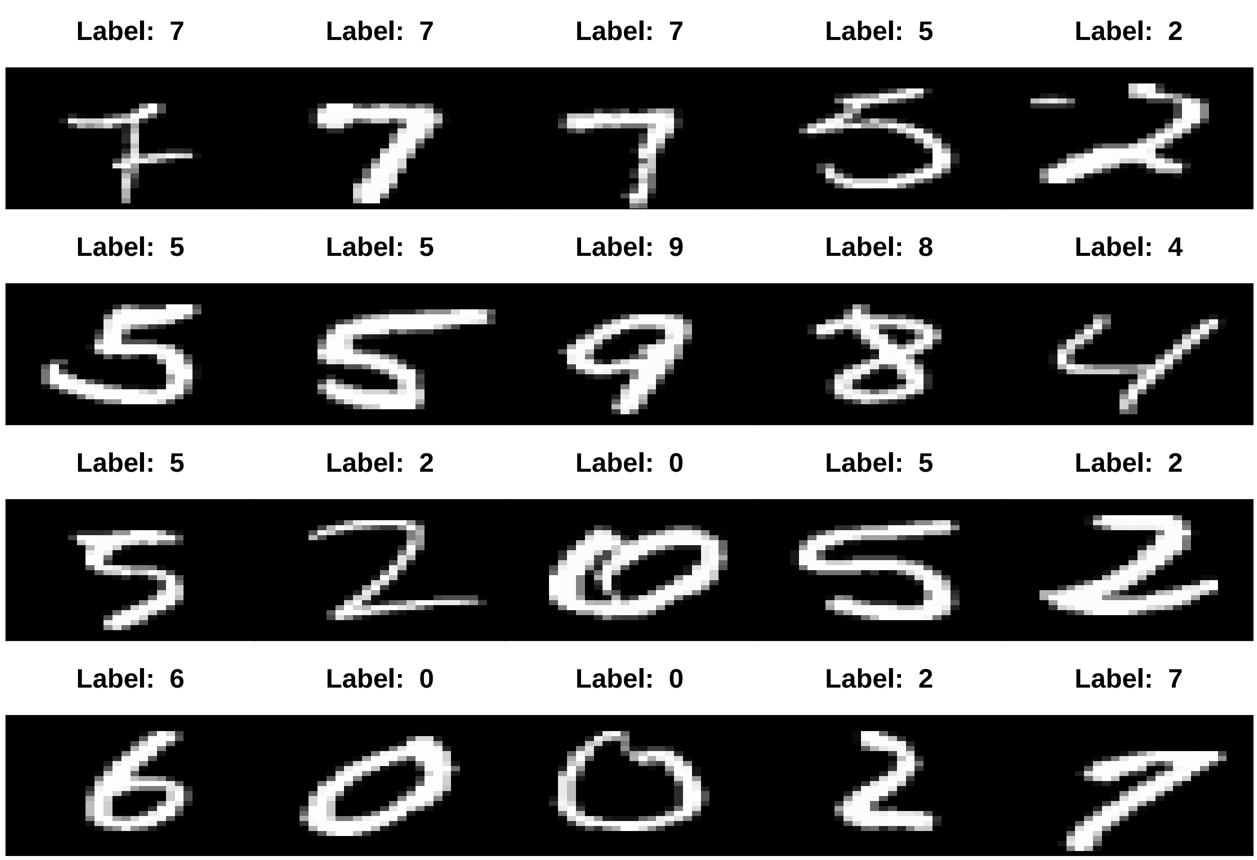
```r
# Get predictions on test set
Y_pred <- model %>% predict_classes(X_test)
```

```
# Show correct predictions
correct_idx <- Y_pred == Y_test
X_correct <- X_test[correct_idx,]
Y_correct <- Y_test[correct_idx]

par(mfcol = c(4, 5))
par(mar = c(0, 0, 3, 0), xaxs = 'i', yaxs = 'i')

for (idx in 1:20) {
  x <- X_correct[idx,] %>% rev()
  y <- Y_correct[idx]
  dim(x) <- c(28, 28)
  x <- apply(x, 2, rev)

  image(1:28, 1:28, x, col = gray((0:255) / 255),
    xlab = '', ylab = '', xaxt = 'n', yaxt = 'n',
    main = paste('Label: ', y)
  )
}
```

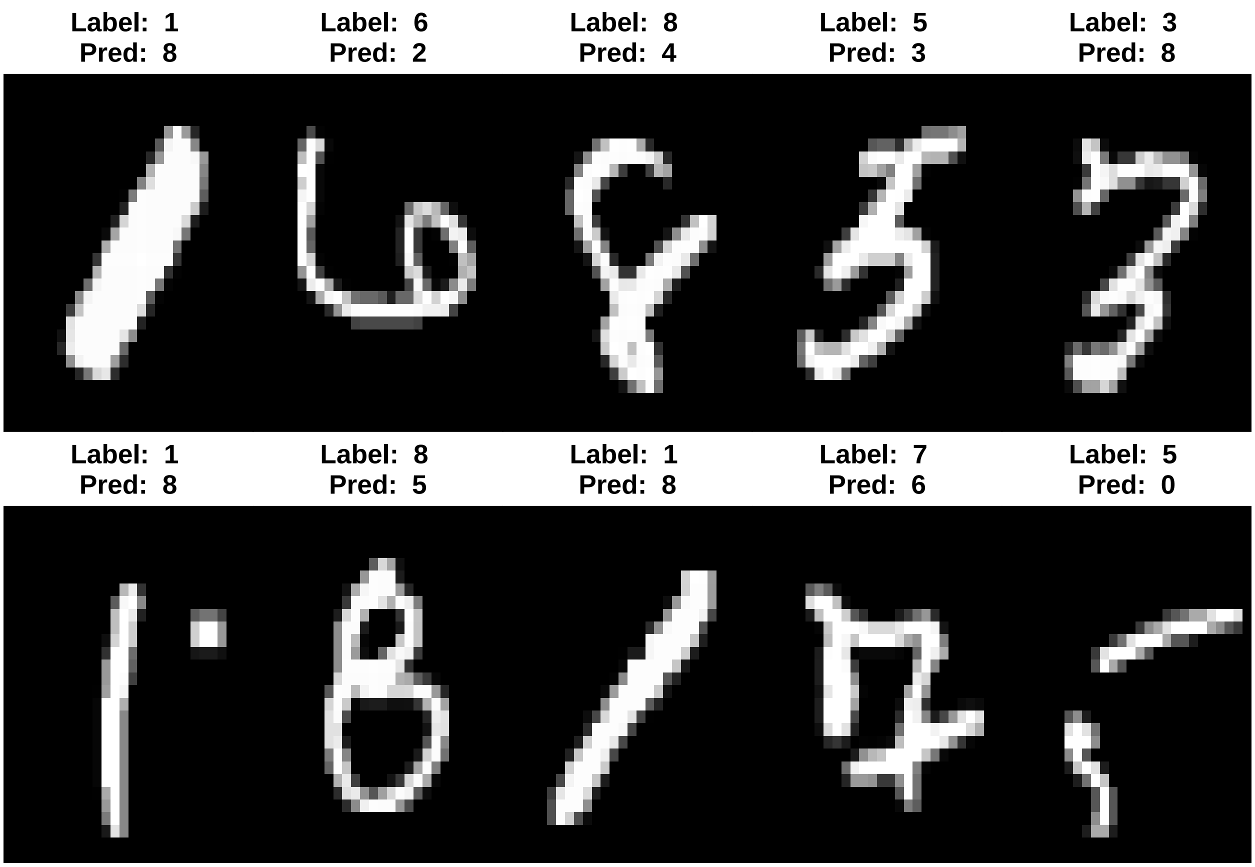



```
# Show incorrect predictions
X_incorrect <- X_test[!correct_idx,]
Y_incorrect <- Y_test[!correct_idx]
Y_pred_incorrect <- Y_pred[!correct_idx]
```

```
par(mfcol = c(2, 5))
par(mar = c(0, 0, 3, 0), xaxs = 'i', yaxs = 'i')

for (idx in 1:10) {
  x <- X_incorrect[idx,] %>% rev()
  y_true <- Y_incorrect[idx]
  y_pred <- Y_pred_incorrect[idx]
  dim(x) <- c(28, 28)
  x <- apply(x, 2, rev)

  image(1:28, 1:28, x, col = gray((0:255) / 255),
    xlab = '', ylab = '', xaxt = 'n', yaxt = 'n',
    main = paste('Label: ', y_true, '\nPred: ', y_pred)
  )
}
```



Part 9