Χαλαντζούκας Φοίβος Τμήμα ΗΜΤΥ AM:1066579

Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας – Εργαστηριακές Ασκήσεις
Μέρος Β

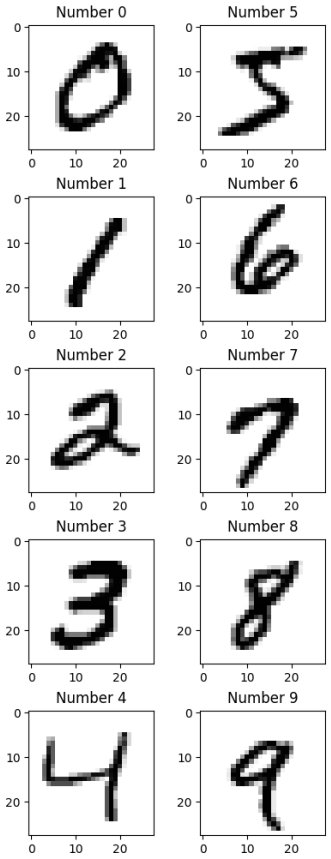
ΘΕΜΑ 1 – Κατηγοριοποίηση Εικόνων

Σκοπός αυτής της εργασίας είναι η κατηγοριοποίηση εικόνων με τη χρήση 2 διαφορετικών μεθόδων και η σύγκριση αυτών. Η πρώτη μέθοδος αφορά τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks – CNN) και η δεύτερη μέθοδος αφορά τη χρήση HOG (Histogram of Oriented Gradients) και Support Vector Machines (SVM).

Μέρος Α

Αντικείμενο του μέρους Α είναι η Κατηγοριοποίηση Εικόνων με χρήση Συνελικτικών Νευρωνικών Δικτύων. Τα CNNs (γνωστά και ως Deep CNNs) είναι το κυριότερο είδος Νευρωνικών Δικτύων που χρησιμοποιείται για την επεξεργασία πολυδιάστατων σημάτων. Το όνομα τους οφείλεται στην πράξη της συνέλιξης που εκτελείται σε τουλάχιστον ένα επίπεδο εντός του δικτύου, και το όνομα «βαθιά» (deep) προκύπτει από το μεγάλο πλήθος των επιπέδων που αυτά συνήθως περιέχουν.

Αρχικά κατεβάζουμε το σύνολο δεδομένων του MNIST από τον δοθέντα σύνδεσμο, και αποσυμπιέζουμε σε έναν κοινό υποφάκελο – με το όνομα data – τα συμπιεσμένα αρχεία του. Τα αποσυμπιεσμένα αρχεία είναι της μορφής ".idxa-ubyte", όπου a=3 για τα αρχεία εικόνων και a=1 για τα αρχεία των labels. Με τη χρήση της βιβλιοθήκης idx2numpy μετατρέπουμε τα δεδομένα τους σε numpy.ndarrays, πίνακες δηλαδή που μπορούν να διαβάσουν οι βιβλιοθήκες matplotlib και tensorflow-keras. Στη συνέχεια, για κάθε έναν από τους 10 αριθμούς/κλάσεις που υπάρχουν στο training dataset (0-9) βρίσκουμε την πρώτη εικόνα του και την τυπώνουμε σε ένα subplot, μαζί με τις φωτογραφίες των υπόλοιπων αριθμών (1 ανά αριθμό/κλάση).



Εικόνα 1: Εμφάνιση 1 εικόνας ανά κλάση

Αφού σβήσουμε το figure που εμφανίζεται, περνάμε στο κομμάτι της κατασκευής και εκπαίδευσης του νευρωνικού δικτύου. Αρχικά κανονικοποιούμε τις εικόνες του dataset διαιρώντας τις τιμές τους με το 255, ώστε το σύνολο τιμών τους να περιέχεται από το 0 ως το 1 για την απλοποίηση των υπολογισμών που εκτελεί το δίκτυο. Ακολούθως τροποποιούμε τις διαστάσεις τους ώστε να είναι βεβαιωθούμε ότι θα είναι συμβατές με την είσοδο του δικτύου (28x28x1), και μετατρέπουμε τα labels των εικόνων σε δυαδικούς πίνακες με τη χρήση της συνάρτησης to_categorical(), για να μπορεί να τις χρησιμοποιήσει κατά την εκπαίδευση του το δίκτυο. Στη συνέχεια

επιλέγουμε τις τιμές που επιθυμούμε για το batch size και για το πλήθος των epochs που θέλουμε, και δημιουργούμε ένα ακολουθιακό μοντέλο CNN στο keras, με την αρχιτεκτονική που περιγράφεται στην Εικόνα 1 της εκφώνησης. Επιλέγουμε ως συνάρτηση σφάλματος το Μέσο Τετραγωνικό Σφάλμα, ορίζουμε το batch size στο 50 και το πλήθος των epochs στο 200, στα πρότυπα της βιβλιογραφικής πηγής [1] της εκφώνησης.

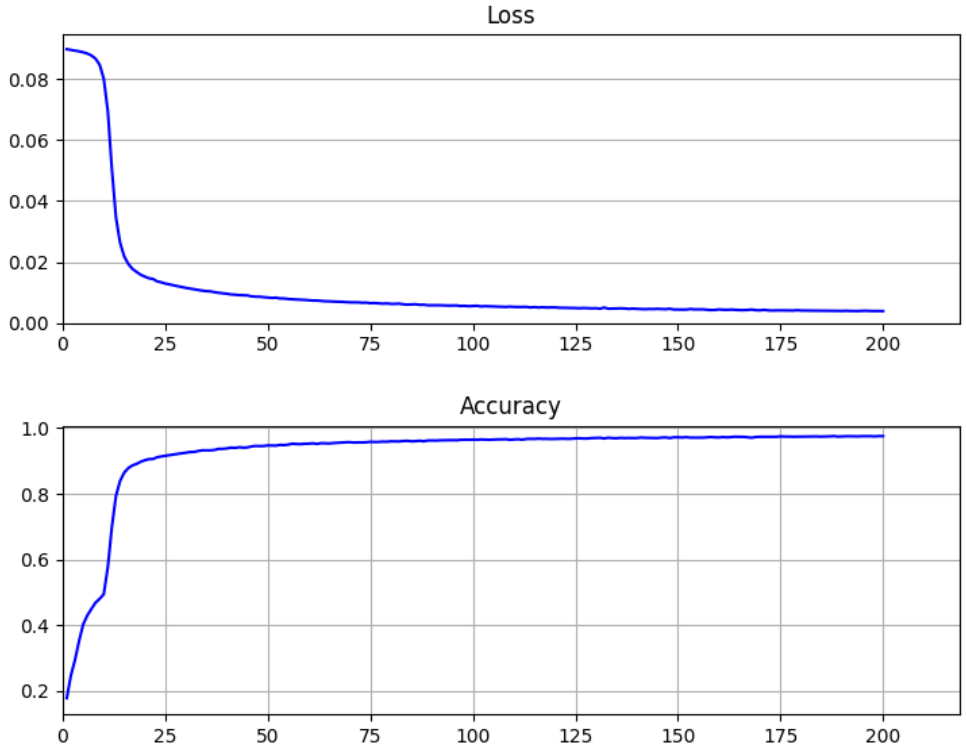
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         60

 average_pooling2d (AverageP  (None, 14, 14, 6)        0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 14, 14, 16)        880

 average_pooling2d_1 (Averag  (None, 7, 7, 16)         0
 ePooling2D)

 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 120)               94200

 dense_1 (Dense)             (None, 84)                10164

Total params: 106,154
Trainable params: 106,154
Non-trainable params: 0
```

Εικόνα 2: Η δομή του δικτύου και το πλήθος παραμέτρων από το model.summary()

Σε κάθε epoch ελέγχουμε την ακρίβεια (val_accuracy) και το σφάλμα κατηγοριοποίησης (val_loss) με τη χρήση του testing dataset. Οι τιμές αυτές αποθηκεύονται σε πίνακες στη δομή “history”, και μετά το τέλος της εκπαίδευσης τυπώνονται σε 2 γραφικές παραστάσεις. Για κάθε epoch τυπώνεται ο αύξων αριθμός του (1 ως 200) και οι μετρικές σφάλματος – ακρίβειας, καθώς και η διάρκειά του σε δευτερόλεπτα.

```
Epoch 146/200
1200/1200 - 5s - loss: 0.0042 - accuracy: 0.9742 - val_loss: 0.0045 -
val_accuracy: 0.9712 - 5s/epoch - 4ms/step
```

Εικόνα 3:Στατιστικά του 146ου epoch

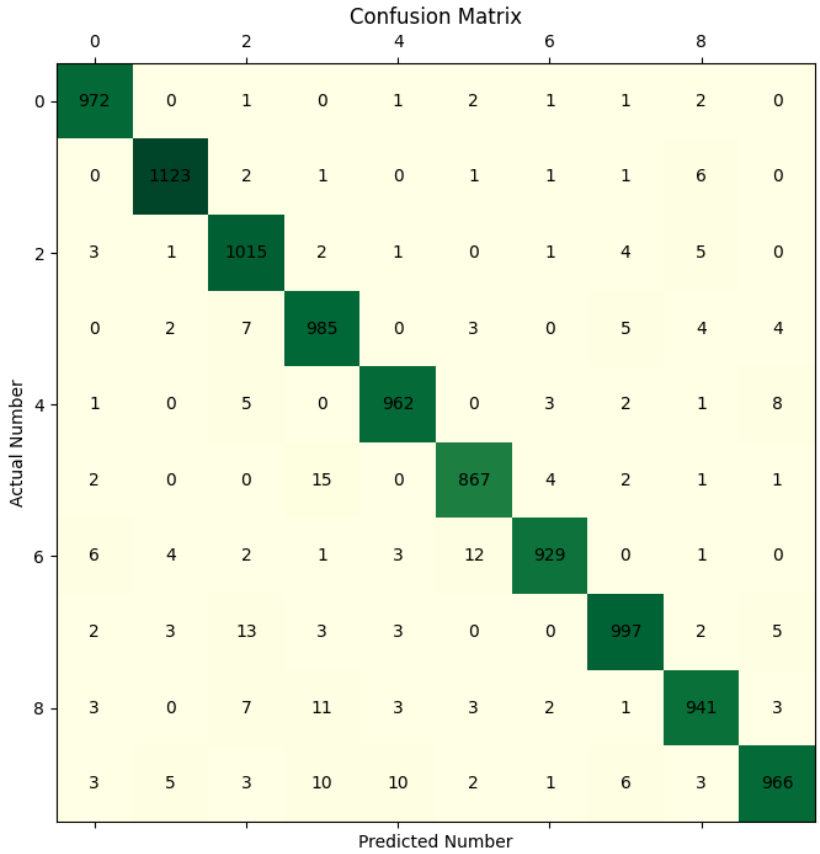


Εικόνα 4: Σφάλμα κατηγοριοποίησης και ακρίβεια μοντέλου ανά epoch

Παρατηρούμε ότι χρειάζονται περίπου 20 epochs για να φτάσουμε σε μια ακρίβεια της τάξης του 90%, και περίπου 150 epochs για να παγιωθεί το σφάλμα στην ελάχιστη τιμή του, και η ακρίβεια αντίστοιχα στη μέγιστη τιμή της.

Αφού κλείσουμε το παράθυρο με τα διαγράμματα, το δίκτυο προβλέπει τα labels του testing dataset, και τα συγκρίνουμε με τα έτοιμα labels, ώστε να κατασκευάσουμε τον confusion matrix. Με κίτρινο σημειώνονται οι τιμές με χαμηλή συχνότητα, και με πράσινο αυτές με υψηλή συχνότητα. Όσο πιο σκούρο είναι το χρώμα, τόσο μεγαλύτερη η συχνότητα.

Εικόνα 5: YlGn colorbar



Εικόνα 6: Confusion Matrix για 200 epochs, με 97.57% ακρίβεια

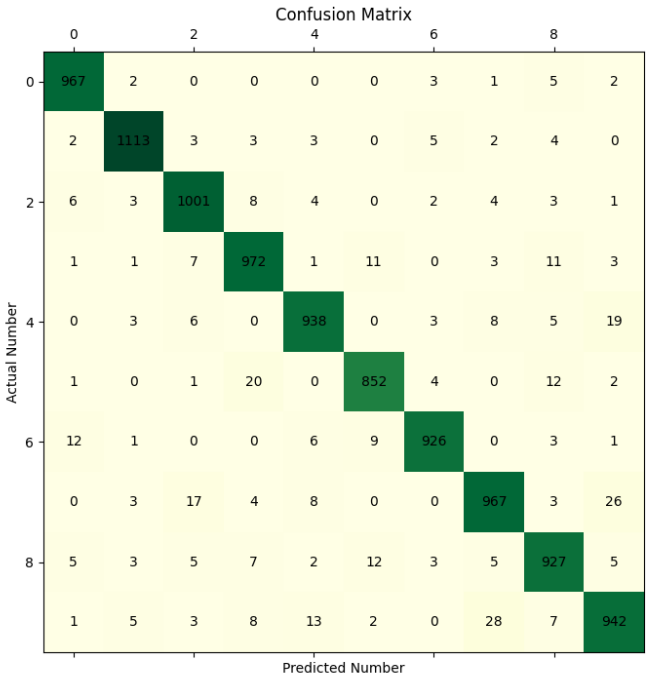
Μέρος Β

Αντικείμενο του μέρους Α είναι η Κατηγοριοποίηση Εικόνων με χρήση Ιστογραμμάτων “Histogram of Oriented Gradients” και κατηγοριοποιητών “Support Vector Machines”.

Όπως και στο μέρος Α, εισάγουμε τα δεδομένα του MNIST και κανονικοποιούμε τις εικόνες και τροποποιούμε τις διαστάσεις τους. Για την εξαγωγή των χαρακτηριστικών HOG των εικόνων χρησιμοποιούμε τη βιβλιοθήκη skimage (scikit-image). Το block size καθορίζεται από τις παραμέτρους pixels_per_cell και cells_per_block.

Αφού γίνει ο υπολογισμός των χαρακτηριστικών HOG τόσο για το training όσο και για το testing dataset, εκπαιδεύουμε έναν γραμμικό ταξινομητή. Μέσω αυτού προβλέπουμε τα labels των test images, και δημιουργούμε τον confusion matrix με τον ίδιο τρόπο με το πρώτο ερώτημα.

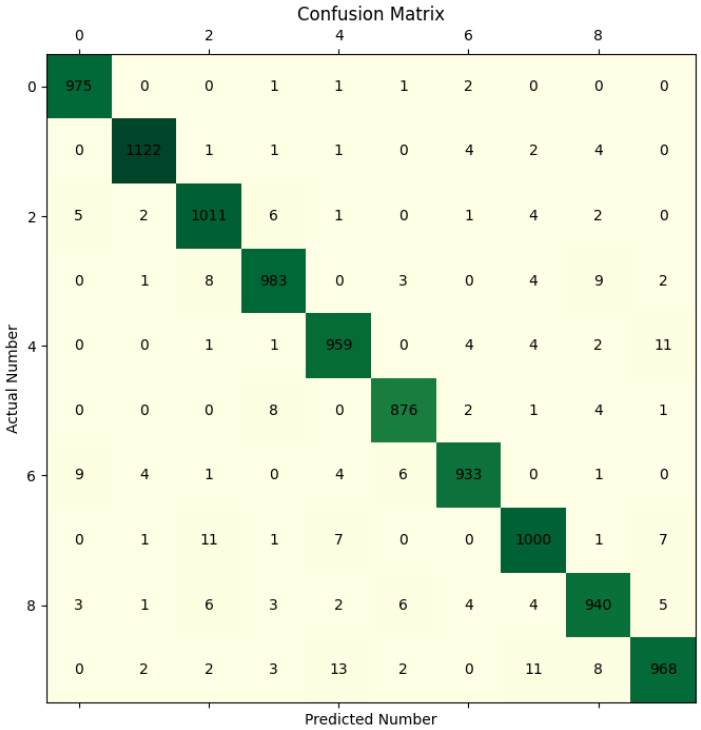
Για 7x7 pixels/cell και 1 cell/block παίρνουμε ακρίβεια 96.05%.



75.92 seconds
96.05% accuracy

Εικόνα 7: 7x7 p/c, 1 c/b

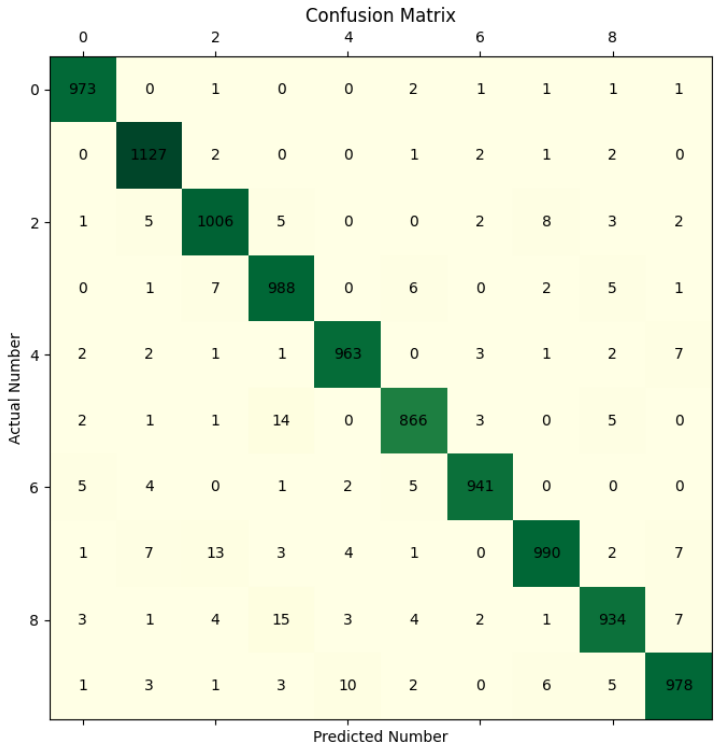
Για 7x7 pixels/cell και 2x2 cells/block

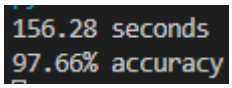


80.99 seconds
97.67% accuracy

Εικόνα 8: 7x7 p/c, 2x2 c/b

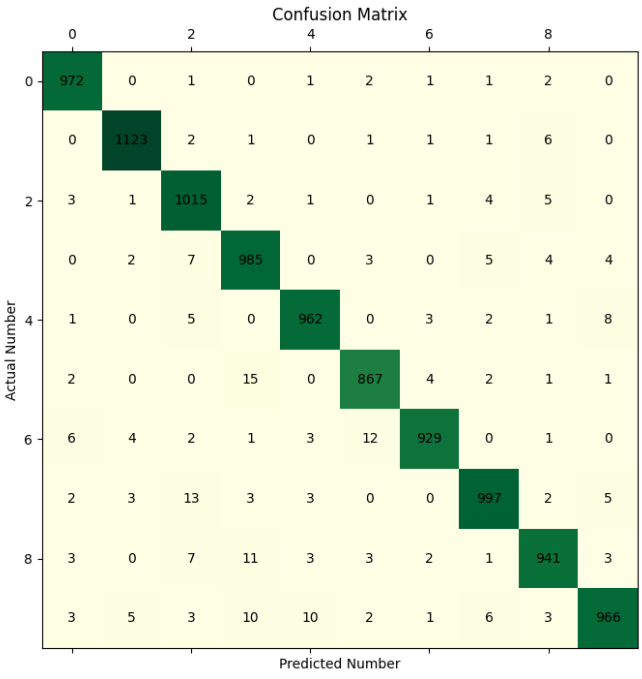
Για 4x4 pixels και 1 cell:



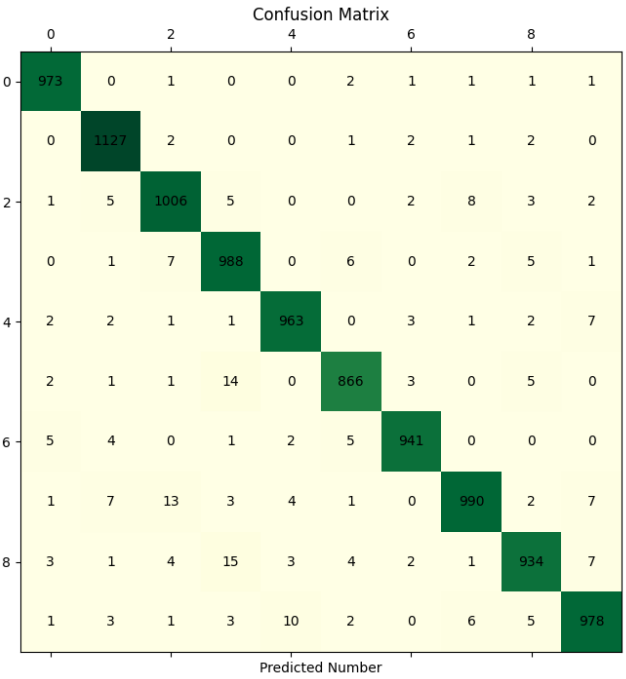


Εικόνα 9: 4x4 p/c, 1 c/b

Συγκρίνουμε τα τελευταία αποτελέσματα με αυτά του CNN:



Εικόνα 11: CNN, 97.57%



Εικόνα 10: SVM, 97.66%

Παρατηρούμε ότι τα αποτελέσματα είναι παραπλήσια, καθώς πέρα από το ότι και τα 2 μοντέλα έχουν κοντινή ακρίβεια στην σωστή αναγνώριση ανά αριθμό, υπάρχει στους περισσότερους αριθμούς αντιστοιχία στην πιο συχνή λανθασμένη πρόβλεψη ανά αριθμό. Για παράδειγμα, όταν έχουμε λάθος εκτίμηση της κλάσης ενός αριθμού, ο αριθμός 7 τείνει και στους 2 ταξινομητές να αναγνωρίζεται ως 2, ενώ ο αριθμός 5 τείνει να ερμηνεύεται ως 3 και στους 2 ταξινομητές.

Το Νευρωνικό Δίκτυο χρειάζεται αρκετά περισσότερο χρόνο να εκπαιδευτεί (4s ανά epoch κατά μέσο όρο) για το επιλεγμένο batch size , ακόμα και με τη χρήση κάρτας γραφικών για τους υπολογισμούς του (με τη χρήση των CUDA, cuDNN toolkits της NVIDIA). Η εκπαίδευση όμως αρκεί να γίνει μια φορά και το μοντέλο μπορεί να αποθηκευτεί, όποτε η καθυστέρηση δεν είναι σημαντική παράμετρος.

Κώδικας

Μέρος Α (partA.py)

```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from keras import layers
from conf import *
from data import *

#Read MNIST data from idx files
(trainimages, trainlabels, testimages, testlabels) = importData()
classes=10

#Plot 1 image per class
figure, axis = plt.subplots(5, 2, constrained_layout = True)
figure.set_figwidth(4)
figure.set_figheight(10)
for i in range(classes):
    j=0
    while int(trainlabels[j])!=i:
        j+=1
    axis[i%5,i//5].imshow(trainimages[j], cmap=plt.cm.binary)
    axis[i%5,i//5].set_title("Number "+str(i))
plt.show()

#Resize and Normalize images
trainimages = trainimages.astype("float32")/255
testimages = testimages.astype("float32")/255
trainimages = np.expand_dims(trainimages, -1)
testimages = np.expand_dims(testimages, -1)
```

```
#Define classes and parameters
trainlabels = keras.utils.to_categorical(trainlabels, classes)
testlabelscat = keras.utils.to_categorical(testlabels, classes)
batch_size=50
epochs=200

#Neural Network Architecture
model = keras.Sequential(
    [
        keras.Input(shape=(28,28,1)),
        layers.Conv2D(6, kernel_size=(3, 3), strides=1, padding='same',
activation="relu"),
        layers.AveragePooling2D(pool_size=(2, 2), strides=2),
        layers.Conv2D(16, kernel_size=(3, 3), strides=1,
padding='same', activation="relu"),
        layers.AveragePooling2D(pool_size=(2, 2), strides=2),
        layers.Flatten(),
        layers.Dense(120, activation="relu"),
        layers.Dense(84, activation="relu"),
        layers.Dense(classes, activation="softmax"),
    ])
model.summary()

#Train Model and Plot Stats
model.compile(loss="mean_squared_error", optimizer="SGD",
metrics=["accuracy"])
history = model.fit(trainimages, trainlabels, batch_size=batch_size,
epochs=epochs, verbose=2, validation_data=(testimages, testlabelscat))
figure, axis = plt.subplots(2, 1)
x=np.linspace(1,epochs,epochs)
axis[0].plot(x,history.history['val_loss'], '-b')
axis[0].set_title("Loss")
axis[0].grid()
axis[1].plot(x,history.history['val_accuracy'], '-b')
axis[1].set_title("Accuracy")
axis[1].grid()
figure.tight_layout()
plt.show()

#Predict the labels of the test set and convert prediction matrices to
single integers
predictions=model.predict(testimages).argmax(axis=1)

#Confusion Matrix
(diff, confusion) = confusionMatrix(classes, predictions, testlabels)
print('{:.2f}% accuracy'.format(100-100*diff/len(predictions)))
fig, ax = plt.subplots()
ax.matshow(confusion, cmap='YlGn')

ax = addText(classes, confusion, ax)
```

```
fig.set_figwidth(8)
fig.set_figheight(8)
plt.ylabel('Actual Number')
plt.xlabel('Predicted Number')
plt.title('Confusion Matrix')

plt.show()
```

Μέρος Β (partB.py)

```
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn import svm, metrics
from conf import *
from data import *

#Read MNIST data from idx files
(trainimages, trainlabels, testimages, testlabels) = importData()

#Resize and Normalize images
trainimages = trainimages.astype("float32")/255
testimages = testimages.astype("float32")/255
trainimages = np.expand_dims(trainimages, -1)
testimages = np.expand_dims(testimages, -1)

#Parameters
classes=10
pixels_per_cell=(2, 2)
cells_per_block=(2, 2)

#HOG extraction
start_time = time.time()
hogTrainImages=extractHogFeatures(trainimages, pixels_per_cell,
cells_per_block)
hogTestImages=extractHogFeatures(testimages, pixels_per_cell,
cells_per_block)

#Classification
classifier = svm.SVC(kernel='linear')
classifier.fit(hogTrainImages, trainlabels)
preds=classifier.predict(hogTestImages)
print('{:.2f} seconds'.format(time.time()-start_time))

#Confusion Matrix
(diff, confusion) = confusionMatrix(classes, preds, testlabels)
print('{:.2f}% accuracy'.format(100-100*diff/len(preds)))
fig, ax = plt.subplots()
```

```
ax.matshow(confusion, cmap='YlGn')

ax = addText(classes, confusion, ax)
fig.set_figwidth(8)
fig.set_figheight(8)
plt.ylabel('Actual Number')
plt.xlabel('Predicted Number')
plt.title('Confusion Matrix')

plt.show()
```

Βοηθητικές Συναρτήσεις (data.py)

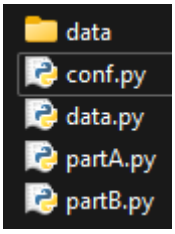
```
import idx2numpy
from skimage.feature import hog

#Reads the idx-ubyte files and returns numpy arrays
def importData():
    file1 = 'data/train-images.idx3-ubyte'
    file2 = 'data/train-labels.idx1-ubyte'
    tfile1 = 'data/t10k-images.idx3-ubyte'
    tfile2 = 'data/t10k-labels.idx1-ubyte'
    trainimages = idx2numpy.convert_from_file(file1)
    trainlabels = idx2numpy.convert_from_file(file2)
    testimages = idx2numpy.convert_from_file(tfile1)
    testlabels = idx2numpy.convert_from_file(tfile2)
    return (trainimages, trainlabels, testimages, testlabels)

#Returns a list with the HOG Features per image
def extractHogFeatures(images, pixels_per_cell, cells_per_block):
    hogImages=[]
    for image in images:
        hogImage=hog(image, orientations=9,
pixels_per_cell=pixels_per_cell,
                     cells_per_block=cells_per_block, channel_axis=-1)
        hogImages.append(hogImage)
    return hogImages
```

Βοηθητικές Συναρτήσεις (conf.py)

```
import numpy as np

#Returns a confusion matrix and the number of negative predictions
def confusionMatrix(classes, predictions, testlabelslist):
    diff=0
```

```
    confusion=np.zeros((classes,classes))
    for i in range(len(predictions)):
        if predictions[i]!=testlabelslist[i]:
            diff+=1
        confusion[testlabelslist[i]][predictions[i]]+=1
    return (diff, confusion)

#Fills the initial colored confusion matrix with numbers
def addText(classes, confusion, ax):
    for i in range(classes):
        for j in range(classes):
            val = int(confusion[j, i])
            ax.text(i, j, str(val), va='center', ha='center')
```

Δομή του φακέλου



Στο φάκελο data πρέπει να βρίσκονται τα ubyte αρχεία του MNIST που αναφέρθηκαν παραπάνω.