Project
Εργασία 3

Μέλη (Ομάδα 32):
Σπυρίδων Μπριάκος (1115201700101)
Αποστολοπούλου Αλεξάνδρα (1115201700005)

-Μεταγλώττιση:

- *Για το Α ερώτημα:*
  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
- *Για το Β ερώτημα:*
  ./scripts/lsh.sh
- *Για το Γ ερώτημα:*
  python src/emd.py -d data/train/train-images-idx3-ubyte -q data/test/t10k-images-idx3-ubyte -l1 data/train/train-labels-idx1-ubyte -l2 data/test/t10k-labels-idx1-ubyte -o output -EMD
- *Για το Δ ερώτημα:*
  ./scripts/cluster.sh
  Στο συγκεκριμένο script μπορείτε να σχολιάσετε/απασχολιάσετε την 2η και 3η γραμμή αντιστοίχως, ούτως ώστε να δώσετε σαν input dataset το "train dataset", με 60.000 εικόνες ή το "test dataset", με 10.000 εικόνες. Εμείς δοκιμάσαμε τις περισσότερες των περιπτώσεων το μικρό dataset για λόγους εξοικονόμησης χρόνου.

-Ερώτημα Α

Σε αυτό το ερώτημα κατασκευάσαμε ένα νευρωνικό δίκτυο αυτοκωδικοποίησης εικόνων το οποίο περιλαμβάνει στρώματα συμπίεσης και αποσυμπίεσης (“bottleneck”). Πρέπει να σημειωθεί ότι δεν πραγματοποιήσαμε ξανά εκτενή πειράματα εκπαίδευσης του δικτύου με διαφορετικές τιμές υπερπαραμέτρων όπως τον αριθμό συνελικτικών στρωμάτων, μεγέθους συνελικτικών φίλτρων, αριθμού συνελικτικών φίλτρων ανά στρώμα, αριθμού εποχών εκπαίδευσης (epochs), μεγέθους δέσμης (batch size), καθώς είχαμε πειραματιστεί με αυτές τις τιμές σε προηγούμενη εργασία.

Έτσι, είχαμε κάποιες “σιγουρες” τιμές γι’ αυτές τις υπερπαραμέτους, οι οποίες θα μας έδιναν τα βέλτιστα αποτελέσματα, και πειραματιστήκαμε κυρίως με την διάσταση συμπίεσης (latent dimension) ώστε να ελαχιστοποιήσουμε το σφάλμα (loss) αποφεύγοντας την υπερπροσαρμογή (overfitting). Τα δεδομένα του συνόλου εισόδου τα χωρισαμε κατάλληλα σε σύνολο εκπαίδευσης (training set) και σε σύνολο επικύρωσης (validation set). Γενικά, ακολουθήσαμε την δομή σημίεσης που προτάθηκε από τους διδάσκοντες, μιας και παρουσίαζε αρκετά καλά αποτελέσματα χωρίς overfitting. Όρίστε μερικά αποτελέσματα από διάφορους πειραματισμούς μας:

	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

Για τα αποτελέσματα αυτά, δώσαμε τις εξής τιμές στις διάφορες υπερπαραμέτρους:

- Epochs = 20 (Αν και σε λιγότερες επαναλήψεις ο αλγόριθμος ήδη αρχιζε να συγκλίνει)
- Batch size = 128 (Αρκετά καλά αποτελέσματα και σε λίγο χρόνο)
- Number of Convolution Layers = 3
- Kernel size = (3,3)
- Number of filters by each Layer = [32,64,128]

Τα units του Dense Layer είναι η πιο σημαντική παράμετρος του, καθώς δηλώνει το πόσα neurons έχει το κάθε layer, το output size του layer και καθορίζει το μέγεθος του weight matrix και του bias vector.

Όπως έδειξαν και τα αποτελέσματα, λογικό είναι, όσο περισσότερα units έχουμε τόσο πιο περίπλοκο γίνεται το μοντέλο μας και άρα έχουμε λιγότερο loss και καλύτερα αποτελέσματα. Ωστόσω, θεωρήσαμε ότι το loss της τάξης του 0.012 είναι ήδη πολύ μικρό κι επειδή τα learning curves ήταν ελάχιστα πιο smooth για dense units 10 έναντι με τα 15 dense units, αποφασίσαμε να κρατήσουμε τα 10 units, καθώς το dataset ήταν ήδη πολυ απλό και όσο αυξάναμε την πολυπλοκότητα με τα dense units ελλόχευε ο κίνδυνος για overfitting.

Έφόσον, λοιπόν, καταλήξαμε και στον αριθμό των 10 dense units (όπως προτάθηκε και από τους διδάσκοντες), το μόνο που έμενε ήταν να "κόψουμε" το μοντέλο μας στον encoder, έτσι ώστε να μπορέσουμε να συμπιέσουμε τα δεδομένα μας σε 10 μόνο διαστάσεις. Έπειτα, κανονικοποιήσαμε τις τιμές των δεδομένων μας σε 0-25500, για περισσότερη ακρίβεια, καθώς η συμπίεση απο 28*28 σε 10*1 ήταν μεγάλη.

Τέλος, αποθηκεύσαμε τα δεδομένα σε δυο καινούρια αρχεία με παρόμοια μορφή με το αρχείο εισόδου, με αριθμό γραμμών 1 και αριθμό στηλών ίσο με το πλήθος των διαστάσεων (10). Οι συντεταγμένες αποθηκέυτηκαν σε 2 διαδοχικά bytes με την βοήθεια της συνάρτησης to_bytes() της python που μας επέτρεπε να αποθηκεύσουμε έναν ακέραιο αριθμό σε όσα διαδοχικα bytes θέλουμε και με όποια byteorder θέλουμε.

-Ερώτημα Β

Όπως ζητήθηκε στην εκφώνηση της εργασίας στο συγκεκριμένο ερώτημα είμαστε σε θέση να συγκρίνουμε τις αναζητήσεις (όπως αυτές διατυπώθηκαν ρητά στην εκφώνηση) που έγιναν στο αρχικό και στον νέο διανυσματικό χώρο, έχοντας βέβαια διαλέξει κάποιες σταθερές τιμές σε σημαντικές μεταβλητές, όπως W=20.000 και m=423255 (από την πρώτη εργασία).

Πιο συγκεκριμένα εκτελέσαμε πειράματα με διαφορετικά μεγέθη training set και test set μέσω των οποίων είδαμε μικρές διαφορές, όμως για να είμαστε πιο αντιπροσωπευτικοί θα παρουσιάσουμε τα αποτελέσματα με όσο δυνατόν περισσότερη πληροφορία! Επομένως, στο training set υπάρχουν 60.000 εικόνες και στο test set υπάρχουν 10.000 εικόνες.

	LSH	***Reduced***
Approximation Factor	1.15	1.35

Παρατηρήθηκε στο συγκεκριμένο πείραμα πως τα Approximation Factors (Afs) έλαβαν τιμή περίπου 1.15 για LSH και περίπου 1.35 για Reduced. Κάτι τέτοιο, καταλαβαίνουμε ότι είναι πολύ ικανοποιητικό σκορ αν αναλογιστεί κανείς πως στο πρώτο σκορ η κάθε φωτογραφία είχε πληροφορία μέσω ενός διανύσματος 784 διαστάσεων, ενώ στο δεύτερο μόλις 10 διαστάσεων!

Όσον αφορά τον χρόνο αυτό που παρατηρήσαμε σε γενικές γραμμές είναι πως ο tReduced ήταν 2-3 φορές μεγαλύτερος από τον tLSH, αλλά ακομά εξαιρετικά μικρότερος (6-8 φορές μικρότερος) από τον εξαντλητικό χρόνο αναζήτησης tTrue στον αρχικό χώρο αναζήτησης. Πράγμα το οποίο σημαίνει πως η εξαντλητική αναζήτηση στον νέο διανυσματικό χώρο πέτυχε τον σκοπό της, αφού έχει και αρκετά μειωμένο χρόνο, άρα είναι χρονικά αποδοτικότερη η αναζήτηση, αλλά και σαν ακρίβεια απέδωσε αρκετά τίμια με μικρό ποσοστό λάθους συγκριτικά με τον LSH.

-Ερώτημα Γ

Σε αυτό το ερώτημα βρήκαμε τη μετρική *Earth Mover's Distance (EMD)* που ανάγεται σε επίλυση προβλήματος Γραμμικού Προγραμματισμού (*Linear Programming*). Η *Earth Mover's Distance* αφορά μία μέτρηση μεταξύ δύο διανομών και βασίζεται στο ελάχιστο κόστος που πρέπει να καταβληθεί για να μετατραπεί η μία διανομή στην άλλη. Σημαντικό είναι το γεγονός ότι η EMD σαν μέθοδος έχει αποδειχθεί ότι είναι καλύτερη από άλλες αποστάσεις που χρησιμοποιούνται για την ανάκτηση/ομοιότητας εικόνων.

Η EMD ανήκει στην γενική κατηγορία των *Transportation Problems*, όπου υπάρχουν οι προμηθευτές, με δεδομένη ποσότητα αγαθών, και οι καταναλωτές, με δεδομένη περιορισμένη χωρητικότητα. Στο πρόβλημά μας, οι προμηθευτές είναι τα (υπο)clusters των train images, ενώ οι καταλαλωτές είναι τα (υπο)clusters των test images.

Μία εικόνα χωρίζεται σε ένα σύνολο από clusters, είτε σε 7x7 είτε σε 4x4 για να μπορέσουν να γίνουν οι κατάλληλοι υπολογισμοί, εφόσον δεν διαθέτουμε την απαραίτητη υπολογιστική δύναμη για να βρούμε την μετρική EMD ανάμεσα σε pixels. Για κάθε ζεύγος προμηθευτή-καταναλωτή υπολογίσαμε το κόστος μεταφοράς μίας μονάδας εμπορευμάτων, το οποίο είναι η Ευκλείδια απόσταση μεταξύ των κέντρων του κάθε υποκλάστερ. Το πρόβλημα μεταφοράς είναι να βρούμε την λιγότερο ακριβή ροή αγαθών από τους προμηθευτές στους καταναλωτές που ικανοποιεί τη ζήτηση των καταναλωτών.

Στο πρόβλημά μας με τις εικόνες (αντιστοίχιση υπογραφών) αυτό μεταφράζεται στο να καταβληθεί το ελάχιστο ποσό "δουλειάς" για την μετατροπή της μίας υπογραφής στην άλλη. Σύμφωνα με τις διαφάνειες του μαθήματος, αυτό μπορεί να αποτυπωθεί ως το παρακάτω πρόβλημα γραμμικού προγραμματισμού:

Έστω $P = \{(p_1, w_{p1}),\ldots,(p_m, w_{pm})\}$ είναι η πρώτη υπογραφή με m clusters, όπου p_i η αναπαράσταση του cluster και w_{pi} τα βάρη του κάθε cluster. Έστω $Q = \{(q_1, w_{q1}),\ldots,(q_n, w_{qn})\}$ η δεύτερη υπογραφή με n clusters και $D = [d_{ij}]$ η απόσταση μεταξύ των clusters p_i και q_j.
Αυτό που θέλουμε, είναι να βρούμε την ροή εκείνη $F = [f_{ij}]$, όπου f_{ij} η ροή μεταξύ των p_i και q_j, η οποία ελαχιστοποιεί το συνολικό κόστος:

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

Προκειμένου να επιτευχθεί αυτό, θα πρέπει να ισχύουν οι παρακάτω περιορισμοί (σύμφωνα με την Wikipedia):

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

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

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

(4) $\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} = min(\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{n} w_{qj})$

Ο περιορισμός (1) επιτρέπει τη μετακίνηση προμηθειών από P σε Q και όχι αντίστροφα. Ο περιορισμός (2) περιορίζει το ποσό προμηθειών που μπορούν να σταλούν από τις συστάδες του P να μην ξεπερνάει τα βάρη τους. Ο περιορισμός (3) περιορίζει τις συστάδες του Q να μην λαμβάνουν περισσότερες προμήθειες από τα βάρη τους. Ο περιορισμός (4) αναγκάζει να μετακινείται η μέγιστη δυνατή ποσότητα προμηθειών. Αυτό το ποσό ονομάζουμε το total flow. Η EMD προκύπτει εάν κονονικοποιήσουμε την Work με το total flow:

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

Όλα αυτά τα εφαρμόσαμε στον κώδικά μας με την βοήθεια της βιβλιοθήκης Pulp, η οποία μας επέτρεψε να ορίσουμε ένα LpProblem με όλα τα clusters μιας train και test image. Επίσης για κάθε εικόνα υπολογίσαμε τα βάρη της και τα κόστοι μεταφοράς μιας προμήθειας καθώς και όλα τα διαφορετικά routes που μπορουν να υπάρξουν μεταξύ των υποκλάστερ. Αφού τα ορίσαμε όλα αυτά, προσθέσαμε τους παραπάνω περιορισμούς, αποθηκεύσαμε την βέλτιστοποιημένη τιμή που μας γύρισε η objective function σε μία ουρά προτεραιότητας για κάθε σύγκριση μιας query εικόνας με μία train εικόνα, έτσι ώτσε να είμαστε σε θέση να τραβήξουμε τους 10 κοντινότερους γείτονες από κάθε query.

Παράλληλα, κάθε φορά αποθηκεύαμε τις αποστάσεις και τους γείτονες που βρίσκαμε με την Manhattan Distance, όπως ζητήθηκε απ' την εκφώνηση, προκειμένου να προβούμε σε σύγκριση των ποσοστών επιτυχίας. Ορίστε μερικά από τα αποτελέσματα για κάποιους από τους συνδυασμούς που τρέξαμε:

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%

Όπως παρατηρούμε, όσο αυξάνουμε τα δεδομένα μας αλλά και τον αριθμό των queries, τόσο μεγαλύτερη ακρίβεια δίνει ο EMD αλλά και η Manhattan distance, πράγμα αναμενόμενο.

Μία σημαντική παρατήρηση είναι ότι όσο μειώνεται το μέγεθος των συστάδων, τόση μεγαλύτερη ακρίβεια μας δίνει το EMD. Αυτό συμβαίνει, διότι όσο "σπάμε" την εικόνα σε μικρότερα "παράθυρα", τόσο μεγαλύτερη λεπτομέρια υπάρχει και γι' αυτό υπάρχει καλύτερο performance. Βέβαια κάτι τέτοιο, γίνεται αντιληπτό, ότι στοίχιζε πολύ σε χρόνο. Για παράδειγμα, τα 5 queries με 1000 trains με μέγεθος 4x4 έναντι το ίδιων δεδομένων με μέγεθος συστάδων 7x7 ήταν κατά πολύ πιο χρονοβόρο. Κάτι τέτοιο καταλαβαίνουμε πως μας απέτρεψε από τον πειραματισμό με μικρότερο μέγεθος συστάδων, γιατί δεν διαθέταμε την κατάλληλη υπολογιστική δύναμη.

Μία ακόμα, ζωτικής σημασίας, παρατήρηση που βλέπουμε από τα παραπάνω πινακάκια, είναι ότι ακόμα κι αν μειώσουμε το μέγεθος των συστάδων και αυξήσουμε τα δεδομένα μας, η ακρίβεια του EMD πλησιάζει αλλά δεν φτάνει (και προπάντων δεν ξεπερνάει) ποτέ την ακρίβεια που μας δίνει η Manhattan Distance. Η Manhattan Distance είναι μία απλή μετρική η οποία στην ουσία κάνει Brute Force στα δεδομένα μας και γι αυτό φαίνεται να έχει περισσότερη ακρίβεια. Όμως, επειδή τα πινακάκια δείχνουν ότι όσο αυξάνονται τα δεδομένα μας και όσο μειώνεται το μέγεθος των συστάδων, τόσο μειώνεται η ψαλίδα απόκλισης της ακρίβειας EMD-Manhattan, θα μπορούσαμε να εξάγουμε το συμπέρασμα ότι εάν πχ η σύγκριση του EMD γινόταν με πολύ μικρό μέγεθος παραθύρων (πχ pixel-pixel) και με περισσότερα δεδομένα, η μετρική EMD, διαισθητικά θα ξεπέρναγε την ακρίβεια της Manhattan Distance. Κάτι τέτοιο όμως, δυστυχώς, δεν μπορούμε να το εξακριβώσουμε λόγω έλλειψης κατάλληλης υπολογιστικής δύναμης.

Σημείωση: Η σύγκριση μεταξύ αυτού του ερωτήματος με το ερώτημα Β δεν θα είχε πολύ νόημα, καθώς το ερώτημα Γ είναι υλοποιημένο σε γλώσσα python ενώ το Β ερώτημα σε C++. Εκτός αυτού όμως δεν γνωρίζουμε τους ακριβείς αλγορίθμους και την πολυπλοκότητα αυτών που χρησιμοποιεί η Pulp βιβλιοθήκη για να υπολογίσει την EMD μετρική, οπότε θα ήταν ανούσιο να προβούμε σε τέτοιου είδους σύγκριση, χωρίς να ξέρουμε όλα τα δεδομένα μας.

-Ερώτημα Δ

Όσον αφορά αυτό το ερώτημα, ακολουθήσαμε και εδώ ρητά τις απαιτήσεις της εκφώνησης και έτσι εκτελέσαμε τρεις διαφορετικές συσταδοποιήσεις, στις οποίες δεν παρατήρηθε μεγάλη διαφορά μεταξύ των Values of Objective Function.

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

Αρχικά η πρώτη συσταδοποίηση που έγινε στον νέο διανυσματικό χώρο συνέκλινε με μόλις 6 ανανεώσεις κεντροειδών, ενώ ως συνθήκη τερματισμού είχαμε θέσει πως ο ρυθμός μεταβολής της συνάρτησης objective function να είναι μικρότερος από μια σταθερά ε=0.001. Επίσης η προαναφερθείσα συσταδοποίηση είχε τελική σιλουέτα 0.16 στο σύνολο 10.000 εικόνων.

Από την άλλη πλευρά η δεύτερη συσταδοποίηση, που έγινε στον αρχικό διανυσματικό χώρο (των 784 διαστάσεων) με όλους τους άλλους περιορισμούς να παραμένουν ίδιοι, συνέκλινε σε μόλις 7 ανανεώσεις κεντροειδών (βάσει της k-median τεχνικής) και είχε τελική σιλουέτα 0.10.

Τέλος, η συσταδοποίηση η οποία έγινε στηριζόμενη στα labels του παραδοτέου της δεύτερης εργασίας μας είχε παρόμοια αποτελέσματα με αυτά της συσταδοποίησης στον νέο διανυσματικό χώρο (με τελική σιλουέτα 0.16). Στη συγκεκριμένη συσταδοποίηση μπορούμε να δικαιολογήσουμε τη βελτιωμένη σιλουέτα συγκριτικά με τον αρχικό διανυσματικό χώρο, αφού τα labels στα οποία ανατέθηκαν οι εικόνες ήταν αποτέλεσμα της κατηγοριοποιήσης, την οποία είχαμε υλοποιήσει στην δεύτερη εργασία, ενός νευρωνικού δικτύου, το οποίο εκπαιδεύτηκε σε ακριβώς αυτά τα δεδομένα. Με την τακτική αυτή συμπεραίνουμε πως αφού το νευρωνικό δίκτυο μας

'γνώριζε' τα δεδομένα μας πολύ καλά, θα είχε και εξαιρετικά ακριβείς προβλέψεις, που μετέπειτα θα οδηγούσε σε καλή συσταδοποίηση των εικόνων άρα και ικανοποιητική σιλουέτα.

Σημείωση: Όσον αφορά τα ερωτήματα Β και Δ, υπήρξε ιδιαίτερη μεταχείριση στο διάβασμα του αρχείου που παρήγαγε το ερώτημα Α, καθώς πλέον η κάθε φωτογραφία αναπαραρίσταντο από 10 αριθμούς μεγέθους 2 bytes. Για αυτό τον λόγο δημιουργήθηκε ξεχωριστή συνάρτηση read_binary_file η οποία διάβαζε σταδιακά με το κατάλληλο endianess 2 bytes σειριακά. Τέλος, όλα τα αποτελέσματα εκτύπωνονται προσωρινά στο εξωτερικό αρχείο output.