Μηχανική Μάθηση

Εργασία 2

Όνομα : Ζαχαράκης Αλέξανδρος

AM:1066662

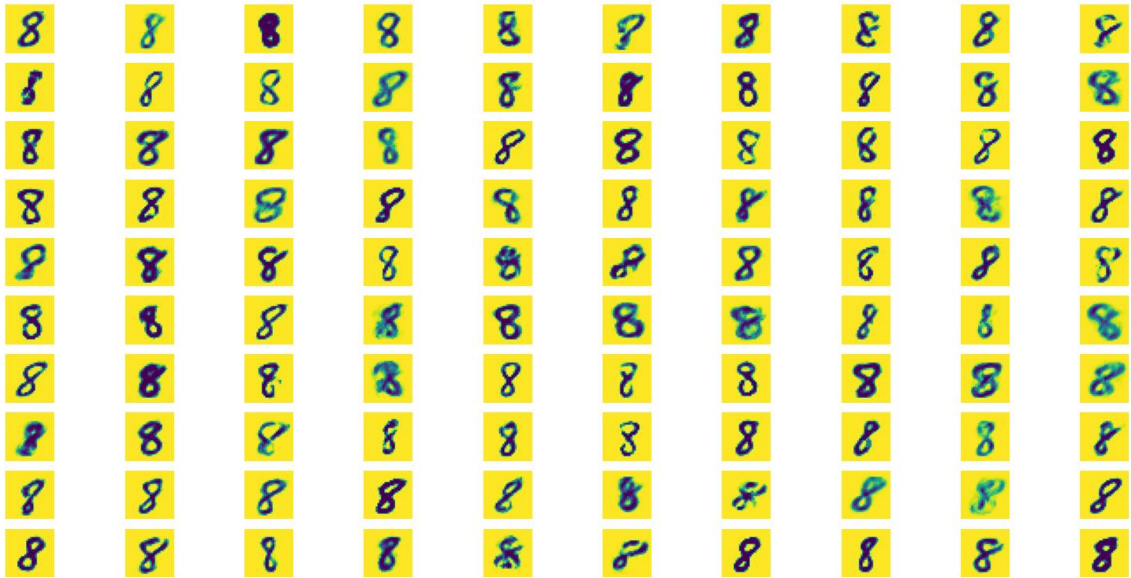
Τμήμα: Ηλεκτρολόγων Μηχανικών και τεχνολογίας υπολογιστών

Ερώτημα 1:

Για το πρώτο ερώτημα δημιουργήσαμε 100 10αδες από τυχαίες υλοποιήσεις με Gaussian κατανομή ώστε να τις περάσουμε ως είσοδο σε ένα Generative Model για την δημιουργία χειρόγραφων 8. Οι πίνακες που χρειαζόμαστε μας δίνονται και η μορφή του δικτύου είναι η παρακάτω.

$$
\begin{aligned}
W_1 &= A_1 * Z + B_1 \\
Z_1 &= \max\{W_1, 0\} \text{ (ReLU)} \\
W_2 &= A_2 * Z_1 + B_2 \\
X &= 1./(1 + \exp(W_2)) \text{ (Sigmoid).}
\end{aligned}
$$

Το ερώτημα ήταν αρκετά απλό και στην ουσία είναι το πρώτο στάδιο forward passing ενός Νευρωνικού δικτύου. Στην περίπτωση μας το Generative model ήταν ήδη εκπαιδευμένο να δημιουργεί 8αρια και γι’ αυτό τα αποτελέσματα είναι πολύ καλά εξ αρχής. Παρακάτω παρουσιάζονται τα 100 8αρια.



Ερώτημα 2:

Σε αυτό το ερώτημα βλέπουμε πώς μπορούμε να κάνουμε inpainting σε εικόνες δηλαδή να ανακτήσουμε χαμένη πληροφορία, από τις οποίες μας έχουν αφαιρεθεί δεδομένα. Στην περίπτωση μας, μας δίνονται 4 εικόνες για testing και 4 εικόνες στις οποίες έχει προστεθεί τυχαίος θόρυβος. Στις εικόνες που έχει προστεθεί ο θόρυβος θέλουμε να τους αφαιρέσουμε στοιχεία ώστε να περάσουμε την επεξεργασμένη εικόνα σε ένα Neural network το οποίο θα μας δώσει σαν έξοδο την αρχική εικόνα.

Αρχικά υπολογίσαμε τον πίνακα T δηλαδή τον μετασχηματισμό στον οποίο θέλουμε να υποβληθούν οι εικόνες με τον θόρυβο. Ο πίνακας T είναι της μορφής T=[I 0] δηλαδή είναι ένας πίνακας διαστάσεων Nx784 (όσα και τα συνολικά pixel μιας 28x28 εικόνας) ο οποίος περιέχει 1 στην κύρια διαγώνιο του υποπίνακα NxN και μηδενικά σε όλα τα άλλα στοιχεία του. Με αυτόν τον τρόπο κρατάμε πληροφορία διάστασης N από την εικόνα μας. Στο πρόγραμμα έγιναν test για να βρούμε ποια τιμή του N είναι η ελάχιστη που μπορούμε να έχουμε ώστε το δίκτυο μας να λειτουργεί ικανοποιητικά. Στην συνέχεια δημιουργήσαμε 20 10άδες από τυχαίες υλοποιήσεις με gaussian κατανομή για να τις περάσουμε ανά μία 10άδα σαν είσοδο στο δίκτυο μας. **<u>Αυτό έγινε ώστε να δούμε για ποια 10άδα έχουμε το μικρότερο κόστος και να δείξουμε τα καλύτερα αποτελέσματα.</u>**

Όσον αφορά το δίκτυο, ανάλογα τον αριθμό N κάνουμε επαναληπτικά gradient descent ως προς Z για να βρούμε την βέλτιστη έξοδο και το καλύτερο κόστος.

Για τον λόγο αυτό χρειάστηκε να υπολογίσουμε την παράγωγο του κόστους J(Z) ως προς Z, όπου J(Z)=$Nlog(\|TX - Xn\|^2) + \|Z\|^2$ μέσω της διαδικασίας του backpropagation. Για τον υπολογισμό αυτής της παραγώγου ως προς την είσοδο εφαρμόσαμε τον κανόνα της αλυσίδας στο κομμάτι που περιέχει ο λογάριθμος καθώς εκεί έγκειται η δυσκολία στον υπολογισμό της παραγώγου.

Ορίσαμε ως Φ(X)= $log(\|TX - Xn\|^2)$ και υπολογίσαμε την παράγωγο αυτήν ως προς X.

Οπότε u2 $= \nabla x\Phi(X)=\frac{2(TX-Xn)T}{\|TX-Xn\|^2}$

Στην συνέχεια πολλαπλασιάσαμε elementwise το u2 με τον πίνακα που δημιουργήθηκε από την παράγωγο της sigmoid όταν σε αυτήν βάλαμε ως μεταβλητή το W2 που έχουμε υπολογίσει από το forward pass. Η παράγωγος της σιγμοειδούς είναι $f_2'(x) = -\frac{e^x}{(1+e^x)^2}$

Άρα v2 $=$ u2 $\circ f_2'(x)$

Στην συνέχεια κάναμε πολλαπλασιασμό μεταξύ των πινάκων v2 και ${A_2}^T$ και υπολογίσαμε u1 = ${A_2}^T * v2$

Έπειτα υπολογίσαμε την παράγωγο της ReLu ως $f_1'(x) = \begin{cases} 0\,, x \leq 0 \\ 1\,, x > 0 \end{cases}$ για χ=W1 και πολλαπλασιάσαμε elementwise με το u1

Άρα v1 $=$ u1 $\circ f_1'(x)$

Τέλος υπολογίσαμε το u0=${A_1}^T * v1$ το οποίο είναι το$\nabla_z[\log(\|TX - Xn\|^2)]$

Εν τέλη το u0 έχει μέγεθος 1χ10 και ∇_zJ(Z)=N*u0+2*Z.

Επιπλέον στην τελική παράγωγο εφαρμόστηκε και η μέθοδος ADAMS κατά την οποία υπολογίσαμε την ισχύ της παραγώγου και διαιρέσαμε το gradient για πιο ομαλή σύγκλιση. Την χρονική στιγμή 0 η ισχύς αρχικοποιείται ως το τετράγωνο τ
ης παραγώγου του J(Z) και στην συνέχεια ορίζεται ως

$P(J(Z))_t = P(J(Z))_{t-1} + \lambda J(Z)^2$

Εν τέλη το gradient descend έχει την μορφή

Z=Z-learning_rate*∇_zJ(Z)/$\sqrt{power + c}$ όπου c= 0.0000001 ώστε να αποφύγουμε την διαίρεση με το 0.

Παρακάτω θα παρουσιάσουμε τα αποτελέσματα του κώδικα.

Τα αποτελέσματα θα έχουν την εξής μορφή:

1. Φωτογραφία σύγκλισης συνάρτησης κόστους.
2. Φωτογραφία με αριστερά την ιδανική εικόνα στην μέση την επεξεργασμένη και δεξιά την έξοδο του Νευρωνικού δικτύου.

Εικόνα 1:

N=500 , learning rate=0.001 repetitions για το νευρωνικό=2000 και λ_adams=0.1

Συνάρτηση κόστους:

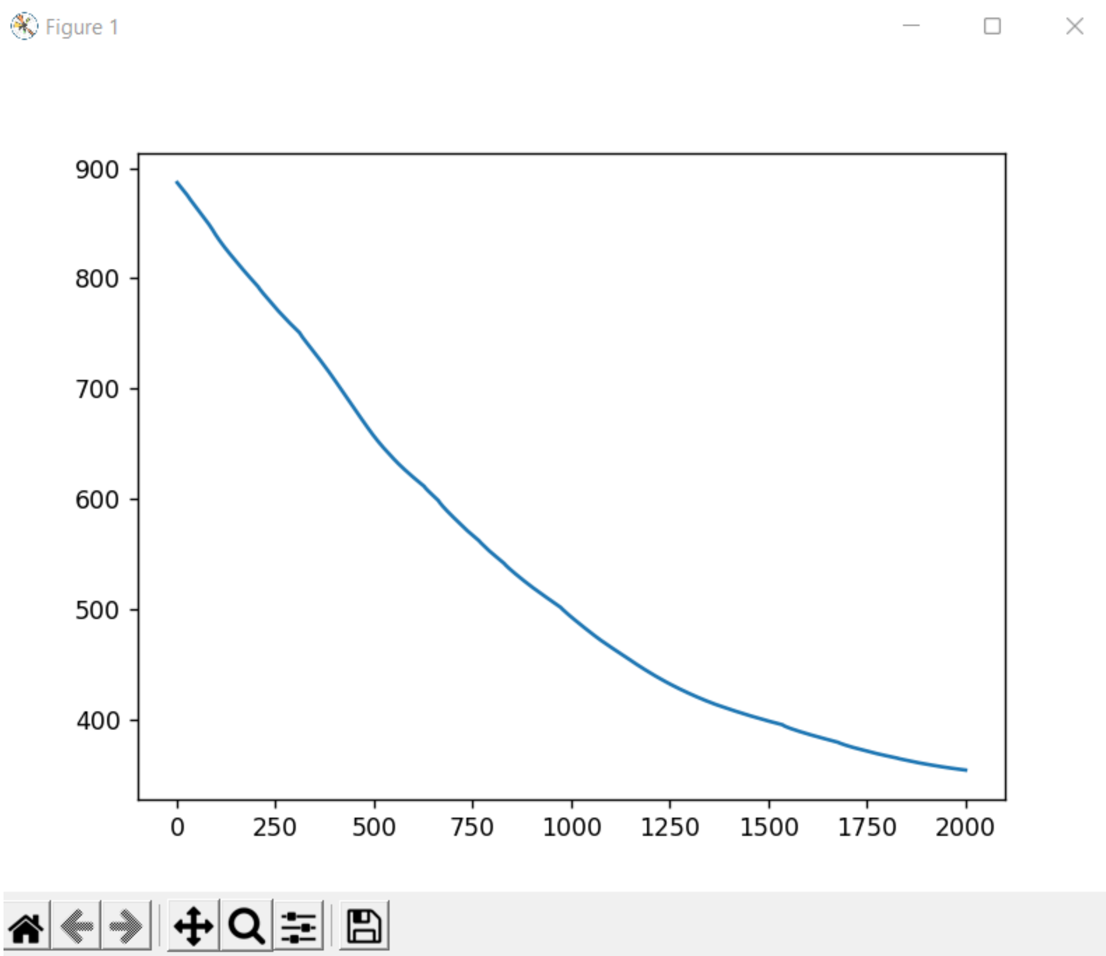
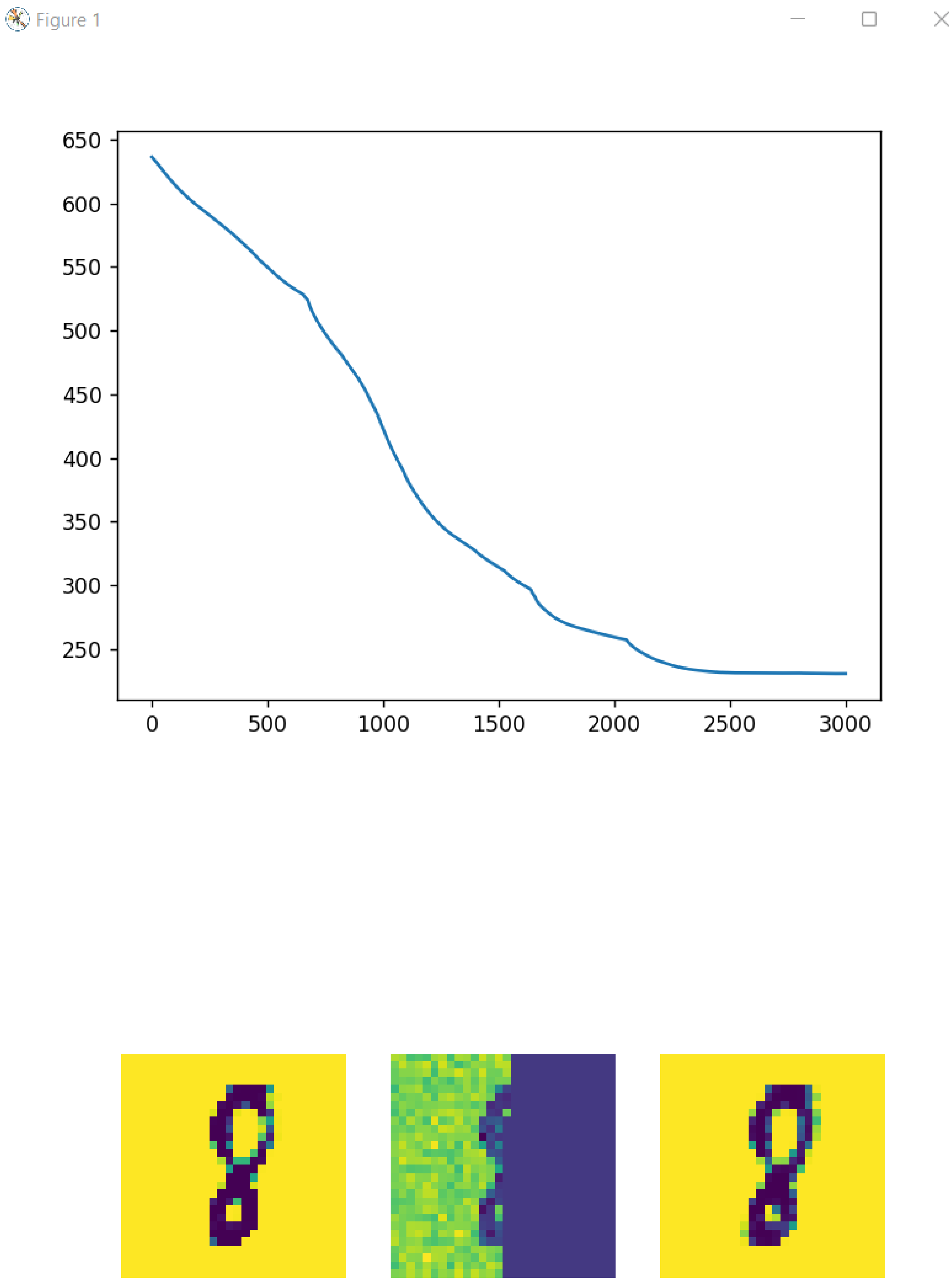
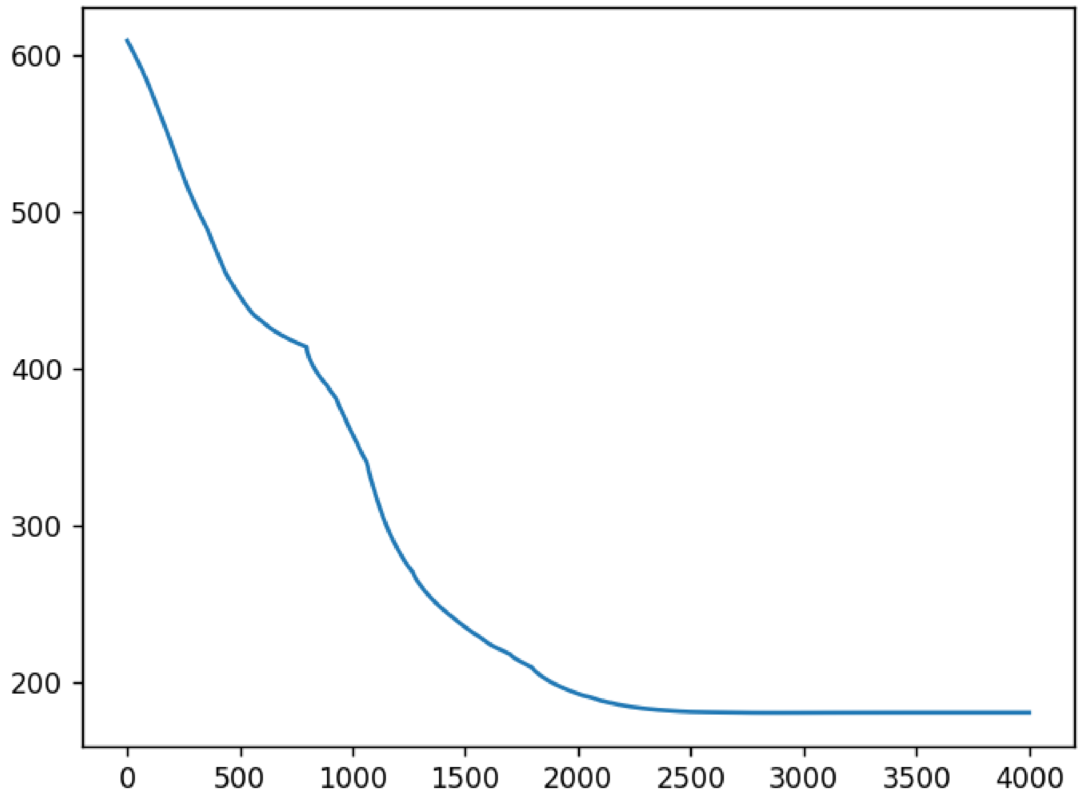


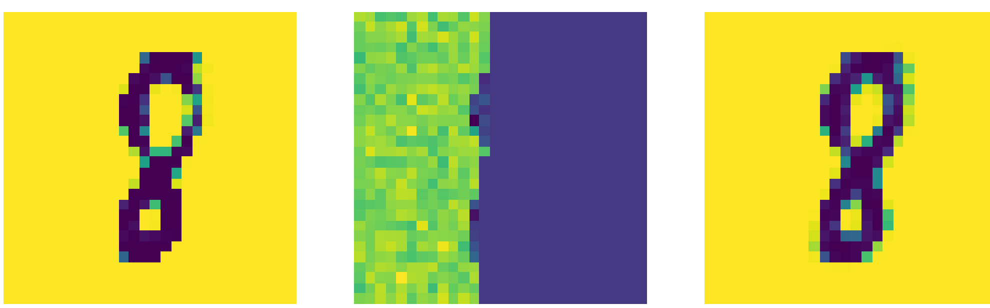

Figure 1

N=400 , learning rate=0.001 repetitions = 3000 και λ_adams=0.1

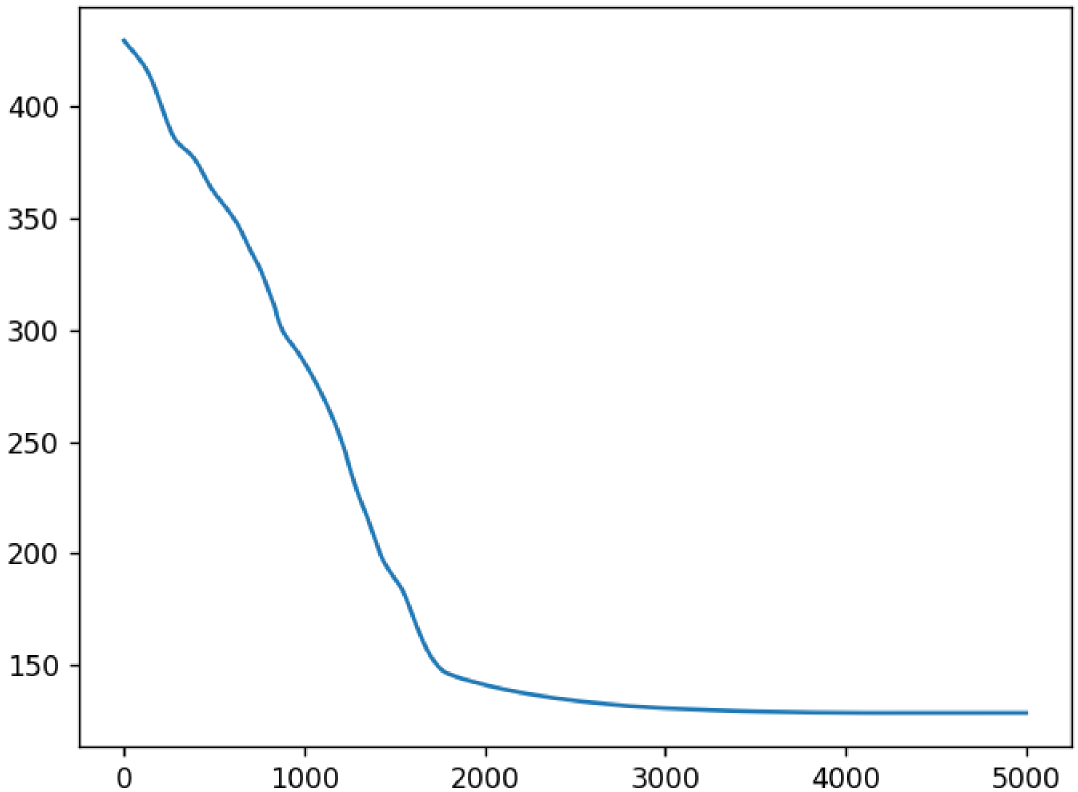


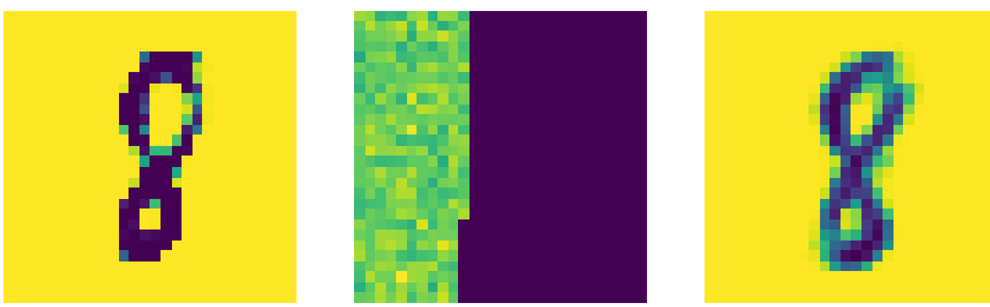
N=350 , learning rate=0.001 repetitions = 4000 και λ_adams=0.1





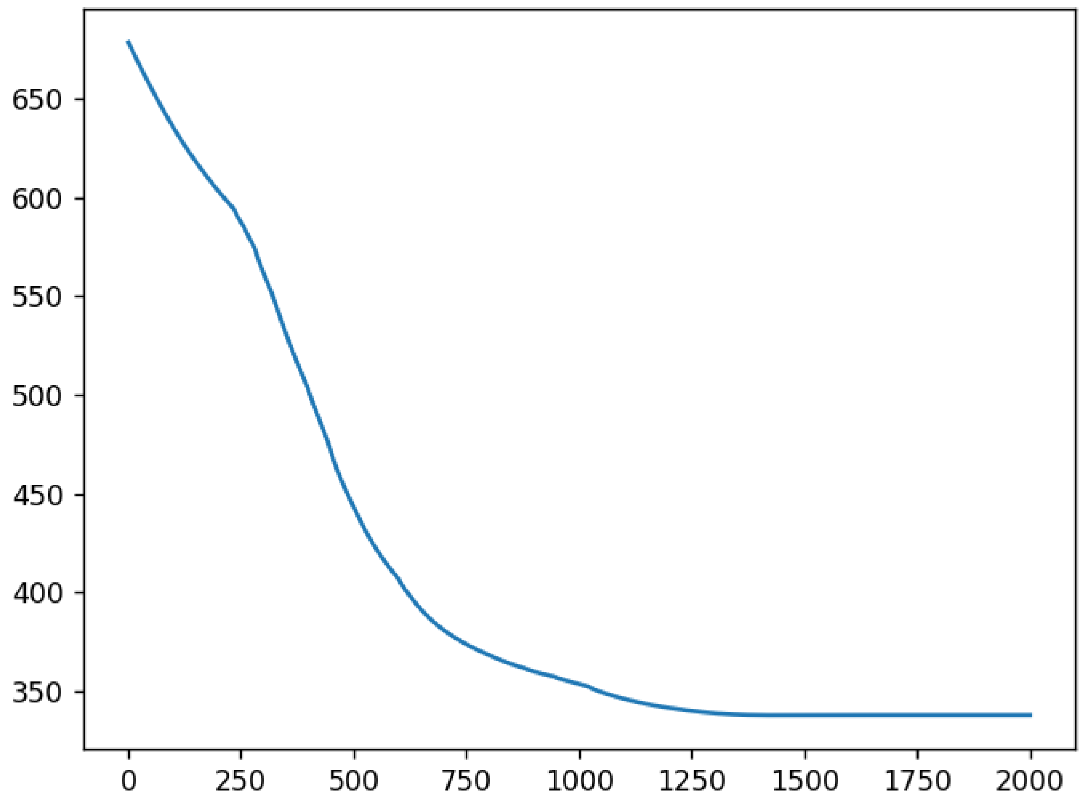
N=300 , learning rate=0.001 repetitions = 5000 και λ_adams=0.1

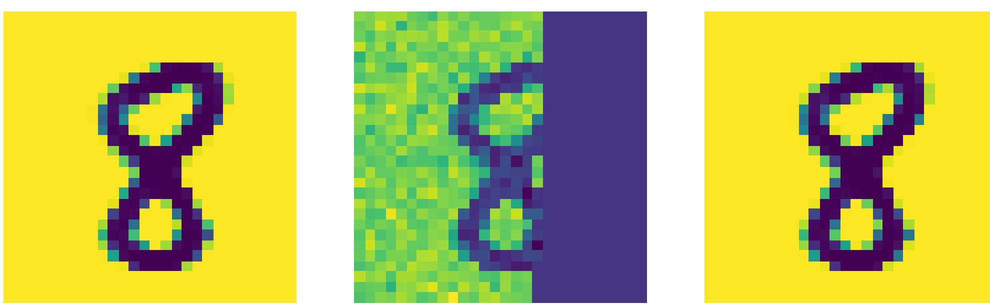




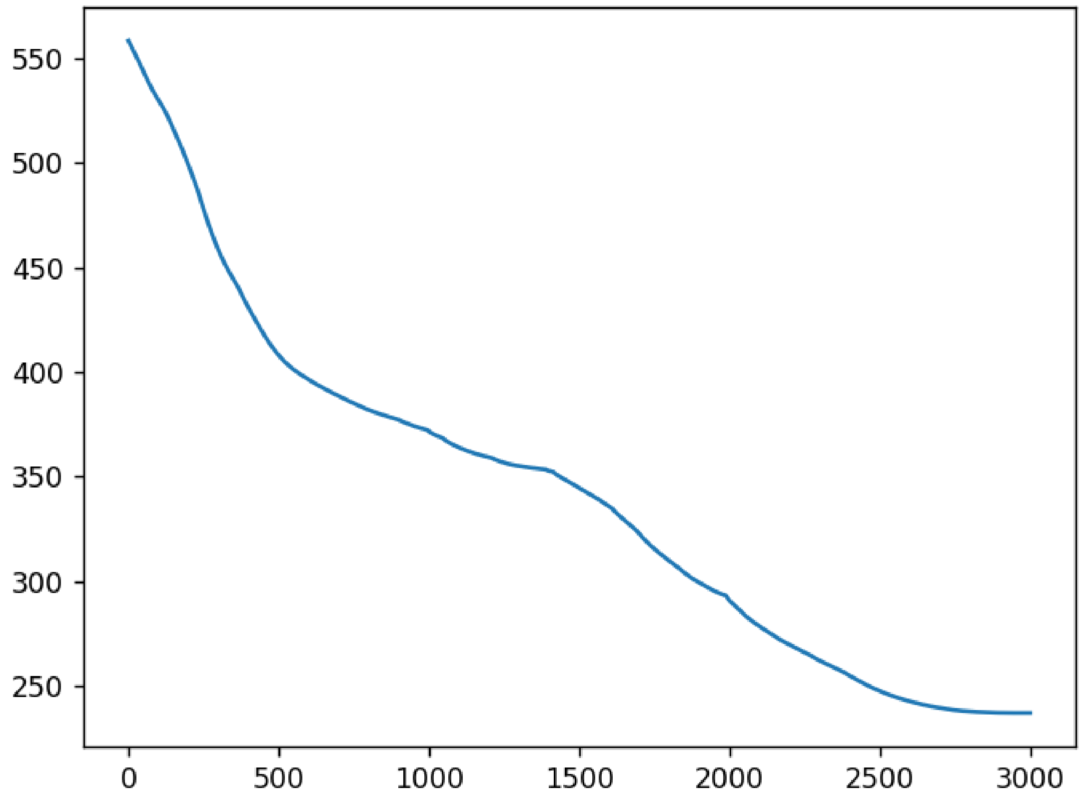
Εικόνα 2:

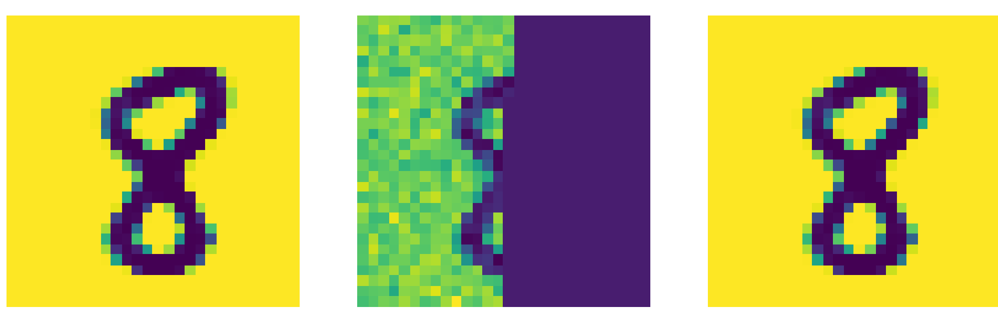
N=500 , learning rate=0.001 repetitions για το νευρωνικό=2000 και λ_adams=0.1



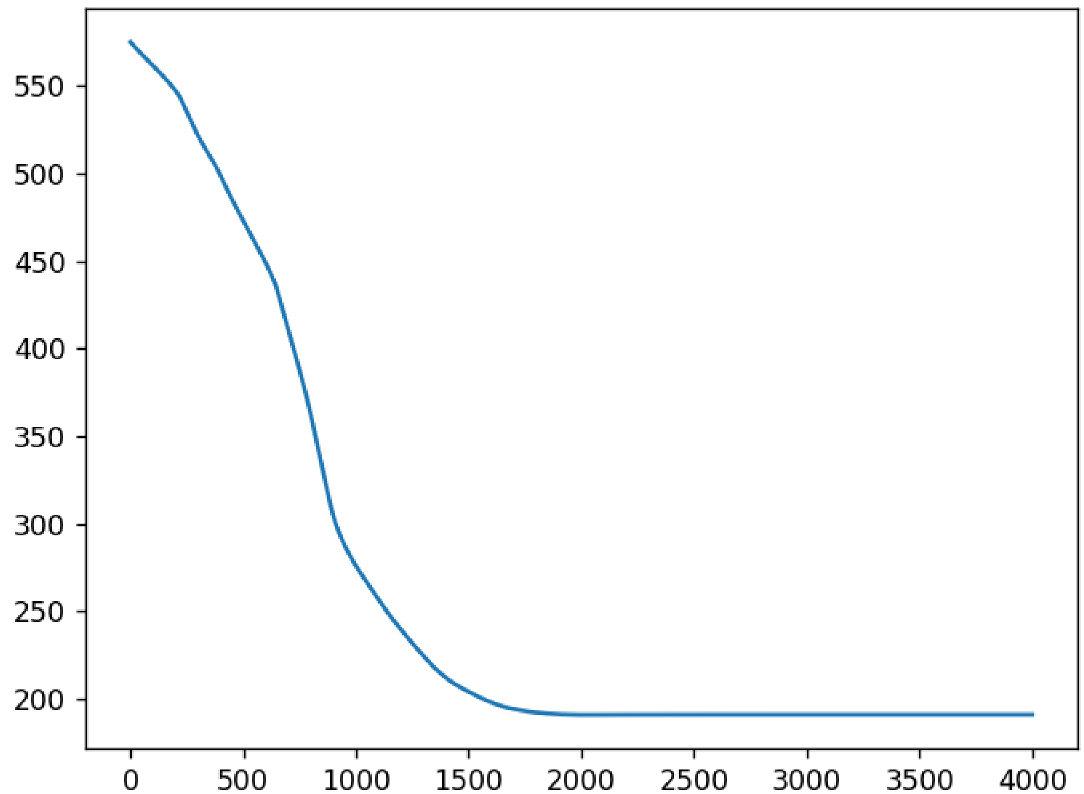


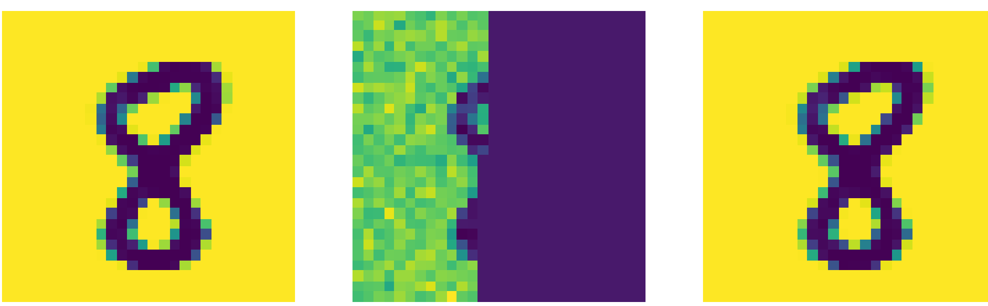
N=400 , learning rate=0.001 repetitions για το νευρωνικό=3000 και λ_adams=0.1



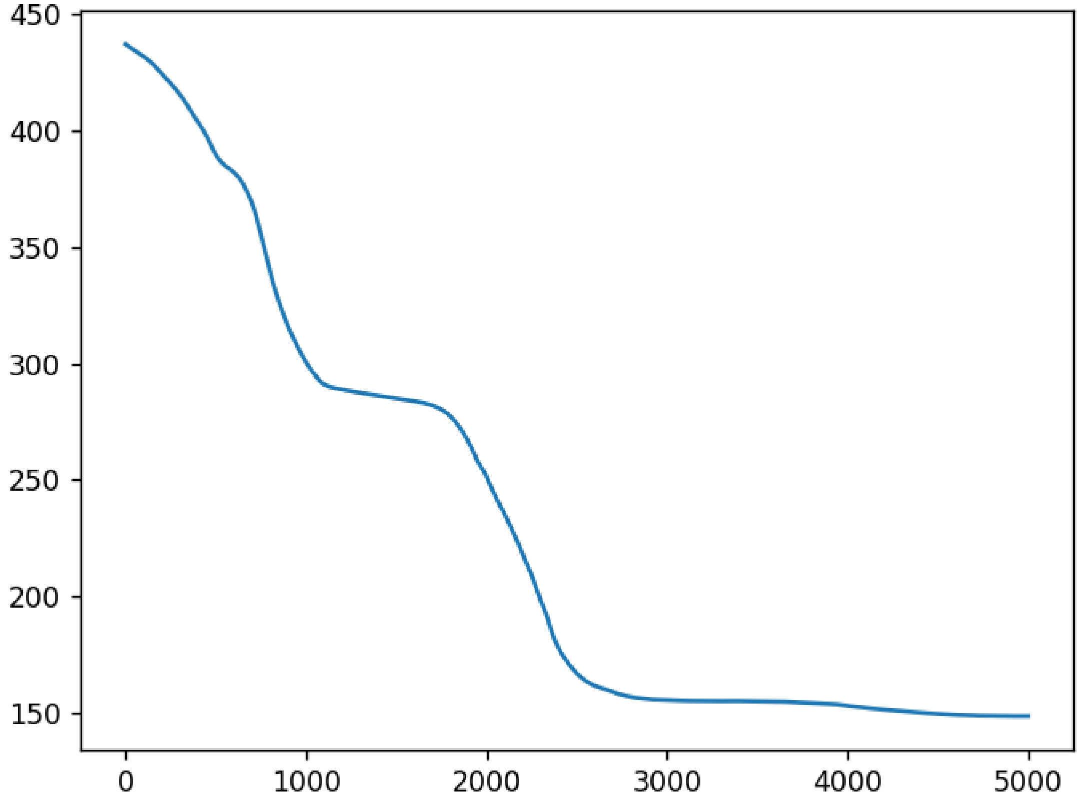


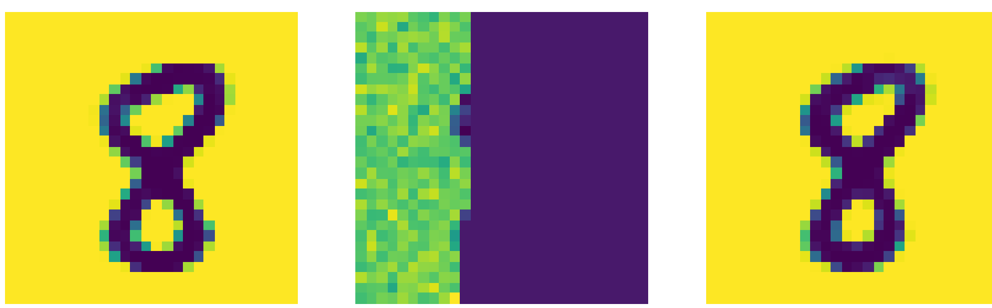
N=350 , learning rate=0.001 repetitions για το νευρωνικό=4000 και λ_adams=0.1





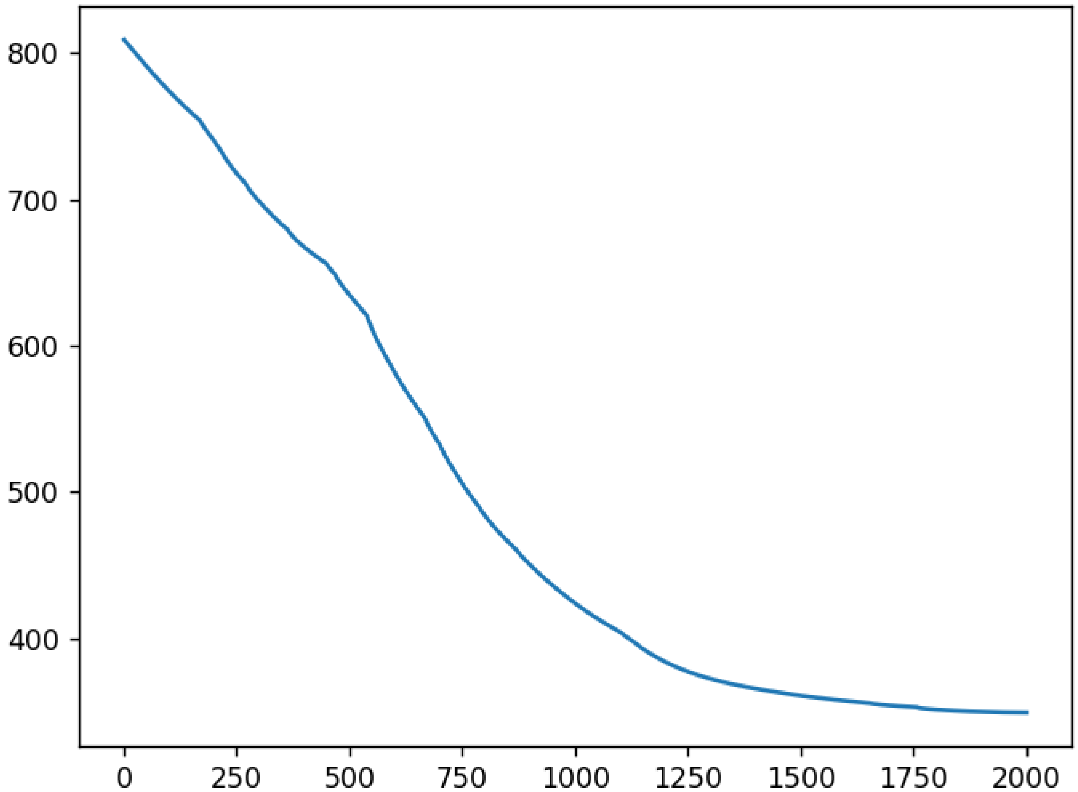
N=300 , learning rate=0.001 repetitions για το νευρωνικό=5000 και λ_adams=0.1

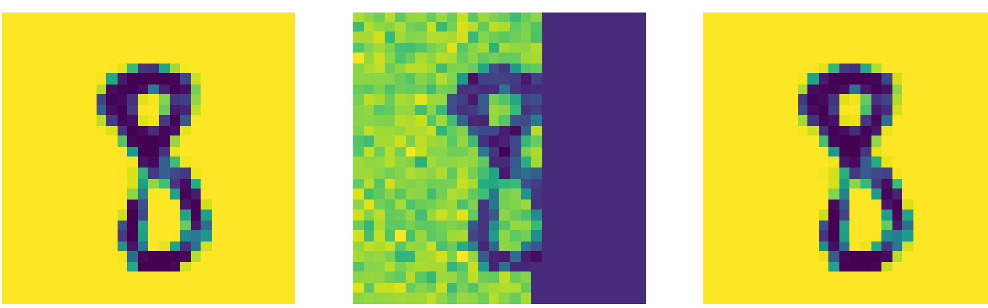




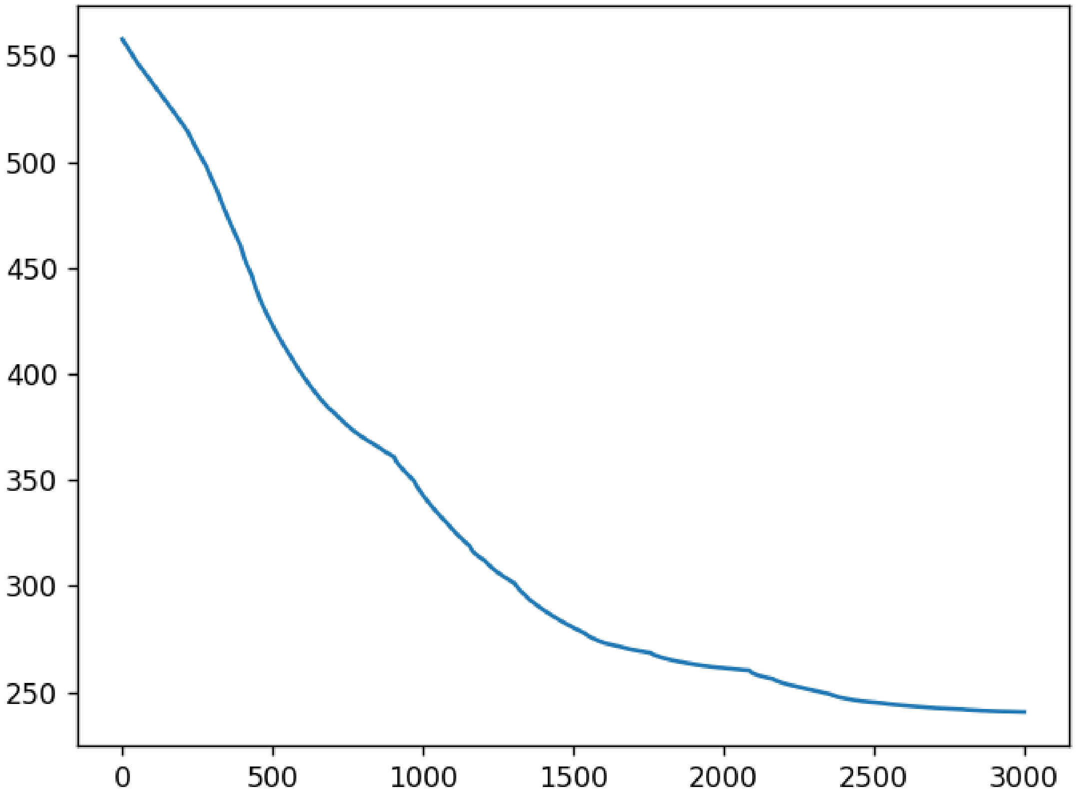
Εικόνα 3:

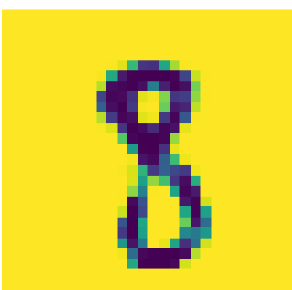
N=500 , learning rate=0.001 repetitions = 2000 και λ_adams=0.1

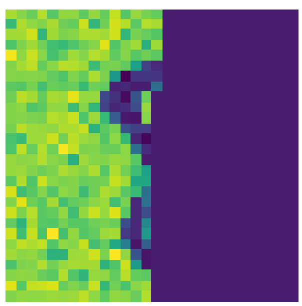


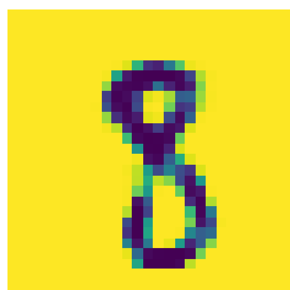


N=400 , learning rate=0.001 repetitions = 3000 και λ_adams=0.1

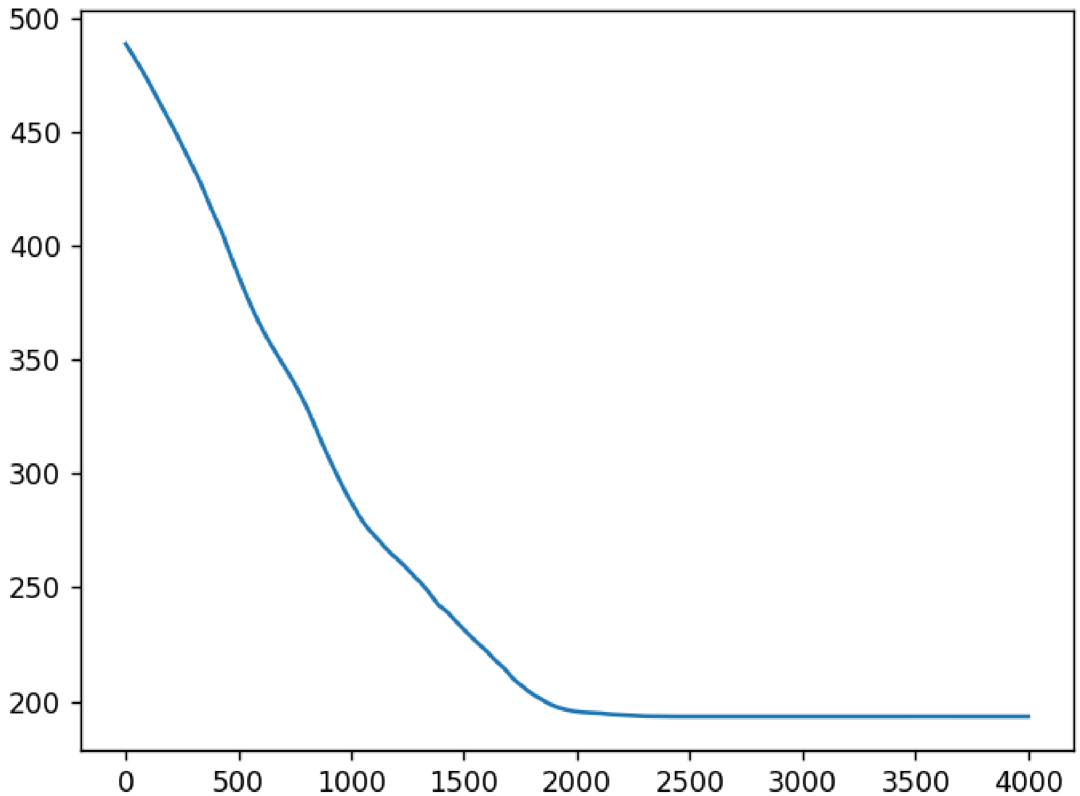


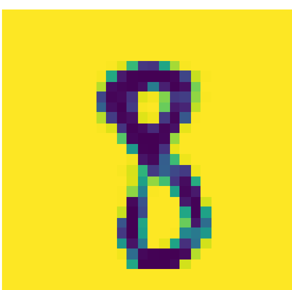


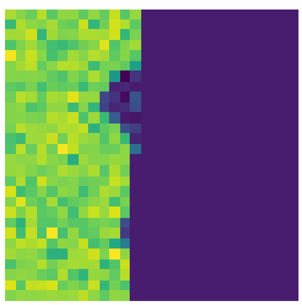


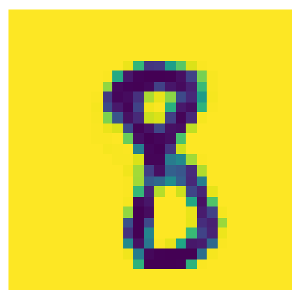


N=350 , learning rate=0.001 repetitions = 4000 και λ_adams=0.1

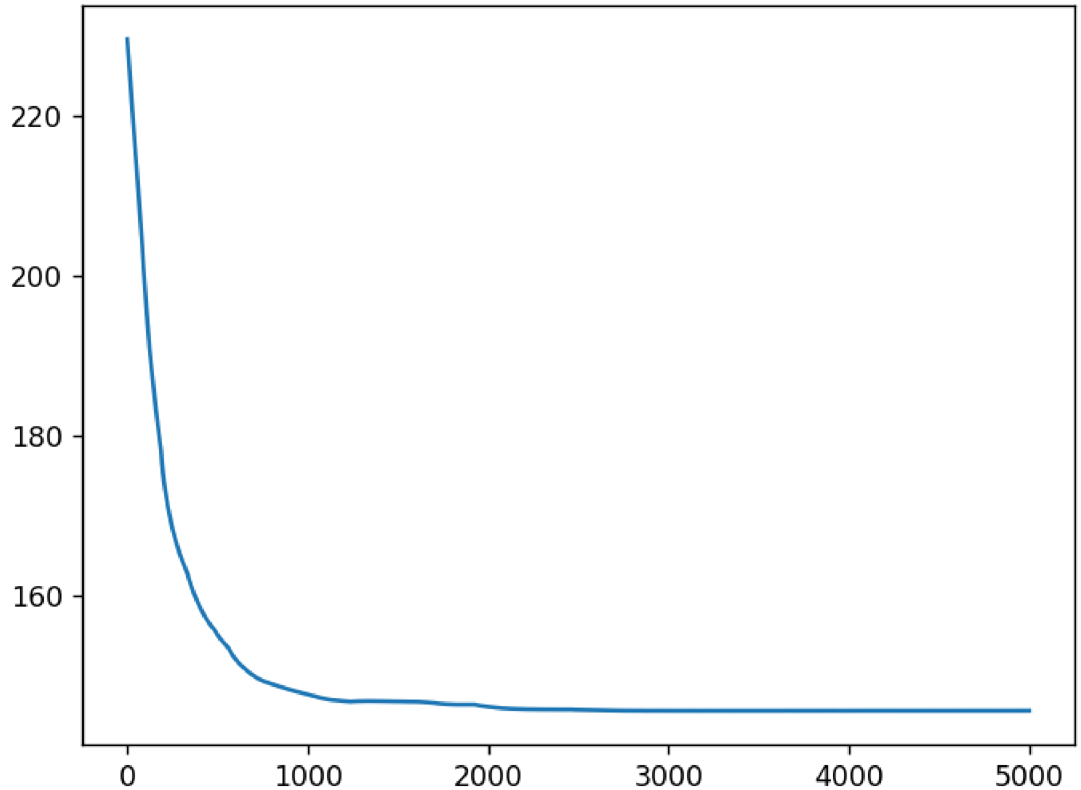


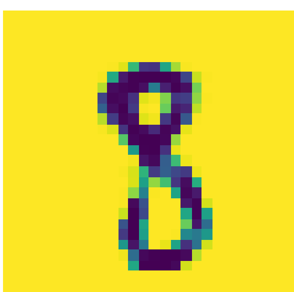


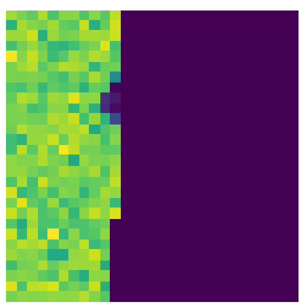


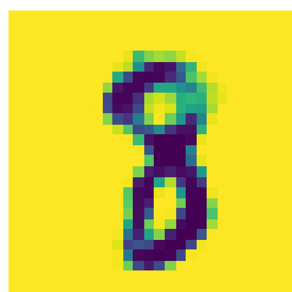


N=300 , learning rate=0.001 repetitions = 5000 και λ_adams=0.1



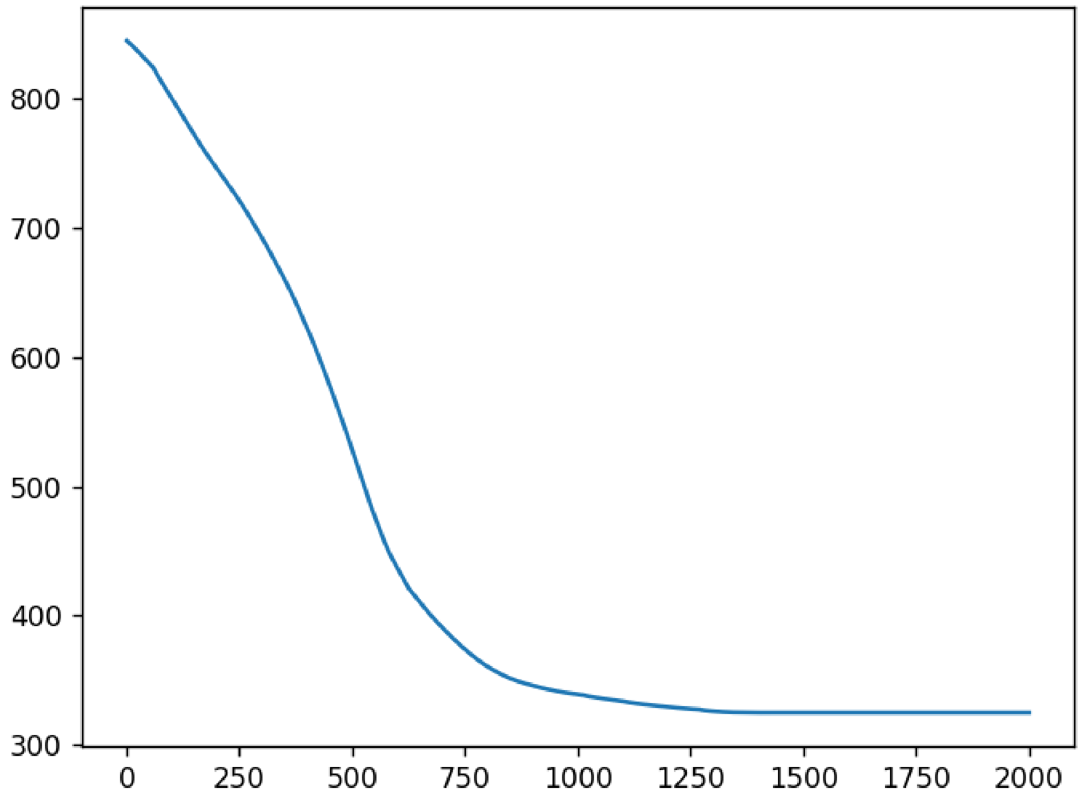


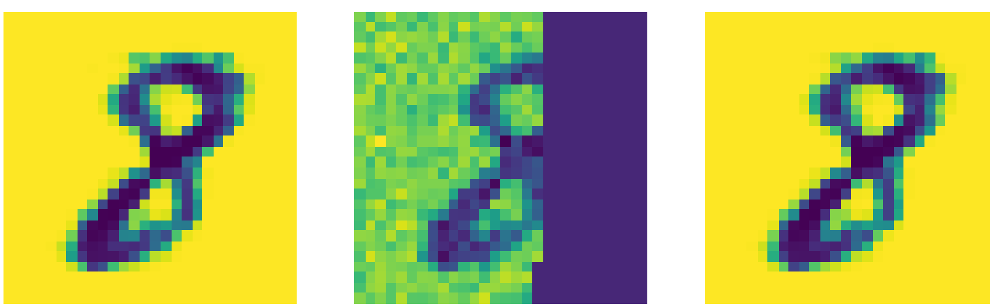




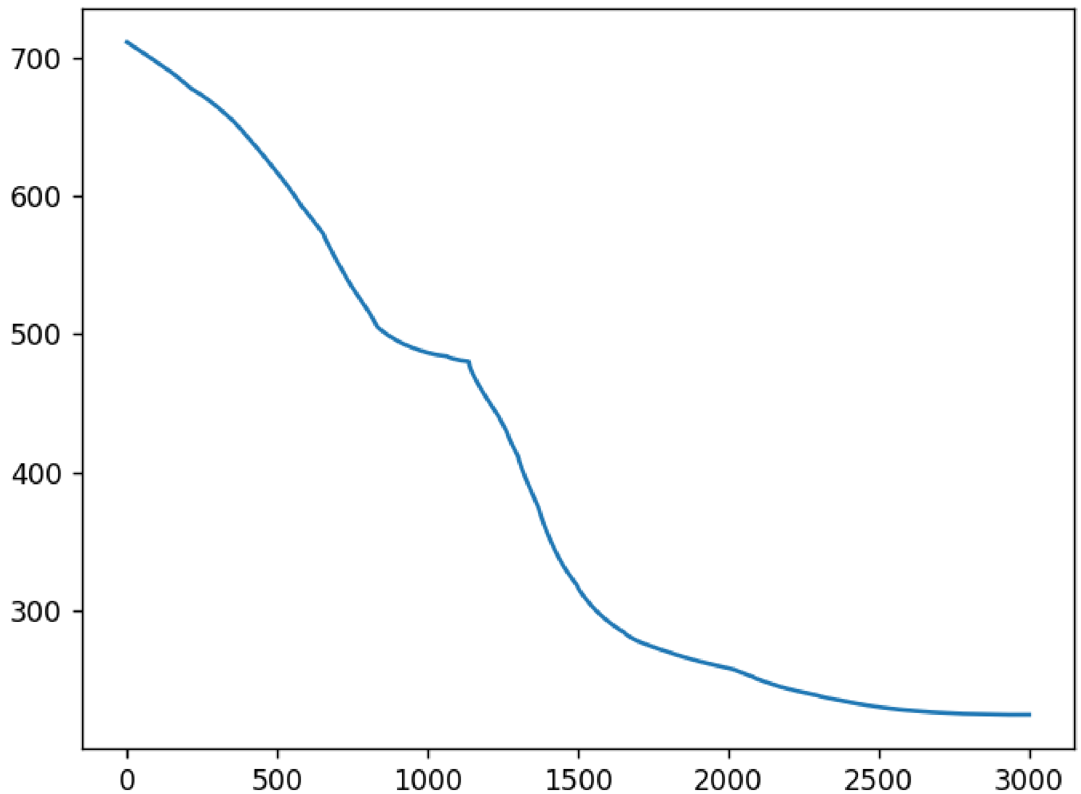
Εικόνα 4:

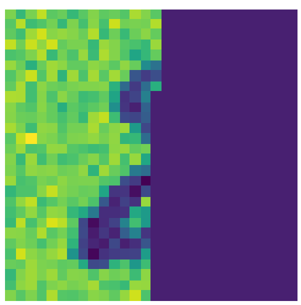
N=500 , learning rate=0.001 repetitions = 2000 και λ_adams=0.1



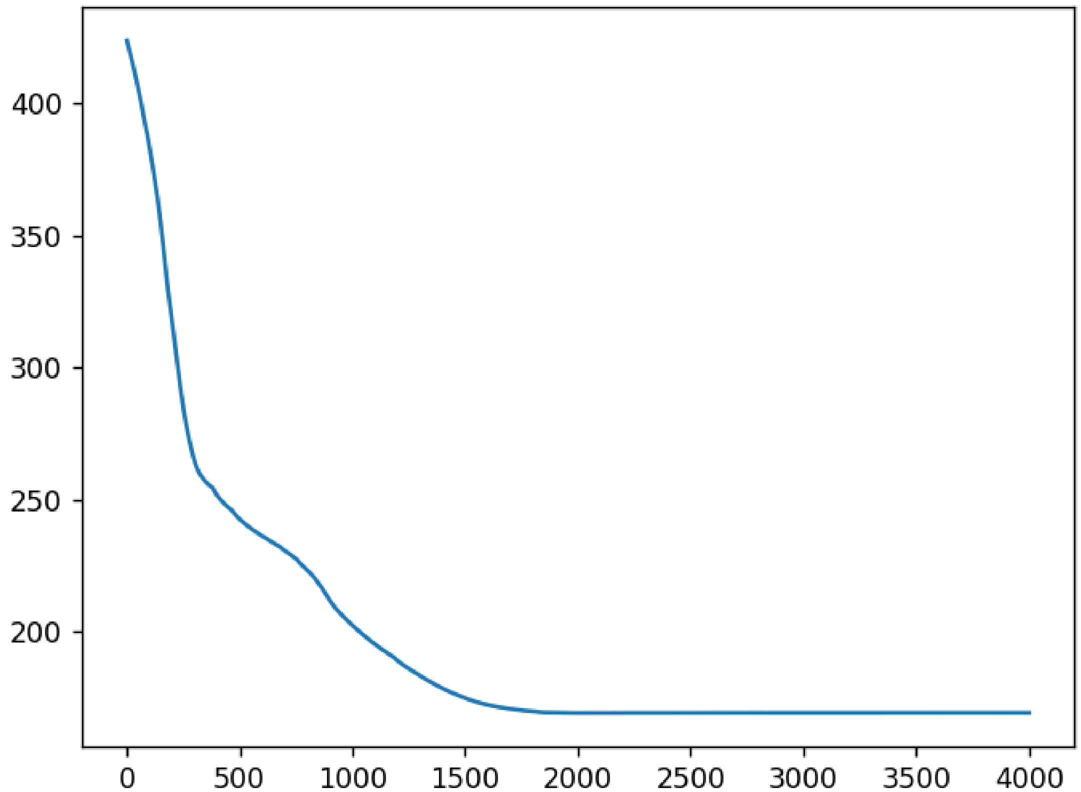


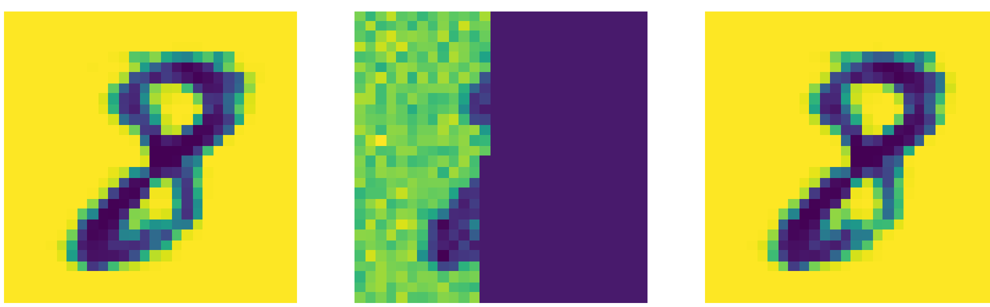
N=400 , learning rate=0.001 repetitions = 3000 και λ_adams=0.1



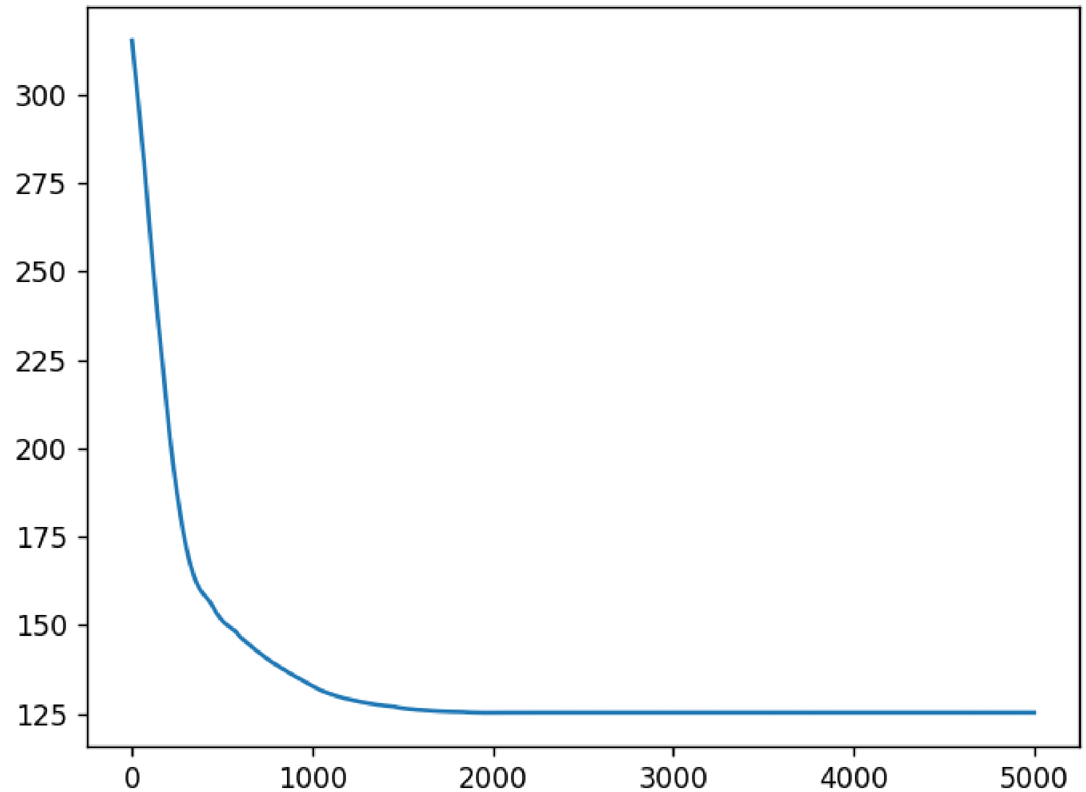


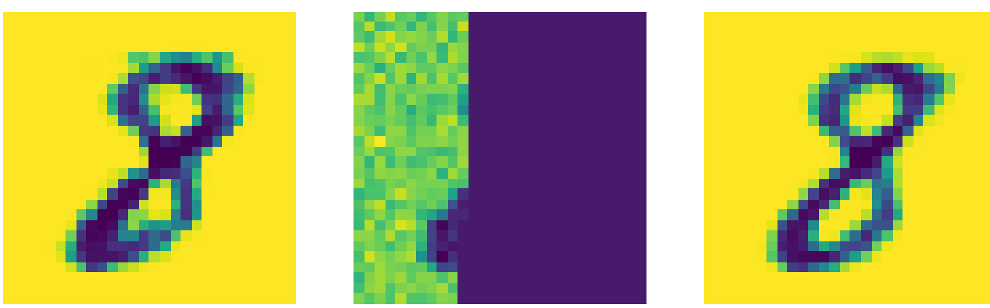
N=350 , learning rate=0.001 repetitions = 4000 και λ_adams=0.1





N=300 , learning rate=0.001 repetitions = 5000 και λ_adams=0.1





Παρατηρήσεις:

- Παρατήρησα ότι όταν χάνουμε παραπάνω από την μισή πληροφορία τότε η ανακατασκευή αρχίζει να γίνεται πιο τυχαία.
- Επιπλέον παρατήρησα ότι με την μέθοδο ADAMS το κόστος συγκλίνει πιο ομαλά απ' ότι χωρίς την εφαρμογή της μεθόδου.
- Τέλος παρατήρησα ότι ο βέλτιστος αριθμός επαναλήψεων είναι κοντά στο 3000 και πως για N<350 παρόλο που το κόστος συγκλίνει πολύ πιο γρήγορα το αποτέλεσμα είναι λάθος.

Ερώτημα 3:

Για το τρίτο ερώτημα ακολουθήθηκε παρόμοια διαδικασία. Η μόνη διαφορά ήταν στον πίνακα T. Για να βρούμε τις σωστές θέσεις των τιμών 1/16 που πρέπει να πάρει ο T έπρεπε να σκεφτούμε πως όταν μία εικόνα 28x28 γίνεται διάνυσμα μπαίνει η κάθε 28αδα η μία κάτω από την άλλη. Επιπλέον από την στιγμή που δημιουργούμε grids 4άδων στην εικόνα 28χ28 για κάθε 7^{o} grid που δημιουργείται θα πρέπει να μετατοπίζουμε τα σετ των 1/16 κατά 112 στον πίνακα T ώστε να μετακινηθούμε στις επόμενες 4 σειρές της εικόνας 28x28 καθώς 28x4 = 112.

Παραθέτω τον κώδικα σε αυτό το σημείο για καλύτερη κατανόηση της προγραμματιστικής διαδικασίας που ακολουθήθηκε για την δημιουργία του πίνακα T.

```
for i in range(49):
    if(i%7==0 and i!=0):
        k+=112
    for j in range(4):
        for t in range(4):
            T[i][((28*j)+(t+4*(i%7)))+k]=1/16
```

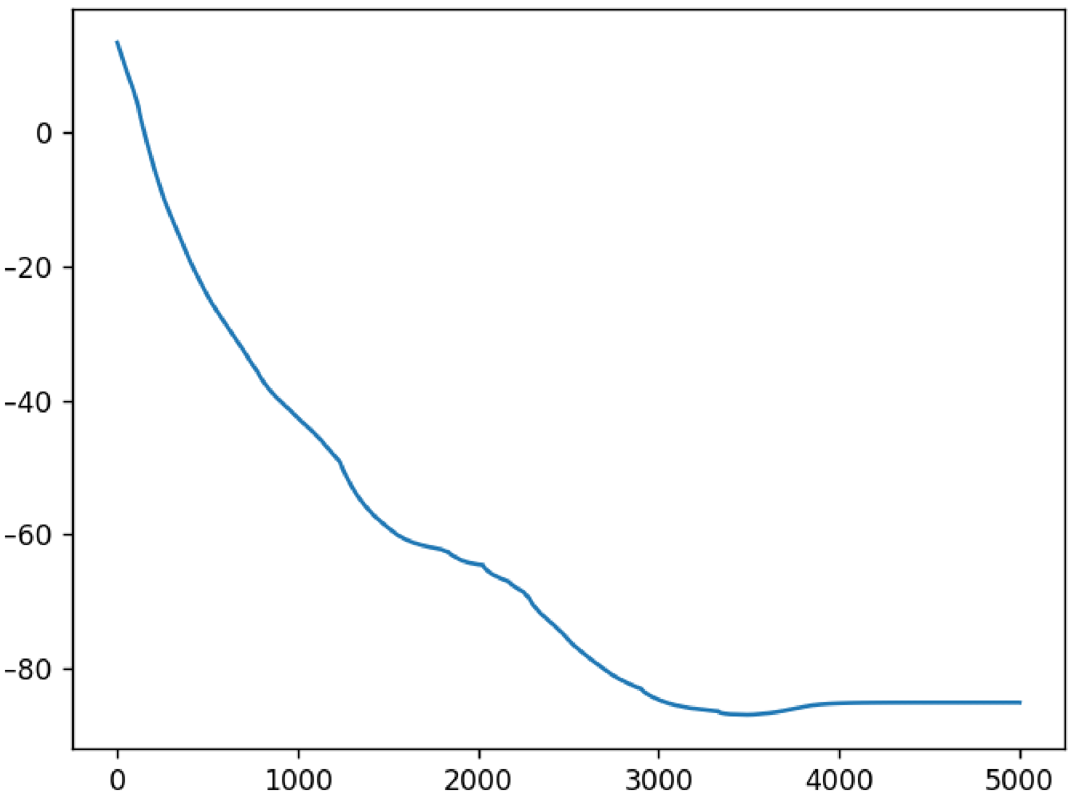
Παρακάτω θα παρουσιάσουμε τα αποτελέσματα του κώδικα.

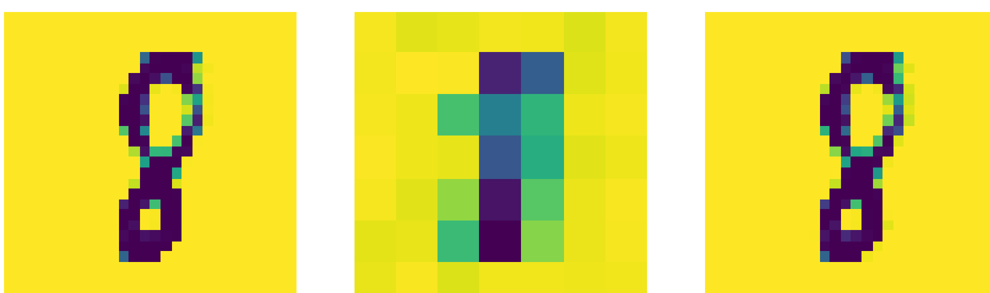
Τα αποτελέσματα θα έχουν την εξής μορφή:

1. Φωτογραφία σύγκλισης συνάρτησης κόστους.
2. Φωτογραφία με αριστερά την ιδανική εικόνα στην μέση την επεξεργασμένη και δεξιά την έξοδο του Νευρωνικού δικτύου.

Εικόνα 1:

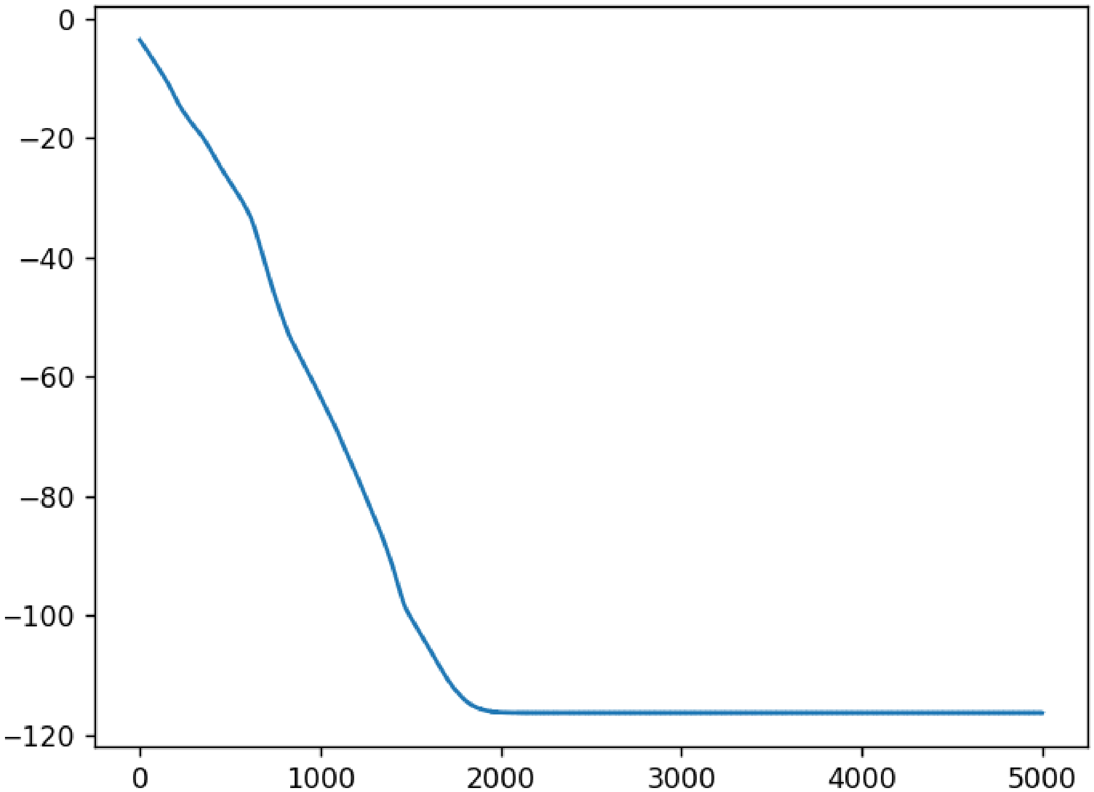
learning rate=0.001 repetitions = 5000 και λ_adams=0.1

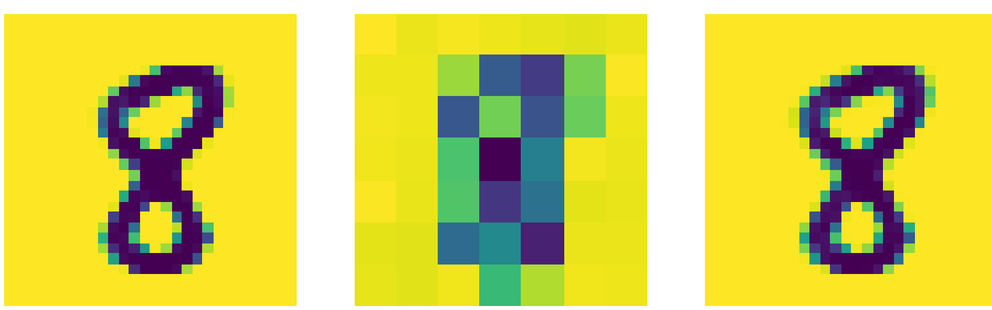




Εικόνα 2:

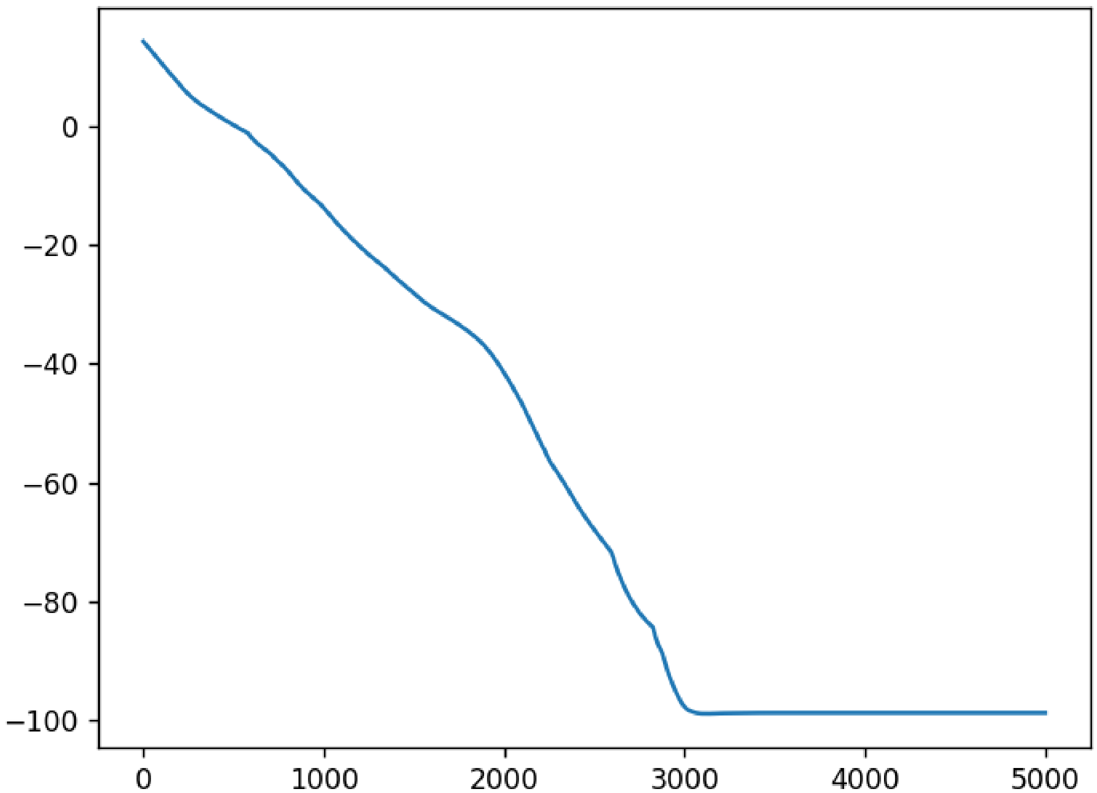
learning rate=0.001 repetitions = 5000 και λ_adams=0.1

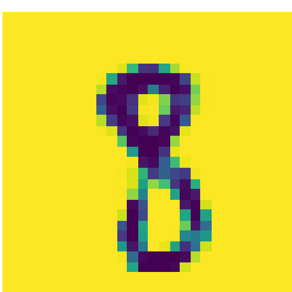


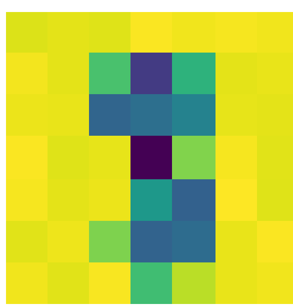


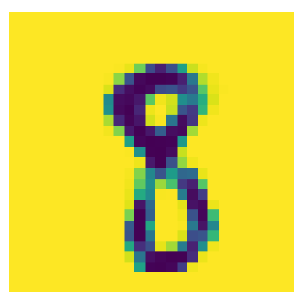
Εικόνα 3:

learning rate=0.001 repetitions = 5000 και λ_adams=0.1



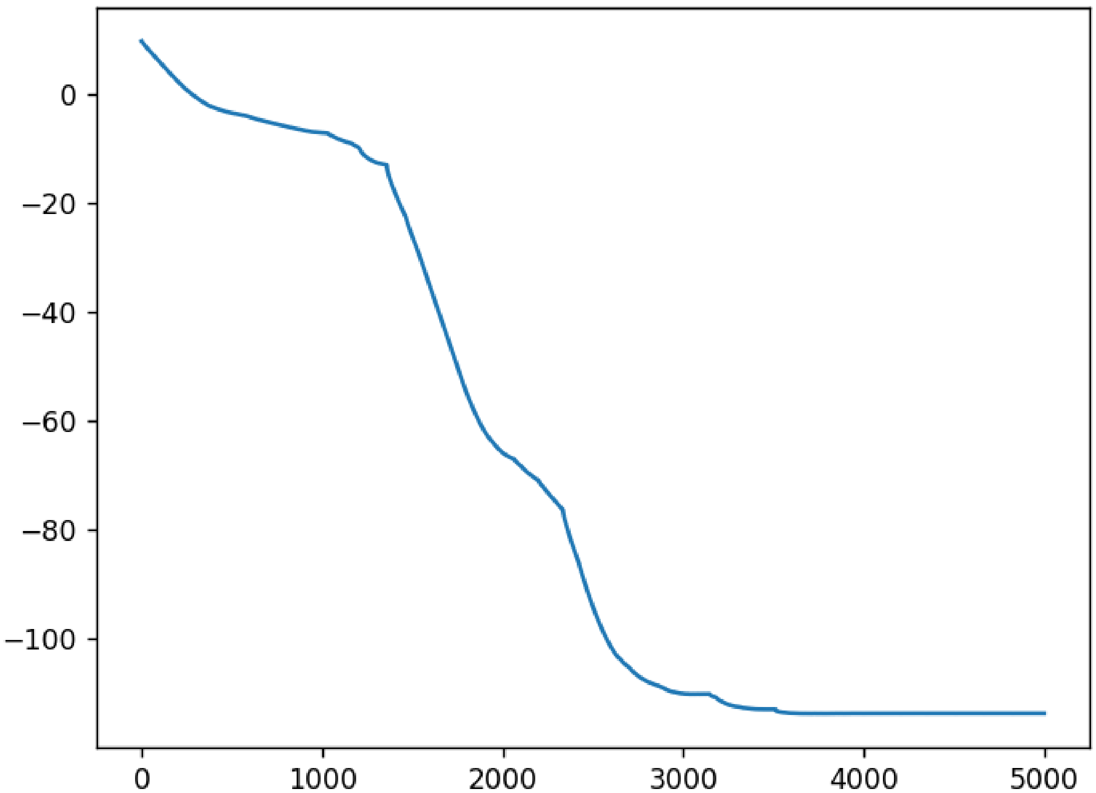


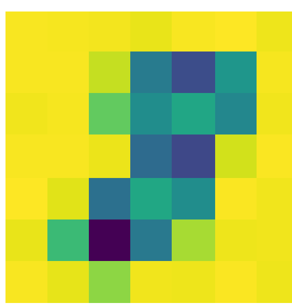




Εικόνα 4:

learning rate=0.001 repetitions = 5000 και λ_adams=0.1





<u>Παρακάτω παρατίθεται ο κώδικας για εκτενέστερη ανάλυση:</u>

```
import numpy as np

import matplotlib.pyplot as plt

import math

import scipy.io

from numpy import random



class NN():


    def __init__(self):
        return


    def ReLu(self, x):
        temp = np.where(x > 0, x, 0)
        return temp


    def ReLu_der(self, x):
        temp = np.where(x > 0, 1, 0)
        temp = np.reshape(temp, (len(temp), 1))
        return temp


    def Sigmoid(self, output):
        return 1 / (1 + np.exp(output))


    def Sigmoid_der(self,x):
        return -np.exp(x)/(np.power((1+np.exp(x)),2))
```

```
def forward_pass(self,Z,A1,A2,B1,B2):
    W1 = np.dot(A1,Z)+B1
    Z1 = self.ReLu(W1)
    W2 = np.dot(A2,Z1)+B2
    X = self.Sigmoid(W2)
    return W1,Z1,W2,X

def calc_fi_x(self,T,X,Xn,N):
    temp = np.reshape(Xn,(1,N))
    norm_power = np.sum(np.power((np.dot(T,X).T-temp),2))
    fi_x = np.log10(norm_power)
    return fi_x

def calc_fi_x_der(self,T,X,Xn,N):
    temp = np.reshape(Xn,(1,N))
    num = 2*np.dot((np.dot(T,X).T-temp),T)
    den = np.sum(np.power((np.dot(T,X).T-temp),2))
    return num/den

def calc_Jz(self,N,T,X,Xn,Z):
    fi_x=self.calc_fi_x(T,X,Xn,N)
    Jz = N*fi_x+np.sum(np.power(Z,2))
    return Jz

def calc_Jz_der(self,N,u0,Z):
    return N*u0+2*Z
```

```
def training(self,Xn,A1,A2,B1,B2,T,Z,N,rep):
    learnig_rate=0.001
    cost=[]
    l_adams=0.1
    for i in range(rep):
        Parameters=self.forward_pass(Z,A1,A2,B1,B2)
        W1=Parameters[0]
        W2=Parameters[2]
        X =Parameters[3]

        Jz = self.calc_Jz(N,T,X,Xn,Z)
        u2=self.calc_fi_x_der(T,X,Xn,N)
        v2=np.multiply(u2.T,self.Sigmoid_der(W2))
        u1 = np.dot(A2.T,v2)
        v1 = np.multiply(u1,self.ReLu_der(W1))
        u0 = np.dot(A1.T,v1)
        Jz_der = self.calc_Jz_der(N,u0,Z)
        if(i==0):
            power=np.power(Jz_der,2)
        else :
            power=(1-l_adams)*power+l_adams*np.power(Jz_der,2)

        Z= Z-learnig_rate*Jz_der/(np.sqrt(power+0.0000001))
        cost.append(Jz)
    return X,Jz,cost
```

```
while True:
    print("Πατήστε 1 για το πρώτο ερώτημα, 2 για το δεύτερο και 3 για το τρίτο.\nΠατήστε 0 
για έξοδο")
    Nn=NN()
    data21 = scipy.io.loadmat('data21.mat')
    A1 = data21['A_1']
    B1 = data21['B_1']
    A2 = data21['A_2']
    B2 = data21['B_2']
    choice=input()
    if choice=='0':
        break
    if choice=='1':
        N=10
        Z=np.random.randn(N,100)
        X = Nn.forward_pass(Z,A1,A2,B1,B2)
        X=X[-1].T
        for i in range(1,101):
            X_2D = np.reshape(X[i-1],(28,28))
            plt.subplot(10,10,i)
            plt.axis('off')
            plt.imshow(X_2D.T)
        plt.show()



    if choice=='2':
```

```
data22 = scipy.io.loadmat('data22.mat')
Xi = data22['X_i']
Xn = data22['X_n']
print("Διαλέξτε αριθμό φωτογραφίας μεταξύ 1-4")
im_number=int(input())
while True:
   if im_number<1 or im_number>4:
      print("Λάθος αριθμός παρακαλώ εισάγεται αριθμό από 1-4")
      im_number=int(input())
   if im_number>=1 and im_number<=4:
      break

print('Διαλέξτε ποσότητα πληροφορίας που θέλετε να κρατήσετε\nΑποδεκτές τιμές 500,400,350,300')
N=int(input())
if N==500:
   rep=2000
elif N==400:
   rep=3000
elif N==350:
   rep=4000
elif N==300:
   rep=5000
else:
   rep=5000
Set=10
```

```
Xn=Xn.T
Xi=Xi.T
I=np.eye(N,dtype=float)
zeros=np.zeros((N,784-N))
T=np.concatenate((I,zeros),axis=1)
Xn_0=np.dot(T,Xn[im_number-1])
Jz=[]
for i in range(20):
    Z=np.random.randn(Set,1)
    X = Nn.training(Xn_0,A1,A2,B1,B2,T,Z,N,rep)
    if i ==0:
        Z_all=Z
        X_all=X[0]
        costs=[X[2]]
    else:
        Z_all=np.concatenate((Z_all,Z),axis=1)
        X_all=np.concatenate((X_all,X[0]),axis=1)
        costs=np.concatenate((costs,[X[2]]),axis=0)
    Jz.append(X[1])
minn=Jz[0]
min_index=0
for i in range(1,len(Jz)):
    if(Jz[i]<minn):
        minn=Jz[i]
        min_index=i
print(Z_all.T[min_index])
print(Jz[min_index])
```

```
    x = np.linspace(1, rep,rep )
    plt.plot(x, costs[min_index])
    plt.show()
    X_2D = np.reshape(Xi[im_number-1],(28,28))
    plt.subplot(1,3,1)
    plt.axis('off')
    plt.imshow(X_2D.T)

    zero_pad=np.zeros(784-N)
    Xn_0_pad=np.concatenate((Xn_0,zero_pad))
    X_2D = np.reshape(Xn_0_pad,(28,28))
    plt.subplot(1,3,2)
    plt.axis('off')
    plt.imshow(X_2D.T)

    X_2D = np.reshape(X_all.T[min_index],(28,28))
    plt.subplot(1,3,3)
    plt.axis('off')
    plt.imshow(X_2D.T)

    plt.show()

if choice=='3':

    N=49
    Set=10
    data23 = scipy.io.loadmat('data23.mat')
```

```
Xi = data23['X_i']

Xn = data23['X_n']

print("Διαλέξτε αριθμό φωτογραφίας μεταξύ 1-4")

im_number=int(input())

while True:

   if im_number<1 or im_number>4:

      print("Λάθος αριθμός παρακαλώ εισάγεται αριθμό από 1-4")

      im_number=int(input())

   if im_number>=1 and im_number<=4:

      break

Xi=Xi.T

Xn=Xn.T

T=np.zeros((49,784))

k=0


for i in range(49):

   if(i%7==0 and i!=0):

      k+=112

   for j in range(4):

      for t in range(4):

         T[i][((28*j)+(t+4*(i%7)))+k]=1/16


Jz=[]

rep=5000

for i in range(20):

   Z=np.random.randn(Set,1)

   X = Nn.training(Xn[im_number-1],A1,A2,B1,B2,T,Z,N,rep)
```

```
    if i ==0:
        Z_all=Z
        X_all=X[0]
        costs=[X[2]]
    else:
        Z_all=np.concatenate((Z_all,Z),axis=1)
        X_all=np.concatenate((X_all,X[0]),axis=1)
        costs=np.concatenate((costs,[X[2]]),axis=0)
    Jz.append(X[1])
minn=Jz[0]
min_index=0
for i in range(1,len(Jz)):
    if(Jz[i]<minn):
        minn=Jz[i]
        min_index=i
print(Jz[min_index])
print(Z_all.T[min_index])
x = np.linspace(1, rep,rep )
plt.plot(x, costs[min_index])
plt.show()

X_2D = np.reshape(Xi[im_number-1],(28,28))
plt.subplot(1,3,1)
plt.axis('off')
plt.imshow(X_2D.T)

X_2D = np.reshape(Xn[im_number-1],(7,7))
```

```
plt.subplot(1,3,2)
plt.axis('off')
plt.imshow(X_2D.T)


X_2D = np.reshape(X[0],(28,28))

plt.subplot(1,3,3)
plt.axis('off')
plt.imshow(X_2D.T)
plt.show()
```