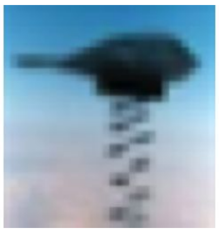
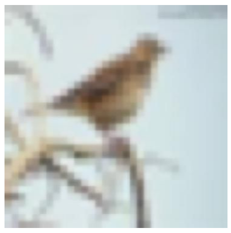
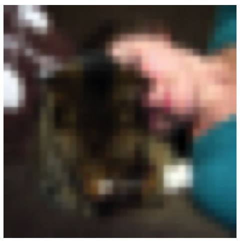
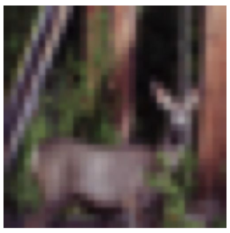
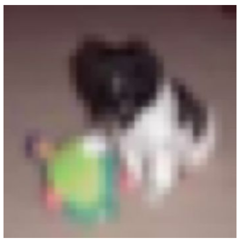
Final Project

STA141A

Instructor: Shuva Gupta

Jacob Fraysher
Haochen Zhan
Yuhui Li

3.
The table below shows the 10 different classes of images (and their numeric values) in this project along with an example from each group.

 Airplane (0)	Automobile (1)	 Bird (2)
 Cat (3)	 Deer (4)	 Dog (5)
Frog (6)	Horse (7)	Ship (8)
	Truck (9)	

After we calculated the standard deviation for each pixel in each of the three colors, we find that:

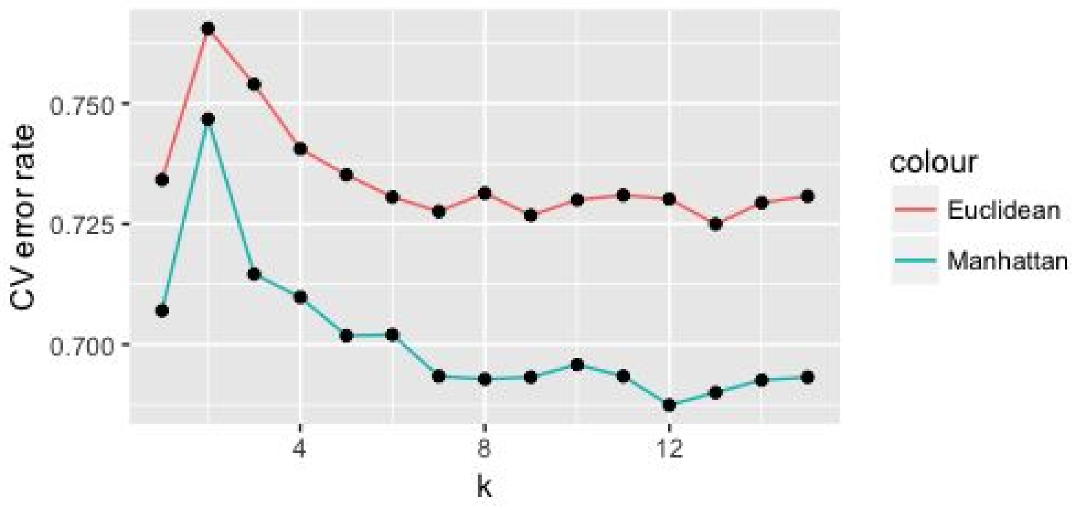
For the red channel:
Pixels in the region (1,1) are the *most* useful in classification.
Pixels in the region (12,15) are the *least* useful.

For the green channel:
Pixels in the region (1,1) are the *most* useful.
Pixels in the region (12,15) are the *least* useful

For the blue channel:
Pixels in the region (1,1) are the *most* useful
Pixels in the region (23,17) are the *least* useful

5.
We worked on the distance matrix by computing the 10-fold validation by dividing the indexes to ten equal size instead of dividing the whole training set to ten parts. It ran faster based on the method of using distance matrix.

6.



The above line plot shows that it's favorable to choose Manhattan method over the Euclidean method due to the lower CV error rates. The lower error rate would then help the efficiency of the predictions the program is created to make.

7.
We get three best k from CV error rate are 12,13,14 using Manhattan method to form distance matrices.
For:

k = 12, the confusion matrix indicates an accuracy rate of 0.3126
k = 13, accuracy rate = 0.31
k = 14, accuracy rate = 0.3074.

Note: The rows represent the predicted values of the images shown to the program while the the columns represent the values the image actually was. For example, the cell (0,0) in the first table shows that 264 images of airplanes were accurately predicted by the program to be airplanes while cell (0,1) shows that 58 pictures of automobiles were mistakenly seen as airplanes by the program.

	0	1	2	3	4	5	6	7	8	9
0	264	58	66	47	37	39	21	60	117	70
1	1	70	1	5	2	3	5	2	6	33
2	58	56	228	110	139	113	163	93	26	52
3	3	23	21	69	10	44	25	21	10	19
4	22	115	113	120	248	122	125	166	37	49
5	1	11	6	38	2	100	6	14	4	4
6	18	37	26	74	30	48	140	29	7	29
7	5	9	6	8	12	5	9	77	5	28
8	125	92	30	24	18	25	5	26	281	130
9	3	29	3	5	2	1	1	12	7	86

(k = 12)

	0	1	2	3	4	5	6	7	8	9
0	263	52	67	39	37	40	17	54	123	62
1	1	64	1	3	2	6	6	1	6	36
2	55	64	218	119	128	117	161	100	25	55
3	4	18	22	69	10	45	21	20	9	19
4	23	113	116	117	251	115	132	167	39	41
5	1	7	6	36	3	96	7	15	5	5
6	20	44	28	71	36	45	141	24	5	31
7	6	9	7	13	13	7	8	80	6	25
8	124	99	32	27	18	27	5	28	275	133
9	3	30	3	6	2	2	2	11	7	93

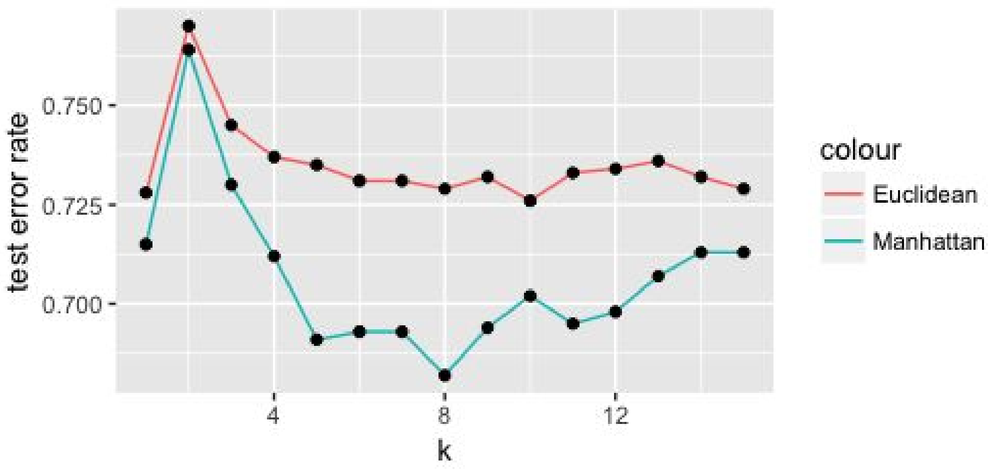
(k = 13)

	0	1	2	3	4	5	6	7	8	9
0	261	50	66	39	36	40	22	55	114	62
1	2	60	1	3	2	6	7	1	5	33
2	55	64	217	121	131	114	165	100	23	54
3	4	17	21	69	8	47	25	21	9	15
4	29	112	117	119	246	116	128	172	40	45
5	1	6	6	36	3	94	7	15	7	5
6	17	50	29	69	39	45	135	22	5	30
7	7	5	8	12	12	10	6	77	6	26
8	122	103	32	27	21	25	4	27	283	135
9	2	33	3	5	2	3	1	10	8	95

(k = 14)

8.
Refer to the confusion matrix in Q7 when k = 12 since it is the best combination overall. After recording the misclassified observations from the training set, we are able to see that our classifier is very accurate to some extent except when classifying the class “deer,” “bird,” “airplane,” and “ship.”

9.
The least CV error rate by using Manhattan method is 0.6874 compares to the least test error rate by the same method 0.682, which are very close. However, we prefer to use k=8 on test sets instead of k=12 which is dominant in cross validation. Overall, our classifier is better when applies on the test set than train set.



10. Contributions to our final project:

Haochen Zhan:
- Created the original R code for the questions.
- Generating ideas for the project.
- Created the original starting point for the report

Yuhui Li:
- Provided the idea and methods for the questions.
- Cooperated with creating and debugging the R code.

Jacob Fraysher:
- Troubleshooted the code for errors and to attain reproducibility of the results.
- Tidied the paper and code for easier viewing
- Corrected grammar and word choice for a better flow and display of information.

```
################# Appendix #################


is_installed = function(mypkg) is.element(mypkg, installed.packages()[, 1])

if(!is_installed("rmarkdown")){
  install.packages("rmarkdown")
}
if(!is_installed("ggplot2")){
  install.packages("ggplot2")
}
if(!is_installed("grid")){
  install.packages("grid")
}
if(!is_installed("parallelDist")){
  install.packages("parallelDist")
}
if(!is_installed("gridExtra")){
  install.packages("gridExtra")
}
library("rmarkdown") ; library("ggplot2") ; library("grid") ; library("parallelDist");
library("gridExtra")


#learned the way computer works in
http://l3d.cs.colorado.edu/courses/CSCI1200-96/binary.html
#read and write binary files in ?readBin
#https://stats.idre.ucla.edu/r/faq/how-can-i-read-binary-data-into-r/

###1.
load_training_images = function(in_dir = "C:/Users/Jacob/Documents/STA 141A",
                                out_file = "C:/Users/Jacob/Documents/STA 141A/training.rds"){
  list1 = c("data_batch_1.bin", "data_batch_2.bin", "data_batch_3.bin",
            "data_batch_4.bin", "data_batch_5.bin")
  setwd(in_dir)
  bin = list()
  train = list()
  for(i in list1){
    bin[[match(i, list1)]] = as.integer(readBin(con = i,what = "raw",
                                        n = 3073 * 10000, size = 1, endian = "big"))
  }
  for(i in 1:5){
    train[[i]] = matrix(bin[[i]], nrow = 10000, ncol = 3073, byrow = T)
  }
  bin_t = rbind(train[[1]], train[[2]], train[[3]], train[[4]], train[[5]])
  saveRDS(bin_t, out_file)

}

load_training_images()

#directory in function. refer to discussion note 10 by Jiahui Guan

#test
```

```
load_testing_images = function(in_dir = "C:/Users/Jacob/Documents/STA 141A",
                               out_file = "C:/Users/Jacob/Documents/STA 141A/testing.rds"){
  bin6 = as.integer(readBin(con = "test_batch.bin", what = "raw", n = 3073*10000, size =
1, endian = "big"))
  bin_te = matrix(bin6, ncol = 3073, nrow = 10000, byrow = T)
  saveRDS(bin_te, out_file)
}

load_testing_images()



#2
training = readRDS("training.rds")
testing = readRDS("testing.rds")
data_rescale = function(labels, k = 500)sort(as.vector(sapply(unique(labels),
function(i)which(labels == i))[1:k,]))
train2 = training[data_rescale(training[,1], k = 500),]
test2 = testing[data_rescale(testing[,1], k = 100),]

train2 = saveRDS(train2, "train2.rds")
test2 = saveRDS(test2, "test2.rds")

#getting the class names for image function.
dat.names = readLines("batches.meta.txt", n = 10)

test2 = readRDS("test2.rds")
train2 = readRDS("train2.rds")
view_image = function(x){
  img = train2[x, -1]
  r = matrix(img[1:1024], ncol = 32, byrow = TRUE)
  g = matrix(img[1025:2048], ncol = 32, byrow = TRUE)
  b = matrix(img[2049:3072], ncol = 32, byrow = TRUE)
  img_col = rgb(r, g, b, maxColorValue = 255)
  dim(img_col) = dim(r)
  grid.raster(img_col, interpolate = FALSE)
  classnum = train2[x, 1]
  print(dat.names[classnum + 1])
}



##3.
train2 = as.data.frame(train2)
train2 = train2[order(train2$V1),]
set.seed(6969)
view_image(sample(1:500, size = 1))
dev.copy2pdf()

#random select images in each class group. Save current plots refer to:
#https://stackoverflow.com/questions/26034177/r-saving-multiple-ggplots-using-a-for-loo
p/26078489?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=googl
e_rich_qa
```

```
for(i in 1:9){
  index = sample((i * 500 + 1):((i + 1) * 500), size = 1)
  file_name = paste("plot_", i, ".pdf", sep = "")
  pdf(file_name)
  view_image(index)
  dev.off()
}

#extract the data for three channels that are in 1024 pixels each.
#red
red = train2[, 2:1025]
names(red) = as.character(c(1:1024))
pix = function(x){c((x%/%32)+1,x%%32)}#determine the pixel location.
sapply(as.integer(names(tail(sort(sapply(red, sd)), 1))), pix)#larger sd() values are in tail.
sapply(as.integer(names(head(sort(sapply(red, sd)), 1))), pix)

#green
green = train2[, 1026:2049]
names(green) = as.character(c(1:1024))
sapply(as.integer(names(tail(sort(sapply(green, sd)), 1))), pix)
sapply(as.integer(names(head(sort(sapply(green, sd)), 1))), pix)

#blue
blue = train2[, 2050:3073]
names(blue) = as.character(c(1:1024))
sapply(as.integer(names(tail(sort(sapply(blue,sd)), 1))), pix)
sapply(as.integer(names(head(sort(sapply(blue,sd)), 1))), pix)



#4.
#change test2 and train2 back to matrix format.
test2 = readRDS("test2.rds")
train2 = readRDS("train2.rds")

########distance function.Consume about 20mins.
dist_mat = parDist(rbind(train2, test2)[, -1], method = "euclidean")
dist_mat = as.matrix(dist_mat)
##########

#change row and column names to observation classes.
rownames(dist_mat) = c(train2[, 1], test2[, 1])
colnames(dist_mat) = c(train2[, 1], test2[, 1])
saveRDS(dist_mat, "dist_mat.rds")
dist_mat = readRDS("dist_mat.rds")

#select test observations as prediction points.
predict_knn = function(test_index = 1:1000, train_index = 1:5000, dist_mat1 =
dist_mat[5001:6000,1:5000], k = 5){
  c(sapply(test_index, function(x) names(which.max(table(names(sort(dist_mat1[x,
train_index])[1:k]))))))
}
```

```r
#5
labels = names(dist_mat[, 1])
cv_error_knn = function(dist_mat1 = dist_mat[1:5000, 1:5000], k){
  mse = rep(0, 10)
  for (i in 0:9) {
    mse[i + 1] = sum(predict_knn(test_index = (500 * i + 1):(500 * (i + 1)),
                              train_index = -((500 * i + 1):(500 * (i + 1))),
                              dist_mat1, k) != labels[(500 * i + 1):(500 * (i + 1))]) / 500#10-fold
  }
  return(mean(mse))
}



#6
###euclidean method
cvknn = sapply(1:15, function(i) cv_error_knn(k = i))
cvknn

###method "manhattan"
dist_matm = as.matrix(dist(rbind(train2, test2)[, -1], method = "manhattan"))
##### 20mins consumption.

rownames(dist_matm) = c(train2[, 1], test2[, 1])
colnames(dist_matm) = c(train2[, 1], test2[, 1])
saveRDS(dist_matm, "dist_matm.rds")
dist_matm = readRDS("dist_matm.rds")
labels = names(dist_matm[, 1])
cvknnm = sapply(1:15, function(i) cv_error_knn(dist_mat1 = dist_matm[1:5000, 1:5000], k
= i))
cvknnm

#start plot two lines on ggplot2.
#https://stackoverflow.com/questions/3777174/plotting-two-variables-as-lines-using-ggplo
t2-on-the-same-graph?utm_medium=organic&utm_source=google_rich_qa&utm_campai
gn=google_rich_qa
cv6 = as.data.frame(cbind(c(1:15), cvknn, cvknnm))
ggplot(aes(x = V1), data = cv6) +
  geom_line(aes(y = cvknn, colour = "Euclidean")) +
  geom_line(aes(y = cvknnm, colour = "Manhattan")) +
  geom_point(aes(y = cvknn)) +
  geom_point(aes(y = cvknnm)) +
  labs(x = "k", y = "CV error rate")
order(cvknnm)



#7. confusion matrix. refer to https://www.youtube.com/watch?v=FAr2GmWNbT0
#prefer Manhattan method to Euclidean refers to Q6 plot.
head(order(cvknnm), 3)#best three k by CV error rate.

con = function(dist_mat1 = dist_matm[1:5000, 1:5000] , k){
  pred = c()
  for (i in 0:9) {
    pred_i = predict_knn(test_index = (500 * i + 1):(500 * (i + 1)),
                    train_index = -((500 * i + 1):(500 * (i + 1))), dist_mat1,k)
    pred = c(pred, pred_i)
}
  pred <<- pred #predictions are accessible from outside.
}

#when k = 12
con(k = 12)
test_l = names(dist_matm[1:5000, 1])

#grid the confusion matrix.
pred1 = pred
grid.table(table(pred1, test_l))
ar12 = sum(diag(table(pred1, test_l))) / 5000      #accuracy rate
ar12

#k = 13
con(k = 13)
pred2 = pred
grid.table(table(pred2, test_l))
ar13=sum(diag(table(pred2, test_l))) / 5000
ar13

#k = 14
con(k = 14)
pred3 = pred
grid.table(table(pred3, test_l))
ar14 = sum(diag(table(pred3, test_l))) / 5000
ar14



#8.Found in the Q7 that when k=12 with Manhattan method is the best combination
overall.
miscla = rep(0, 10)
tab = as.matrix(table(pred1, test_l))

for(i in 0:9){
  miscla[i + 1] = sum(tab[i + 1,]) - tab[i + 1, i + 1]
}
names(miscla) = dat.names
sort(miscla)



#9.
#By Euclidean method
#writing a function for the test error calculation.
test_error_knn = function(dist_mat1, k){
  test_lab = names(dist_mat[5001:6000, 1])
  sum(test_lab != predict_knn(test_index = 1:1000, train_index = 1:5000, dist_mat1, k)) /
1000
```

```
}

test_error_eud = sapply(1:15, function(i) test_error_knn(dist_mat1 = dist_mat[5001:6000,
1:5000], k = i))
test_error_eud

#Manhattan method
test_error_man = sapply(1:15, function(i) test_error_knn(dist_mat1 =
dist_matm[5001:6000, 1:5000], k = i))
test_error_man

#A plot similar to Q6
tt9 = as.data.frame(cbind(c(1:15), test_error_eud, test_error_man))
ggplot(aes(x = V1), data = tt9) +
  geom_line(aes(y = test_error_eud, colour = "Euclidean")) +
  geom_line(aes(y = test_error_man, colour = "Manhattan")) +
  geom_point(aes(y = test_error_eud)) +
  geom_point(aes(y = test_error_man)) +
  labs(x = "k",y = "test error rate")
sort(test_error_man)
sort(cvknnm)


################# End of Code #################
```