Project 3
Briakos Spyros (1115201700101)
Apostolopoulou Alexandra (1115201700005)

Compilation:

- ***For the first question:*** python src/reduce.py -d data/train/train-images-idx3-ubyte -q data/test/t10k-images-idx3- ubyte -od data/new/newtrain-images-idx3-ubyte -oq data/new/newt10k-images-idx3-ubyte

- ***For the second question:*** ./scripts/lsh.sh

- ***For the third question:*** python src/emd.py -d data/train/train-images-idx3-ubyte -q data/test/t10k-images-idx3-ubyte -l1 data/train-labels-idx1-ubyte -l2 data/test/t10k-labels-idx1-ubyte -o output -EMD

- ***For the fourth question:*** ./scripts/cluster.sh

Query A

In this question we have built a neural image self-coding network which includes compression and decompression layers ("bottleneck"). It should be noted that no we have conducted extensive network training experiments again with different values of hyperparameters such as the number of layers, filter size, number of filters per stratum, number of training seasons (epochs), size batch size, as we experimented with these values in previous work. So, we had some "certain" values for these overriding ones, which we'll be able to they gave the best results, and we experimented mainly with the compression dimension (latent dimension) to minimize loss by avoiding overfitting. The input set data was properly divided into training set and validation set. General, we followed the structure proposed by the teachers, as it presented quite good results without overfitting. Here are some results from our various experimentation:

	Training Loss	Validation Loss
Units of Dense Layer = 5	0.0242	0.0238
Units of Dense Layer = 10	0.0129	0.0127
Units of Dense Layer = 15	0.0087	0.0084

For these results, we have given the following values to hyperparameters:
- Epochs = 20 (although in fewer occurrences the algorithm was already converging)
- Batch size = 128 (Pretty good results and in a short time)
- Number of Convolution Layers = 3
- Kernel size = (3.3)
- Number of filters by each Layer = [32,64,128]

Units of Dense Layer is its most important parameter, indicating how many neurons has each layer, output size of layer, and determines the size of weight matrix and bias vector.

As the results have shown, it makes sense, the more units we have, the more our model becomes complex, so we have less loss and better results. However, we thought that the loss of 0.012 was already too small, because learning curves were slightly more smooth for dense units 10

versus 15 dense units, we decided to keep 10 units, as the dataset was already very simple and as we increased the complexity with dense units the risk of overfitting was increasing.

So after selected the the value of 10 dense units (as proposed by the teachers), all we had to do was cut our model in encoder so that we can compress our data in only 10 dimensions. Then, we normalized our data values to 0-25500, more precisely, as the compression from 28*28 to 10*1 was too strong.

Finally, we saved the data in two new files in a similar format to input file, with number of rows 1 and number of columns equal to number of dimensions (10). Coordinates saved in 2 consecutive bytes using function to_bytes() python that allowed us to store a whole number in as many bytes as we want any byteorder we want.

Question B

As requested in project's announcement, we are able to compare the searches (which were clearly formulated in announcement) which were made in the initial and in the new vector space, having previously opted fixed values for significant variables, such as W=20,000 and m=423255.

In particular, we have conducted experiments with different sizes of training set and test set through which we have seen slight differences, but in order to be more representative we will present the results with as much information as possible! The training set is made up of 60.000 images and there are 10.000 images in the test set.

	LSH	Reduced
Approximation Factor	1.15	1.35

It was observed in this experiment that approximation Factors (AFs) received a value about 1.15 for LSH and about 1.35 for Reduced. We clearly understand that it is very satisfactory score when you consider that in the first score each photo was information through a 784-dimensional vector, and in the second just 10-dimensional!

As regards time, what we have seen in general terms is that TReduced it was 2-3 times larger than tLSH, but it's still way smaller (6-8 times less) than the brute-force, tTrue, search time in the original search space. This lead us to the result that the exhaustive search in the new vector space has achieved its goal, due to the fact that it has much reduced time, so timely speaking it is the most efficient searching, also note that difference is not so big compared to LSH!

Query C

In this query we found the Earth Mover’s Distance (EMD) metric that aims to give solution to a Linear Programming Problem. Earth Mover's Distance concerns a measurement between two distributions and is based on the minimum cost that must be paid to convert one distribution into another. It is important that EMD as a method it has been shown to be better than other distances used for retrieve/match images.

EMD belongs to the general category of Transportation Problems, where suppliers, with a given quantity of goods, and consumers, with a given limited capacity. In our problem, the suppliers are the subclusters of the train images, and the consumers are the (sub)clusters of test images.

An image is divided into a set of clusters, either 7x7 or 4x4 , in order to be able to make the appropriate calculations, based on the fact that we do not have the necessary computational power to find the EMD metric between pixels. For each pair of supplier-consumer we have calculated the cost of transporting a unit of goods, which is simply the Euclidean distance between each classroom's centers. The problem of transport is to find the least expensive flow of goods from suppliers to consumers, that satisfy consumer demand.

In our image problem (signature mapping) this is translated into the minimum amount of work so as to convert one signature into another. According to the slides of the course, this can be captured as the following problem linear programming:

Suppose $P = \{(p_1, w_{p1}), \ldots, (p_m, w_{pm})\}$ is the first signature with m clusters, where the representation of the cluster and w_{pi} the weights of each cluster. Suppose $Q = \{(q_1, w_{q1}), \ldots, (q_n, w_{qn})\}$ or second signature with n clusters and $D=[d_{ij}]$the distance between clusters p_i and q_j. What we want is to find that flow $F = [f_{ij}]$, where f_{ij} the flow between the p_i and q_j ,which minimises the total cost of:

$$\text{WORK}(P,Q,F) = \sum_{i=1}^{m}\sum_{j=1}^{n} \square_{ij} f_{ij}$$

In order to achieve this, the following restrictions (according to Wikipedia):

(1) $f_{ij} \geq 0$, όπου $1 \leq i \leq m$ και $1 \leq j \leq n$

(2) $\sum_{j=1}^{n} \square_{ij} \leq w_{pi} \quad 1 \leq i \leq m$

(3) $\sum_{i=1}^{m} \square_{ij} \leq \square_{qj} \quad 1 \leq j \leq n$

(4) $\sum_{i=1}^{m}\sum_{j=1}^{n} \square_{ij} = \square\left(\sum_{i=1}^{m} \square_{pi}, \sum_{j=1}^{n} \square_{qj}\right)$

Restriction (1) allows supplies to be moved from P to Q and not vice versa. Therestriction (2) limits the amount of supplies that can be sent from the clusters of P not to exceed their weights. Restriction (3) restricts Q clusters not to receive more supplies than their weights. Restriction (4) forces you to the maximum possible quantity of supplies is moved. This is what we call total flow.

EMD occurs if we pulverize work with total flow:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{n} d_{ij} f_{ij}}$$

We applied all this to our code with the help of the Pulp library, which allowed us to set an LpProblem with all the clusters of a train and test image. Also for each image we calculated its weights and costs of a transportation as well as all different routes that can exist between sub-clusters. After we've defined all this, we added the above restrictions, saved the optimized value that we turned the objective function to a priority queue for each comparison of a query image with a train picture, to be able to pull the 10 closest neighbors out of every query.

At the same time, each time we stored the distances and neighbors we found with Manhattan Distance, as requested by the pronunciation, in order to make a comparison of success rates. Here are some of the results for some of the where we ran:

Για size of cluster = 7x7:

	EMD Success Rate	*Manhattan Success Rate*
10 train images – 2 queries	5%	5%
100 train images – 2 queries	20%	30%
1000 train images – 2 queries	30%	75%

	EMD Success Rate	*Manhattan Success Rate*
10 train images – 5 queries	14%	14%
100 train images – 5 queries	36%	56%
1000 train images – 5 queries	56%	80%

Για size of cluster = 4x4:

	EMD Success Rate	*Manhattan Success Rate*
10 train images – 2 queries	5%	5%
100 train images – 2 queries	25%	30%
1000 train images – 2 queries	70%	75%

	EMD Success Rate	*Manhattan Success Rate*
10 train images – 5 queries	14%	14%
100 train images – 5 queries	42%	56%
1000 train images – 5 queries	76%	80%
10000 train images – 5 queries	93%	96%

As we observe, the more we increase our data and the number of queries, the more precision is given by EMD and Manhattan distance, which is to be expected.

One important observation is that the smaller the size of the clusters, the more precision gives us the EMD. This is because as long as we "break" the image into smaller "windows", the more detail there is and that's why there's greater performance. Of course, this, as expected, costed a lot in time. For example, 5 queries with 1000 trains with a size of 4x4 versus the same data with a cluster size of 7x7 was much more time-consuming. That's what we understand that prevented us from experiment with smaller cluster size, because we did not have the desirable computational power.

Another vital observation we see from the above signs, is that even if we reduce the size of the clusters and increase our data, the accuracy of EMD approaches but does not reach (and above all does not exceed) the accuracy Manhattan Distance gives us. Manhattan Distance is a simple metric that in substance makes Brute Force in our data and that's why it seems to have more accuracy. But because the signs show that the more our data grows and the size of clusters, so

decreases the difference of EMD-Manhattan accuracy, and so we can conclude that if, for example, the EMD was compared with a very small window size (e.g. pixel-pixels) and with more data, the EMD metric will intuitively would be over the accuracy of Manhattan Distance. But unfortunately, we cannot do that due to a lack of adequate computing power.

Note: Comparing this query to query B would not make much sense, as query C is implemented in python language while B query in C++. In addition, we do not really know the exact algorithms and complexity of the Pulp library in order to calculate the EMD metric, so it would be pointless to make such a comparison, without knowing all our data.

Question D

With regard to this question, we have also explicitly followed the requirements of the pronunciation and so we performed three different approaches of clustering, in which there was no big difference between values of objective function.

	Σ1	***Σ2***	***Σ3***
Silhouette	0.16	0.10	0.16

Initially the first clustering that took place in the new vector space converged with just 6 centroid updates, while as a termination condition we had set that the rate of change of objective function to be less than a constant e=0.001. Also clustering, which mentioned right previously, had a final silhouette of 0.16 out of a total of 10,000 images.

On the other hand, the second clustering, which took place in the original vector space (of 784 dimensions) with all other restrictions remaining the same, converged on just 7 centroid updates (based on k-median technique) and had a final silhouette of 0.10.

Finally, clustering which was based on the labels of deliverable of our second assignment had similar effects to those of clustering in the new vector space (with final silhouette 0.16). In this particular clustering we can justify the improved silhouette compared to the original vector space, since the labels to which the images were assigned were the result of the classification, which we had implementation in the second assignment, of a neural network, which was trained in exactly this data. With this approach we conclude that since our neural network 'knew' our data very well, it would also have extremely accurate predictions, which would lead to good clustering of images and therefore a satisfactory silhouette.

Note: *With regard to queries B and D, there was special treatment in reading file that generated query A, as each photo was now represented by 10 number of 2 bytes. This is why a separate read_binary_file exist, which its goal is to read gradually with the appropriate endianess 2 bytes serially. Finally, all results are printed temporarily in the external output file.*