1. **Take the standard MNIST handwritten digit dataset, and as usual, split it into training and testing data. Treat each image as a vector. For all the test images, calculate the nearest neighbor from the training data, and report this label as the prediction.**
    - **How accurate is this method?**
    - **What metric did you use for distance?**
    - **How fast/slow is your implementation? can you analyze the bottlenecks and speed things up?**
    - **Any ideas on improving accuracy?**

Above problem is solved by Sequential Neural Network method

Here 3 layers are used

1st layer: It converts the greyscale image into list of array

2nd layer: It is the hidden layer which labels the data

3rd layer: It is the image itself

- How accurate is this method?

By using above method we got accuracy as 97.7% which is approximately equal to 98%

- What metric did you used for distance?

- How fast/slow is your implementation? can you analyse the bottlenecks and speed things up?

  Time required to reach above accuracy was 20sec

- Any ideas on improving accuracy?

  To increase the accuracy we can increase the no of epochs

  Also we can increase the number of dense layers

2)Look up what the Arithmetic-Geometric mean of two numbers is.

- **Given two numbers, calculate their AGM**
- **Decide on a range of numbers and for every pair in this range, calculate their AGM and plot it**

Approach

So the first thing we did:divide up the given task.We have divided the task into 4

a.Finding the Arithmetic-Geometric mean

b.Given a range find all the possible pairs in that range

c.Plotting the AGM for each pair

d.Optimizing the step 1 and 2

Step 1:

From all the information we collected ,

We found out 2 methods to compute the Arithmetic-Geometric mean

- Using the series of approximations
- Using Integral-form expression

Approaching Method 1:

1.a)AGM for given pair is equal to the limit of the sequence of Arithmetic mean and Geometric mean:Ex:Source:Wikipedia

To find the arithmetic–geometric mean of $a_0 = 24$ and $g_0 = 6$, iterate as follows:

$$\begin{aligned} a_1 &= \tfrac{1}{2}(24+6) &&= 15 \\ g_1 &= \sqrt{24\cdot 6} &&= 12 \\ a_2 &= \tfrac{1}{2}(15+12) &&= 13.5 \\ g_2 &= \sqrt{15\cdot 12} &&= 13.416\,407\,8649\ldots \\ &\vdots \end{aligned}$$

The first five iterations give the following values:

n	a_n	g_n
0	24	6
1	15	12
2	13.5	13.416 407 864 998 738 178 455 042...
3	13.458 203 932 499 369 089 227 521...	13.458 139 030 990 984 877 207 090...
4	13.458 171 481 745 176 983 217 305...	13.458 171 481 706 053 858 316 334...
5	13.458 171 481 725 615 420 766 820...	13.458 171 481 725 615 420 766 806...

The number of digits in which a_n and g_n agree (underlined) approximately doubles with each iteration. The arithmetic–geometric mean of 24 and 6 is the common limit of these two sequences, which is approximately 13.458 171 481 725 615 420 766 813 156 974 399 243 053 838 8544.[1]

After surveying more about series of Approximation for AGM-we ended up defining the following function for AGM

```
def agm(x, y, tol=pow(10,10)):
    xn, yn = (x + y) / 2.0, sqrt(x * y)
    while abs(xn - yn) > tol:                    #looping  until abs(x-y)>10^10
        xn, yn = (xn + yn) / 2.0, sqrt(xn * yn)
    return xn
```

1.b)we did this by using 2 for-loops and iterating it in the given range

1.c)plotted using matplotlib's pyplot

1.d) The computations here are ofO(n^2) ,so we tried optimizing it using another method(method 2) i.e.,Integral form

2.a) Defined a function for AGM using the procedure as in Resource :[Wikipedia]

The geometric mean of two positive numbers is never bigger than the arithmetic mean (see inequality of arithmetic and geometric means). As a consequence, for $n > 0$, (g_n) is an increasing sequence, (a_n) is a decreasing sequence, and $g_n \le M(x, y) \le a_n$. These are strict inequalities if $x \ne y$.

$M(x, y)$ is thus a number between the geometric and arithmetic mean of x and y; it is also between x and y.

If $r \ge 0$, then $M(rx,ry) = r\,M(x,y)$.

There is an integral-form expression for $M(x,y)$:

$$M(x,y) = \frac{\pi}{2}\left(\int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{x^2\cos^2\theta + y^2\sin^2\theta}}\right)^{-1}$$
$$= \frac{\pi}{4}\cdot\frac{x+y}{K\left(\frac{x-y}{x+y}\right)}$$

where $K(k)$ is the complete elliptic integral of the first kind:

$$K(k) = \int_0^{\frac{\pi}{2}} \frac{d\theta}{\sqrt{1-k^2\sin^2(\theta)}}$$

Indeed, since the arithmetic–geometric process converges so quickly, it provides an efficient way to compute elliptic integrals via this formula. In engineering, it is used for instance in elliptic filter design.[3]

```
def f(theta):      # declaring f(θ) for ∫ f(θ) dθ :K(k)
        return (1/math.sqrt((math.pow(k,2)*math.pow(sin(theta),2))))

def K(k):       #single integration using quad()-numpy over range 0,pi/2)
        res,err=quad(f,0,pi/4,limit=50)
        return res
def agm():
    if(K(k)>0):
        return ((pi/4)*_sum/K(k))
```

We got a error here , for a given pair we got integration waring as follows:
/home/lenovo/.local/lib/python3.5/site-packages/ipykernel_launcher.py:14: IntegrationWarning: The maximum number of subdivisions (50) has been achieved.
If increasing the limit yields no improvement it is advised to analyze the integrand in order to determine the difficulties. If the position of a local difficulty can be determined (singularity, discontinuity) one will probably gain from splitting up the interval and calling the integrator on the subranges. Perhaps a special-purpose integrator should be used.

In order to solve this issue we worked on the number of division and increased "limit=100","limit=200",etc.Though we eliminated the error at one time we found out that our answer was changing throughout the process.

So, we proceeded further to work on the line highlighted above in the warning:
In the analysis of the integrand we found out that $K(k)$ is the complete elliptic integral of the first kind and that inorder to get the more accuracy we need to implement it using complete elliptic integral of the second kind.Well, we are still working on the function K(k) and trying search on the best K(k) which could suit our purpose and thus optimize the system to its best needs.
So, would conclude it by saying that though we got the accuracy using the Method 1, i.e.series of approximations ,which is more a brute force approach.We are still working on optimization of AGM using method 2 ,i.e., integral form.