**PSEUDOCODE FOR BARCODE GENERATOR ALGORITHM**

**Description**

**Openimgs() iterates through each image in the MNIST folder and crops it to make a 18 by 18 image. This is to get rid of any whitespaces. This is now converted to a array and appended to an empty there dimensional array. The function then returns this three dimensional array. This is with indices 0-9 representing 0 , 10-19 are ones and so forth.**

**The barcode generator function then takes the radon projections of the image using The sci-kit package in python and a loop. They will thus increase the degrees of the projections by 15 degrees. The projection is saved in a array and referenced to the threshold value. The threshold value is calculated uing the mean of pixels in a projection. A 1 is then placed in the barcode otherwise a 0 will be placed. Finally the barcode is converted to a string data type and returned**

**The barcodeGen function then creates barcodes for the dataset and opens a text file to save the barcodes to . The text file is called barcode.txt. This is where each line corresponds to a image in the MNIST Folder. O**

**Barcodes.txt is overwritten at every code execution.**

**These are all to determine the hit ratio for a query image and return the most close match**

**Function** Openimgs()  
 initialized a cropped numpy array

**For all** images in MNIST dataset

Crop images to rows and column 4-22

Append image to cropped numpy array

**Return** image

**Function** Barcode\_generator()

Initialize numpy array

**For** angle 0-165 degrees with a 15 degree increment

Projection<-radon projection of angle

Threshold <- mean of the array of radon projections

**For** i in 0 to projection length

If (projection[i] >= threshold)

Projection[i] <-1

Else

Projection[i]<-0

Append projection to array

stringcode <- empty string

**For** size of code arr

convert code array entries to string and append

return stringcode

**Function** Barcodesetgen():

//This is to create a binary code for the dataset in the case of the barcode.txt file is nonexistent

Images = Openimgs()

**For** each image in the dataset

Code <- Barcode\_generator()

Write code to end of barcodes.txt file

**Close file**

Pseudocode for the search algorithm

**Function findalgo()**

**Queryimg<-** cropped to rows and columns 4-22

Open and accesss text fiile with codes of images

Strokes<\_ array of barcodes in txt

Check difference

**Close** txt

Distancefrh<- 217(len(Distancfrh)+1)

Indx<\_None

For j from 0- lengthofStrokes

Strokes 🡨 Strokes[j]

If len(Distancefrh)!+ len(Stroke)

Throw(exception)

Else:

For j from 0 to length of barcode

If code[j] ==stroke[j]

Distancefrh—

If distancefrh<-stored.d and Distancefrh not 0

Stored.d <-distancefrH

Index<-j

Return DistancefrH,code,Strokes[indx]

Matchingnum 🡨 (ind/10) //Since the position must be divided by 10

Function findall ()  
for img in MNIST

Findalgo(img)

Explanation :

We first access the function with the query image and store the created barecode in a string variable. This then accesses the barcode.txt file that will allow us to work with. We then compare the Hamming distances of the 2 barcodes found in the dataset and we use it to find similairities between them. Each barcode is then compared against the image query to find the closest match. The closest match is then saved.