# HandWritten Digit Recognition

Data Science Process (Assignment)

10-09-2020

Group Members
Ansuman Biswas
RA1811027010036

Rishav Kumar Jha
RA1811027010039

## Problem Description

It is easy for the human to perform a task accurately by practicing it repeatedly and memorizing it for the next time. Human brain can process and analyse images easily. Also, recognize the different elements present in the images.

In this competition, the goal is to correctly identify digits from a dataset of tens of thousands of handwritten images and experiment with different algorithms to learn first-hand what works well and how techniques compare.

## About the Dataset

**MNIST ("Modified National Institute of Standards and Technology")** is the de facto "hello world" dataset of computer vision. Since its release in 1999, this classic dataset of handwritten images has served as the basis for benchmarking classification algorithms. As new machine learning techniques emerge, MNIST remains a reliable resource for researchers and learners alike.

A handwritten digit's recognition system was implemented with the famous MNIST data set.

Handwritten digits recognition has been studied by researchers since 1998 with almost all the algorithms designed by then and even until now. The test error rate decreased from 12% in 1988 by linear classifier to 0.23% in 2012 by convolutional nets, and these days more and more data scientists and machine learning experts are trying to develop and validate unsupervised learning methods such as auto-encoder and deep learning models. Besides, there is also a Kaggle project based on MNIST, which is a perfect resource to begin with.

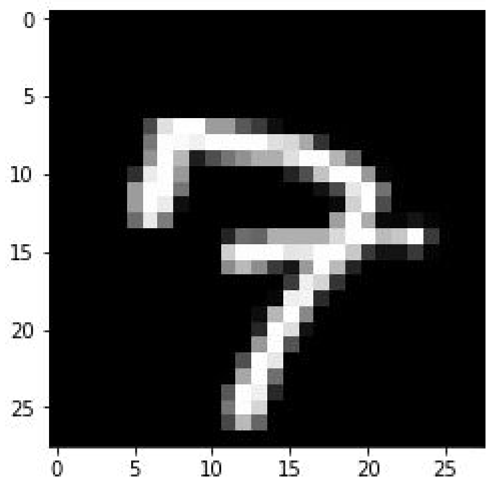
## Code

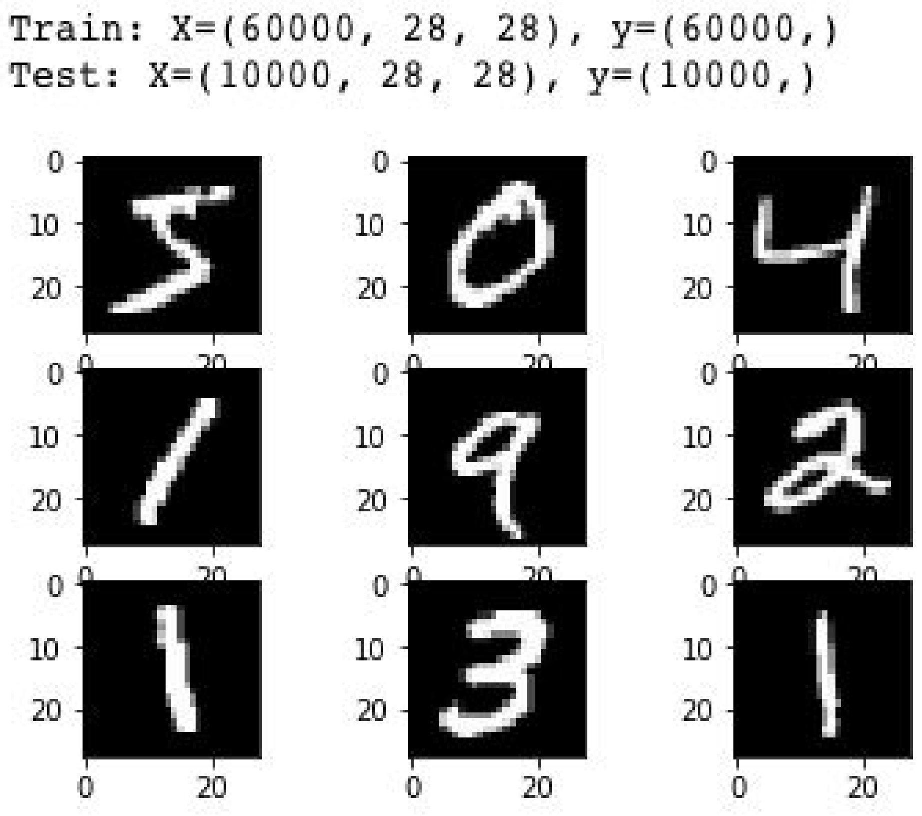
```
# MNIST data
library(keras)
mnist <- dataset_mnist()
trainx <- mnist$train$x
trainy <- mnist$train$y
testx <- mnist$test$x
testy <- mnist$test$y

# Plot images
par(mfrow = c(3,3))
for (i in 1:9) plot(as.raster(trainx[i,,], max = 255))
par(mfrow= c(1,1))

# Five
a <- c(1, 12, 36, 48, 66, 101, 133, 139, 146)
par(mfrow = c(3,3))
for (i in a) plot(as.raster(trainx[i,,], max = 255))
par(mfrow= c(1,1))
```

```
# Reshape & rescale
trainx <- array_reshape(trainx, c(nrow(trainx), 784))
testx <- array_reshape(testx, c(nrow(testx), 784))
trainx <- trainx / 255
testx <- testx /255

# One hot encoding
trainy <- to_categorical(trainy, 10)
testy <- to_categorical(testy, 10)

# Model
model <- keras_model_sequential()
model %>%
    layer_dense(units = 512, activation = 'relu', input_shape = c(784)) %>%
    layer_dropout(rate = 0.4) %>%
    layer_dense(units= 256, activation = 'relu') %>%
    layer_dropout(rate = 0.3) %>%
    layer_dense(units = 10, activation = 'softmax')

# Compile
model %>%
    compile(loss = 'categorical_crossentropy',
        optimizer = optimizer_rmsprop(),
        metrics = 'accuracy')

# Fit model
history <- model %>%
    fit(trainx,
        trainy,
        epochs = 30,
        batch_size = 32,
        validation_split = 0.2)

# Evaluation and Prediction - Test data
model %>% evaluate(testx, testy)
pred <- model %>% predict_classes(testx)
table(Predicted = pred, Actual = mnist$test$y)

prob <- model %>% predict_proba(testx)
cbind(prob, Predicted_class = pred, Actual = mnist$test$y)[1:5,]

# New data
library(EBImage)
setwd("~/Desktop/numbers")
temp = list.files(pattern = "*.jpg")
mypic <- list()
for (i in 1:length(temp)) {mypic[[i]] <- readImage(temp[[i]])}

par(mfrow = c(3,2))
for (i in 1:length(temp)) plot(mypic[[i]])
par(mfrow = c(1,1))

for (i in 1:length(temp)) {colorMode(mypic[[i]]) <- Grayscale}
for (i in 1:length(temp)) {mypic[[i]] <- 1-mypic[[i]]}
for (i in 1:length(temp)) {mypic[[i]] <- resize(mypic[[i]], 28, 28)}

for (i in 1:length(temp)) {mypic[[i]] <- array_reshape(mypic[[i]], c(28,28,3))}

new <- NULL

for (i in 1:length(temp)) {new <- rbind(new, mypic[[i]])}

newx <- new[,1:784]

newy <- c(7, 5,2,0, 5, 3)


# Prediction

pred <- model %>% predict_classes(newx)

table(Predicted = pred, Actual = newy)


prob <- model %>% predict_proba(newx)

cbind(prob, Predicted = pred, Actual = newy)
```

## Snapshot

```
Predicted 7
<matplotlib.image.AxesImage at 0x7f9b14242668>
```





```
Train: X=(60000, 28, 28), y=(60000,)
Test: X=(10000, 28, 28), y=(10000,)
```

## Result

The project was successfully completed and the digits were correctly recognised with an accuracy of 94%.