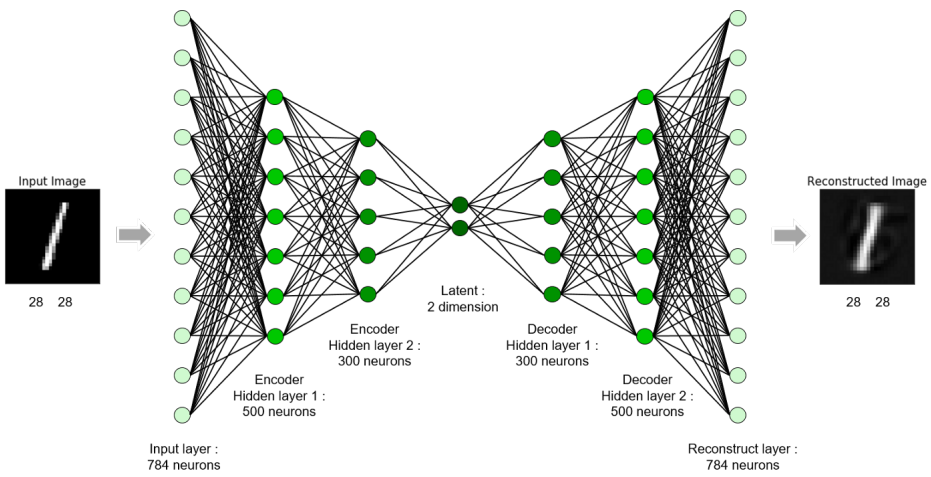
Δίκτυα Αυτοκωδικοποίησης MNIST

Autoencoders MNIST



Αριστοστέλειο Πανεπιστήμιο Θεσσαλονίκης

10ο Εξάμηνο ECE AUTh

7ο Εξάμηνο CSD AUTh

Νευρωνικά Δίκτυα - Βαθιά Μάθηση

Χάρης Φίλης ΑΕΜ: 9449

GitHub Repository Link:

Autoencoders-Url/Urn:

Janouary,2023

Contents

List of Figures

1 Εισαγωγή - **Introduction**

Στην παρούσα εργασία έγιναν υλοποιήσεις δικτύων Αυτο-κωδικοποίησης δεδομένων *Autoencoder Neual Networks* αλλά και άλλων μοντέλων μείωσης διάστασης δεδομένων όπως η ανάλυση κύριων συνιστωσών *Principal Component Analysis*, με την απώτερο σκοπό την επιλογή καλύτερου μοντέλου ανακατασκευής δεδομένων από τις συμπιεσμένες αναπαραστάσεις που δημιουργούν.

Σήμερα η ανάγκη συμπίεσης δεδομένων είναι πιο επιτακτική από ποτέ. Σχεδόν όλο το ποσοστό το πληροφοριών σήμερα έχει ψηφιοποιήθει και ορισμένες μορφές όπως η εικόνα και το βίντεο κατ' επέκταση είναι άπληστα στην κατανάλωση χώρου μιας και ατόφια περιέχουν πολλή πληροφορία μιας και η αποτύπωση του φυσικού κόσμου σε μια εικόνα είναι δαπανηρή ψηφιακά διαδικασία.

Τα μοντέλα αυτοκωδικοποίησης και μείωσης διάστασης προσπαθούν μέσω ενός νευρωνικού συστήματος να μειώσουν αυτόν τον χώρο των ατόφιων δεδομένων εφαρμόζοντας με μορφή υπολογιστικού γράφου περίπου παρόμοια συμπίεση με αυτη που εφαρμόζεται και σε ένα αρχείο ήχου για παράδειγμα mp3 ή σε μια εικόνα JPEG (βέβαια αυτά είναι συγκεκριμένα πρότυπα κωδικοποίησης και δεν βασίζονται σε νευρωνικά δίκτυα αλλά σε μετασχηματισμούς πχ. DCT (*discrete cosine transform*). Η δυνατότητα να έχουμε μια πιο συμπιεσμένη μορφή του μέσου αποθηκευμένη και να μπορούμε μέσω μια συνάρτησης που εκφράζεται στην συγκεκριμένη περίπτωση μη ντετερμινιστικά από ένα νευρωνικό δίκτυο, δεν είναι ο κύριος λόγος που μελετάμε τους Auto-Encoders. Μορφές αυτών των μοντέλων μπορούν να μοντελοποιήσουν θόρυβο αλλά και ατέλειες στις εικόνες και να τα αφαιρέσουν. Παράλληλα μπορούν να λειτουργήσουν ως γεννήτριες νέων εικόνων χρησιμοποιώντας μόνο το κομμάτι του εκπαιδευμένου αποκωδικοποιητή.

Όπως θα δούμε και στην συνέχεια η εκπαίδευση που εφαρμόζεται στα μοντέλα που μελετούμε είναι "αυτο-επιβλεπόμενη" μιας και δεν εμφανίζεται πουθενά η ανάγκη σύγκρισης εκτιμήσεων του δικτύου με κάποιο *annotation* δηλαδή κάποια ετικέτα που εκφράζει το δεδομένο κατηγορικά. Αντιθέτως η έξοδος είναι η εκτιμώμενη εικόνα και συγκρίνεται με την είσοδο στην κλασσική περίπτωση ενός Αυτοκωδικοποιητή.

2 Επιλογή και προ επεξεργασία δεδομένων - **Dataset**

Στο συγκεκριμένο σκέλος της εργασίας μια και οι πολύχρωμες εικόνες του CIFAR10 εισάγουν επιπλέον πολυπλοκότητα στο δίκτυο, μετατρέποντας την εκπαίδευση σε μεγαλύτερο γολγοθά αποφάσισα να επιλέξω το κλασσικό MNIST dataset για να εκπαιδεύσω το δίκτυο

2.0.1 Λίγα λόγια για το **MNIST**

Το MNIST dataset απότελεί ένα σύνολο δεδομένων εικόνων με χειρόγραφα ψηφία. Οι εικόνες αυτές είναι ασπρόμαυρες και έχουν διάστασης 28x28 pixels. Το dataset προσφέρεται από την pytorch annotated δηλαδή έχοντας ετικέτες. Στο αρχείο *utils.py* που βρίσκεται στο *parent dir* της εργασίας, εφαρμόζονται μέθοδοι λήψης και προ επεξεργασίας των δεδομένων, αποθήκευσης μοντέλου μετά την εκπαίδευση

καθώς ορίζονται και καίριοι βοηθητικοί αλγόριθμοι και δομές για την εκπαίδευση τα οποία χρησιμοποιήθηκαν και στην πρώτη εργασία.Τα δεδομένα φορτώνονται σε *iterators* και συγκεκριμένα στους DataLoaders που προσφέρει η pytorch ώστε να μπορούν εύκολα να μεταφερθούν σε gpu για επιτάχυνση του training.

2.0.2 Προ επεξεργασία **MNIST**

Οι προαναφερθείς *Dataloaders* χωρίζουν την βάση δεδομένων σε δεδομένα εκπαίδευσης, και δεδομένα αξιολόγησης κατά αναλογία 60% και 40% αντίστοιχα. Όπως αναφέρεται στο στο 4ο μέρος του [6], η προσέγγιση προβλημάτων βελτιστοποίησης με την αφαίρεση πληροφορίας στην είσοδο (εδώ πρακτικά μετατροπή της ασπρόμαυρης εικόνας σε δυαδική) είναι συχνή τεχνική. Σκοπός αυτής της τεχνικής είναι να επικεντρώσουμε την προσοχή του δικτύου στις ακμές της εικόνας του MNIST. Έτσι, τιμές με φωτεινότητα κάτω από 0.5 πρακτικά αφαιρούνται μέσω ενός *binarization masking transformer* και την χρήση της αντίστοιχης κλάσης που προσφέρει η pytorch. Αυτή η τεχνική είναι ένα *global thresholding* που εφαρμόζεται στην είσοδο και μετατρέπει τις εικόνες σε εικόνες με πιο σαφή άκρα μιας και η φωτεινότητα πέφτει στα pixel που βρίσκονται στο έξω πλαίσιο κάθε γραμμής. Σε άλλη συνθήκη εφαρμόζεται *gaussian* θόρυβος στα δεδομένα μέσω αντίστοιχου transformer και με επιλογή που δίνεται στο χρήστη αυτό έχει ως απότέλεσμα ορισμένα μοντέλα που πρακτικά δέχονται ως είσοδο συστατικά κατανομών με βάση το *variational inference formulation* [5]το οποίο θα αναλυθεί αργότερα δίνει μια δυνατότητα στο μοντέλο να μοντελοποιήσει αυτόν τον θόρυβο και να τον αφαιρέσει ώστε να φτάσει στο αρχικό dataset. Τέλος, υπάρχει και η επιλογή που το dataset απλώς κανονικοποιείται μεταξυ 0,1 και μετατρέπεται σε τανυστή.

3 Παρόμοια Μοντέλα με τον Αυτοκωδικοποιητή

3.1 Μείωση Διάστασης με **PCA**

Τα δίκτυα αυτοκωδικοποίησης είναι πολύ στενά συνδεδεμένα με την ανάλυση κύριων συνιστωσών (*Principal Component Analysis*). Στην πραγματικότητα αν η συνάρτηση ενεργοποίησης μεταξύ κάθε layer είναι γραμμική το μοντέλο αυτό-κωδικοποίησης και συγκεκριμένα το κομμάτι κωδικοποίησης *Encoder* και η έξοδός του συγκεκριμένα (ουσιαστικά το μικρότερο σε αριθμό νευρώνων layer) αντιστοιχεί απευθείας στις κύριες συνιστώσες του PCA.

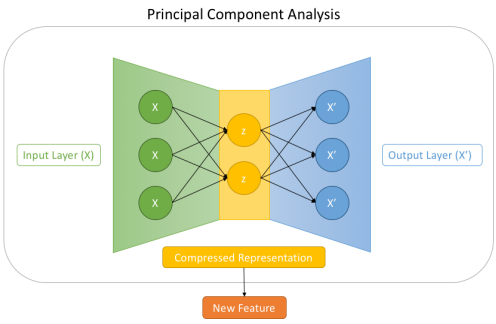


Figure 1: PCA reconstruction model

Έτσι κύριο μέτρο σύγκρισης των δικτύων αποκωδικοποίησης, είναι η ανακατασκευή μέσω PCA. Για αυτό λοιπόν υλοποιήθηκε μοντέλο μέσω της βιβλιοθήκης *sklearn.decomposition.PCA*, το οποίο μειώνει τις διαστάσεις των δεδομένων της MNIST και ταυτόχρονα τρέχει τον αντίστροφο μετασχηματισμό ώστε να επανέλθει στην αρχική διάσταση δεδομένων. Η PCA λειτουργεί με ντετερμινιστικό τρόπο υπολογίζοντας συμμεταβλητότες μεταξύ δεδομένων έτσι είναι σχετικά απλό και το μοντέλο του αντίστροφου μετασχηματισμού της. Ωστόσο, αυτό αναφέρεται γιατί όπως είπαμε θα μπορούσαμε να φτιάξουμε μια PCA φτιάχνοντας έναν Autoencoder με συναρτήσεις ενεργοποιήσεις μεταξύ των layer οι οποίες ειναι γραμμικές.

Ως μετρική αξιολόγησης ορίστηκε το μέσο τετραγωνικό σφάλμα μεταξύ της εισόδου και της εξόδου μιας και είναι αυτό που ορίζει την απόσταση των δύο εικόνων πρακτικά. Το πλήθος των pca components που χρησιμοποιήθηκε ήταν [0.8,0.9,0.95] όπως φαίνεται παρακάτω. Το αρχείο models/PCA_denoise.py περιέχει το pipeline αυτού του μοντέλου το οποίο δέχεται και ως command line argument τον πίνακα των συνιστωσών.

```
[22:31:02] ❯ python .\models\PCA_denoise.py
torch.Size([60000, 28, 28])
torch.Size([60000, 28, 28])
Time taken for PCA with 0.8 components: 5.827985525131226
For 44 components
The RMSE of the train set is: 23.877350758777983
The RMSE of the test set is: 23.877350758777983
Time passed: 12.513728857040405 seconds.
Time taken for PCA with 0.9 components: 5.871204376220703
For 87 components
The RMSE of the train set is: 17.48213757700727
The RMSE of the test set is: 17.48213757700727
Time passed: 12.516777276992798 seconds.
Time taken for PCA with 0.95 components: 6.119304656982422
For 154 components
The RMSE of the train set is: 12.62115827352506
The RMSE of the test set is: 12.62115827352506
Time passed: 12.616967678070068 seconds.
Model saved to ./saved_models/PCA
```

Figure 2: PCA training/inference backlog

(a) Data/Reconstructions

(b) Data/Reconstructions 44 components

(c) Reconstructions 87 components

(d) Reconstructions 154 components

Figure 3: Ανακατασκευές εικόνων για διαφορετικό αριθμό pca_components

4 Εισαγωγή στους Αυτοκωδικοποιητές - **Autoencoders Preliminaries**

Οι Autoencoder, όπως ήδη αναφέραμε έχουν διάφορες χρήσεις κάποιες από αυτές είναι οι εξής:

- Image Generation
- Image Augmentation
- Image Blending

παράλληλα μπορούν να λειτουργούν και σε άλλους τύπους δεδομένων πέραν της εικόνας απλά η χρήση της εικόνας ως είσοδο είναι αρκετά διαδεδομένη.

Τα δίκτυα αυτοκωδικοποίησης, παρουσιάζουν πολύ απλή αρχιτεκτονική ωστόσο καλό θα ήταν να μελετήσουμε αρχικά ένα κλασσικό δίκτυο Autoencoder προτού πάμε στα πιο σύνθετα μιας και τα σύνθετα έχουν πολλές έννοιες θεωρίας στατιστικής καθώς και κάνουν επιπλέον πράγματα από τον κλασσικό autoencoder.

4.1 Είδη Δικτύων Αυτοκωδικοποίησης

4.1.1 Traditional Autoencoder

Αυτού του είδους ο αυτοκωδικοποιητής έχει ήδη περιγραφεί εκτενώς.Πρόκειται για είτε ένα Dense είτε συνελικτικό δίκτυο, το οποίο από layer σε layer μειώνει την διάσταση των δεδομένων μειώνοντας τον αριθμό των νευρώνων. Στόχος είναι από το τελευταίο layer το οποίο λέγεται και latent να μπορεί να αναπαραχθεί η είσοδος. Έτσι λέμε οτι το δίκτυο είναι ένα δίκτυο που μιμείται την είσοδο ατόφια. Η ανάγκη ανακατασκευής κωδικών πλέον - εξαιρετικά χαμηλής μη διακριτής διάστασης εικόνων ή γενικότερα δεδομένων είναι πλέον επιτακτική στην εποχή μας όπως αναφέρθηκε στην αρχή. Εκεί έγκειται και η αναγκαιότητα αυτών των δικτύων που μοντελοποιούν την μορφή των δεδομένων στο σώμα του δηλαδή στην έκτασή τους.

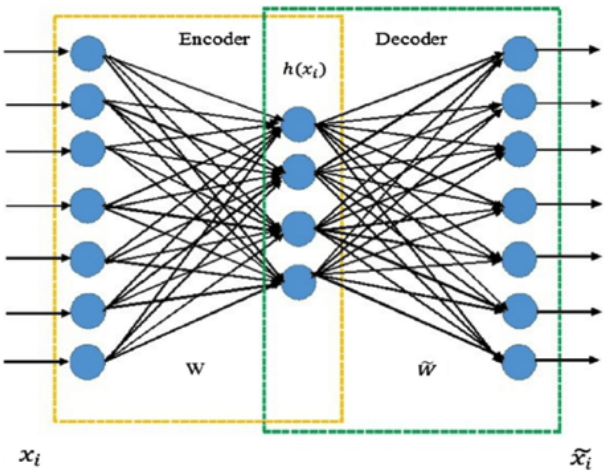


Figure 4: Απλός Autoencoder

Όπως αναφέρθηκε το κόστος ανακατασκευής, ή αλλιώς όπως το αναφέρουμε στα νευρωνικά δίκτυα η αντικειμενική συνάρτηση βελτιστοποίησης δίνεται από τον παρακάτω τύπο στον κλασσικό Autoencoder δεδομένου σιγμοειδούς συνάρτησης στο layer εξόδου:

Έστω
φ: X $\xrightarrow{encoder} F$
$\psi : F \xrightarrow{decoder} X$
$aim\,\varphi,\ aim\psi : \arg\min_{(\varphi_i,\psi_i)} ||x - \psi \circ \varphi(x)||$
$L(x, \hat{x}) = ||x - \hat{x}||^2 = ||x - \dot{\sigma}(\dot{W}(\sigma(W * x + b) + \dot{b})||^2$

Στόχος ενός αυτοκωδικοποιητή είναι να επιλέξει τέτοια βάρη W και biases b για τον κωδικοποιητή και τον απόκωδικοποιητή, έτσι ώστε να απαιτείται η ελάχιστη το δυνατόν πληροφορία, για να μπορέσει να ανακατασκευάσει την κωδικοποιημένη εικόνα. Ουσιαστικά στόχος είναι το μηδαμινό latent layer size με όσο δυνατό καλύτερη ανακατασκευή δηλαδή όπως φαίνεται παραπάνω τις παραμέτρους των φ και ψ.

4.2 Άλλα είδη Αυτοκωδικοποιητών

4.2.1 Denoising Autoencoder

Αυτός είναι ο πιο διαδεδομένος τύπος δικτύου αυτοκωδικοποίησης. Χρησιμοποιεί λευκό θόρυβο στα δεδομένα πριν την εκπαίδευση και υπολογίζει το σφάλμα μεταξύ της πραγματικής εικόνας και της ανακατασκευής που παράγεται με τα σηματοθορυβικά δεδομένα. Έτσι καταφέρνει μαζί με την εικόνα να μοντλεοποιήσει και τον λευκό θόρυβο που συχνά υπάρχει στις εικόνες προερχόμενο από motion-blurring για παράδειγμα. Παράλληλα, η σύγκριση που πραγματοποιείται τον αναγκάζει να μην κάνει overfit στα δεδομένα όπως συμβαίνει συχνά με τον κλασσικό Autoencoder και έτσι έχουμε ένα εργαλείο που ξέρει να διορθώνει μη σταθερές λήψεις ή λήψεις κακών συνθηκών φωτισμού που δημιουργού ψηφιακό θόρυβο.

4.2.2 Sparse Autoencoder

[1] Αυτοί οι Autoencoders σε αντίθεση με του προηγούμενους έχουν μεγαλύτερη διάσταση latent layer από οτι η είσοδος. Παρόλα αυτά, κάθε φορά ένα μικρός μέρος αυτών πυροδοτεί/τρέχει έτσι ονομάζεται και αχανής. Η δουλειά του είναι παρόμοια με ενός denoising autoencoder με την έννοια ότι, η τμηματική του λειτουργία ένα είδος κανονικοποίησης για μην κάνει overfit.

4.2.3 Contratctive Autoencoder

Μοιάζει με τα δύο τελευταία δίκτυα χωρίς όμως να μειώνει ή να αυξάνει την διάσταση στο latent size.

4.3 Πρόβλημα των μοντέλων των Αυτοκωδικοποιητών

4.3.1 Gaps in the latent size

Τα κενά στο τελευταίο layer κωδικοποίησης ισοδυναμούν με κλειστούς νευρώνες και ουσιαστικά έλλειψη δεδομένων σε μια αυτο-επιβλεπόμενη μάθηση όπως είπαμε. Αυτό δείχνει οτι το δίκτυο δεν έχει μάθει όλες τις πιθανές εισόδους και υπάρχουν δεδομένα που δεν ξέρει να τα ανακατασκευάζει σωστά.

4.3.2 Seperability

Η διαχωρισιμότητα αντιστοιχεί στο πρόβλημα ότι τα δείγματα που ανακατασκευάζονται δεν είναι γραμμικά διαχωρίσιμα μεταξύ τους. Πρακτικά τα δείγματα ανακατασκευής μοιάζουν αρκετά μεταξύ τους και πιθανόν να φέρουν θόρυβο σε σχέση με τα πραγματικά δεδομένα.Αυτός συσχετίζεται με τα κλασσικά MLP ως χαμηλό accuracy και στην συγκεκριμένη περίπτωση effectiveness του Autoencoder.

4.3.3 Λοιπά προβλήματα

- Διακριτό latent space
- Αδυναμία ανακατασκευής που εμφανίζεται ως θόρυβος ή ως μη δυνατότητα αναγνώρισης της εξόδου ως δεδομένο του dataset

5 Κλήση με στοιχεία διακύμανσης - Variational Inference Formulation

[4] Μέχρι στιγμής είδαμε μοντέλα τα οποία λειτουργούν με μη πιθανοτικό τρόπο. Ωστόσο τα πραγματικά δεδομένα εμπερίχουν θόρυβο ο οποίος μπορεί να οριστεί και ως τυχαιότητα. Ουσιαστικά μπορεί σε μια εικόνα να έχουμε τον ίδιο αριθμό αλλά η εικόνα να έχει τραβηχτεί με διαφορετικές συνθήκες με αποτέλεσμα να μην φαίνονται τα όρια, ή ακμές τις εικόνας αν θέλετε καλά. Σε πολλές συνθήκες λοιπόν χρειάζονται πιθανοκρατικά μοντέλα που να παράγουν δεδομένα. Τα πιο διαδεδομένα μοντέλα που εμφανίστηκαν ως πιθανοκρατικά είναι οι Variational Autoencoders[2](VAE) και τα GANs (Generative Adversarial Networks). Αυτά τα μοντέλα μαθαίνουν την κατανομή των δεδομένων p(x) και επιτρέπουν την δειγματοληψία από αυτήν την κατανομή ως είσοδο. Για αυτό τον σκοπό το σφάλμα ή η συνάρτηση βελτιστοποίησης του μοντέλου εμπεριέχει και το ποια κατανομή ταιριάζει καλύτερα στα δεδομένα εκπαίδευσης.Σε αυτό ακριβώς αναφέρεται η μέθοδος Variational Inference Formulation.

6 Απόκωδικοποιητές Διακύμανσης - Variational Autoencoders

Ο απόκωδικοποιητής διακύμανσης που υλοποιείται και στην εργασία, πρώτο διατυπώθηκε από τους (Kingma Welling, 2013) και είναι ένα προσανατολισμένο μοντέλο παραγωγής δεδομένων από latent variables πλέον και οχι δεδομένα. Συγκεκριμένα το layer εξόδου του κωδικοποιητή μέσω ενός *reparametrization trick* το οποίο υλοποιείται στο αρχείο vi_utils.py μπορεί από layers που εκφράζουν στοιχεία μια gaussian κατανομής συγκεκριμένα στην περίπτωσή μας την μέση τιμή (mu) και την λογαριθμική τιμη της διακύμανσης της κατανομής (logvar) να παράξουν από δείγματα gaussian δείγματα το z το οποίο αντιστοιχεί σε δεδομένο του dataset αλλά από τυχαία πλέον κατανομή η οποία έχει επιλεγεί από το μοντέλο. Για να συμβεί αυτό βέβαια χρειάζεται κατάλληλη και τροποποιημένη συνάρτηση.

6.1 Sum of Kullback-Leibler Divergence and Reconstruction Loss

Σύμφωνα με το paper [2], το variational inference ειναι μια τεχνική προσέγγισης σύνθετων κατανομών μέσω παραμετρικών κατανομών μεγιστοποιώντας το marginal likelihood και δηλαδή με βάση την παράγραφο 2.2 περί variational bound ελαχιστοποιώντας το όρο που ονομάζεται KL divergence. Σε αυτό το σφάλμα προστίθεται το reconstruction loss.

$$D_{KL}(P||Q) = -\sum_{x \in X} P(x) * \log \frac{P(x)}{Q(x)} \tag{1}$$

7 Υλοποίηση

7.1 Autoencoder

Ο κλασσικός Autoencoder αποτελείται απο απλά BasicLayers τα οποία έχω φτιάξει μέσω της pytorch και υλοποιούν το forward pass δηλαδή και την συμπίεση και την αποκωδικοποίησης της εικόνας. Η κωδικοποίηση γίνεται με δύο layer τα οποία πρώτα κατεβάζουν την διάσταση στην μια κρυφή και μετά στο latent_space. Η συνάρτηση ενεργοποίηση που χρησιμοποιείται για τα layer είναι η ReLU για να υπάρχει μη γραμμικότητα ενώ υλοποιούνται οι συναρτήσεις encode και generate και ξεχωριστά. Η backward συνάρτηση υλοποεία backward pass μόνο στο decoder εκπαιδεύοντας αυτό το κομμάτι(αν και δεν χρησιμοποιείται μονομερής εκπαίδευση του decoder). Τέλος, η βελτιστοποίησης ειναι το reconstruction loss το οποίο αναφέρθηκε ήδη και υλοποιείται μέσω της torch.mean(torch.square()) δηλαδή υπολογίζεται και εδώ το RMSE μεταξύ της παραγόμενης και της εισαγόμενης εικόνας.

7.2 Variational Autoencoder

Ο συγκεκριμένος Autoencoder υλοποιεί όλες τις συναρτήσεις με παρόμοια layers με τον απλό encoder ωστόσο το loss function εδώ υπολογίζεται με βάση το Variational Inference Formulation [3].

7.3 Convolutional Variational Autoencoder

Ο συγκεκριμένος Autoencoder διαθέτει συνελικτικά δίκτυα και στο κομμάτι τις κωδικοποίησης καθώς και στο κομμάτι της αποκωδικοποίησης. Παράλληλα χρησιμοποιούνται batch normalization layers για να εξομαλύνουν τον μετασχηματισμό των δεδομένων στα feature maps αλλά average pooling layer στην έξοδο του κωδικοποιητή για να γίνει ομοιόμορφα μείωση της διάστασης και ταυτόχρονα με μείωση υπολογιστικού κόστους τελικής συνέλιξης στο latent_space. Να υπενθυμίσω πως επειδή η είσοδος έρχεται flatten πρέπει να γίνει unflatten στο αποκωδικοποιητή. Έτσι, έχουμε γραμμικά μέρη να παρεμβάλλονται μεταξύ συνελικτικών μερών του δικτύου. Κατά τα άλλα μοιάζει με τον VAE.

8 Αποτελέσματα Εκπαίδευσης και Αξιολόγησης Μοντέλων

Για την εκπαίδευση και την αξιολόγηση των μοντέλων έχουν γραφεί κατάλληλα script που δέχονται command line arguments με βάση την δομή της εκπαίδευσης που ακολουθήθηκε και στο MLP. Πρακτικά χρησιμοποιώ όλες τις καλές μεθόδους mini-batch training, adjustable learning rate, optimizer select αλλά και επιλογή εποχών καθώς και αποθήκευση των μοντέλων. Παρόλα αυτά, δεν μπόρεσαν να εκπαιδευτούν πάνω όλα τα μοντέλα πάνω από μια φορά μιας και η διαδικασία εκπαίδευσής τους ήταν χρονοβόρα. Συγκεκριμένα τα δίκτυα με τα πυκνά layers εκπαιδεύονται στην μισή ώρα περίπου όπως θα φανεί και από τις εικόνες του τερματικού παρακάτω, ενώ τα συνελικτικά μοντέλα θέλουν τουλάχιστον μια ώρα για 100 εποχές και δεδομένο batch_size=32 (μικρό ώστε να γίνουν πολλές αναβαθμίσεις του μοντέλου και να αποφευχθούν gaps). Παρακάτω παρατίθενται όσα αποτελέσματα συλλέχθηκαν για τα δίκτυα αυτοκωδικοποίησης.

8.1 AutoEncoder

8.1.1 Learning Curves

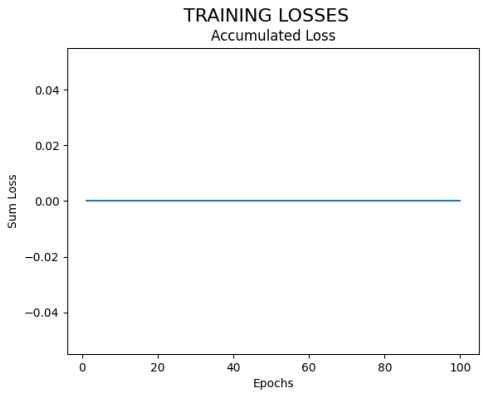


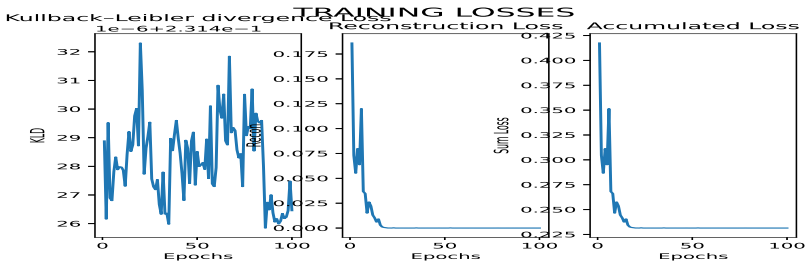
Figure 5: Autoencoder Overfit Learning Curve

8.2 PCA_denoise

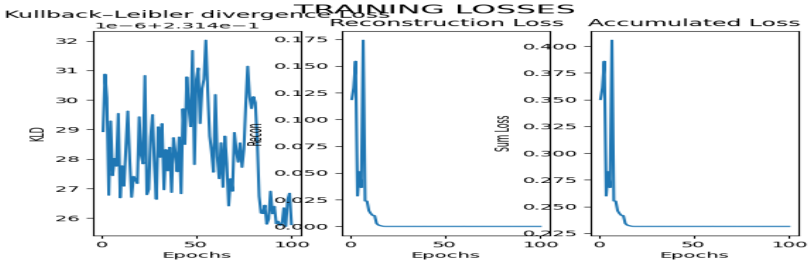
Τα αποτελέσματα του PCA_denoise βρίσκονται στην αρχή.

8.3 VAE

8.3.1 Latent Space Train Curve Comparison

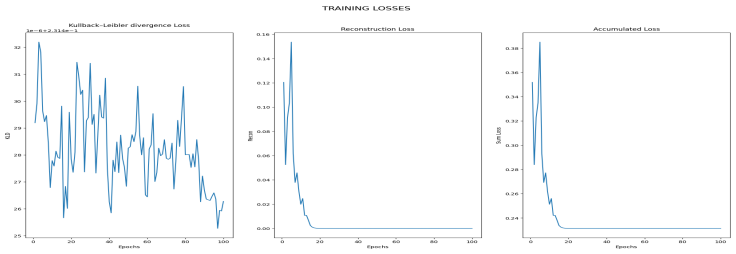


(a) Latent Space 85

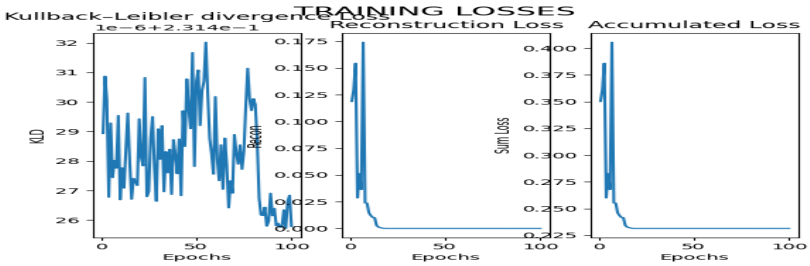


(b) Latent Space 20

8.3.2 Input Maksing Train Curve Comparison



(a) Bernoulli



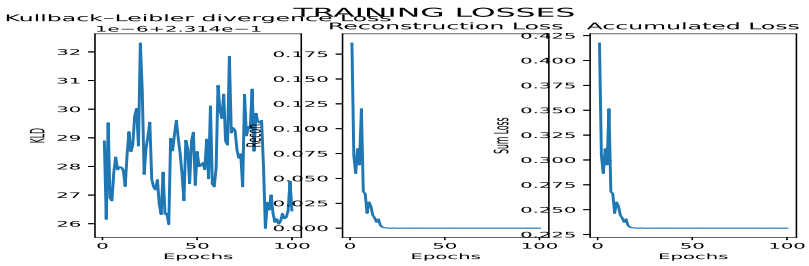
(b) Gaussian

8.3.3 Σχολιασμος

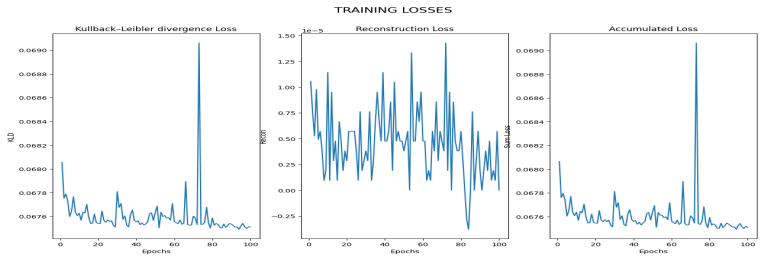
Παρατηρείται οτι το μεγαλύτερο μέρος του σφάλματος εισάγεται από το KLD loss (Kullback-Leibler Divergence) αυτό δείχνει οτι το ο σηματοθορυβικός δείκτης δεν είναι αυτός που επηρεάζει την σύγκλιση περισσότερο του μοντέλου μιας και το συσσωρευμένο σφάλμα μοιάζει να έχει την πορεία του reconstruction loss που παραμένει πολύ μεγαλύτερο από ότι η διασπορά στις κατανομές.Τέλος φαίνεται πως σε bernoulli είσοδο η σύγκλιση του μοντέλου είναι πιο αργή από οτι σε gaussian το οποίο δικαιολογείται μιας και η προσέγγιση vi γίνεται με gaussian κατανομές.

8.4 VAE_cnn

8.4.1 Latent Space Train Curve Comparison

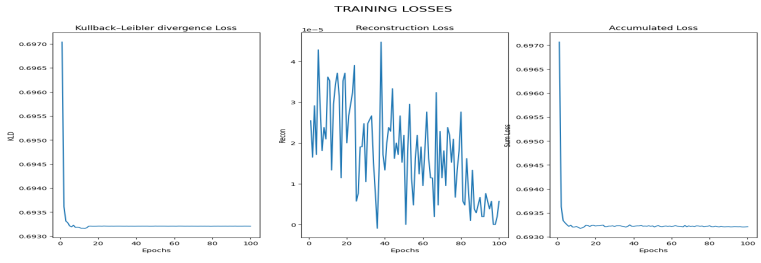


(a) Latent Space 85

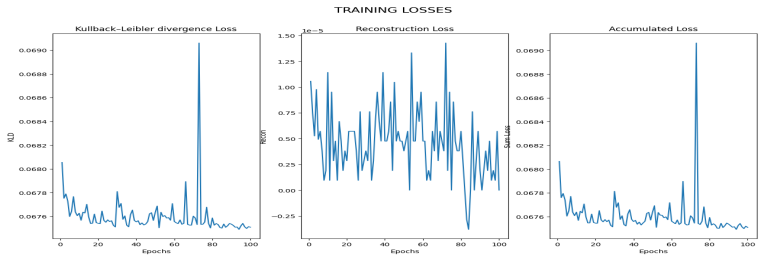


(b) Latent Space 20

8.4.2 Input Maksing Train Curve Comparison



(a) Bernoulli



(b) Gaussian

8.4.3 Σχολιασμός

Στο κομμάτι με την αλλαγή του latent space παρατηρούμε οτι το μοντέλο αρχίζει να δυσκολεύεται να συγκλινει με μικρότερο latent space το οποίο είναι λογικό μιας και μικρό latent space ισοδυναμεί με λιγότερο αποδοτικό μοντέλο ή τουλάχιστον με μοντέλο που χρειάζεται περισσότερο training.Στο κομμάτι του διαφορετικού masking απλά να σημειώσω οτι εδώ το reconstruction loss είναι αυτό που παίζει περισσότερο δηλαδή δουλεύει καλύτερα το KLD loss με το συνελικτικό μοντέλο.

8.5 Timings

Παρατίθεται teminal Backlog για διάφορα μοντέλα καθώς και παραδείγματα training.

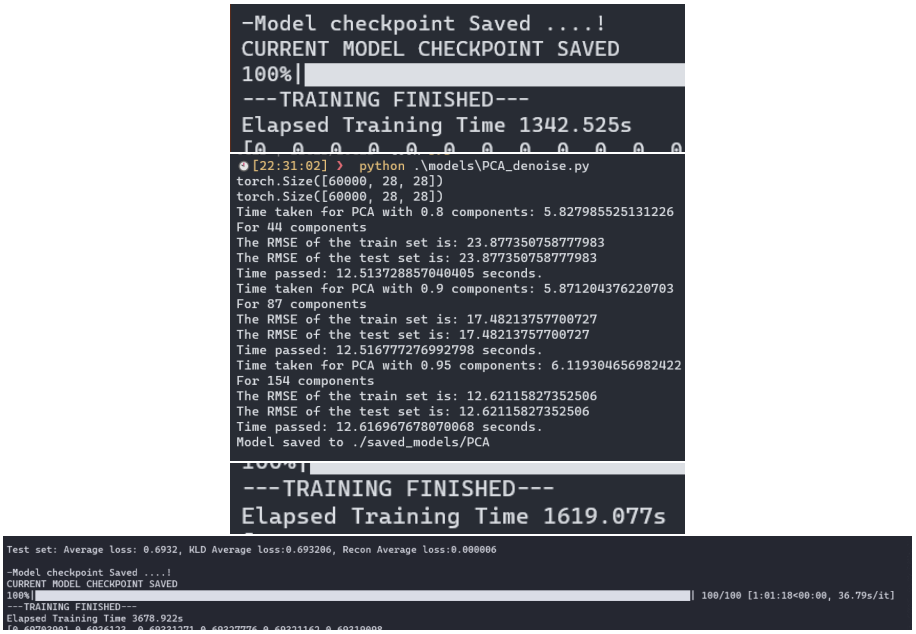


Figure 10: Terminal Backlogs

9 Παραγόμενες Εικόνες **AE**

Σχετικά με τι παραγόμενες εικόνες τα πιθανοκρατικά μοντέλα δεν μου έβγαλαν κατάλληλες εξόδους γιατί μάλλον εφάρμοσα ανακατασκευή όχι σε interpolated gaussian samples αλλά στα ίδια τα δεδομένα και αυτό είναι λάθος. Όπως και να έχει ο autoencoder και το pca δούλεψαν καλά και αυτό που έγινε φανερό είναι πως η PCA δεν βελτιώνεται πολύ ούτε μπορεί να φτάσει σε πολύ μικρό αριθμό components ώστε να κάνεις ανάποδο μετασχηματισμό σε αντίθεση με τα δίκτυα αυτοκωδκοποίησης.

9.0.1 Autoencoder

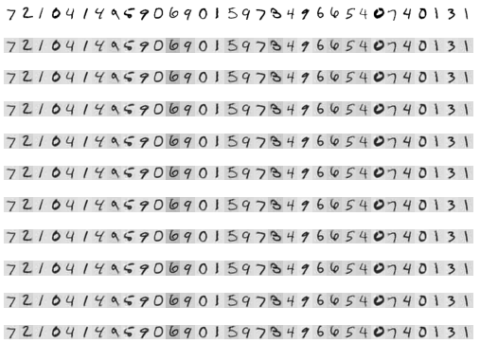


Figure 11: Reconstructed Autoencoder Images from 2nd row

(a) Input (b) Recon

Figure 12: Autoencoder Inference

9.0.2 VAE

(a) bin (b) gaussian

Figure 13: VAE recon

Παρόμοια φαινόμενα βλέπουμε και στο VAE_cnn που μάλλον εμφανίζονται επειδή έγινε λάθος στο inference για τα πιθανοκρατικα μοντέλα.

9.1 Τέλος αναφοράς

Η τελική μορφή της εργασίας βρίσκεται και το github σύνδεσμο που υπάρχει και στην αρχική σελίδα.

References

[1] A. Ng, "Cs229 lecture notes - supervised learning", 2012.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes", *arXiv preprint arXiv:1312.6114*, 2013.

[3] L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks", in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 2391–2400. [Online]. Available: https://proceedings.mlr.press/v70/mescheder17a.html.

[4] O. Ivanov, M. Figurnov, and D. Vetrov, "Variational autoencoder with arbitrary conditioning", *arXiv preprint arXiv:1806.02382*, 2018.

[5] L. Zhang, D. M. Blei, and C. A. Naesseth, "Transport score climbing: Variational inference using forward kl and adaptive neural transport", *arXiv preprint arXiv:2202.01841*, 2022.

[6] M. Paramasivam, R. Sabeenian, and P. Dinesh, "The effect of binarization algorithms considering color-to-gray scale conversion methods on historic document images",