Νευρωνικά Δίκτυα

Εργασία 1η

ΧΡΥΣΙΚΟΣ ΧΡΗΣΤΟΣ ΑΕΜ 9432 ΤΜΗΜΑ ΗΜΜΥ

ΕΞΑΜΗΝΟ 11ο

> Αντικείμενα εργασίας

- 1. Περιγραφή Αλγορίθμου
- 2. Παραδείγματα Σωστής και εσφαλμένης κατηγοριοποίησης
- 3. Ποσοστά επιτυχίας κατά τα στάδια εκπαίδευσης και ελέγχου
- 4. Χρόνος Εκπαίδευσης και ποσοστά επιτυχίας για διαφορετικούς αριθμούς νευρώνων στο hidden layer
- 5. Σύγκριση με την κατηγοριοποίηση Nearest Neighbor και Nearest Centroid
- 6. Σχολιασμός αποτελσμάτων και κώδικας

Βασικός αλγόριθμος εκπαίδευσης Convolutional Neural Network για την αναγνώριση ψηφιών από το dataset MNIST(60.000 training samples and 10.000 testing samples):

Import Libraries

```
#@title Import Libraries
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow import keras
from keras.layers import Dense, Flatten
from keras.models import Sequential
from keras.utils import to_categorical
from keras.datasets import mnist
```

Prepare MNIST data

```
#@title Prepare MNIST data
(x_train, y_train), (x_test, y_test) = mnist.load_data()
num_features = 784 # data features - img shape = 28*28
```

Data Shape

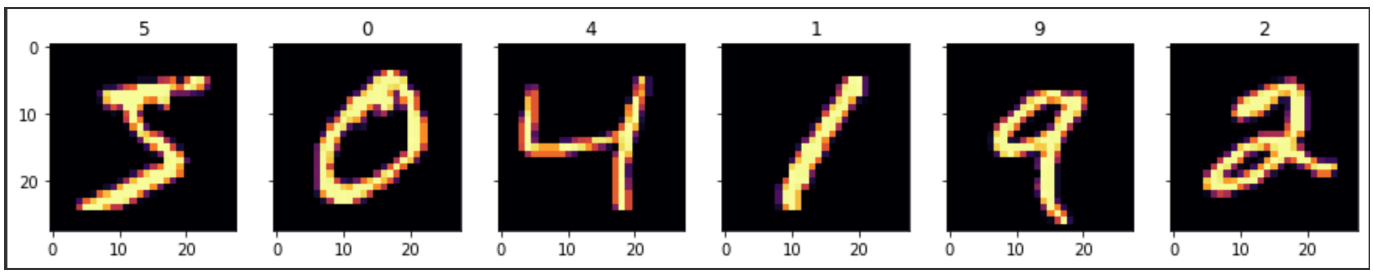
```
#@title Data Shape
print("Xtrain Shape:", x_train.shape)
print("Xtest Shape:", x_test.shape)

print("Ytrain Shape:", y_train.shape)
print("Ytest Shape:", y_test.shape)

# We have to transform Y data to the suitable format for the Neural Network
```

```
Xtrain Shape: (60000, 28, 28)
Xtest Shape: (10000, 28, 28)
Ytrain Shape: (60000,)
Ytest Shape: (10000,)
```

Κάνουμε μια βασική ανάγνωση των πρώτων 6 στοιχείων του Dataset για να δούμε την μορφή τους.

Display the first six images of mnist digit database

```
#@title Display the first six images of mnist digit database
fig, axes = plt.subplots(ncols=6, sharex=False,
    sharey=True, figsize=(15, 6))
for i in range(6):
    axes[i].set_title(y_train[i])
    axes[i].imshow(x_train[i], cmap='inferno')
    axes[i].get_xaxis().set_visible(True)
    axes[i].get_yaxis().set_visible(True)
plt.show()
```



Convert DATA into suitable format

```
#@title Convert DATA into suitable format

# Convert to float32.
x_train, x_test = np.array(x_train, np.float32), np.array(x_test, np.float32)
# Flatten images to 1-D vector of 784 features (28*28).
x_train, x_test = x_train.reshape([-1, num_features]), x_test.reshape([-1, num_feat
# Normalize images value from [0, 255] to [0, 1].
x_train, x_test = x_train / 255., x_test / 255.
```

Η βασική δομή του νευρωνικού μας δικτύου αποτελείται από ένα κρυφό στρώμα με 128 νευρώνες. Ο κατηγοριοποιητής που αρχικά επιλέγεται είναι ο stochastic gradient descent ενώ ως συνάρτηση απώλειας έχει επιλεχθεί η sparse categorical crossentropy ο οποίος προτιμάται έναντι της categorical crossentropy καθώς κάθε δείγμα ανήκει σε ακριβώς μια κλάση. Βασική μετρική αξιολόγησης είναι η ακρίβεια ενώ αρχικά εκπαιδεύουμε το μοντέλο μας αρχικα για 100 εποχές. By default το batch size μένει στο 32.

Create simple Neural Network model Neurons 128

```
#@title Create simple Neural Network model Neurons 128
model = Sequential()
model.add(Flatten(input_shape=(num_features, )))
model.add(Dense(128, activation='relu'))
model.add(Dense(10))

#Compile the Neural Network
model.compile(optimizer='SGD',loss =keras.losses.SparseCategoricalCrossentropy(fror

#See the details of our architecture
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_9 (Flatten)         (None, 784)               0

 dense_18 (Dense)            (None, 128)               100480

 dense_19 (Dense)            (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

Fit the train and testing data to the Neural Network Epochs 100

```
#@title Fit the train and testing data to the Neural Network Epochs 100
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_si:
```

```
Epoch 1/100
1875/1875 [==============================] - 9s 4ms/step - loss: 0.0386 - accuracy: 0.9908 - val_loss: 0.0726 - val_accuracy: 0.9785
Epoch 2/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0378 - accuracy: 0.9912 - val_loss: 0.0719 - val_accuracy: 0.9786
Epoch 3/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0371 - accuracy: 0.9912 - val_loss: 0.0716 - val_accuracy: 0.9783
Epoch 4/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0364 - accuracy: 0.9916 - val_loss: 0.0713 - val_accuracy: 0.9787
Epoch 5/100
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0358 - accuracy: 0.9919 - val_loss: 0.0723 - val_accuracy: 0.9776
Epoch 6/100
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0350 - accuracy: 0.9919 - val_loss: 0.0720 - val_accuracy: 0.9784
Epoch 7/100
1875/1875 [==============================] - 10s 6ms/step - loss: 0.0345 - accuracy: 0.9923 - val_loss: 0.0707 - val_accuracy: 0.9783
Epoch 8/100
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0338 - accuracy: 0.9924 - val_loss: 0.0709 - val_accuracy: 0.9783
Epoch 9/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0333 - accuracy: 0.9927 - val_loss: 0.0708 - val_accuracy: 0.9784
```

```
Epoch 92/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0102 - accuracy: 0.9993 - val_loss: 0.0691 - val_accuracy: 0.9794
Epoch 93/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0101 - accuracy: 0.9993 - val_loss: 0.0690 - val_accuracy: 0.9789
Epoch 94/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0100 - accuracy: 0.9993 - val_loss: 0.0691 - val_accuracy: 0.9793
Epoch 95/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0099 - accuracy: 0.9993 - val_loss: 0.0691 - val_accuracy: 0.9793
Epoch 96/100
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0098 - accuracy: 0.9994 - val_loss: 0.0691 - val_accuracy: 0.9794
Epoch 97/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0097 - accuracy: 0.9994 - val_loss: 0.0694 - val_accuracy: 0.9795
Epoch 98/100
1875/1875 [==============================] - 7s 4ms/step - loss: 0.0096 - accuracy: 0.9994 - val_loss: 0.0694 - val_accuracy: 0.9791
Epoch 99/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0095 - accuracy: 0.9994 - val_loss: 0.0693 - val_accuracy: 0.9790
Epoch 100/100
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0093 - accuracy: 0.9995 - val_loss: 0.0696 - val_accuracy: 0.9792
```

Παρατηρούμε στο διάγραμμα ότι μετά από περίπου 30 εποχές, η ακρίβεια πρόβλεψης στο σετ επικύρωσης σταθεροποιείται στο περίπου 97,9% και στη συνέχεια δεν έχουμε περαιτέρω άνοδο.

Reprsentation of the performance with a graph of a function for 128 Neurons in first hidden layer

```
#@title Reprsentation of the performance with a graph of a function for 128 Neurons
plt.figure(1)
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```

Στη συνέχεια δοκιμάζοντας να μειώσουμε τον αριθμό των εποχών σε 50 ενώ το πλήθος νευρώνων στο κρυφό στρώμα αυξάνεται στα 256 έχουμε τα παρακάτω αποτελέσματα:

Create simple Neural Network model neurons 256

```
#@title Create simple Neural Network model neurons 256
model = Sequential()
model.add(Flatten(input_shape=(num_features, )))
model.add(Dense(256, activation='relu'))
model.add(Dense(10))

#Compile the Neural Network
model.compile(optimizer='SGD',loss =keras.losses.SparseCategoricalCrossentropy(fron

#See the details of our architecture
model.summary()
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_7 (Flatten)         (None, 784)               0

 dense_14 (Dense)            (None, 256)               200960

 dense_15 (Dense)            (None, 10)                2570

=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
_________________________________________________________________
```

Fit the train and testing data to the Neural Network Epochs 50

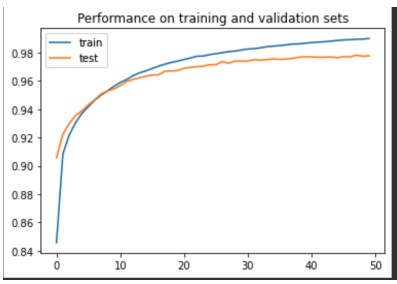
```
#@title Fit the train and testing data to the Neural Network Epochs 50
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_siz
```

```
Epoch 1/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.6237 - accuracy: 0.8457 - val_loss: 0.3470 - val_accuracy: 0.9058
Epoch 2/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.3276 - accuracy: 0.9085 - val_loss: 0.2830 - val_accuracy: 0.9225
Epoch 3/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2791 - accuracy: 0.9219 - val_loss: 0.2534 - val_accuracy: 0.9299
Epoch 4/50
1875/1875 [==============================] - 16s 8ms/step - loss: 0.2480 - accuracy: 0.9307 - val_loss: 0.2290 - val_accuracy: 0.9357
Epoch 5/50
1875/1875 [==============================] - 10s 6ms/step - loss: 0.2244 - accuracy: 0.9372 - val_loss: 0.2116 - val_accuracy: 0.9390
Epoch 6/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2058 - accuracy: 0.9420 - val_loss: 0.1949 - val_accuracy: 0.9432
Epoch 7/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1902 - accuracy: 0.9465 - val_loss: 0.1848 - val_accuracy: 0.9468
Epoch 8/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1771 - accuracy: 0.9501 - val_loss: 0.1723 - val_accuracy: 0.9509
Epoch 9/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1656 - accuracy: 0.9531 - val_loss: 0.1626 - val_accuracy: 0.9529
Epoch 10/50
```

```
Epoch 41/50
1875/1875 [==============================] - 12s 7ms/step - loss: 0.0508 - accuracy: 0.9872 - val_loss: 0.0774 - val_accuracy: 0.9771
Epoch 42/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0496 - accuracy: 0.9876 - val_loss: 0.0771 - val_accuracy: 0.9768
Epoch 43/50
1875/1875 [==============================] - 10s 6ms/step - loss: 0.0484 - accuracy: 0.9878 - val_loss: 0.0764 - val_accuracy: 0.9769
Epoch 44/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0473 - accuracy: 0.9883 - val_loss: 0.0753 - val_accuracy: 0.9771
Epoch 45/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0462 - accuracy: 0.9887 - val_loss: 0.0750 - val_accuracy: 0.9765
Epoch 46/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0452 - accuracy: 0.9891 - val_loss: 0.0742 - val_accuracy: 0.9773
Epoch 47/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0440 - accuracy: 0.9893 - val_loss: 0.0744 - val_accuracy: 0.9771
Epoch 48/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0431 - accuracy: 0.9895 - val_loss: 0.0730 - val_accuracy: 0.9783
Epoch 49/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0422 - accuracy: 0.9898 - val_loss: 0.0733 - val_accuracy: 0.9776
Epoch 50/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0412 - accuracy: 0.9901 - val_loss: 0.0719 - val_accuracy: 0.9778
```

Reprsentation of the performance with a graph of a function for 256 Neurons in first hidden layer

```
#@title Reprsentation of the performance with a graph of a function for 256 Neurons
plt.figure(2)
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```



Απο τα αποτελεσματα δεν παρατηρουμε διαφορές στην ακρίβεια εκπαίδευσης ενώ σε σχεση με τους 128 νευρώνες ο χρόνος που χρειάστηκε ήταν περισσότερος Σε επόμενη δοκιμή διαλέγουμε ως συνάρτηση ενεργοποίησης στην έξοδο την softmax ενώ αλλαζουμε σε loss function την Categorical Crossentropy.

Create simple Neural Network model neurons 256 Categorical and Output Softmax

```
#@title Create simple Neural Network model neurons 256 Categorical and Output Softm
model = Sequential()
model.add(Flatten(input_shape=(num_features, )))
model.add(Dense(256, activation='relu'))
model.add(Dense(10, activation='softmax'))

#Compile the Neural Network
model.compile(optimizer='SGD', loss =keras.losses.CategoricalCrossentropy(from_logi
```

```
#See the details of our architecture
model.summary()
```

```
Model: "sequential_18"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_18 (Flatten)        (None, 784)               0

 dense_36 (Dense)            (None, 256)               200960

 dense_37 (Dense)            (None, 10)                2570

=================================================================
Total params: 203,530
Trainable params: 203,530
Non-trainable params: 0
_________________________________________________________________
```

Fit the train and testing data to the Neural Network Epochs 50

```
#@title Fit the train and testing data to the Neural Network Epochs 50
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_si
```

Double-click (or enter) to edit

```
1875/1875 [==============================] - 13s 7ms/step - loss: 0.6200 - accuracy: 0.8466 - val_loss: 0.3488 - val_accuracy: 0.9063
Epoch 2/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.3293 - accuracy: 0.9081 - val_loss: 0.2851 - val_accuracy: 0.9211
Epoch 3/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2824 - accuracy: 0.9208 - val_loss: 0.2597 - val_accuracy: 0.9272
Epoch 4/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2523 - accuracy: 0.9293 - val_loss: 0.2319 - val_accuracy: 0.9363
Epoch 5/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2289 - accuracy: 0.9369 - val_loss: 0.2128 - val_accuracy: 0.9419
Epoch 6/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.2096 - accuracy: 0.9414 - val_loss: 0.2011 - val_accuracy: 0.9446
Epoch 7/50
1875/1875 [==============================] - 12s 7ms/step - loss: 0.1938 - accuracy: 0.9464 - val_loss: 0.1844 - val_accuracy: 0.9478
Epoch 8/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.1800 - accuracy: 0.9507 - val_loss: 0.1747 - val_accuracy: 0.9507
Epoch 9/50
1875/1875 [==============================] - 12s 7ms/step - loss: 0.1681 - accuracy: 0.9535 - val_loss: 0.1643 - val_accuracy: 0.9522
Epoch 10/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.1575 - accuracy: 0.9561 - val_loss: 0.1538 - val_accuracy: 0.9540
Epoch 11/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1488 - accuracy: 0.9589 - val_loss: 0.1473 - val_accuracy: 0.9560
Epoch 12/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1405 - accuracy: 0.9611 - val_loss: 0.1404 - val_accuracy: 0.9581
Epoch 13/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.1332 - accuracy: 0.9632 - val_loss: 0.1344 - val_accuracy: 0.9599
Epoch 14/50
1875/1875 [==============================] - 13s 7ms/step - loss: 0.1267 - accuracy: 0.9654 - val_loss: 0.1296 - val_accuracy: 0.9618
Epoch 15/50
```

```
Epoch 40/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0530 - accuracy: 0.9870 - val_loss: 0.0783 - val_accuracy: 0.9776
Epoch 41/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0518 - accuracy: 0.9871 - val_loss: 0.0777 - val_accuracy: 0.9770
Epoch 42/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0506 - accuracy: 0.9876 - val_loss: 0.0759 - val_accuracy: 0.9780
Epoch 43/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0494 - accuracy: 0.9876 - val_loss: 0.0760 - val_accuracy: 0.9772
Epoch 44/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0483 - accuracy: 0.9881 - val_loss: 0.0751 - val_accuracy: 0.9778
Epoch 45/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0471 - accuracy: 0.9886 - val_loss: 0.0751 - val_accuracy: 0.9773
Epoch 46/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0460 - accuracy: 0.9890 - val_loss: 0.0744 - val_accuracy: 0.9775
Epoch 47/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0451 - accuracy: 0.9889 - val_loss: 0.0741 - val_accuracy: 0.9779
Epoch 48/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0441 - accuracy: 0.9894 - val_loss: 0.0729 - val_accuracy: 0.9782
Epoch 49/50
1875/1875 [==============================] - 11s 6ms/step - loss: 0.0430 - accuracy: 0.9899 - val_loss: 0.0730 - val_accuracy: 0.9783
Epoch 50/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0421 - accuracy: 0.9900 - val_loss: 0.0723 - val_accuracy: 0.9786
```

Reprsentation of the performance with a graph of a function for 256 Categorical and Output Softmax

```
#@title Reprsentation of the performance with a graph of a function for 256 Categor
plt.figure(2)
```

```
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```

Δεν παρατηρούμε σχεδόν καμία διαφορά με την αλλαγή συναρτήσεων ενεργοποίησης και απώλειας.

Στη συνέχεια αλλάζουμε μόνο τον αριθμό των νευρώνων από 256 σε 32 στο κρυφό στρώμα.

Create simple Neural Network model neurons 32 Categorical and Output Softmax

```
#@title Create simple Neural Network model neurons 32 Categorical and Output Softma
model = Sequential()
model.add(Flatten(input_shape=(num_features, )))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))

#Compile the Neural Network
model.compile(optimizer='SGD', loss =keras.losses.SparseCategoricalCrossentropy(fro

#See the details of our architecture
model.summary()
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_9 (Flatten)         (None, 784)               0

 dense_16 (Dense)            (None, 32)                25120

 dense_17 (Dense)            (None, 10)                330

=================================================================
Total params: 25,450
Trainable params: 25,450
Non-trainable params: 0
_________________________________________________________________
```

Fit the train and testing data to the Neural Network Epochs 50

```
#@title Fit the train and testing data to the Neural Network Epochs 50
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_siz
```

```
1875/1875 [==============================] - 5s 3ms/step - loss: 0.7316 - accuracy: 0.8102 - val_loss: 0.3816 - val_accuracy: 0.8968
Epoch 2/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3589 - accuracy: 0.8995 - val_loss: 0.3138 - val_accuracy: 0.9126
Epoch 3/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3105 - accuracy: 0.9123 - val_loss: 0.2833 - val_accuracy: 0.9216
Epoch 4/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2834 - accuracy: 0.9194 - val_loss: 0.2624 - val_accuracy: 0.9260
Epoch 5/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2634 - accuracy: 0.9251 - val_loss: 0.2479 - val_accuracy: 0.9292
Epoch 6/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2480 - accuracy: 0.9299 - val_loss: 0.2363 - val_accuracy: 0.9328
Epoch 7/50
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2349 - accuracy: 0.9333 - val_loss: 0.2238 - val_accuracy: 0.9364
Epoch 8/50
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2230 - accuracy: 0.9370 - val_loss: 0.2174 - val_accuracy: 0.9388
Epoch 9/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2128 - accuracy: 0.9393 - val_loss: 0.2071 - val_accuracy: 0.9412
```


```
1875/1875 [==============================] - 5s 3ms/step - loss: 0.7316 - accuracy: 0.8102 - val_loss: 0.3816 - val_accuracy: 0.8968
Epoch 2/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3589 - accuracy: 0.8995 - val_loss: 0.3138 - val_accuracy: 0.9126
Epoch 3/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.3105 - accuracy: 0.9123 - val_loss: 0.2833 - val_accuracy: 0.9216
Epoch 4/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2834 - accuracy: 0.9194 - val_loss: 0.2624 - val_accuracy: 0.9260
Epoch 5/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2634 - accuracy: 0.9251 - val_loss: 0.2479 - val_accuracy: 0.9292
Epoch 6/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2480 - accuracy: 0.9299 - val_loss: 0.2363 - val_accuracy: 0.9328
Epoch 7/50
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2349 - accuracy: 0.9333 - val_loss: 0.2238 - val_accuracy: 0.9364
Epoch 8/50
1875/1875 [==============================] - 6s 3ms/step - loss: 0.2230 - accuracy: 0.9370 - val_loss: 0.2174 - val_accuracy: 0.9388
Epoch 9/50
1875/1875 [==============================] - 5s 3ms/step - loss: 0.2128 - accuracy: 0.9393 - val_loss: 0.2071 - val_accuracy: 0.9412
```


Reprsentation of the performance with a graph of a function for 32 Categorical and Output Softmax

```
#@title Reprsentation of the performance with a graph of a function for 32 Categori
plt.figure(4)
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```

Όπως είναι αναμενόμενο, η ακρίβεια μειώνεται σε κάποιο μικρό βαθμό λόγω της μείωσης του αριθμού των νευρώνων αλλά έχουμε και σημαντική μείωση του χρόνου εκπαίδευσης(σχεδόν 2 λεπτά).

Στη συνέχεια αυξάνουμε το batch size από 32 σε 256 και αναμένουμε περαιτέρω μείωση του χρόνου εκπαίδευσης αλλά και πιθανή μείωση της μετρικής της ακρίβειας, γεγονός που τελικά συμβαίνει:

Fit the train and testing data to the Neural Network Epochs 50

```
#@title Fit the train and testing data to the Neural Network Epochs 50
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_si
```

```
Epoch 1/50
235/235 [==============================] - 2s 9ms/step - loss: 0.0808 - accuracy: 0.9775 - val_loss: 0.1064 - val_accuracy: 0.9686
Epoch 2/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0806 - accuracy: 0.9775 - val_loss: 0.1064 - val_accuracy: 0.9678
Epoch 3/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0804 - accuracy: 0.9774 - val_loss: 0.1066 - val_accuracy: 0.9679
Epoch 4/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0803 - accuracy: 0.9774 - val_loss: 0.1065 - val_accuracy: 0.9681
Epoch 5/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0801 - accuracy: 0.9774 - val_loss: 0.1063 - val_accuracy: 0.9678
Epoch 6/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0800 - accuracy: 0.9775 - val_loss: 0.1063 - val_accuracy: 0.9678
Epoch 7/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0798 - accuracy: 0.9777 - val_loss: 0.1061 - val_accuracy: 0.9683
Epoch 8/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0797 - accuracy: 0.9775 - val_loss: 0.1062 - val_accuracy: 0.9675
Epoch 9/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0796 - accuracy: 0.9774 - val_loss: 0.1061 - val_accuracy: 0.9679
Epoch 10/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0795 - accuracy: 0.9776 - val_loss: 0.1060 - val_accuracy: 0.9681
```

```
Epoch 40/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0759 - accuracy: 0.9785 - val_loss: 0.1040 - val_accuracy: 0.9680
Epoch 41/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0758 - accuracy: 0.9788 - val_loss: 0.1043 - val_accuracy: 0.9680
Epoch 42/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0757 - accuracy: 0.9789 - val_loss: 0.1040 - val_accuracy: 0.9678
Epoch 43/50
235/235 [==============================] - 1s 6ms/step - loss: 0.0756 - accuracy: 0.9787 - val_loss: 0.1042 - val_accuracy: 0.9679
Epoch 44/50
235/235 [==============================] - 2s 7ms/step - loss: 0.0755 - accuracy: 0.9787 - val_loss: 0.1040 - val_accuracy: 0.9679
Epoch 45/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0754 - accuracy: 0.9790 - val_loss: 0.1043 - val_accuracy: 0.9683
Epoch 46/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0753 - accuracy: 0.9790 - val_loss: 0.1037 - val_accuracy: 0.9682
Epoch 47/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0751 - accuracy: 0.9789 - val_loss: 0.1038 - val_accuracy: 0.9682
Epoch 48/50
235/235 [==============================] - 2s 6ms/step - loss: 0.0750 - accuracy: 0.9790 - val_loss: 0.1038 - val_accuracy: 0.9683
Epoch 49/50
235/235 [==============================] - 2s 10ms/step - loss: 0.0749 - accuracy: 0.9791 - val_loss: 0.1039 - val_accuracy: 0.9684
Epoch 50/50
235/235 [==============================] - 1s 5ms/step - loss: 0.0748 - accuracy: 0.9791 - val_loss: 0.1038 - val_accuracy: 0.9680
```

Reprsentation of the performance with a graph of a function for 32 Categorical and Output Softmax

```
#@title Reprsentation of the performance with a graph of a function for 32 Categori
plt.figure(4)
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```

Επιστρέφοντας στον αρχικό μας αλγόριθμο, οι εποχές θα ρυθμιστούν στις 50 ενώ θα προσθέσουμε και άλλο ένα κρυφό στρώμα 128 νευρώνων:

Create simple Neural Network model Neurons 128

```
#@title Create simple Neural Network model Neurons 128
model = Sequential()
model.add(Flatten(input_shape=(num_features, )))
model.add(Dense(128, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(10))

#Compile the Neural Network
model.compile(optimizer='SGD',loss =keras.losses.SparseCategoricalCrossentropy(fror

#See the details of our architecture
model.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_10 (Flatten)        (None, 784)               0

 dense_18 (Dense)            (None, 128)               100480

 dense_19 (Dense)            (None, 128)               16512

 dense_20 (Dense)            (None, 10)                1290

=================================================================
Total params: 118,282
Trainable params: 118,282
Non-trainable params: 0
_________________________________________________________________
```

Fit the train and testing data to the Neural Network Epochs 50

```
#@title Fit the train and testing data to the Neural Network Epochs 50
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_si:
```

```
Epoch 1/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.6295 - accuracy: 0.8368 - val_loss: 0.3158 - val_accuracy: 0.9076
Epoch 2/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2922 - accuracy: 0.9162 - val_loss: 0.2563 - val_accuracy: 0.9274
Epoch 3/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2419 - accuracy: 0.9307 - val_loss: 0.2166 - val_accuracy: 0.9373
Epoch 4/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2093 - accuracy: 0.9402 - val_loss: 0.1927 - val_accuracy: 0.9431
Epoch 5/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1844 - accuracy: 0.9471 - val_loss: 0.1717 - val_accuracy: 0.9503
Epoch 6/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1646 - accuracy: 0.9531 - val_loss: 0.1566 - val_accuracy: 0.9537
Epoch 7/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1487 - accuracy: 0.9575 - val_loss: 0.1419 - val_accuracy: 0.9587
Epoch 8/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1352 - accuracy: 0.9610 - val_loss: 0.1329 - val_accuracy: 0.9622
```

```
Epoch 42/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0219 - accuracy: 0.9954 - val_loss: 0.0761 - val_accuracy: 0.9772
Epoch 43/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0213 - accuracy: 0.9957 - val_loss: 0.0754 - val_accuracy: 0.9776
Epoch 44/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0203 - accuracy: 0.9962 - val_loss: 0.0772 - val_accuracy: 0.9773
Epoch 45/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0194 - accuracy: 0.9963 - val_loss: 0.0789 - val_accuracy: 0.9768
Epoch 46/50
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0184 - accuracy: 0.9964 - val_loss: 0.0770 - val_accuracy: 0.9773
Epoch 47/50
1875/1875 [==============================] - 8s 5ms/step - loss: 0.0179 - accuracy: 0.9966 - val_loss: 0.0760 - val_accuracy: 0.9770
Epoch 48/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0170 - accuracy: 0.9970 - val_loss: 0.0752 - val_accuracy: 0.9774
Epoch 49/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0163 - accuracy: 0.9973 - val_loss: 0.0748 - val_accuracy: 0.9770
Epoch 50/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0155 - accuracy: 0.9975 - val_loss: 0.0772 - val_accuracy: 0.9764
```

Παρατηρούμε την μείωση ακρίβειας και στα 2 σετ ,ωστόσο ο χρόνος εκπαίδευσης μειώνεται

Reprsentation of the performance with a graph of a function for 32 Categorical and Output Softmax

```
#@title Reprsentation of the performance with a graph of a function for 32 Categori
plt.figure(5)
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```

Create simple Neural Network model Neurons 128

```
#@title Create simple Neural Network model Neurons 128
model = Sequential()
model.add(Flatten(input_shape=(num_features, )))
model.add(Dense(128, activation='relu'))
model.add(Dense(10))

#Compile the Neural Network
model.compile(optimizer='adam',loss =keras.losses.SparseCategoricalCrossentropy(fr

#See the details of our architecture
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_11 (Flatten)        (None, 784)               0

 dense_21 (Dense)            (None, 128)               100480

 dense_22 (Dense)            (None, 10)                1290

=================================================================
```

```
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

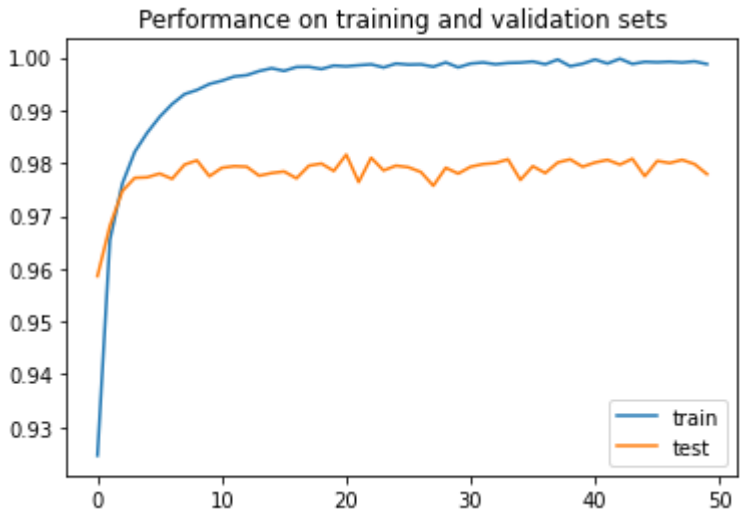
Fit the train and testing data to the Neural Network Epochs 50

```
#@title Fit the train and testing data to the Neural Network Epochs 50
history = model.fit(x_train, y_train, validation_data = (x_test, y_test), batch_si
```

```
Epoch 1/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2620 - accuracy: 0.9246 - val_loss: 0.1411 - val_accuracy
Epoch 2/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1170 - accuracy: 0.9654 - val_loss: 0.1061 - val_accuracy
Epoch 3/50
1875/1875 [==============================] - 9s 5ms/step - loss: 0.0805 - accuracy: 0.9761 - val_loss: 0.0873 - val_accuracy
Epoch 4/50
1875/1875 [==============================] - 12s 6ms/step - loss: 0.0595 - accuracy: 0.9821 - val_loss: 0.0788 - val_accurac
Epoch 5/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0465 - accuracy: 0.9858 - val_loss: 0.0744 - val_accuracy
Epoch 6/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0365 - accuracy: 0.9887 - val_loss: 0.0780 - val_accuracy
Epoch 7/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0291 - accuracy: 0.9912 - val_loss: 0.0776 - val_accuracy
Epoch 8/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0231 - accuracy: 0.9930 - val_loss: 0.0766 - val_accuracy
Epoch 9/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0195 - accuracy: 0.9938 - val_loss: 0.0719 - val_accuracy
Epoch 10/50
1875/1875 [==============================] - 8s 4ms/step - loss: 0.0166 - accuracy: 0.9950 - val_loss: 0.0855 - val_accuracy
```

Reprsentation of the performance with a graph of a function for 32 Categorical and Output Softmax

```
#@title Reprsentation of the performance with a graph of a function for 32 Categori
plt.figure(6)
plt.plot(history.history['accuracy'], label = 'train')
plt.plot(history.history['val_accuracy'], label = 'test')
plt.legend()
plt.title('Performance on training and validation sets')
plt.show()
```



Από την παραπάνω εικόνα βλέπουμε πως μετά από ιδιαίτερα μικρό πλήθος εποχών(περίπου 10 εποχές) η ακρίβεια του τεστ επικύρωσης σταματάει να αυξάνεται και σε κάποιες περιπτώσεις μειώνεται, με την ακρίβεια του σετ εκπαίδευσης να 'τελειοποιείται' μετά από περίπου 25 εποχές(περίπτωση overfitting). Σημαντική θεωρείται και η μείωση του χρόνου εκπαίδευσης

Comparison of CNN with K-NN and Nearest Centroid

Συγκρίνοντας βάσει ακρίβειας τις επιδόσεις του CNN με τους κατηγοριοποιητές της ενδιάμεσης εργασίας(1-KNN, NCC), βγάζουμε το συμπέρασμα πως το CNN κρίνεται μάλλον πιο αποτελεσματικό λόγω της μεγαλύτερης ακρίβειας που προσφέρει 2-3% στις καλύτερες υλοποίησεις του CNN. Η παραπάνω διαπίστωση είναι λογική αν σκεφτούμε ότι στο KNN είχαμε χρησιμοποιήσει αρκετά μικρό K με αποτέλεσμα το ενδεχόμενο σφάλματος να είναι μεγαλύτερο.

↳ *2 cells hidden*