Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Συστηματων Τεχνητης Νοημοσυνης και Μαθησης

Embedding-Based Variants of Deep Generative Models

Exploring GANs, VAEs and Diffusion Architectures with Dense Representations for Categorical Attributes

Diploma Thesis

KOUTENTAKIS STAVROS

Supervisor: Panayiotis Tsanakas

Professor, NTUA

Athens, September 2025

Εθνικο Μετσοβιο Πολυτεχνειο

Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Τομεας Τεχνολογιας Πληροφορικης και Υπολογιστων

Εργαστηριο Συστηματων Τεχνητης Νοημοσυνης και Μαθησης

Embedding-Based Variants of Deep Generative Models

Exploring GANs, VAEs and Diffusion Architectures with Dense Representations for Categorical Attributes

Diploma Thesis

KOUTENTAKIS STAVROS

Supervisor: Panayiotis Tsanakas
Professor, NTUA

Approved by the three-member committee on 19/09/25

…….	…….	…….
Panayiotis Tsanakas	Andreas-Georgios Stafylopatis	Georgios Alexandridis
Professor, NTUA	Professor, NTUA	Assistant Professor, NKUA

.....

Koutentakis Stavros

Graduate of School of Electrical and Computer Engineering, National Technical University of Athens

Περίληψη

Τα παραγωγικά μοντέλα Μηχανικής Μάθησης (ΜΜ), όπως τα Παραγωγικά Ανταγωνιστικά Δίκτυα (GAN), οι Εναλλασσόμενοι Αυτόματοι Κωδικοποιητές (VAE) και τα μοντέλα διάχυσης (Diffusion Models), έχουν επιτύχει αξιοσημείωτα αποτελέσματα στο πεδίο των εικόνων. Ωστόσο, η απόδοση τους σε δεδομένα πίνακα, ειδικά σε σύνολα δεδομένων με πολλές κατηγορικές κολώνες, δεν παρουσιάζει ιδανικά αποτελέσματα. Η αυξανόμενη ζήτηση για υψηλής ποιότητας δεδομένα, τα οποία αποκρύπτουν και ιδιωτικές πληροφορίες των δειγμάτων, έχει τονίσει την ανάγκη για βελτίωση σε αυτόν τον τομέα. Μία σημαντική πρόκληση που ενέχει αυτός ο σκοπός είναι η αναπαράσταση των κατηγορικών μεταβλητών, οι οποίες συνήθως κωδικοποιούνται με τη μέθοδο one-hot, από την οποία προκύπτουν αραιές αναπαραστάσεις που δεν περιέχουν χρήσιμη πληροφορία και μπορούν να κάνουν τη μοντελοποίηση ασταθή και δύσκολη. Στην παρούσα διπλωματική εργασία, αντιμετωπίζουμε αυτές τις προκλήσεις με τη χρήση πυκνών ενσωματώσεων (embeddings) ως εναλλακτική αναπαράσταση των κατηγορικών χαρακτηριστικών στην παραγωγική μοντελοποίηση. Κατασκευάζουμε ένα ειδικά σχεδιασμένο μοντέλο ενσωμάτωσης, εμπνευσμένο από την αρχιτεκτονική skip-gram και προσαρμοσμένο σε δεδομένα πίνακα, για να μάθει τις διανυσματικές αναπαραστάσεις των κατηγορικών τιμών βασισμένο στη συνύπαρξή τους ή μη. Εφαρμόζουμε τις παραγόμενες ενσωματώσεις στη συνέχεια σε τρία παραγωγικά μοντέλα και κατασκευάζουμε τις καινοτόμες αρχιτεκτονικές των eGAN, eVAE και eDDPM. Η αξιολόγησή τους γίνεται τόσο σε μικτά δεδομένα όσο και σε αποκλειστικά κατηγορικά σύνολα αντλούμενα από τα σύνολα δεδομένων Adult Income και Mushroom. Τα συνθετικά σύνολα δεδομένων που δημιουργούνται από όλα τα μοντέλα αξιολογούνται με βάση μετρήσεις που λαμβάνουν υπόψη την πιστότητα και την ιδιωτικότητα. Τα μοντέλα ενισχυμένα με ενσωμάτωση επιδεικνύουν ανταγωνιστική απόδοση, ιδιαίτερα στην κατανόηση των εξαρτήσεων μεταξύ των κατηγορικών κολώνων και στην αντιμετώπιση προκλήσεων που ενέχει η παραγωγή διακριτών δεδομένων. Αυτή η διπλωματική εργασία αναδεικνύει τις προοπτικές των αναπαραστάσεων ενσωμάτωσης προς βελτίωση των παραγωγικών μοντέλων για διακριτά δεδομένα καθώς και τον πιο αποδοτικό και φυσικό χειρισμό των δεδομένων αυτών.

Λέξεις Κλειδιά: Δημιουργία Συνθετικών Δεδομένων, Μοντελοποίηση Δεδομένων Πίνακα, Παραγωγικά Ανταγωνιστικά Δίκτυα, Εναλλασσόμενοι Αυτόματοι Κωδικοποιητές, Μοντέλα Διάχυσης, Ενσωματώσεις Δεδομένων

Abstract

Generative Machine Learning Models such as GANs, VAEs and Diffusion Models have achieved remarkable results in image domains. However, their performance on tabular data, especially datasets with many categorical features, has not seen such excellent results. The increasing demand for high-quality, diverse and privacy-preserving data has highlighted the need for improvement in this area. A major challenge arising in this task is the representation of categorical variables, which are commonly one-hot encoded, leading to sparse, non-informative inputs that can make modeling unstable and difficult. This thesis investigates the use of dense, learned embeddings for categorical attributes as an alternative representation for generative modeling. A custom embedding model, inspired by the skip-gram architecture and adapted to tabular data, is introduced to learn co-occurrence-based vector representations of categorical values. These embeddings are then integrated into three generative models, resulting in the novel architectures of eGAN, eVAE and eDDPM. The proposed models are evaluated against their highly appreciated and used baseline counterpart models: CTGAN, TVAE and TabDDPM. Tests are conducted on both mixed-type and categorical-only versions of the Ault Income and Mushroom datasets. The generated synthetic datasets of all models are assessed by metrics that account for fidelity and privacy. The embedding-augmented models demonstrate competitive performance, especially in capturing categorical dependencies and mitigating some of the challenges associated with discrete data generation. This work highlights the potential of embedding-based representations to improve generative modeling of tabular data and opens new directions for handling discrete features more naturally and effectively.

Keywords: Synthetic Data Generation, Tabular Data Modeling, Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), Diffusion Models, Data Embeddings

Table of Contents

Chapter 1

Εκτεταμένη Περίληψη στα Ελληνικά

1.1 Εισαγωγή

1.1.1 Κίνητρα

Η αυξανόμενη ζήτηση για δεδομένα υψηλής ποιότητας έχει προκαλέσει έντονο ενδιαφέρον για τη δημιουργία συνθετικών δεδομένων. Έχουν αναπτυχθεί πολλές τεχνικές, με καινοτομίες που αποδίδουν εξαιρετικά σε συγκεκριμένα πεδία, ενώ σε άλλα να υστερούν ελαφρώς. Συγκεκριμένα, τα δεδομένα πινάκων έχουν πολλές εφαρμογές και προσελκύουν μεγάλο ενδιαφέρον στον τομέα της πληροφορικής, ωστόσο τα παραγωγικά μοντέλα συχνά δυσκολεύονται να διαχειριστούν ζητήματα όπως η προστασία της ιδιωτικότητας, η ανισορροπία των κατηγοριών, οι μικτοί τύποι δεδομένων και πολλές άλλες προκλήσεις. Ενώ τα γενετικά μοντέλα, όπως τα Παραγωγικά Ανταγωνιστικά Δίκτυα (GANs), [1] και οι Εναλλασσόμενοι Αυτόματοι Κωδικοποιητές (VAEs) [2], έχουν επιτύχει αξιοσημείωτα αποτελέσματα στο πεδίο της εικόνας, η απόδοσή τους σε δεδομένα σε μορφή πίνακα (ειδικά σε δεδομένα αποκλειστικά κατηγορικού τύπου) παραμένει περιορισμένη.

Τα περισσότερα μοντέλα καταφεύγουν στην one-hot και ordinal κωδικοποίηση[3] για την αναπαράσταση κατηγορικών χαρακτηριστικών. Ωστόσο, αυτές οι τεχνικές οδηγούν σε αραιές και άκαμπτες αναπαραστάσεις χωρίς ουσιαστικό νόημα και πληροφορίες στις τιμές τους. Ο στόχος που διαμορφώθηκε ήταν να καταστεί δυνατή η φυσική λειτουργία των αρχικών μοντέλων πάνω στα κατηγορικά χαρακτηριστικά, όπως ακριβώς γίνεται στα αριθμητικά. Κατά αυτόν τον τρόπο μπορούν να εκμεταλλευθούν οι ακατέργαστες δυνατότητες των παραγωγικών μοντέλων, οι οποίες υποκρύπτονται χάρη στις αραιές και διακριτές αναπαραστάσεις.

Για να αντιμετωπιστεί αυτό, εξετάστηκε η ιδέα της χρήσης πυκνών ενσωματώσεων. Πρόκειται για μια τεχνική δανεισμένη από την επεξεργασία φυσικής γλώσσας (NLP), όπου οι ενσωματώσεις λέξεων έχουν χρησιμοποιηθεί επανειλημμένα με επιτυχία για τη μοντελοποίηση σημασιολογικών σχέσεων. Με την εκπαίδευση ενός εξειδικευμένου μοντέλου ενσωμάτωσης με βάση τη συνύπαρξη μεταξύ κατηγορικών τιμών, κάθε μία από αυτές μετατρέπεται σε ένα συνεχή διάνυσμα σε έναν κοινό λανθάνοντα χώρο. Αυτή η αναπαράσταση επιτρέπει στα γενετικά μοντέλα να λειτουργούν σε έναν συνεχή χώρο, επιτρέποντας την οπισθοδρόμηση (backpropagation) σε ολόκληρο το δίκτυο.

Για να δοκιμαστεί αυτή η προσέγγιση, πραγματοποιήθηκαν πειράματα σε τρία δημοφιλή γενετικά μοντέλα: GAN, VAE και DDPM (Denoising Diffusion Probabilistic Model). Ως βάση για τα πειράματα παραγωγής μικτών συνθετικών δεδομένων χρησιμοποιήθηκαν τα ευρέως γνωστά CTGAN, TVAE και TabDDPM. Για την αξιολόγηση της προσέγγισης των ενσωματώσεων, κατασκευάσαμε τα ομόλογά τους eGAN, eVAE και eDDPM πάνω στα οποία έγιναν τα ίδια πειράματα με αυτά που έγιναν στα βασικά μοντέλα. Πρώτα, αξιολογήθηκε η απόδοση τους σε δεδομένα μικτού τύπου, προτού συνεχιστούν τα πειράματα σε σύνολα δεδομένων αποκλειστικά κατηγορικού τύπου.

1.1.2 Συνεισφορές

Η παρούσα διπλωματική εργασία παρουσιάζει τις ακόλουθες βασικές συνεισφορές:

- Ενσωμάτωση για διακριτά δεδομένα σε πίνακες: Εισήχθη μια νέα στρατηγική ενσωμάτωσης, εμπνευσμένη από το skip-gram με αρνητική δειγματοληψία (negative sampling), προσαρμοσμένη ειδικά για δεδομένα σε πίνακες χωρίς χωρική σειρά. Το μοντέλο μαθαίνει ενσωματώσεις βασισμένες στη συνύπαρξη σε κατηγορικές στήλες.
- Νέες παραγωγικές αρχιτεκτονικές: Με βάση τις προαναφερθείσες ενσωματώσεις, αναπτύχθηκαν τρία νέα μοντέλα:
    - eGAN: μία γεννήτρια βασισμένη στο GAN που λειτουργεί με ενσωματωμένα διανύσματα καθώς και αριθμητικές στήλες
    - eVAE: ένα μοντέλο VAE που ανακατασκευάζει ενσωματωμένα διανύσματα και τα αποκωδικοποιεί σε κατηγορικές τιμές
    - eDDPM: ένα μοντέλο DDPM, επίσης εκπαιδευμένο σε δεδομένα μικτού τύπου, που ανακατασκευάζει τα ενσωματωμένα διανύσματα σε κατηγορικές τιμές.
- Αξιολόγηση σε μικτά αλλά και αποκλειστικά κατηγορικά δεδομένα: Πραγματοποιήθηκαν πειράματα σε μικτές και αποκλειστικά κατηγορικές εκδοχές των συνόλων δεδομένων Adult Income και Mushroom του αποθετηρίου UCI, χρησιμοποιώντας πολλαπλές μετρικές ιδιωτικότητας και πιστότητας.

Διαπιστώσεις κα προοπτικές: Τα αποτελέσματα των μοντέλων αξιολογούνται σε δύο διαφορετικούς τύπους δεδομένων. Προκύπτουν συμπεράσματα και τονίζονται οι προοπτικές των μοντέλων καθώς και η μελλοντική έρευνα σε αυτό τον τομέα.

1.2 Σύντομη Ιστορία και Εμπόδια της Επιστήμης των Δεδομένων

Η επιστήμη των υπολογιστών από τη δημιουργία της αποτέλεσε έναν όρο που συνένωνε πολλά θεωρητικά πεδία, όπως οι αλγόριθμοι και η θεωρία πληροφορίας, καθώς και εφαρμοσμένους τομείς όπως η μηχανική υλικού και λογισμικού. Ωστόσο, πολλές πρωτοποριακές ιδέες, όπως για παράδειγμα η κρυπτογραφία δημόσιου κλειδιού και τα παράλληλα υπολογιστικά συστήματα, έμειναν για χρόνια μόνο στη θεωρία, καθώς η έλλειψη υπολογιστικών πόρων δεν μπορούσε να ικανοποιήσει τις ανάγκες τους. Η Τεχνητή Νοημοσύνη (ΤΝ), αν και θεμελιώθηκε τη δεκαετία του 1950, γνώρισε πραγματική ανάπτυξη μόνο στις αρχές του 21ου αιώνα, χάρη στην εξέλιξη των υπολογιστικών πόρων, γεγονός που επέτρεψε εφαρμογές όπως η υπολογιστική όραση, η αναγνώριση ομιλίας, η επεξεργασία φυσικής γλώσσας καθώς και την άνοδο των 'Big Data' [4].

Παράλληλα, η ραγδαία ανάπτυξη των Μεγάλων Γλωσσικών Μοντέλων (LLMs) συνοδεύεται από σημαντικούς περιορισμούς, καθώς η εκπαίδευσή τους απαιτεί τεράστιους ενεργειακούς πόρους. Ενδεικτικά, μελέτες έδειξαν ότι η εκπαίδευση μοντέλων όπως το BERT και το GPT-2 παρήγαγε έως και 626.000 λίβρες διοξειδίου του άνθρακα [5]. Παρά τη χρήση καρτών γραφικών που το 2010 πολλαπλασίασαν την ταχύτητα εκπαίδευσης κατά 10-15 φορές σε σχέση με τους επεξεργαστές [6], η συνεχής αύξηση του αριθμού παραμέτρων των μοντέλων, αρχίζει να οδηγεί ξανά σε υπολογιστικό αδιέξοδο. Επιπλέον, η αυξανόμενη ανάγκη για δεδομένα εκπαίδευσης καθιστά την εξεύρεση επαρκών και ποιοτικών συνόλων δεδομένων κρίσιμη αλλά δύσκολη, καθώς υπάρχουν ζητήματα ιδιωτικότητας, περιορισμός δειγμάτων αλλά και υψηλό κόστος συλλογής τους.

1.2.1 Προκλήσεις για τη Συλλογή Πραγματικών Δεδομένων

Υπάρχουν μεγάλες προκλήσεις στη συλλογή πραγματικών δεδομένων για την εκπαίδευση μοντέλων μηχανικής μάθησης. Πρώτον, τίθεται το ζήτημα της ιδιωτικότητας, καθώς η απόκτηση μεγάλων συνόλων δεδομένων οδηγεί συνήθως στη χρήση προσωπικών πληροφοριών. Αυτό συχνά φέρνει οργανισμούς αντιμέτωπους με νομικούς περιορισμούς ή κινδύνους παραβίασης νομοθεσίας περί πνευματικών δικαιωμάτων και προστασίας προσωπικών δεδομένων [7]. Η διεθνής νομοθεσία απαιτεί τη ρητή συγκατάθεση από τα άτομα και αυστηρή συμμόρφωση με τους κανονισμούς. Δεύτερον, τα δεδομένα παρουσιάζουν συχνά μεροληψία (bias), είτε επειδή δεν αντικατοπτρίζουν την πραγματικότητα με ακρίβεια είτε επειδή αποτυπώνουν μια αλλοιωμένη από εξωτερικούς παράγοντες πραγματικότητα, όπως οι χρόνιοι κοινωνικοοικονομικοί αποκλεισμοί και προκαταλήψεις [8]. Επί παραδείγματι, δεδομένα από νοσοκομείο που εξυπηρετεί κυρίως μια συγκεκριμένη κοινωνική τάξη δεν είναι αντιπροσωπευτικά του γενικού πληθυσμού. Επιπλέον, μεροληψίες μπορούν να εισαχθούν από ανθρώπινα λάθη κατά την επισήμανση δεδομένων ή από σφάλματα στην προεπεξεργασία,

οδηγώντας σε αντιστοίχως μεροληπτικά μοντέλα [9]. Τρίτον, παρά την άνοδο των Μεγάλων Δεδομένων, η διαθεσιμότητα ορισμένων κρίσιμων τύπων δεδομένων δεν είναι εξασφαλισμένη. Για παράδειγμα, η ανάπτυξη αυτόνομων οχημάτων προϋποθέτει δεδομένα με συγκρούσεις και αποφυγές πεζών, κάτι εξαιρετικά δύσκολο να συλλεχθεί με συνέπεια μεταξύ διαφορετικών πηγών αλλά και σε επαρκή ποσότητα [8].

Παράλληλα, η αξιοπιστία των δεδομένων που επισημαίνονται χειροκίνητα είναι συχνά αμφισβητήσιμη αφού γίνεται από εργαζομένους υπό κακές συνθήκες, κάτι που καθιστά δύσκολη την αποφυγή λαθών [10]. Τέλος, προκύπτει ζήτημα εργασιακής εκμετάλλευσης και ηθικής, καθώς πολλές εταιρίες, στην προσπάθεια τους να συγκεντρώσουν μεγάλους όγκους δεδομένων, καταφεύγουν σε φθηνή εργασία υπό εξουθενωτικές συνθήκες προσφέροντας ελάχιστα δικαιώματα [10].

1.2.2 Τεχνικές Παραγωγής Δεδομένων και Προστασίας Ιδιωτικότητας

Για την αντιμετώπιση ζητημάτων ιδιωτικότητας και περιορισμένης διαθεσιμότητας δεδομένων έχουν αναπτυχθεί διάφορες τεχνικές. Η αποκεντρωμένη μάθηση (federated learning) επιτρέπει την εκπαίδευση μοντέλων σε κατανεμημένα δεδομένα χωρίς αυτά να εγκαταλείπουν τις δομές στις οποίες είναι αποθηκευμένα, με το τελικό αποτέλεσμα να προκύπτει από τη συνένωση των μερικών αποτελεσμάτων [11]. Η μέθοδος k-anonymity ανωνυμοποιεί δεδομένα ομαδοποιώντας παρόμοιες εγγραφές και γενικεύοντας ευαίσθητα χαρακτηριστικά, αν και συνήθως παρατηρείται απώλεια πληροφορίας [12]. Τέλος, η τεχνική SMOTE στοχεύει στη διόρθωση της ανισορροπίας κλάσεων μέσω τεχνητής υπερδειγματοληψίας. Οι παραλλαγές της (SMOTE-NC) επιτρέπουν την ταυτόχρονη διαχείριση συνεχών και κατηγορικών χαρακτηριστικών, με χρήση κατάλληλων τεχνικών και απλοποιημένων κανόνων [13], [14].

1.2.3 Συνθετικά Δεδομένα

Τα συνθετικά δεδομένα είναι τεχνητά παραγόμενα σύνολα δεδομένων που μιμούνται πραγματικά δεδομένα μέσω αλγορίθμων. Στόχος τους είναι η διατήρηση των στατιστικών ιδιοτήτων των πραγματικών δεδομένων, επιτρέποντας τη χρήση τους χωρίς παραβίαση της ιδιωτικότητας ή αποκάλυψη προσωπικών δεδομένων [15]. Η αξία τους είναι ιδιαίτερα σημαντική σε περιπτώσεις όπου η πρόσβαση σε πραγματικά δεδομένα είναι περιορισμένη, δαπανηρή και ηθικά απαγορευτική, όπως σε εφαρμογές υγείας.

1.3 Θεωρητικό Υπόβαθρο

1.3.1 Βασικές Αρχές

Η Τεχνητή Νοημοσύνη (AI) στοχεύει στην προσομοίωση της ανθρώπινης νοημοσύνης σε μηχανές. Βασίζεται σε αλγορίθμους που εφαρμόζονται σε δεδομένα για την αναγνώριση προτύπων και τη βελτίωση με την πάροδο του χρόνου, ενώ περιλαμβάνει επιμέρους τομείς όπως η επεξεργασία φυσικής γλώσσας και η όραση υπολογιστών. Η Μηχανική Μάθηση (ML) είναι το σημαντικότερο πεδίο της Τεχνητής Νοημοσύνης. Δίνει τη δυνατότητα στους υπολογιστές, μέσω αλγορίθμων και στατιστικών μοντέλων, να μαθαίνουν από δεδομένα και να βελτιώνουν την απόδοσή τους όσο αποκτούν εμπειρία. Η ικανότητα μάθησης μέσω επανάληψης, λαθών και επιτυχιών καθιστά τη Μηχανική Μάθηση κινητήριο μοχλό της προόδου της Τεχνητής Νοημοσύνης, με πληθώρα εφαρμογών.

Η ιδέα των Νευρωνικών Δικτύων εμπνεύστηκε από τη λειτουργία του ανθρώπινου εγκεφάλου, στον οποίο οι νευρώνες μεταδίδουν σήματα μέσω συνάψεων. Το θεμελιώδες δομικό στοιχείο των Νευρωνικών Δικτύων είναι ο perceptron, που μιμείται τον ανθρώπινο νευρώνα, λαμβάνει κάποιες εισόδους, τους εφαρμόζει βάρη και τις επεξεργάζεται σε έναν κόμβο μέσω συναρτήσεων ενεργοποίησης. Οι τιμές αυτές προσαρμόζονται ώστε να μοιάζουν με τα ηλεκτρικά σήματα του εγκεφάλου, με τις συνήθεις συναρτήσεις ενεργοποίησης να παράγουν τιμές στο διάστημα [-1,1] ή [0,1]. Όταν πολλοί perceptrons συνδυαστούν σε στρώματα, σχηματίζουν δίκτυα που μπορούν να μιμηθούν πολύπλοκες συμπεριφορές και να μάθουν μέσω επαναλαμβανόμενων παραδειγμάτων, επιτρέποντας έτσι την ανάπτυξη συστημάτων, ικανών να προσαρμόζονται και να εκτελούν σύνθετες εργασίες.

Τα Πολυεπίπεδα Νευρωνικά Δίκτυα (Multilayer Neural Networks) βασίζονται στη διαδικασία της οπισθοδιάδοσης σφάλματος (backpropagation), κατά την οποία το δίκτυο αρχικά ξεκινά με τυχαία βάρη και παράγει προβλέψεις που αξιολογούνται με βάση κάποιο κριτήριο σφάλματος. Το σφάλμα αυτό υπολογίζεται ως διαφορά μεταξύ των προβλέψεων $\hat{Y}$ και των πραγματικών τιμών Y, ενώ στη συνέχεια υπολογίζονται οι παράγωγοι του σφάλματος ως προς τα βάρη, οι τιμές των οποίων καθορίζουν την ενημέρωση των βαρών στις νέες τους τιμές. Για παράδειγμα, αν η αύξηση ενός βάρους οδηγεί στην αύξηση του σφάλματος, το βάρος θα μειωθεί, και αντίστροφα, με αποτέλεσμα το δίκτυο να συγκλίνει προοδευτικά σε σωστότερες προβλέψεις [16]. Η διαδικασία αυτή επαναλαμβάνεται πολλές φορές ώσπου το σφάλμα να ελαχιστοποιηθεί και το μοντέλο να αποκτήσει ικανοποιητική προβλεπτική ικανότητα αλλά και ικανότητα γενίκευσης.

Βασικό εργαλείο στον υπολογισμό των παραγώγων είναι ο κανόνας της αλυσίδας (chain rule), που επιτρέπει την ανίχνευση της επίδρασης ενός βάρους στο τελικό αποτέλεσμα, λαμβάνοντας υπόψη όλους τους ενδιάμεσους υπολογισμούς.

Έστω η σύνθετη συνάρτηση $f_1 \circ (f_2 \circ \dots (f_{n-1} \circ f_n)$, μπορούμε να υπολογίσουμε την παράγωγο της f_1ως προς το x με τον κανόνα της αλυσίδας ως εξής:

$$\frac{df_1}{x} = \frac{df_1}{df_2}\frac{df_2}{df_3}\dots\frac{df_n}{x} \tag{1.3.1}$$

Επιπλέον, οι συναρτήσεις ενεργοποίησης (activation functions) παίζουν καθοριστικό ρόλο, καθώς εισάγουν μη γραμμικότητα και καθιστούν δυνατή τη μοντελοποίηση πολύπλοκων σχέσεων. Για να είναι αποτελεσματική η εκπαίδευση με κατιούσα κλίση (gradient descent), οι συναρτήσεις αυτές πρέπει να είναι διαφορίσιμες. Η επιλογή τους εξαρτάται από την αρχιτεκτονική του δικτύου και τα δεδομένα, καθώς επηρεάζει προβλήματα όπως η εξαφάνιση ή η υπερβολική όξυνση των κλίσεων.

1.3.2 Μέθοδοι Μηχανικής Μάθησης

Από μία οπτική, η μηχανική μάθηση χωρίζεται σε δύο βασικές κατηγορίες: εποπτευόμενη μάθηση, όπου τα μοντέλα εκπαιδεύονται με επισημασμένα δεδομένα και συγκρίνουν τις προβλέψεις τους με τα σωστά αποτελέσματα ώστε να βελτιώνονται, και μη εποπτευόμενη μάθηση, όπου χρησιμοποιούνται μη επισημασμένα δεδομένα με στόχο την εύρεση προτύπων, ομάδων ή τη μείωση διαστασιμότητας [17]. Η επιλογή της μεθόδου εξαρτάται τόσο από τη φύση του προβλήματος όσο και από τη φύση και τη διαθεσιμότητα των δεδομένων, με κάθε προσέγγιση να έχει πλεονεκτήματα αλλά και περιορισμούς.

Από μία διαφορετική οπτική, τα μοντέλα μηχανικής μάθησης μπορούν να διακριθούν σε ντετερμινιστικά και πιθανοτικά, ανάλογα με το αν ενσωματώνουν ή όχι τυχαιότητα στα αποτελέσματά τους. Τα ντετερμινιστικά μοντέλα παράγουν πάντα το ίδιο αποτέλεσμα για δεδομένη είσοδο, γεγονός που τα καθιστά κατάλληλα για καλά ορισμένα προβλήματα που χρίζουν ακριβών προβλέψεων. Αντίθετα, τα πιθανοτικά μοντέλα εκφράζουν τα αποτελέσματα τους ως κατανομές πιθανότητας, εισάγοντας έτσι ποικιλία και επιτρέποντας διαφορετικές εκβάσεις ακόμη και για ίδιες εισόδους.

Στη δημιουργία συνθετικών δεδομένων, η ποικιλομορφία είναι κρίσιμη για την ποιότητα, καθώς ένα μοντέλο πρέπει να παράγει διαφορετικά δείγματα και όχι να επαναλαμβάνει τα ίδια. Για τον λόγο αυτό, τα πιθανοτικά μοντέλα υπερτερούν στα παραγωγικά έργα, ενώ οι ντετερμινιστικές προσεγγίσεις συχνά χρειάζονται πρόσθετες τεχνικές που καταλήγουν να ενσωματώνουν στοιχεία πιθανοτικής μοντελοποίησης.

1.3.3 Κατανομή Δεδομένων και Πινθανότητες

Η ποιότητα των συνθετικών δεδομένων εξαρτάται από το πόσο πιστά κληρονομούν τα χαρακτηριστικά των πραγματικών δεδομένων, τόσο σε ποικιλία όσο και σε ακρίβεια. Κεντρικό ρόλο στη δημιουργία υψηλής ποιότητας δεδομένων παίζει η θεωρία πιθανοτήτων, η οποία παρέχει το μαθηματικό πλαίσιο για την αποτύπωση της αβεβαιότητας και της τυχαιότητας. Τα περισσότερα σύγχρονα γεννητικά μοντέλα ανήκουν στην κατηγορία των πιθανοτικών μοντέλων, με κύριο στόχο την προσέγγιση πολύπλοκων κατανομών πιθανότητας. Η πιο συχνά χρησιμοποιούμενη κατανομή είναι η κανονική (Γκαουσιανή) $X \sim N(\mu,\sigma)$, που ορίζεται από τον

μέσο όρο μ και την τυπική απόκλιση σ. Ο μέσος καθορίζει το σημείο κεντροποίησης των τιμών, ενώ η τυπική απόκλιση ορίζει την πυκνότητα και τη διασπορά τους στον άξονα x: μικρότερη σ συνεπάγεται πιο «αιχμηρή» καμπύλη, ενώ μεγαλύτερη οδηγεί σε πιο επίπεδη.

Μαθηματικά, η κανονική κατανομή ορίζεται ως:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \quad (1.3.3.1)$$

Ωστόσο, τα πραγματικά δεδομένα σπάνια περιγράφονται με ακρίβεια από μια μονοτροπική Γκαουσσιανή. Για τον λόγο αυτό, τα σύγχρονα πιθανοτικά παραγωγικά μοντέλα συχνά υιοθετούν πολυτροπικές κατανομές (multimodal distributions), δηλαδή προσεγγίσεις που συνδυάζουν πολλαπλές Γκαουσσιανές κατανομές. Με αυτόν τον τρόπο αποτυπώνεται καλύτερα η πολυπλοκότητα και η ποικιλία των δεδομένων, επιτυγχάνοντας πιο ρεαλιστική και αξιόπιστη δημιουργία συνθετικών δειγμάτων. Βασικό παράδειγμα αποτελεί το Μοντέλο Μείγματος Γκαουσσιανών (GMM) του οποίου η συνάρτηση πυκνότητας πιθανότητας εκφράζεται ως:

$$p(x) = \sum_{\mathrm{i}}^{\mathrm{K}} \varphi_i N(x|\mu_i, \sigma_i) \quad (1.3.3.2)$$

όπου φ_i είναι τα βάρη κάθε συνιστώσας του μείγματος.

1.3.4 Βασικά Μοντέλα, Προκλήσεις και Λύσεις

1.3.4.1 Εναλλασσόμενοι Αυτόματοι Κωδικοποιητές

Οι Εναλλασσόμενοι Αυτόματοι Κωδικοποιητές (VAEs) αποτελούν εξέλιξη των κλασικών αυτόματων κωδικοποιητών, καθώς ενσωματώνουν πιθανοτική προσέγγιση στο λανθάνοντα χώρο (variational space) . Διατηρούν την ίδια βασική αρχιτεκτονική με κωδικοποιητή και αποκωδικοποιητή, όμως σε αντίθεση με τους ντετερμινιστικούς autoencoders, οι VAEs παράγουν κατανομές πιθανοτήτων αντί για διακριτές μεταβλητές. Αυτό τους δίνει ισχυρές παραγωγικές δυνατότητες, καθώς μπορούν να μοντελοποιήσουν την πραγματική κατανομή δεδομένων $p(x)$ εισάγοντας λανθάνουσες μεταβλητές z και προσεγγίζοντας την κοινή κατανομή $p_\theta(x, z)$. Ο κωδικοποιητής προσεγγίζει την κατανομή του λανθάνοντα χώρου, ο οποίος μοντελοποιείται σαν Γκαουσσιανή κατανομή. Ο στόχος του δικτύου είναι η μεγιστοποίηση του Evidence Lower Bound (ELBO), που περιλαμβάνει τον όρο απόκλιση Kullback-Leibler (KL divergence) για την ελαχιστοποίηση της απόστασης μεταξύ του παραγόμενου λανθάνοντα χώρου και της Γκαουσσιανής κατανομής [2], [20].

Παρόλα αυτά, παρατηρείται ότι μεταξύ του κωδικοποιητή και του αποκωδικοποιητή, υπάρχει ένα βήμα δειγματοληψίας τυχαίου z από την κατανομή που παράγει ο κωδικοποιητής. Για να είναι εφικτή η εκπαίδευση μέσω οπισθοδιάδοσης σφάλματος, εισάγεται μια βοηθητική

μεταβλητή θορύβου $\epsilon \sim N(0,1)$ και γίνεται δειγματοληψία του z ως $z = \mu + \sigma * \epsilon$. Αυτή η μέθοδος ονομάζεται reparameterization trick και μετατρέπει τη μη διαφορίσιμη διαδικασία της δειγματοληψίας ενός τυχαίου z, επιτρέποντας τη ροή των παραγώγων, και επομένως την οπισθοδιάδοση σφάλματος, σε όλο το μήκος του δικτύου.

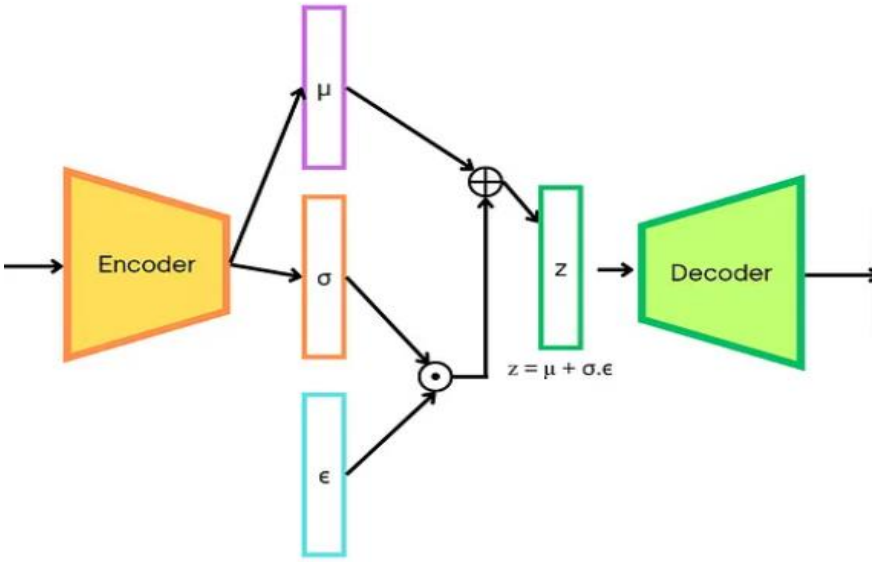


Figure 1: Αρχιτεκτονική δικτύου VAE

1.3.4.2 Παραγωγικά Ανταγωνιστικά Δίκτυα

Τα Παραγωγικά Ανταγωνιστικά Δίκτυα (GANs) παρουσιάστηκαν για πρώτη φορά το 2014 από τον Ian Goodfellow και την ερευνητική του ομάδα [1], αποτελώντας μια πρωτοποριακή προσέγγιση στα γεννητικά μοντέλα τεχνητής νοημοσύνης. Έκτοτε, έχουν γνωρίσει ευρεία αποδοχή λόγω των εντυπωσιακών αποτελεσμάτων τους, κυρίως στην παραγωγή εικόνων, ενώ έχουν αναπτυχθεί πολλές εξειδικευμένες παραλλαγές, όπως τα medGAN (για ιατρικά δεδομένα), textGAN (για φυσική γλώσσα), cycleGAN, discoGAN κ.ά.

Η βασική αρχιτεκτονική ενός GAN αποτελείται από δύο πολυεπίπεδα νευρωνικά δίκτυα: την γεννήτρια (Generator G) και τον διακριτή (Discriminator D). Η γεννήτρια δέχεται ως είσοδο τυχαίο θόρυβο $z \sim p_z(z)$ και τον αντιστοιχίζει στον χώρο των πραγματικών δεδομένων ως $G(z; \theta_g)$. Τα παραγόμενα δείγματα εισάγονται μαζί με μερικά πραγματικά δεδομένα στον διακριτή, ο οποίο λειτουργεί ως ταξινομητής και εκτιμά την πιθανότητα $D(x; \theta_d)$ ένα δείγμα να είναι πραγματικό ή ψεύτικο. Η διαδικασία εκπαίδευσης μοντελοποιείται ως παιχνίδι minimax, στο οποίο ο D προσπαθεί να μεγιστοποιήσει την ορθότητα των ταξινομήσεων του, ενώ ο G επιδιώκει να ελαχιστοποιήσει την ικανότητα του D να διαχωρίζει τα δείγματα παράγοντας πιο αξιόπιστα δείγματα.

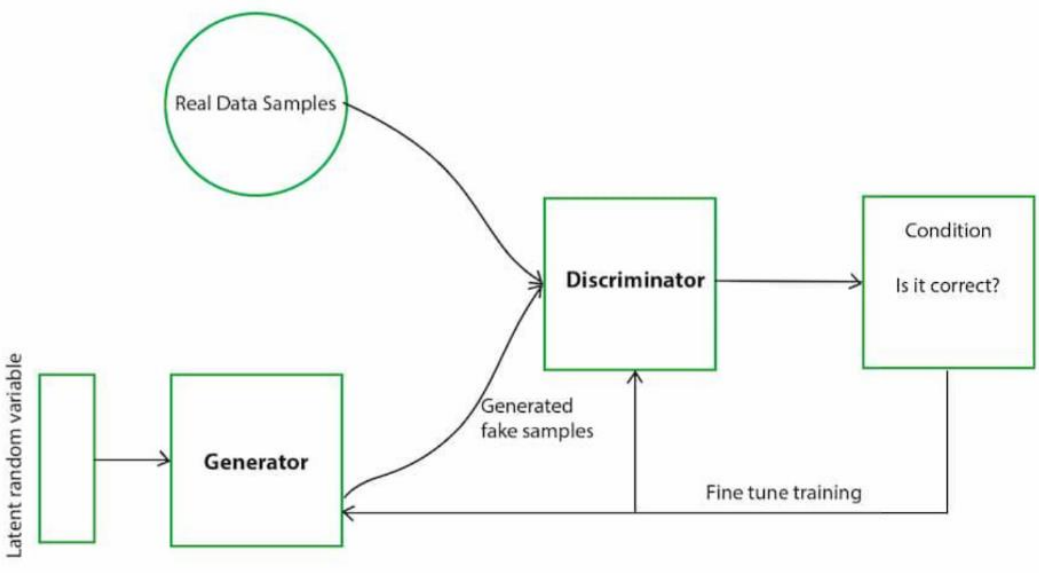


Figure 2: Αρχιτεκτονική GAN

Η εκπαίδευση γίνεται εναλλάξ, ενημερώνοντας διαδοχικά τον G και τον D. Στα αρχικά στάδια, ο G παράγει χαμηλής ποιότητας δείγματα, σχεδόν τυχαία, με αποτέλεσμα ο D να έχει υψηλή ακρίβεια, γεγονός που μπορεί να επιβραδύνει την πρόοδο και των δυο δικτύων. Για να βελτιωθεί η διαδικασία, συχνά η γεννήτρια ξεκινά να εκπαιδεύεται μεγιστοποιώντας το $\log(D(G(x))$ ώστε να παράγονται πιο ισχυρές κλίσεις για την οπισθοδιάδοση.

Ένα από τα συνήθη προβλήματα των GANs είναι το mode collapse, κατά το οποίο τα παραγόμενα δείγματα δεν ποικίλουν αλλά ακολουθούν συγκεκριμένα παρόμοια χαρακτηριστικά. Γι' αυτό είναι σημαντικό τα δύο αντίπαλα δίκτυα να ξεκινούν με παρόμοια 'δύναμη' και να βελτιώνονται ισόρροπα.

1.3.4.3 Μοντέλα Διάχυσης

Τα Μοντέλα Διάχυσης (Diffusion Models) αποτελούν μια από τις πιο πρόσφατες και αποδοτικές μεθόδους στη παραγωγική τεχνητή νοημοσύνη, με εφαρμογές στη σύνθεση εικόνας, στην παραγωγή ήχου και στην επεξεργασία γλώσσας [21]. Ανήκουν στην κατηγορία των πιθανοτικών μοντέλων, όπως και τα VAEs, καθώς ορίζουν ρητά μια συνάρτηση πιθανοφάνειας για την εκτίμηση της κατανομής των δεδομένων. Αυτό τους επιτρέπει να αποφεύγουν προβλήματα όπως το mode collapse που εμφανίζεται σε μοντέλα όπως τα GANs, αν και με τίμημα αυξημένες απαιτήσεις σε πόρους και χρόνο. Η βασική τους έμπνευση προέρχεται από τη διάχυση στη φυσική (diffusion), δηλαδή τη διαδικασία κατά την οποία μόρια κινούνται από περιοχές υψηλής συγκέντρωσης προς περιοχές χαμηλής συγκέντρωσης [22].

Η αρχιτεκτονική των Diffusion Models βασίζεται σε μια διπλή διαδικασία: το forward process και το reverse process. Κατά το forward process, δείγματα δεδομένων αλλοιώνονται προοδευτικά μέσω αλλεπάλληλης προσθήκης Γκαουσσιανού Θορύβου, βάσει ενός προγράμματος απόκλισης $\beta_1, \dots, \beta_T$, μέχρι να καταλήξουν σε καθαρό θόρυβο [23]. Το reverse process επιχειρεί να «αφαιρέσει» τον θόρυβο, μαθαίνοντας μέσω ενός νευρωνικού δικτύου να αντιστρέφει τα βήματα της εμπρόσθιας διαδικασίας (forward process).

Η εκπαίδευση γίνεται μέσω βελτιστοποίησης του variational upper bound της αρνητικής λογαριθμικής πιθανοφάνειας (NLL), με χρήση της απόκλισης Kullback-Leibler (KL divergence) ώστε να ελαχιστοποιηθεί η απόσταση ανάμεσα στο forward και reverse process [21],[23].

1.3.5 Δυσκολίες στην Παραγωγή Διακριτών Δεδομένων Πίνακα

Η δημιουργία συνθετικών δεδομένων πινάκων παρουσιάζει αρκετές δυσκολίες σε σχέση με τις εικόνες, όπου τα παραγωγικά νευρωνικά δίκτυα έχουν σημειώσει εντυπωσιακές επιδόσεις. Στα δεδομένα πινάκων υπάρχουν ποικίλοι τύποι μεταβλητών (αριθμητικά, κατηγορικά, σπάνιες κατηγορίες), γεγονός που αυξάνει τη δυσκολία λόγω υψηλής διαστασιμότητας, αραίωσης και ανισορροπίας συνόλων δεδομένων. Ένα σημαντικό πρόβλημα είναι η μη διαφορισιμότητα των κατηγορικών μεταβλητών, καθώς τα νευρωνικά δίκτυα εκπαιδεύονται με συνεχή δεδομένα μέσω της οπισθοδιάδοσης, ενώ οι διακριτές κατηγορίες (π.χ. πόλη, όνομα, ομάδα αίματος) δεν έχουν ενδιάμεση συνέχεια που να καθοδηγεί την εκπαίδευση. Ένα άλλο κρίσιμο ζήτημα είναι το mode collapse, όπου το μοντέλο παράγει περιορισμένα και επαναλαμβανόμενα αποτελέσματα, με αποτέλεσμα χαμηλή ποικιλία, φαινόμενο ιδιαίτερα έντονο στα κατηγορικά δεδομένα σε σχέση με τα συνεχόμενα.

1.3.6 Ενσωματώσεις Λέξεων

Τα word embeddings αποτελούν θεμελιώδη τεχνική στην επεξεργασία φυσικής γλώσσας (NLP), καθώς μετατρέπουν τις λέξεις σε πυκνές και συνεχείς διανυσματικές αναπαραστάσεις που αποτυπώνουν σημασιολογικές σχέσεις μεταξύ τους. Σε αντίθεση με το one-hot encoding, όπου οι λέξεις θεωρούνται ανεξάρτητες μονάδες και αναπαρίστανται με αραιά και υψηλής διάστασης διανύσματα χωρίς καμία πληροφορία για την ομοιότητα μεταξύ τους, τα embeddings συλλαμβάνουν νοηματικές αναλογίες και μειώνουν σημαντικά την πολυπλοκότητα του μοντέλου. Έτσι, η εκπαίδευση γίνεται πιο αποδοτική και η τελική αναπαράσταση πιο χρήσιμη για εφαρμογές που απαιτούν κατανόηση συμφραζομένων.

Για παράδειγμα, μετά από εκπαίδευση σε ένα μεγάλο σώμα κειμένων, οι λέξεις king, queen, man και woman μπορούν να αναπαρασταθούν με τέτοιο τρόπο ώστε να διατηρούν τις σημασιολογικές τους σχέσεις, με γνωστή ιδιότητα ότι king – queen ≈ man – woman. Αυτή η δυνατότητα προκύπτει φυσικά από τον τρόπο εκμάθησης των embeddings και επιτρέπει τη χρήση τους σε πλήθος εφαρμογών, από συστήματα συστάσεων έως μοντέλα παραγωγής δεδομένων. Ένα από τα πιο σημαντικά μοντέλα είναι το Word2Vec [24], που παρουσιάστηκε το 2013 και χρησιμοποιεί απλά νευρωνικά δίκτυα για να μάθει αναπαραστάσεις λέξεων με βάση τα συμφραζόμενα. Το Word2Vec έχει δύο κύριες αρχιτεκτονικές: το Continuous Bag of Words (CBOW), που προβλέπει μια λέξη από το περιβάλλον της, και το Skip-Gram, που προβλέπει τις γειτονικές λέξεις δεδομένης μιας κεντρικής λέξης.

1.4 Μεθοδολογία

1.4.1 Βασικά Μοντέλα

1.4.1.1 CTGAN

Το CTGAN [25] αποτελεί μια αρχιτεκτονική βασισμένη στα GANs, ειδικά προσαρμοσμένη για την παραγωγή πινάκων μικτών τύπων δεδομένων. Η βασική καινοτομία του αφορά στην προεπεξεργασία και στον τρόπο που τα δεδομένα δίνονται στον generator και τον discriminator. Για τις συνεχείς μεταβλητές χρησιμοποιεί mode-specific normalization με Μοντέλα Γκαουσσιανών Μειγμάτων(GMMs), ώστε κάθε τιμή να αναπαρίσταται μέσω ενός one-hot διανύσματος για τη «λειτουργία» (mode) και ενός πραγματικού αριθμού για τη σχετική θέση της εντός της λειτουργίας. Επιπλέον, ενσωματώνει δεσμευμένη γεννήτρια για την αντιμετώπιση ανισορροπιών κατηγοριών, δηλαδή, αντιστοιχίζει τις διακριτές κολώνες της εισόδου με αυτές της εξόδου. Για τη βελτίωση της ποικιλίας και την αποφυγή mode collapse εφαρμόζει τεχνικές Wasserstein GAN (WGAN) [26] και PacGAN, με το τελευταίο να ενισχύει τον έλεγχο ποικιλίας του discriminator.

1.4.1.2 Tabular VAE

Το Tabular VAE (tVAE) [25] επεκτείνει το κλασικό VAE για την παραγωγή δεδομένων πίνακα. Όπως και το CTGAN, χρησιμοποιεί mode-specific normalization και one-hot encoding για τις κατηγορικές μεταβλητές, ενώ προσαρμόζει τη συνάρτηση απώλειας ώστε να προσθέτοντας όρο cross-entropy για τα διακριτά χαρακτηριστικά. Ο encoder και ο decoder είναι νευρωνικά δίκτυα που βελτιστοποιούν το Evidence Lower Bound (ELBO) όπως στο κλασικό VAE. Ουσιαστικά εκτελεί μια πιο σύνθετη προεργασία των δεδομένων επιτρέποντας στο μοντέλο να χειρίζεται μίξη διακριτών και συνεχών γνωρισμάτων.

1.4.1.3 TabDDPM

Το TabDDPM [27] βασίζεται στα Diffusion Models [28] και αξιοποιεί τον μηχανισμό Multinomial Diffusion [29] για την παραγωγή κατηγορικών δεδομένων. Το forward process αντικαθιστά σταδιακά τα διανύσματα one-hot με τυχαίες κατηγορίες με πιθανότητα β_t, ενώ το reverse process μοντελοποιείται με νευρωνικό δίκτυο που ανακατασκευάζει τα αρχικά δεδομένα. Για τα αριθμητικά χαρακτηριστικά εφαρμόζεται Gaussian diffusion, ενώ για τα κατηγορικά χρησιμοποιείται ξεχωριστή διαδικασία για το καθένα. Το τελικό σφάλμα συνδυάζει το Γκαουσσιανό μέσο τετραγωνικό σφάλμα με τον μέσο όρο των αποκλίσεων KL των κατηγορικών μεταβλητών. Το TabDDPM έχει δείξει ανώτερη απόδοση σε σχέση με μοντέλα όπως το CTGAN και το tVAE, ιδιαίτερα στη διαχείριση μικτών τύπων δεδομένων.

1.4.2 Υλοποίηση Μοντέλου Ενσωμάτωσης

Τα word embeddings χρησιμοποιούνται για να μετατρέπουν λέξεις σε διανύσματα που μαθαίνονται. Στο πλαίσιο της παρούσας μελέτης, τα embeddings εφαρμόζονται σε πινακοειδή δεδομένα αντί για φυσική γλώσσα. Έτσι, κάθε γραμμή δεδομένων αντιμετωπίζεται ως «πρόταση» και οι τιμές των γνωρισμάτων ως «λέξεις». Η διαφορά με τα κείμενα είναι ότι οι πίνακες έχουν σταθερό αριθμό στηλών και η σειρά των γνωρισμάτων είναι προκαθορισμένη, κάτι που απλοποιεί την υλοποίηση.

Η υλοποίηση εμπνέεται από το Continuous Skip-Gram μοντέλο [24], το οποίο μαθαίνει αποδοτικά αναπαραστάσεις λέξεων με χαμηλή υπολογιστική πολυπλοκότητα. Η απλότητα προκύπτει από την απουσία μη γραμμικού κρυφού επιπέδου. Το δίκτυο εκπαιδεύεται κάθε φορά με μια κενρική λέξη (center) και ένα συμφραζόμενο (context), δηλαδή μια λέξη που συναντάται με την κεντρική ή όχι. Ο στόχος του είναι να μεγιστοποιεί την πιθανότητα σωστής πρόβλεψης του context με βάση το center. Στα δεδομένα πίνακα, όμως, τα γνωρίσματα μιας γραμμής δεν έχουν απαραίτητα σχεσιακή σειρά, οπότε η δειγματοληψία των συμφραζόμενων πρέπει να είναι τυχαία ή να περιλαμβάνει όλα τα υπόλοιπα χαρακτηριστικά του δείγματος.

Στόχος είναι η εκπαίδευση ενός πίνακα ενσωμάτωσης E, από τον οποίο κάθε κατηγορική τιμή του συνόλου δεδομένων αντιστοιχίζεται σε ένα μοναδικό διάνυσμα. Για ένα κεντρικό χαρακτηριστικό (center), το διάνυσμα ενσωμάτωσης e_c χρησιμοποιείται για τον υπολογισμό της πιθανότητας του συμφραζόμενου στοιχείου $p(x_{con}|x_t)$ μέσω softmax. Η softmax εφαρμόζεται πάνω στο γινόμενο του e_c με όλα τα διανύσματα του πίνακα Θ, ο οποίος είναι ένας βοηθητικός πίνακας αναπαράστασης δεδομένων εξόδου. Ο Θ έχει ίδιες διαστάσεις με τον πίνακα E, δηλαδή ένα διάνυσμα ενσωμάτωσης για κάθε στοιχείο του λεξιλογίου. Παρά τη χρησιμότητά της, η softmax έχει υψηλή υπολογιστική πολυπλοκότητα, καθώς απαιτεί εκθετικούς υπολογισμούς για όλες τις λέξεις του λεξιλογίου.

Για να μειωθεί η πολυπλοκότητα του Skip-Gram, χρησιμοποιείται το Negative Sampling [30]. Αντί για softmax σε όλο το λεξιλόγιο, το πρόβλημα γίνεται δυαδική ταξινόμηση: για κάθε ζεύγος center–context λέξεων, επιλέγονται τυχαία μερικά αρνητικά δείγματα (λέξεις διαφορετικές από το πραγματικό context). Έτσι, το δίκτυο εκπαιδεύεται με θετικά ζεύγη (ετικέτα 1) και αρνητικά ζεύγη (ετικέτα 0), χρησιμοποιώντας τη σιγμοειδή συνάρτηση ενεργοποίησης στη θέση της softmax.

1.4.2.1 Προσαρμοσμένο Μοντέλο Ενσωματώσεων

Αναπτύχθηκε ένα προσαρμοσμένο μοντέλο για την εκμάθηση ενσωματώσεων από κατηγορικά χαρακτηριστικά σε πίνακες δεδομένων, εμπνευσμένο από το skip-gram με negative sampling. Τα θετικά ζεύγη σχηματίζονται από τυχαία ζεύγη τιμών διαφορετικών στηλών της ίδιας γραμμής, χωρίς να υπάρχει context window λόγω της απουσίας χωρικής διάταξης.

Τα αρνητικά ζεύγη επιλέγονται τυχαία από όλο το λεξιλόγιο. Για μείωση της πολυπλοκότητας, τα θετικά ζεύγη δεν αφαιρούνται από τα αρνητικά δείγματα, αν τυχόν επιλεγούν, ενώ το negative sampling βασίζεται σε στατική unigram κατανομή υψωμένη στη δύναμη 3/4. Επιπλέον, χρησιμοποιείται ποσοστιαία επιλογή θετικών ζευγών αντί για όλες τις διαθέσιμες τιμές της

γραμμής, μειώνοντας υπολογιστικά την διαδικασία και ενισχύοντας τη γενίκευση των αποτελεσμάτων.

1.4.3 Αρχιτεκτονικές Προτεινόμενων Μοντέλων

1.4.3.1 eGAN

Το μοντέλο eGAN προσαρμόζει το κλασικό Generative Adversarial Network ώστε να λειτουργεί πάνω σε ενσωματώσεις κατηγορικών μεταβλητών, αντί για one-hot ή ordinal κωδικοποίηση. Αποτελείται από δύο μέρη: το Προσαρμοσμένο Μοντέλο Ενσωματώσεων και το τροποποιημένο GAN. Οι κατηγορικές στήλες του dataset χρησιμοποιούνται για εκπαίδευση του μοντέλου ενσωματώσεων, από το οποίο προκύπτει ο πίνακας ενσωματώσεων E. Κάθε κατηγορική τιμή αντιστοιχίζεται σε ένα συνεχές διάνυσμα σταθερού μεγέθους, και έτσι το αρχικό μικτό dataset μετατρέπεται σε αμιγώς αριθμητικό. Έτσι, κάθε εγγραφή μετατρέπεται σε μία νέα με μέγεθος $N'_{total} = N_d * n_f + N_c$, όπου N_d είναι το πλήθος των κατηγορικών μεταβλητών, N_c ων αριθμητικών και n_f η διάσταση των διανυσμάτων ενσωμάτωσης.

Το GAN λειτουργεί πάνω σε αυτές τις νέες εγγραφές. Η γεννήτρια παράγει διανύσματα μεγέθους N'_{total} από τυχαίο θόρυβο, ενώ ο διακριτής διακρίνει τα πραγματικά από τα συνθετικά δείγματα. Μετά την εκπαίδευση, οι ενσωματώσεις αντιστρέφονται: κάθε παραγόμενο διάνυσμα, για κάθε κατηγορική στήλη, συγκρίνεται με τις ενσωματώσεις της στήλης μέσω ομοιότητας συνημίτονου, και επιλέγεται η κατηγορική τιμή με τη μεγαλύτερη ομοιότητα. Καθ' όλη τη διαδικασία, τα αριθμητικά χαρακτηριστικά μόνο κανονικοποιούνται.

1.4.3.2 eVAE

Αντίστοιχα με το eGAN, το eVAE αποτελεί εναλλακτική προσέγγιση για παραγωγή συνθετικών δεδομένων, επεκτείνοντας τον Variational Autoencoder (VAE) σε χώρο εισόδου που μετατρέπεται με ενσωματώσεις. Οι κατηγορικές τιμές μετατρέπονται σε διανύσματα μέσω του Προσαρμοσμένου Μοντέλου Ενσωματώσεων, και το νέο σύνολο δεδομένων με μέγεθος $N'_{total} = N_d * n_f + N_c$ ανά εγγραφή, χρησιμοποιείται για εκπαίδευση του eVAE. Ο κωδικοποιητής αντιστοιχίζει τα δεδομένα σε λανθάνοντα χώρο οριζόμενο από τα μ και σ, ενώ μέσω του reparameterization trick παράγεται το λανθάνον διάνυσμα που περνά στον αποκωδικοποιητή για ανακατασκευή της εισόδου.

Η εκπαίδευση βασίζεται στο κλασικό σφάλμα που χρησιμοποιεί το VAE: το σφάλμα ανακατασκευής, δηλαδή το αρνητικό log-likelihood της εξόδου του αποκωδικοποιητή και η απόκλιση KL, που επιβάλλει στον λανθάνοντα χώρο να ακολουθεί κανονική κατανομή. Όπως και στο eGAN, οι έξοδοι που αντιστοιχούν σε κατηγορικές μεταβλητές αντιστρέφονται σε διακριτές τιμές μέσω της ομοιότητας συνημίτονου με τις ενσωματώσεις των κατηγορικών τιμών κάθε στήλης ξεχωριστά.

1.4.3.3 eDDPM

Το eDDPM προσαρμόζει το Denoising Diffusion Probabilistic Model (DDPM) [28] στο πλαίσιο παραγωγής δεδομένων με ενσωματώσεις. Όπως στα eGAN και eVAE, το Μοντέλο

Ενσωματώσεων αντιστοιχίζει τις κατηγορικές τιμές σε συνεχείς διανυσματικές αναπαραστάσεις, οι οποίες μαζί με τα αριθμητικά χαρακτηριστικά κανονικοποιούνται και διατάσσονται σε ενιαίο διάνυσμα.

Στη εμπρόσθια διαδικασία προστίθεται προοδευτικά Γκαουσσιανός θόρυβος σε πολλά βήματα, σύμφωνα με μια σταθερή κατανομή διακύμανσης. Κατά την εκπαίδευση το μοντέλο εστιάζει περισσότερο σε μεταγενέστερα βήματα, δηλαδη πιο κοντά στον απόλυτο θόρυβο, όπου η μάθηση είναι δυσκολότερη. Στην αντίστροφη διαδικασία, ένα νευρωνικό δίκτυο μαθαίνει να αποθορυβοποιεί σταδιακά το δείγμα εισόδου προβλέποντας τον θόρυβο που του είχε προστεθεί σε κάθε βήμα. Μετά την εκπαίδευση, η παραγωγή ξεκινά από τυχαίο θόρυβο και με αλλεπάλληλα βήματα αποθορυβοποίησης παράγονται συνθετικά δεδομένα, τα οποία μετατρέπονται σε κατηγορικές τιμές μέσω της ομοιότητας συνημίτονου με τον πίνακα ενσωματώσεων.

1.5 Αξιολόγηση

Για την αξιολόγηση της ποιότητας των συνθετικών δεδομένων χρησιμοποιήθηκαν τα σύνολα Adult Income [31] και Mushroom [32] από το UCI Machine Learning Repository, μια από τις πιο γνωστές βάσεις δεδομένων στον χώρο της μηχανικής μάθησης. Το Adult Income επιλέχθηκε λόγω του συνδυασμού κατηγορικών και αριθμητικών χαρακτηριστικών, ενώ στη συνέχεια αφαιρέθηκαν τα αριθμητικά γνωρίσματα ώστε να εξεταστεί η απόδοση των μοντέλων μόνο σε κατηγορικά δεδομένα. Επειδή σε αυτή τη μορφή το Adult Income αποτελεί σχετικά απλή πρόκληση, τα μοντέλα δοκιμάστηκαν επιπλέον στο Mushroom, το οποίο διαθέτει αποκλειστικά κατηγορικά γνωρίσματα υψηλότερης πολυπλοκότητας, προσφέροντας πιο απαιτητική αξιολόγηση.

1.5.1 Διαδικασία Εκπαίδευσης

Η εκπαίδευση των μοντέλων απαιτεί προσεκτική ρύθμιση των υπερπαραμέτρων, με πολλαπλούς συνδυασμούς που δοκιμάζουν διαφορετικές παραλλαγές. Η σταθερότητα και η σύγκλιση θεωρήθηκαν κρίσιμες για την ποιότητα των παραγόμενων δεδομένων. Καθώς οι γενετικοί στόχοι διαφέρουν από αυτούς των κλασικών ταξινομητών, η πρόοδος κατά την εκπαίδευση εκτιμήθηκε όχι μόνο μέσω του σφάλματος σε κάθε βήμα αλλά και με παραγωγή 1.000 δειγμάτων ανά κάποιο αριθμό εποχών, συγκρίνοντας τα με πραγματικά δεδομένα με μετρικές όπως η αποστάσεις Wasserstein και L2 των πινάκων συσχέτισης. Για τα προσαρμοσμένα μοντέλα (eGAN, eVAE, eDDPM) οι συγκρίσεις γίνονταν στον ενσωματωμένο χώρο χωρίς αντιστροφή των κατηγορικών τιμών, ώστε να μειωθεί το υπολογιστικό κόστος.

1.5.2 Μοντέλο Ενσωμάτωσης

Το Προσαρμοσμένο Μοντέλο Ενσωμάτωσης εκπαίδευσε αναπαραστάσεις χαμηλών διαστάσεων (5 διαστάσεων) για τις κατηγορικές μεταβλητές. Για την αρνητική δειγματοληψία (negative sampling) σε κάθε βήμα επιλέχθηκε τυχαία το 30% των δυνατών ζευγών στηλών για θετικά δείγματα, ενώ για κάθε θετικό δείγμα επιλέχθηκαν 3–5 αρνητικά βάσει της ομαλοποιημένης κατανομής unigram. Το μοντέλο βελτιστοποιήθηκε με binary cross-entropy loss πάνω στα θετικά και τα αρνητικά ζεύγη, εκπαιδευόμενο σε ομάδες (batches) των 4.000 για καλύτερη αξιοποίηση της GPU και ταχύτερη σύγκλιση.

1.5.3 Μετρικές Πιστότητας

Για την αξιολόγηση της πιστότητας και χρησιμότητας των συνθετικών δεδομένων χρησιμοποιήθηκε το SDMetrics [33], μια βιβλιοθήκη Python ανοιχτού κώδικα από την ομάδα SDV. Το SDMetrics συγκρίνει πραγματικά και συνθετικά δεδομένα μέσω ποικίλων στατιστικών και μετρικών κατανομών, καθώς το Quality Report προσφέρει μια συνολική εικόνα της ομοιότητας. Περιλαμβάνει δύο βασικές ιδιότητες: Column Shape και Column Pair Trends. Το Column Shape μετρά την ομοιότητα των κατανομών στήλη-προς-στήλη, χρησιμοποιώντας για αριθμητικές τιμές το KSComplement, βασισμένο στη στατιστική Kolmogorov-Smirnov, ενώ για τις κατηγορικές στήλες χρησιμοποιεί το TVComplement που υπολογίζει το Total Variation Distance (TVD).

Το Column Pair Trends εξετάζει σχέσεις μεταξύ δύο γνωρισμάτων. Για αριθμητικά δεδομένα χρησιμοποιείται το CorrelationSimilarity, που συγκρίνει και κανονικοποιεί τις συσχετίσεις πραγματικών και συνθετικών δεδομένων. Για κατηγορικά ή boolean ζεύγη εφαρμόζεται το ContingencySimilarity, το οποίο συγκρίνει πίνακες συνάφειας πραγματικών και συνθετικών δεδομένων με TVD. Έτσι, το Quality Report αποτιμά τόσο την πιστότητα επιμέρους στηλών όσο και τη συνέπεια μεταξύ τους, προσφέροντας μια ολοκληρωμένη αξιολόγηση των μοντέλων.

1.5.4 Αξιολόγηση Προστασίας Δεδομένων

Η μέθοδος μέσου Distance to Closest Record (DCR) [33] χρησιμοποιείται για την εκτίμηση του κινδύνου διαρροής πραγματικών δεδομένων από τα συνθετικά. Το DCR μετρά την απόσταση κάθε συνθετικής εγγραφής από την πιο κοντινή πραγματική. Η τιμή 0 σημαίνει πλήρη διαρροή, ενώ υψηλότερες τιμές δείχνουν καλύτερη προστασία ιδιωτικότητας. Η απόσταση υπολογίζεται με το Gower’s Distance [34], που λαμβάνει υπόψη αριθμητικά και κατηγορικά γνωρίσματα.

Μια δεύτερη μέθοδος είναι ο έλεγχος Exact Matches, δηλαδή ο εντοπισμός εγγραφών του συνθετικού συνόλου που ταυτίζονται πλήρως με εγγραφές του πραγματικού. Ένας μεγάλος αριθμός τέτοιων περιπτώσεων δείχνει πιθανή απομνημόνευση και διαρροή πραγματικών δεδομένων από το μοντέλο. Στόχος είναι μια ισορροπία μεταξύ δεδομένων που μοιάζουν με τα πραγματικά αλλά δεν τα αναπαράγουν αυτούσια.

1.6 Αποτελάσματα

Στις επόμενες σελίδες παρουσιάζονται τα αποτελέσματα αξιολόγησης των συνθετικών δεδομένων που παρήχθησαν από κάθε μοντέλο. Η ανάλυση των δεδομένων έγινε με βάση διάφορες μετρικές ομοιότητας με τα πραγματικά δεδομένα, με έμφαση στη πιστότητα των κατανομών, στις συσχετίσεις μεταξύ γνωρισμάτων και στη ποικιλία των δειγμάτων. Αρχικά, η αξιολόγηση πραγματοποιείται στο μικτού τύπου σύνολο δεδομένων Adult, έπειτα σε εκδοχή όπου αφαιρούνται οι αριθμητικές μεταβλητές ώστε να εξεταστεί η απόδοση σε καθαρά κατηγορικά δεδομένα, και τέλος στα πιο απαιτητικά κατηγορικά δεδομένα του συνόλου Mushroom, που διαθέτει περισσότερες εγγραφές και μεγαλύτερη πολυπλοκότητα.

1.6.1 Μετρικές Πιστότητας

1.6.1.1 Σύνολο Δεδομένων Adult

Synthetic Data Model	Column Shapes Score	Column Pair Trends Score
eGAN	0.95	0.90
eVAE	0.86	0.78
eDDPM	0.93	0.88
CTGAN	0.93	0.85
TVAE	0.90	0.85
TabDDPM	0.98	0.97

Table 1: Αποτελέσματα μετρικών για τα δεδομένα Adult

Στο σύνολο δεδομένων Adult, το TabDDPM σημείωσε το υψηλότερο Column Shapes Score (0,98), ακολουθούμενο από eGAN (0,95) και CTGAN/eDDPM (0,93), ενώ τα VAE μοντέλα είχαν χαμηλότερες επιδόσεις (TVAE 0,90, eVAE 0,86). Τα μοντέλα διάζυσης και τα GANs αναπαράγουν καλύτερα τις κατανομές των στηλών, με τα προσαρμοσμένα μοντέλα που αναπτύξαμε να παραμένουν άμεσα ανταγωνιστικά παρά τις απλοποιήσεις. Οι κατηγορικές στήλες αναπαρίστανται καλύτερα από τις αριθμητικές, με τη στήλη ‘hours-per-week’ να είναι η πιο δύσκολη λόγω ακραίας κατανομής.

Επίσης, το TabDDPM σημείωσε κι το υψηλότερο Column Pair Trends Score (0.97). Παρόλα αυτά, τα μοντέλα βασισμένα σε ενσωματώσεις έμεινα πολύ ανταγωνιστικά, με τα eGAN και eVAE να ξεπερνάνε τα ομόλογα τους σε επίδοση

1.6.1.2 Πλήρως Κατηγορικό Σύνολο Adult

Synthetic Data Model	Column Shapes Score	Column Pair Trends Score

eGAN	0.98	0.95
eVAE	0.90	0.85
eDDPM	0.98	0.96
CTGAN	0.93	0.86
TVAE	0.85	0.77
TabDDPM	0.99	0.97

Table 2: Αποτελέσματα Μετρικών στο Κατηγορικό Adult

Μετά τον καθαρισμό του dataset και την αφαίρεση των αριθμητικών στηλών, το TabDDPM παραμένει πρώτο (0,99). Σημαντική είναι η βελτίωση των προσαρμοσμένων μοντέλων με ενσωματώσεις: τα eGAN και eDDPM φτάνουν 0,98, σχεδόν ισοδύναμα με το TabDDPM, ενώ το eVAE αυξάνει την επίδοσή του στο 0,90. Αντίθετα, το TVAE μειώνεται στο 0,85.

Σε ότι αφορά τα Column Pair Trends Scores, αν και το TabDDPM είναι πάλι πρώτο, τα eGAN και eDDPM είναι εξαιρετικά κοντά του σε απόδοση. Γενικότερα, φαίνεται σημαντική βελτίωση των μοντέλων βασισμένων σε ενσωματώσεις πάνω στα αμιγώς κατηγορικά χαρακτηριστικά.

1.6.1.3 Σύνολο Δεδομένων Mushroom

Synthetic Data Model	**Column Pair Trends Score**	**Column Pair Trends Score**
eGAN	0.90	0.90
eVAE	0.78	0.91
eDDPM	0.88	0.92
CTGAN	0.85	0.88
TVAE	0.85	0.92
TabDDPM	0.97	0.96

Table 3: Αποτελέσματα Μετρικών στο Σύνολο Mushroom

Στην αξιολόγηση του dataset ‘Mushroom’, τοTabDDPM σημείωσε ξανά το υψηλότερο Column Shapes Score (0,98). Όλα τα υπόλοιπα μοντέλα είχαν επίσης πολύ καλή απόδοση, δείχνοντας ότι τα μοντέλα βασισμένα σε ενσωματώσεις προσεγγίζουν με ακρίβεια τις κατανομές πολύπλοκων κατηγορικών χαρακτηριστικών.

Εξαιρώντας το TabDDPM που ξεπερνάει όλα τα εξεταζόμενα μοντέλα, τα προτεινόμενα μοντέλα μας, βασισμένα σε ενσωματώσεις, έχουν εξαιρετική απόδοση, ενώ τα eGAN και eVAE ξεπερνούν τα ομόλογά τους CTGAN και TVAE.

1.6.2 Μετρικές Προστασίας Δεδομένων

1.6.2.1 Μέση Απόσταση από την Κοντινότερη Εγγραφή (mean DCR)

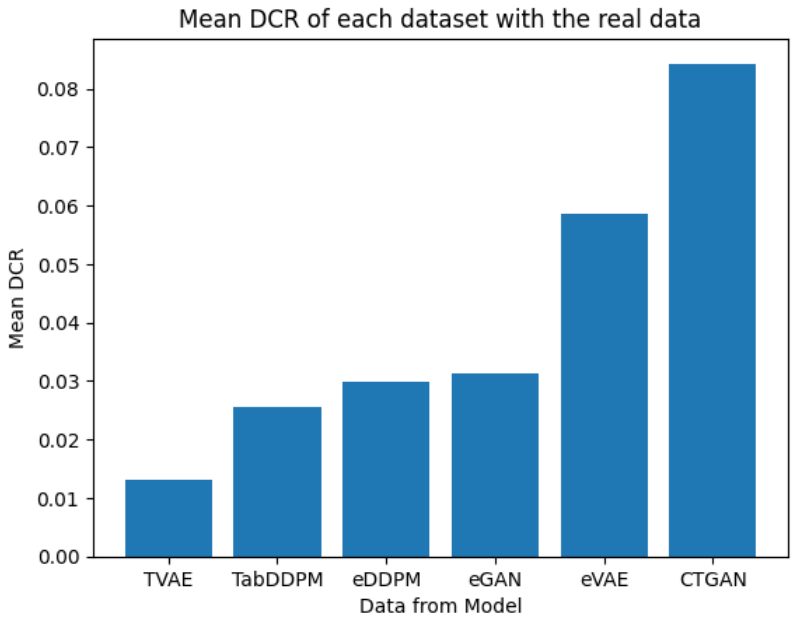


Figure 3: Ιστόγραμμα των μέσων DCR κάθε μοντέλου σε αύξουσα σειρά

Η μέση τιμή του Distance to Closest Record (DCR) δείχνει πόσο απομακρύνονται τα συνθετικά δεδομένα από τα πραγματικά και χρησιμοποιείται ως δείκτης προστασίας ιδιωτικότητας. Το TVAE έχει τη χαμηλότερη DCR (περίπου 0,014), γεγονός που σημαίνει ότι τα συνθετικά του δεδομένα πλησιάζουν πολύ τα πραγματικά, με αποτέλεσμα υψηλό αριθμό ακριβών ταυτίσεων και χαμηλή προστασία ιδιωτικότητας. Το TabDDPM, eDDPM και eGAN έχουν παρόμοια DCR (περίπου 0,03), ενώ το eVAE φτάνει 0,06 και το CTGAN 0,085, υποδεικνύοντας καλύτερη προστασία των ευαίσθητων πληροφοριών στα τελευταία.

1.6.2.2 Ακριβείς Ταυτίσεις

Η χαμηλή μέση τιμή DCR αυξάνει την πιθανότητα ακριβών ταυτίσεων με τα πραγματικά δεδομένα, όπως συμβαίνει στο TVAE, καθιστώντας το ακατάλληλο για προστασία ιδιωτικότητας. Στις μετρήσεις ακριβών ταυτίσεων, η στήλη ‘fnlwgt’ εξαιρέθηκε λόγω της μεγάλης και ποικίλης κλίμακας τιμών της.

Τα υπόλοιπα μοντέλα δείχνουν ικανοποιητική απόδοση: το TabDDPM δεν είχε καμία ακριβή ταύτιση, με μέσο DCR λίγο κάτω από 0,03, ενώ eDDPM και eGAN είχαν περίπου 100 ακριβείς ταυτίσεις με σχετικά υψηλό DCR. Το CTGAN επίσης διατήρησε υψηλό DCR και πολύ λίγες ακριβείς ταυτίσεις, χωρίς να θυσιάζει την ποιότητα της αναπαραγωγής κατανομών.

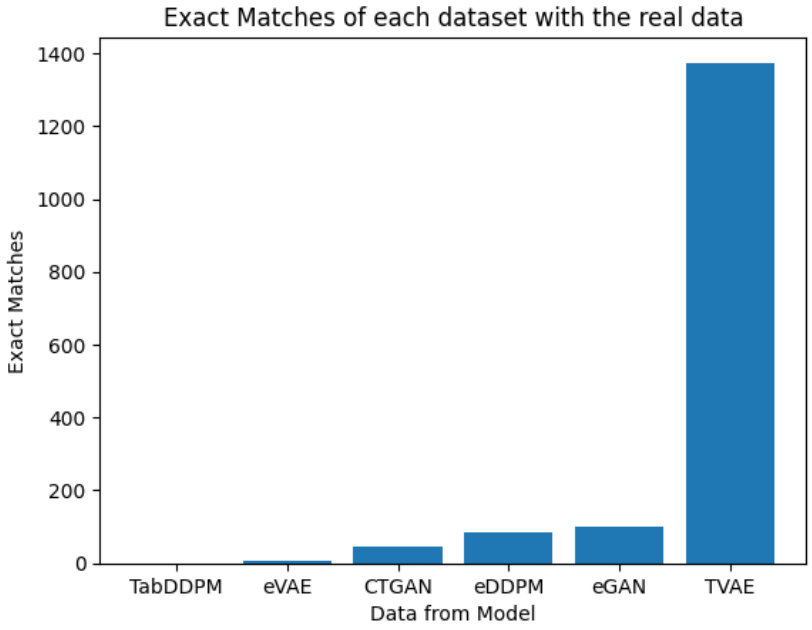


Figure 4: Ιστόγραμμα αριθμού ακριβών ταυτίσεων κάθε μοντέλου σε αύξουσα σειρά

1.7 Συμπεράσματα

1.7.1 Σύνοψη

Ξεκινήσαμε με μια συνολική παρουσίαση της Τεχνητής Νοημοσύνης και της Μηχανικής Μάθησης, φτάνοντας στα πιο σύνθετα πλαίσια των VAEs, GANs και Diffusion Models, αναλύοντας και τα μαθηματικά θεμέλια που στηρίζουν την αρχιτεκτονική και τον στόχο βελτιστοποίησης κάθε μοντέλου.

Σημαντική πρόκληση ήταν η παραγωγή δεδομένων πίνακα, λόγω ετερογενών τύπων (συνεχή, κατηγορικά, δυαδικά) και ανισομερών κατανομών, καθώς και η αποφυγή του mode collapse.

Τα προτεινόμενα προσαρμοσμένα μοντέλα (eGAN, eVAE, eDDPM) αξιοποιούν τα χαρακτηριστικά των βασικών μοντέλων με πιο απλό τρόπο, μετατρέποντας τα κατηγορικά δεδομένα σε ενσωματωμένες αριθμητικές αναπαραστάσεις για να επιτρέπουν την οπισθοδιάδοση σφάλματος με έναν πιο ομαλό και φυσικό τρόπο.

Όλα τα μοντέλα αξιολογήθηκαν στα σύνολα δεδομένων UCI Adult Income και Mushroom χρησιμοποιώντας μετρικές όπως η αποστάσεις Wasserstein και L2 μεταξύ των πινάκων συσχέτισης, δείκτες ποιότητας από το SDMetrics και ευρετικές προστασίας ιδιωτικότητας (DCR και ακριβείς ταυτίσεις).

1.7.2 Αποτελέσματα

Τα αποτελέσματα έδειξαν ότι το TabDDPM υπερέχει σε αναπαραγωγική ικανότητα τόσο των κατανομών κάθε στήλης όσο και των συσχετίσεων μεταξύ χαρακτηριστικών, διατηρώντας ταυτόχρονα υψηλό επίπεδο προστασίας ιδιωτικότητας με υψηλό μέσο DCR και χωρίς ακριβείς ταυτίσεις. Το αντίστοιχο μοντέλο βασισμένο σε ενσωματώσεις, το οποίο προτείναμε, eDDPM, πλησίασε καλά τις κατανομές των στηλών και διατήρησε την ιδιωτικότητα των δεδομένων, αλλά δεν αναπαρήγαγε με την ίδια ακρίβεια τις σχέσεις μεταξύ χαρακτηριστικών.

Τα GAN μοντέλα είχαν επίσης καλή συνολική απόδοση, με το eGAN να είναι δεύτερο καλύτερο, υπερτερώντας ελαφρά του CTGAN στα Column Shapes και Column Pair Trends. Το CTGAN ήταν όμως καλύτερο στην προστασία ιδιωτικότητας με υψηλή μέση DCR και χαμηλές ακριβείς ταυτίσεις.

Τα VAE μοντέλα δεν κατάφεραν να ανταγωνιστούν τα άλλα στο σύνολο δεδομένων Adult. Το eVAE ξεπέρασε το TVAE σε προστασία ιδιωτικότητας, με υψηλή μέση DCR και χαμηλές ακριβείς ταυτίσεις, ενώ η TVAE είχε τη χειρότερη απόδοση σε αυτές τις μετρικές.

Κοινή τροχοπέδη για όλα τα προτεινόμενα μοντέλα ήταν η περιορισμένη ικανότητα αναπαραγωγής συσχετίσεων μεταξύ αριθμητικών και κατηγορικών στηλών. Σε σύνολα δεδομένων μόνο με κατηγορικά χαρακτηριστικά, όπως το Mushroom, τα προτεινόμενα μοντέλα παρουσίασαν πολύ καλή απόδοση, με το eVAE να δείχνει ιδιαίτερα υποσχόμενα αποτελέσματα σε πιο πολύπλοκα δεδομένα συγκριτικά με τα απλούστερα.

1.7.3 Μελλοντική Εργασία

- Βελτίωση μοντελοποίησης αριθμητικών στηλών:
    - Χρήση πιο ολοκληρωμένων ενσωματώσεων, όπου το μοντέλο θα περιλαμβάνει και αριθμητικά χαρακτηριστικά αντί να εκπαιδεύεται μόνο σε κατηγορικά.
    - Πιο σύνθετη επεξεργασία αριθμητικών δεδομένων μέσω πολυτροπικής (multimodal) μοντελοποίησης, όπως Gaussian Mixtures ή Mixture Density Models, όπως κάνουν τα CTGAN και TVAE.
- Μείωση διαστάσεων με ξεχωριστές ενσωματώσεις για κάθε στήλη: Εκμάθηση ενσωματώσεων όπου μόνο τιμές της ίδιας στήλης χαρτογραφούνται στον ίδιο χώρο, διατηρώντας σημαντική σημασιολογική πληροφορία.
- Αύξηση ποικιλίας και προστασίας ιδιωτικότητας: Χρήση πιο ήπιας αντίστροφης μετατροπής από ενσωματωμένα διανύσματα σε κατηγορικές τιμές.
- Transfer learning: Χρήση μοντέλων που έχουν εκπαιδευτεί σε ένα πεδίο και εφαρμογή τους σε άλλο πεδίο με περιορισμένα δεδομένα.

- Ποσοτικοποίηση αβεβαιότητας και σφάλματος ανά δείγμα: Ανίχνευση δειγμάτων που απέχουν από την πραγματική κατανομή, αυξάνοντας την αξιοπιστία στην εκτίμηση ποιότητας δεδομένων.

Chapter 2

Introduction

2.1 Motivation

The growing demand for high-quality data has spurted interest and demand for synthetic data generation. Many techniques have been developed, with technology blooming on certain tasks while slightly trailing in others. Particularly, tabular data has many applications and attract much interest in information technology, however generative models often struggle to manage with privacy, class imbalance, mixed data types and many other challenges such datasets harbor. While generative models such as Generative Adversarial Networks (GANs) [1] and Variational Autoencoders (VAEs) [2] have achieved remarkable success in domains like image, their performance on tabular data (especially categorical-only data) remains limited.

Most tabular data generators resort to one-hot and ordinal encoding [3] to process discrete features. However, these approaches lead to models handling sparse and rigid representations with no substantial meaning and information in their values. Through experimentation, it became clear that when dealing solely with categorical values, such models can significantly underperform. Although that was not the case for all, it sparked an interest in exploring more efficient ways to handle categorical values.

The objective that took form was to make original versions of generative models work naturally on categorical tabular data. Apart from representing categorical values in a meaningful way, it could be very beneficial to take advantage of all raw capabilities of generative models that may be taken away by sparse representations that work only as indexes and do not hold any information about the value they represent. To address this, the idea of using dense, learned embeddings was explored. This is a technique borrowed from natural language processing (NLP), where word embeddings have been repeatedly successfully used to model semantic relationships.

By training a custom embedding model based on co-occurrence between categorical values, every one of them is transformed into a continuous vector in a shared latent space. This representation lets generative models operate in a continuous space enabling backpropagation through the entirety of the network.

To test this approach, experiments were conducted on three popular generative models: GAN, VAE and DDPM (Denoising Diffusion Probabilistic Model). For the purpose of the experiments, we developed three counterparts of the popular base models: eGAN, eVAE and eDDPM. As baseline models for tabular data generation, CTGAN, TVAE and TabDDPM were used. These three models are held on high regard in mixed-type tabular data generation. First, their overall performance on mixed type data was assessed, before continuing experiments on categorical-only datasets.

2.2 Contribution

This diploma thesis presents the following key contributions:

- Embedding Learning for Tabular Data: A novel embedding learning strategy was introduced, inspired by skip-gram with negative sampling adapted specifically for tabular data with no spatial order. The model learns meaningful co-occurrence-based embeddings across categorical columns

- New Generative Architectures: Based on the forementioned embeddings, three new models were developed:
    - eGAN: a GAN-based generator operating on embedding vectors along with numerical attributes
    - eVAE: a VAE model that reconstructs embedded samples and decodes them back to categorical values
    - eDDPM: a DDPM model, also trained on mixed-type data that reconstructs the embedding vectors back to categorical values.

- Evaluation on Mixed and Categorical-Only Data: Experiments were performed on mixed-type and categorical-only versions of the Adult Income and the Mushroom datasets of the UCI repository, using multiple privacy and fidelity metrics.

- Insight and Potential: The results of the models are assessed on these two different dataset types. Conclusions are drawn and potential future work on these specific topics is highlighted.

Chapter 3

Foundations

3.1 A Brief History and the Limitations of Data Science

3.1.1 Computer Science and Artificial Intelligence

Ever since its creation, Computer Science has always been a term unifying many fields of technology spanning from the early days of human history up to this date. It contains theoretical disciplines such as algorithms, information theory and computational theory but it also encompasses applied disciplines like hardware and software engineering. Despite including numerous different fields, each one of them has been closely related to others in order to construct every computing machine found in the modern world.

The truth is, though, that it was never easy for most of the theories to come to life. The ambitions of many computer scientists had to remain only in theory due to other fields of technology not having caught up with their inventions. Such examples are public-key cryptography, parallel computing, peer-to-peer networks and many more. Apart from theory, material engineering, power resources and therefore the so-called computing resources needed to be available. So, many times in history, groundbreaking discoveries have been put on hold.

Artificial Intelligence (AI) was first established in the 1950s but it was not until around the beginning of the 21st century that it attracted extremely growing attention. The past two decades have seen rapid growth in the ability of network and mobile computing systems to gather and transport vast amounts of data, a phenomenon often referred to as “Big Data” [4]. Due to the rise of Big Data and generally the evolution of computing power, that has enabled high resource-demanding data processing, developing practical software for computer vision, speech recognition, natural language processing, robot control and many other applications are now relatively easily and effectively achievable. These applications are products of Machine Learning, a field under the umbrella term of AI.

The rise of AI has been very steep in the last two decades and its applications and potential have just started to unfold. With more and more organizations, companies and individuals putting

effort and money into this field, we can only expect to see many more significant breakthroughs in the coming years. Or maybe not?

3.1.2 Limitations Regarding AI and ML

The most trending and commercialized AI products for the past two years have been Large Language Models (LLMs), or models designed for Natural Language Processing (NLP). However, the industry could be hitting some major bottlenecks. One of them is related to the tremendous amount of energy needed for these models to be trained and to operate. Research on several common large AI models showed in 2019 that their training can emit more than 626,000 pounds of carbon dioxide, which is equivalent to almost five times the lifetime emission of a car [5]. These experiments were on BERT with 110M parameters and other major transformers of around 273M parameters as well as GPT-2, which was even more source demanding. The emissions and the demand in energy and resources of the recent models are much more significant as their parameters can reach up to almost 2 trillion.

3.1.3 Computer Power bottleneck

As mentioned before, the neural networks, which were theorized in the early days of artificial intelligence, demanded computational power that was not available back then. For example, when it was first introduced in 1958 by Frank Rosenblatt from Cornell University, even the most basic application of the single perceptron needed computational resources that could not be achieved with the technology available at that time.

To fulfill the demands of recent models, developers have widely resorted to Graphics Processing Units (GPUs) since the early 2010s. GPU hardware can perform concurrent operations occupying multiple cores, thus accelerating the computational procedures regarding Machine Learning. In early testing, GPUs showed around 10 to 15 times more speedup compared to CPUs, and even higher in certain experiments [6]. However, to train recent large models that can reach up to trillions of parameters and need multiple TBs of training data, current available computing resources have started becoming a bottleneck again.

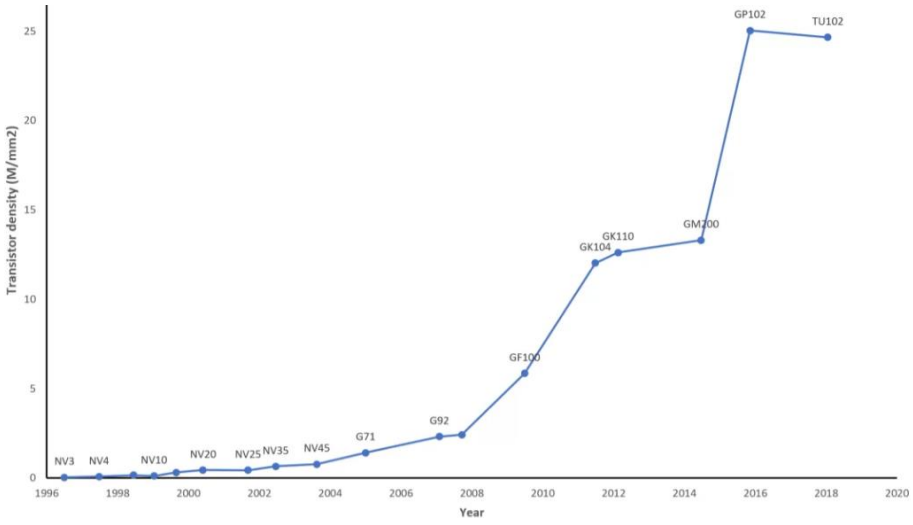


Figure 5: A chart of NVIDIA main GPU transistor density growth until 2020

3.1.4 Need for training data

Machine learning is based on processing data to learn patterns, make predictions, and generalize well to new, unseen scenarios. The growing need for more Machine Learning applications and the increasing complexity and accuracy that is demanded in order to meet the expectations has raised another vital necessity for the field of AI. That is, the demand for more data. AI developers, organizations and companies have struggled in many cases to find satisfying amounts and good quality of training data for many applications. It can be very hard to obtain the dataset that is needed to train a specific model for reasons such as difficulty in collection, privacy policies, limited samples and many more. Therefore, it is a concern for anyone trying to develop new work on Machine Learning and it is a problem that needs a total or partial solution in every single case.

3.2 Challenges of collecting real data

3.2.1 Privacy

In recent years, with the rise of Machine Learning and its deep connection to datasets, there has been an increasing drive to acquire personal data. This demand arises from the necessity of having substantial and high-quality datasets to effectively train ML models [7]. Developers and organizations striving to gather data and personal information to enhance their models often face significant challenges. They can find themselves violating privacy or copyright laws if they are not careful.

Globally, various regulations impose restrictions on the collection, storage, and utilization of personal data. These regulations require organizations to obtain explicit consent from individuals before collecting their data and ensure that individuals' data rights are protected. As a result, acquiring large volumes of high-quality personal data has become a major challenge for companies aiming to deploy effective ML applications. Navigating these legal constraints while attempting to obtain and use personal data for training ML models requires a complex solution as well as careful consideration and adherence to regulatory standards.

3.2.2 Bias

Furthermore, real-world data appears to be biased in many cases, thus resulting in models that attempt decision making based on false norms, as they have not been trained on totally fair and balanced data distributions. Bias in data can mostly origin from two different scenarios. First, training data may not accurately represent reality or second, it may accurately represent a reality that has been altered by external factors, for example chronically discriminating factors [8].

For the second case, we can consider, for example, a hospital that mainly receives patients of a certain economic situation and standard of living. The data that this hospital has collected from its patients is true but does not represent the whole and general reality. So, training models using this data would lead to very specific and untruthful results.

Last but not least, biases don’t only come from the forementioned reasons. There can be mistakes in data collection that do not actually have to do with the validity of the actual data. For example, having humans annotating data and using their own judgement to an extent can eventually lead to biases on the training dataset. Collection errors or slightly wrong data preprocessing can also lead to small biases on datasets. Even these small divergences from absolute reality can amplify the biases in model predictions [9].

3.2.3 Abundancy and liability of data:

We live in the era of Big Data and the availability of most kinds of data is almost never an issue. In some cases of Machine Learning though, the abundance and even the availability of specific data is not a certainty. Consider the need for developing an autonomous vehicle that obviously needs to not hit pedestrians. This entails the need for data of vehicles hitting and avoiding pedestrians. Acquiring sufficient amount of data for this task would be extremely hard, let alone the inconsistencies due to different sources [8].

Additionally, the liability of hand labeled data is highly at question. Consider an individual that labels data manually day by day for several hours straight. It would be practically impossible not to make mistakes. Also, these workers often work under poor conditions and are severely

underpaid [10]. So, there is another concern surrounding the use of real data in artificial intelligence i.e., an ethical concern.

3.2.4 Human labor and ethics

The rise of AI and its overwhelming name has always had people afraid and wondering, at least at some point in their lives, about negative effects and the chances that all this can go out of hand. The actual skepticism of regular people away from this field though is most of the time mainly speculative, baseless and invalid. Even the news and social media present AI as one thing or another: the door to a future utopian world where technology has brought prosperity everywhere or a dystopia created by losing control of AI and letting it take over humanity.

However, the truth is far away from these concerns and scenarios. It is true that AI can be a huge tool for humans and improve many aspects of life but the thing we need to focus on is that there are a great number of problems already created by it. A part of them is about human labor and exploitation happening every day.

Behind the scope and the purpose of technological advancements, corporations are exploiting the global workforce and overstepping legal boundaries. As Machine Learning models are heavily reliant on vast amounts of labeled data, companies are putting people to work in terrible conditions trying to deliver as a group in which everyone has a very specific repeating task. These companies are treating workers like gig-workers and automated machines with the least wage possible [10].

3.3 Techniques regarding data generation and privacy preservation

3.3.1 Federated Learning

One of the ways to tackle problems regarding privacy issues of machine learning is federated learning. Federated learning trains a model using a multi-node cluster. Consider the master node, meaning the main entity of the cluster and the node that eventually has the functional model. The master node has no access to the training data. On the other hand, the data is available to all the worker nodes.

The worker nodes perform gradient descent and the learning process of machine learning. Then, each worker node serves its resulting model to the master worker. The master worker aggregates the results of all the workers resulting in the final model [11]. The main downside of this method is that the model created is non-versatile giving limited options apart from its very specific purpose.

3.3.2 K-anonymity

Another technique that deals with privacy preservation is k-anonymity. It follows a very different approach compared to federated learning as it straight up manipulates the data. K-anonymity, in general, groups similar records and generalizes sensitive attributes [12]. However, such techniques can have major disadvantages. The main goal when using k-anonymity is to ensure the best tradeoff between the utility of the final manipulated data and its information disclosure risk. Generalizing to a certain degree can still not be enough to preserve private information. On the other hand, more generalization, especially on information with granular detail, will lead to poor representation of reality and model accuracy.

3.3.3 SMOTE – Synthetic Minority Oversampling Technique

In some real-life machine learning cases, where data availability is a critical issue, it is substantially impossible to acquire more data for a particular class. This imbalance can lead to major inaccuracies in the model's performance. One way to work around this problem is to under-sample the majority class with the danger of compromising vital information. Another way is to over-sample the minority class, a technique that involves the danger of overfitting due to the fact that same samples are contributing repeatedly. This is the problem that SMOTE is trying to deal with [13].

In General, SMOTE computes the difference between a sample and its nearest neighbor. Then takes the difference and multiplies it with a random number between 0 and 1. After, it adds the resulting number to the initial sample and creates a new one. The process continues to the next nearest neighbors.

However, this model is missing a way of handling categorical values. In presence, of categorical values it is more complicated to find the nearest neighbors and when generating a new sample, the categorical attributes cannot be the weighted average of the corresponding examples because there might be generated a new categorical value that does not exist. SMOTE-NC (Synthetic Minority Over-Sampling Technique – Nominal Continuous) uses a straightforward technique to tackle this problem and handle both continuous and categorical features [13], [14].

After identifying all the minority samples, SMOTE-NE computes the standard deviation of all the continuous attributes. Then, it calculates the median value of all the standard deviations. To find the nearest neighbors of a sample while taking categorical features into consideration, the Euclidean distance is calculated based only on the continuous values available at first. If there is a mismatch on a categorical feature between two samples, the median value of the standard deviations that was previously computed is added to their Euclidean distance. Once finding the nearest neighbors, the synthetic samples are generated using the standard (vanilla) SMOTE

algorithm on continuous attributes, whereas for the categorical feature, the most frequent value of the nearest neighbors is used.

3.4 Synthetic Data

3.4.1 What is synthetic data

Synthetic data refers to artificially generated data that mimics real-world data but is created algorithmically rather than being collected from actual observations. The goal is to retain statistical properties of the original dataset, allowing for analysis and testing without compromising the privacy or confidentiality of sensitive information [15]. The crucial advantage of synthetic data is that it is impossible to acquire real data and sensitive information from it.

Synthetic data is particularly valuable in scenarios where access to real data is limited, expensive, or ethically problematic, enabling researchers, developers, and analysts to train machine learning models, conduct simulations, and perform other data-driven tasks with reduced risks and constraints. Many companies with access to real data aim to create synthetic data with the characteristics that enable developers and other companies to extract useful information or train models without giving them any sensitive information. As privacy regulations arise in many cases, truth holding synthetic data has become a big project in today's market, especially for medical data.

Chapter 4

Theoretical Background

4.1 Basic Principals of AI and ML

4.1.1 Introduction

Artificial Intelligence is the simulation of human-like intelligence in machines, enabling them to perform tasks such as learning, reasoning, problem-solving, and decision making. AI systems apply algorithms on data to recognize patterns, make predictions, and improve over time. It encompasses various subfields, including machine learning, natural language processing, and computer vision. AI aims to create systems that can operate autonomously or assist humans in complex tasks.

Machine learning is a subset of AI, probably the biggest, that focuses on developing algorithms and statistical models that enable computers to learn from and make predictions or decisions based on data. AI is based solely on programming rules. On the other hand, ML algorithms identify patterns in data and improve their performance over time as they are exposed to more data points. The ability to learn from repetition, mistakes and successes makes ML a powerful tool for a wide range of applications, from image recognition and natural language processing to recommendation systems, predictive analytics and decision making. In essence, ML drives much of the progress in AI by allowing systems to adapt and optimize their behavior autonomously.

4.2 The Perceptron

4.2.1 Inspiration from Nature

Throughout history, most of the technological breakthroughs have come through inspiration from nature. Humans have always been finding new ways to advance and make their lives easier by mimicking explicit traits and abilities observed in their environment. Machine Learning and Neural Networks are no different from all those inventions, as the name of the latter implies. The

study of the human brain and the way it functions ignited the concept of computer neural networks. If the human brain learns and operates using signals and pulses, then a computer system could copy this way and learn things itself.

Neurons are the fundamental units of the nervous system. They operate by transmitting electrical and chemical signals and consist of three main parts: the cell body, the dendrites and the axon. Dendrites receive input signals from other neurons. These signals are then processed in the cell body and if they are strong enough, it transmits a new electrical pulse through the axon to other neurons. The neurons communicate with each other through synapses which are connections between axons and dendrites forming a whole nervous system.

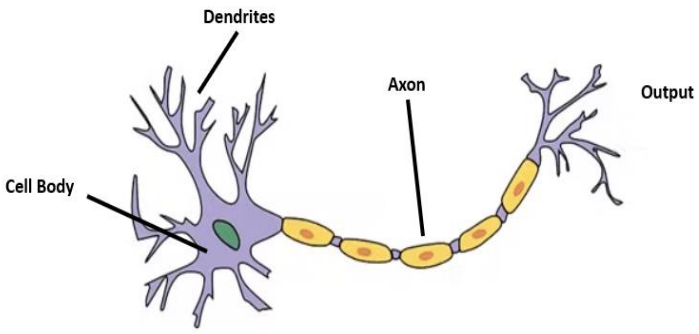


Figure 6: Human Neuron

4.2.2 From nervous system to Neural Networks

Neural Networks are implementing the model of the human brain into computer systems, making it possible for computers to, in a sense, learn and act based on inputs and stimulation. Breaking down a Neural Network, at the base of it is found the perceptron. We think of the perceptron as neuron. It can take input values, apply weights to them and transfer them to its main body, which we can call node. In the node the values are added up forming a new value. This new value is then given as input to a function called activation function. The output of the activation function is the output of the perceptron.

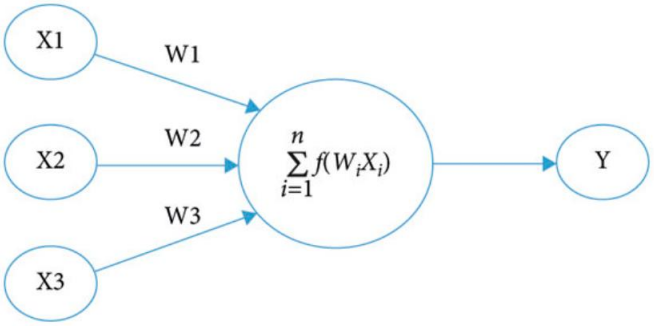


Figure 7: Single Layer Perceptron

For the perceptron to be able to mimic the behavior of a neuron and form a structure able to learn when combined with other nodes, it must receive and produce outputs that resemble the electrical signals that the brain uses. So, it is the most common tactic to use activation functions that produce values between 0 or -1 and 1. However, in many cases, it is very beneficial to just collapse all negative values close to zero. In general, most of the time, a single node takes the input and passes it through a function that adjusts it on a new scale. More details on activations functions will be discussed later.

After constructing the fundamental unit, if put together correctly, many units can create a system that can mimic the behavior of the nervous system and perform complex actions. Neurons are put together in a way that the nodes are organized in layers. The input signal can be of any complexity and form based on the design of the network. Input values are received first, of course, on the input layer. They are then weighted and passed onto the first layer of nodes, the first hidden layer. Depending on the architecture of the network, some input values can be passed to not all but certain nodes. However, it is common to pass all input values to all nodes forming a fully connected Neural Network.

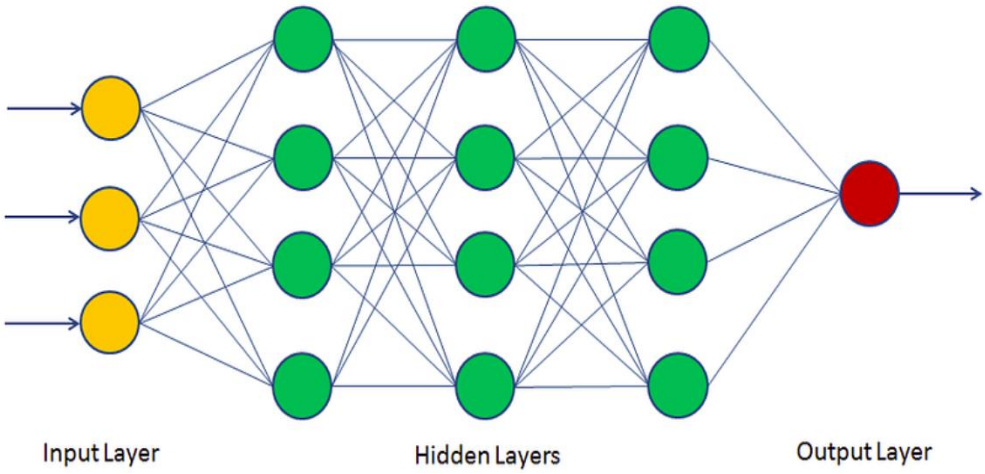


Figure 8: Fully connected neural network

Supposing a Neural Network has been constructed, how does it learn? The structure of a Neural Network resembling a nervous system can produce outputs in a complicated way, but it is not useful unless it is tuned to make these predictions based on certain patterns. Multiple different inputs are fed into the network several times, repeatedly.

4.3 MULTILAYER NEURAL NETWORKS

4.3.1 Backpropagation

Let's consider a multilayer perceptron. First, the network is randomly initialized, meaning the weights that are applied on the values on every layer are random. It is only logical that the model produces completely irrelevant predictions with these random weights. After making predictions for all the inputs, the performance of the model is assessed based on specified metrics. The model then takes feedback from the metrics and changes its weights at every point in a way that next time, it makes more successful predictions. However, in the last sentence lies the most important and difficult part.

The forementioned procedure is called backpropagation. After acquiring the outputs of the network $\hat{Y}$, we compare them to the real target values Y and calculate the error E. The calculation of E is another crucial part determining the behavior of a neural network as it essentially indicates what it is that we actually want the network to approximate. Next, the derivatives of E w.r.t. all the weights are calculated. These give important information as they are the quantities that guide the update of the weights. For example, if the increase of a weight resulted to the increase E, then it would yield better results if we updated this weight with a smaller value and vice versa [16].

After updating all the weights of the network, or at least the ones that took part in forming the output, the process is repeated. The number of repetitions must be sufficient to minimize the error and bring the network in convergence, for it to give meaningful outputs.

4.3.2 The chain rule

The output, hence the error of a neural network, derives from the input passing through a series of multiplications, with the weights of the network, and activation functions. So, we can imagine the final result as a composite function made up of the multiplications and the activation functions stacked the one after the other. To calculate the derivative of the loss with respect to the input, or a certain weight, the concept of chain rule is applied.

Given the composite function $f_1 \circ (f_2 \circ \dots (f_{n-1} \circ f_n)$ using the chain rule, we can compute the derivative of f_1 w.r.t. input x as

$$\frac{df_1}{x} = \frac{df_1}{df_2}\frac{df_2}{df_3}\dots\frac{df_n}{x} \tag{4.3.1}$$

This enables us to track the effect of a weight all the way down to the output taking into account all intermediate calculations. Expanding and applying this concept into a neural network we can get the example of the following figure.

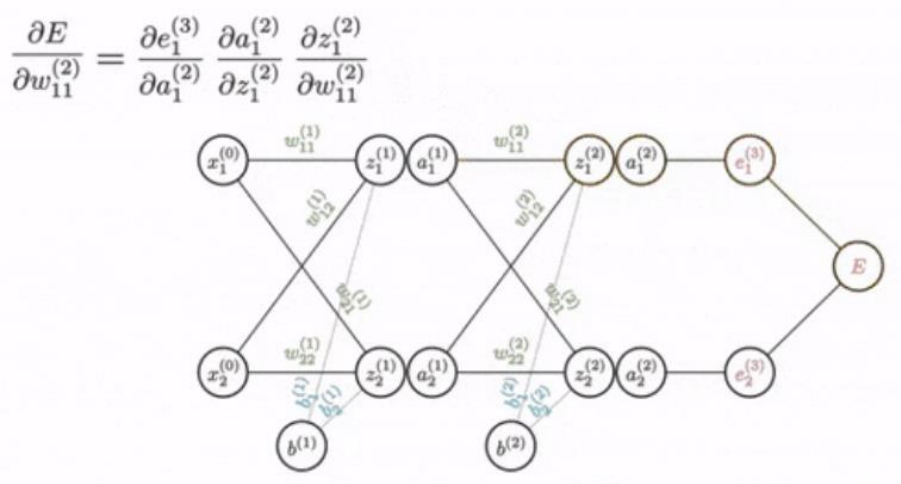


Figure 9: Chain rule for derivative calculation on neural network

4.3.3 Activation Functions

Activation functions are essential components of neural networks. They are attached on the neurons and basically determine whether the neuron should be activated or not by transforming the weighted sum of its inputs into an output signal. They introduce non-linearity as just with sums of multiplications the network would not be able to do that alone. Using activation functions, the network is able to learn to perform complex modeling.

However, as previously shown in backpropagation and chain rule, the activation function has to be differentiable in order for the network to be able to compute the derivatives at each step. Without differentiable activation functions, gradient descent would be ineffective.

An activation function can basically be any imaginable differential function as long as it lets the network learn by the mapping it offers. The best choice for them depends on the network architecture, the domain of the data and probable problems such as potential vanishing or exploding gradients. The most commonly used activation functions are the following:

- ReLU: $f(x) = \max(0, x)$

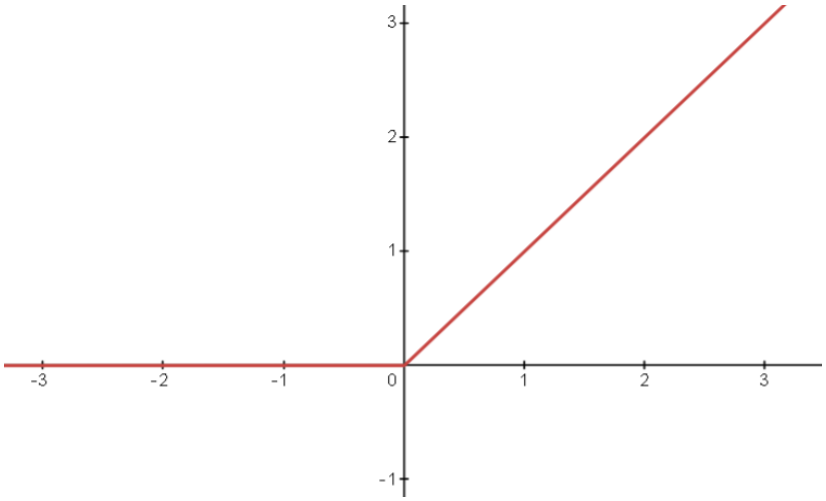


Figure 10: ReLU Activation Function

The Rectified Linear Unit is the most used activation function in neural networks. Even though it Is not differentiable at 0, we assume that its derivative there is equal to 0. This simple function introduces nonlinearity to the network and solves the vanishing gradients problem. However, in some cases, much information can be lost since it does not take into account any negative value. To avoid this, the Leaky ReLU can be used, which is defined as $f(x) = \max(ax, x)$ for small values of a. This function keeps some small information about the negative values as well.

- Sigmoid Activation Function: $f(x) = \frac{1}{1+\exp(-x)}$

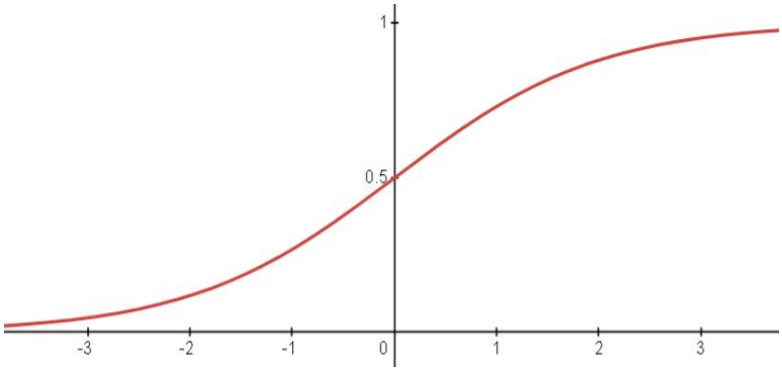


Figure 11: Sigmoid Activation Function

This activation function maps all its inputs between the 0 and 1and is mostly used for binary classification tasks. When using a neural network for such task, the output must indicate the probability of the input belonging in one class or the other. The sigmoid function offers just that as it is also smooth and differentiable. However, in deep networks, it suffers from vanishing gradients. As values begin to settle close to 0 or 1, the gradient becomes smaller step by step, in contrast to values close to 0 where the gradient is significantly larger.

- Hyperbolic tangent: $f(x) = \tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$

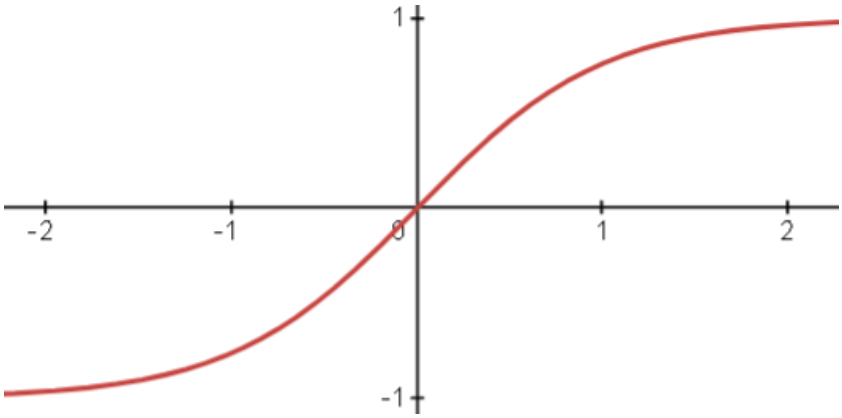


Figure 12: tanh Activation Function

Even though human brain neurons work by sending a signal or not, through the years researchers have found that computer neural networks learn better by keeping neuron outputs centered to 0, returning both negative and positive signals. That is why tanh is a useful activation function, often passing much more information into next layers than other ones. Data is sometimes normalized with mean value 0 which can help the network even more when tanh is used. Even though tanh has stronger gradients than sigmoid and helps the network learn faster, it also suffers from the vanishing gradients problem as values converge close to -1 and 1.

- Softmax Activation Function: $f(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$

Softmax is an extremely popular and useful activation function. It converts a set of prediction scores (logits) into probabilities and all elements of the softmax sum up to 1. It is mostly used for multi-class classification problems as it gives the probability of the input belonging to each one of the classes in a computationally differentiable way.

4.4 Machine Learning Methods

4.4.1 Supervised vs Unsupervised Learning

Machine learning can broadly be categorized into two learning techniques; supervised and unsupervised. These two concepts differ on how they manipulate data and the type of data they learn from. Understanding the differences between these two, the advantages and disadvantages and, hence their applications, is a crucial step to beginning creating a model.

Supervised learning is basically defined as the method of learning from labeled data. That means that for every data point, which can vary from a single number to a whole image, there comes a corresponding output label.

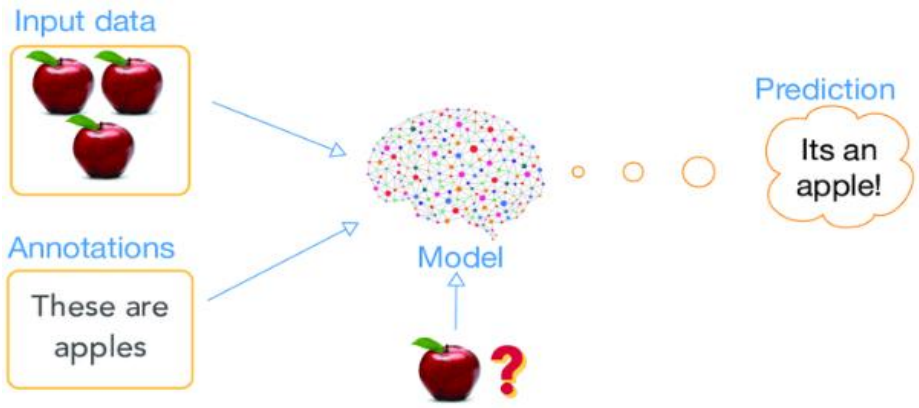


Figure 13: Supervised Learning

The concept is that the model makes predictions given the input data. These predictions are later compared to their corresponding correct labels to acquire the error. The network then updates its parameters based on the error that was calculated at each step of the process.

Unsupervised learning is, opposed to supervised, the concept of learning from unlabeled data. This method refers to tasks such as grouping together data points within a dataset or reducing the dimensionality of the data. Essentially since unlabeled data do not give any clear objective, these algorithms aim to examine and manipulate them in order to extract relationships and patterns within them [17].

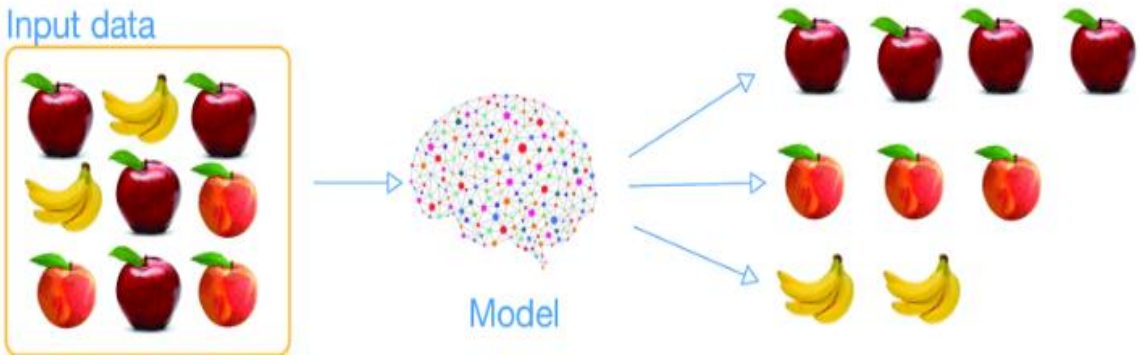


Figure 14: Unsupervised Learning

One can choose a method based on the available data, whether it is labeled or not. However, identifying the nature of the objective and choosing a model that best solves it is the most important. The synthetic data generation task that is analyzed here is one with much complexity and there is no definitive approach. Even though unsupervised learning seems to be ahead, propositions and attempts for supervised learning have also been made successfully.

4.4.2 Probabilistic vs Deterministic Modeling

Apart from the learning method and the type of input data, in order to create better performing models we have to take a look at different modeling techniques from another angle. Another factor that separates machine learning models into two categories is the uncertainty that each model incorporates. Based on whether a model applies randomness in calculating the output or not, we can classify it as deterministic or probabilistic.

Deterministic models produce the same output every time, given the same input. Their results are fully determined by the input and the parameters and equations that comprise the model. Such characteristics are useful for modeling well defined relationships where results need no variability or diversity and require precise predictions.

On the other hand, probabilistic models incorporate a good tradeoff between precision and randomness. Instead of outputting results coming from fixed values, they often model layer outputs as probability distributions. In this way, given the same output, probabilistic models' results can come from a diverse continuous set of options.

For synthetic data generation, one of the most important aspects regarding the quality of the results is the diversity of the output data. It is critical for a model to be able to produce outputs around different data points and not be confined to generating the same results given the same input. This makes it very hard for deterministic approaches to perform well on generating tasks.

There have been many efforts to bridge this gap when using deterministic models, but such methods admittedly lean towards probabilistic approaches regardless of their starting point.

4.5 Basic Data Distributions and Probabilities Background

4.5.1 Distributions and Data Representation

Synthetic data quality is measured on the amount of information it inherits from real data. It needs to be diverse and accurate across data points as well as within individual data points. A very important factor contributing to generating good quality synthetic data is incorporating probability theories in the model's generative and learning process.

Most state-of-the-art generative models fall into the category of probabilistic models. Regardless of the end result and architecture, one of the most crucial tasks is approximating complex probability distributions. Probability theory provides the mathematical framework for modeling uncertainty and randomness.

The most common form of distribution observed in data is the Normal Distribution, alternatively referred to as Gaussian. Normal Distribution ($N(\mu, \sigma)$) is a probability density function (pdf) is a continuous probability distribution defined by two parameters: mean μ and standard deviation σ. The values of a random variable X that follows a Normal Distribution, meaning $X \sim N(\mu, \sigma)$, is centered around its mean value. Near the mean is where the data is more frequent. However, the standard deviation affects how dense the data around the mean is as well as how far from the mean we can observe substantially occurring instances.

Mathematically, normal distribution's pdf is defined as

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{4.5.1}$$

Mean μ sets the point at the x axis where the values are centered. The biggest the mean is, the more far right on the x axis the graph will be. The standard deviation is responsible for the density of the values around the mean and how spread they are across the x axis. As σ gets smaller, more values are observed around the mean and less away from it. This means that the curve of the pdf is pointier and taller as opposed to a larger σ where it becomes smoother and flatter.

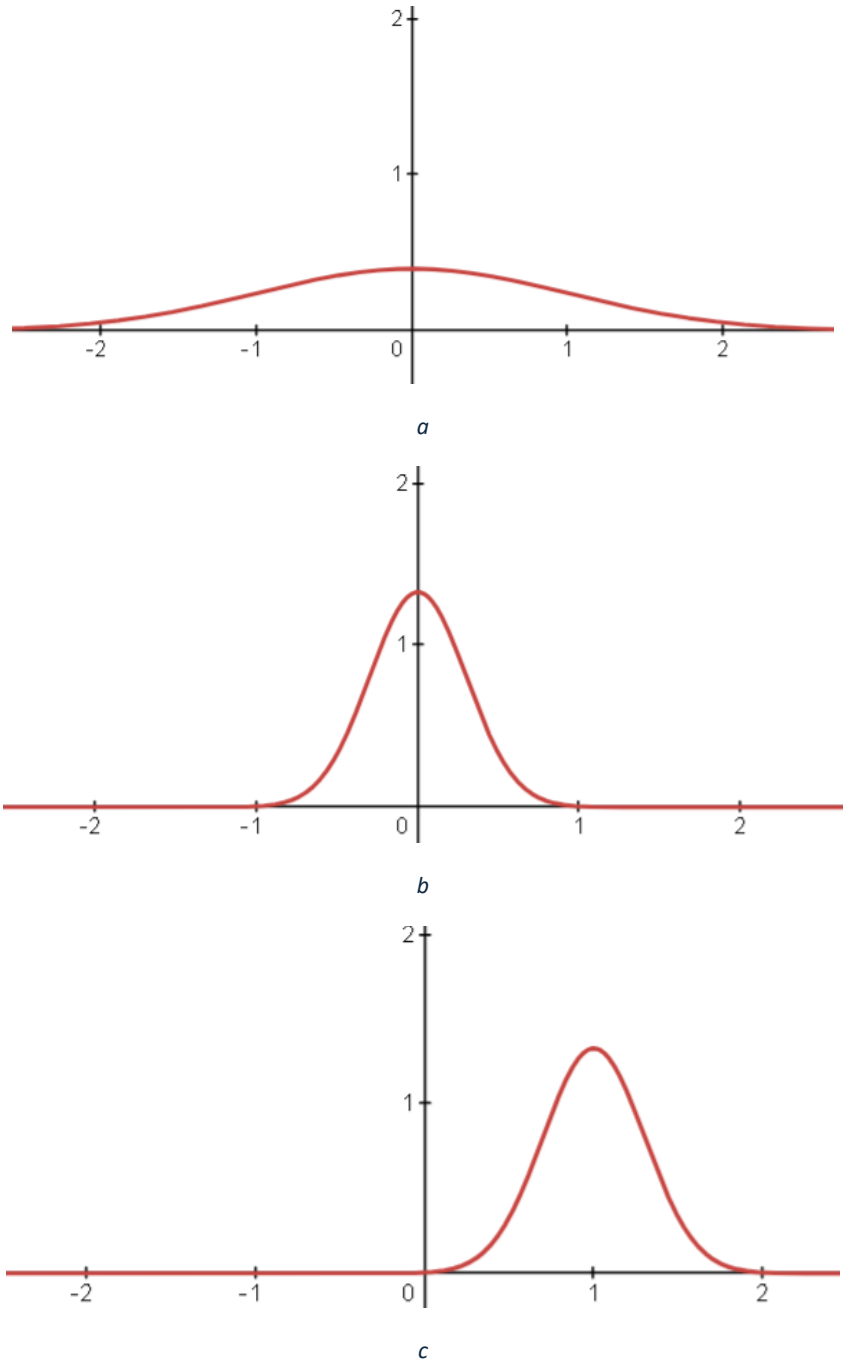


Figure 15: Different examples of Gaussian Distribution. (a) μ = 0, σ = 1.0 (b) μ = 0, σ = 0.4 (c) μ = 1.0, σ=0.4

4.5.2 Multimodal Distributions

In reality, though, data are rarely accurately represented by a simple Gaussian. Using a single Gaussian can be effective as it keeps the complexity low and yields sufficient results in most cases. Seeking better results in probabilistic generative models mainly means trying to approximate

intermediate or final distributions in a more complex way. As a result, the single Gaussian approach is replaced by other distribution approximations. Keeping it as simple and effective as possible, most methods do not deviate much from the unimodal Gaussian Distribution as they just try implementing multimodal gaussian approximations.

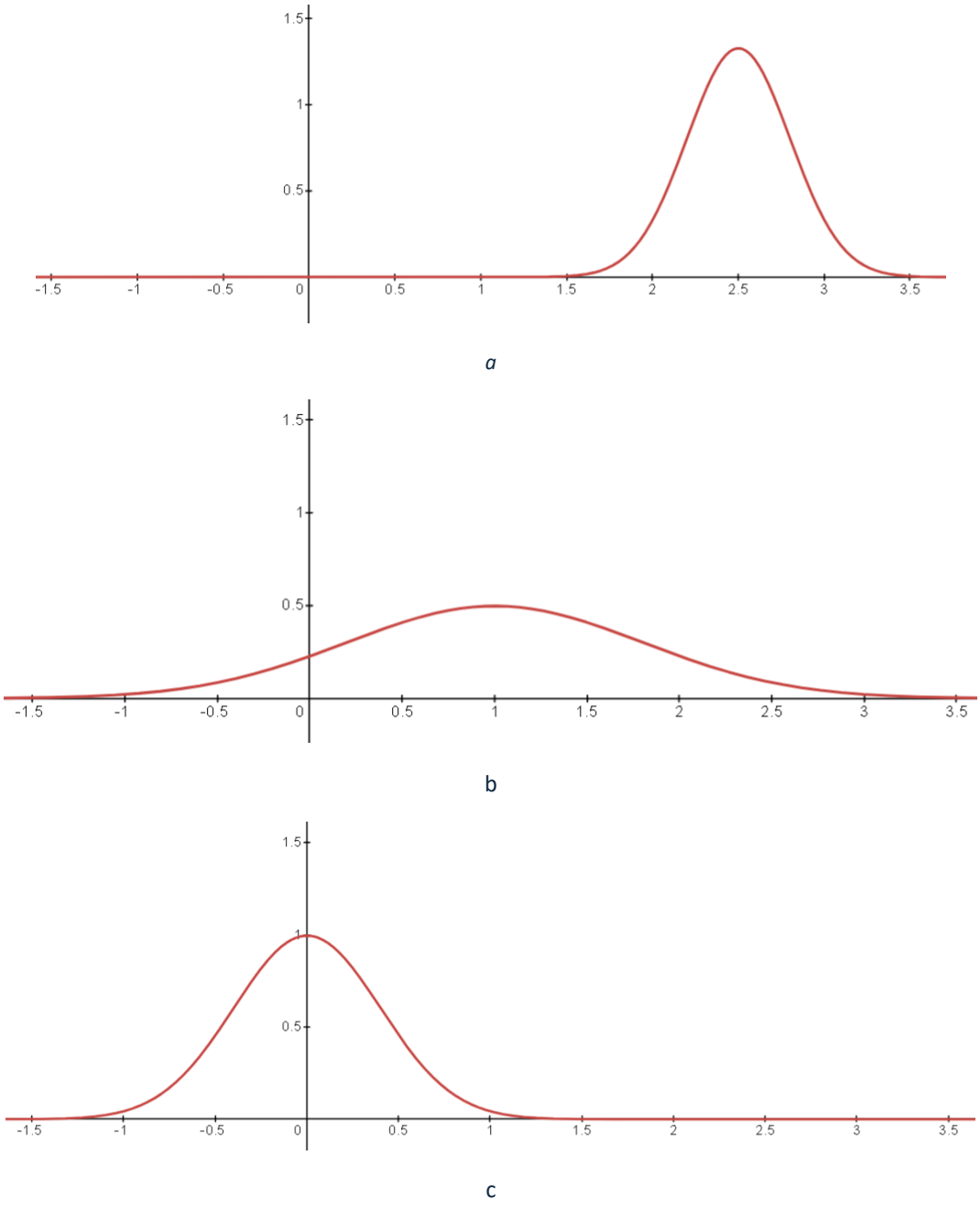


Figure 16: Different unimodal Gaussian Distributions (a) μ = 2.5, σ = 0.3 (b) μ = 1, σ = 0.8 (c) μ = 0, σ = 0.4

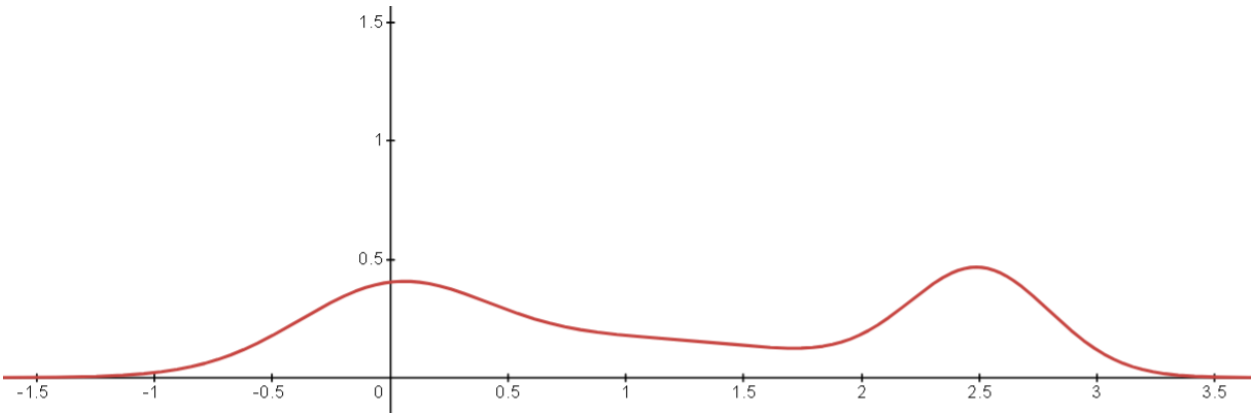


Figure 17: Multimodal Gaussian Distribution deriving from the 3 Gaussians of figure 11 with equal weight coefficients, 0.33 each.

4.5.2.1 Gaussian Mixture Model

The Gaussian Mixture Model (GMM) is a probabilistic unsupervised learning model. It is represented by a weighted sum of a fixed number of Gaussian Distributions and uses the Expectation-Maximization algorithm for training [18]. The pdf of the resulting distribution is

$$p(x) = \sum_{\mathrm{i}}^{\mathrm{K}} \varphi_i N(x|\mu_i, \sigma_i) \tag{4.5.2}$$

where φ_i are the mixture weights.

For training, we consider K gaussian components and analyze the process as an iterative process. Given a training vector x_t, the probability that x_t belongs to the ith gaussian component is calculated as

$$\Pr(i|x_t) = \frac{\varphi_i N(x_t|\mu_i, \sigma_i)}{\sum_j^K \varphi_j N(x_t|\mu_j, \sigma_j)} \tag{4.5.3}$$

At each step, the estimation of each mixture weight φ_i is the average of the probabilities that a training vector belongs to its gaussian $N(\mu_i, \sigma_i)$. So, the estimated value is

$$\bar{\varphi_i} = \frac{1}{T}\sum_t^T \Pr(i|x_t) \tag{4.5.4}$$

The estimated mean value of component i is the weighted average of all the training vectors

$$\bar{\mu_i} = \frac{\sum_t^T \Pr(i|x_t)\, x_t}{\sum_t^T \Pr(i|x_t)} \tag{4.5.5}$$

And similarly, the estimated variance of component i becomes the weighted variance of all training points

$$\bar{\sigma}_i = \frac{\sum_t^T \Pr(i|x_t)\,(x_t - \bar{\mu}_i)}{\sum_t^T \Pr(i|x_t)} \tag{4.5.6}$$

For a smooth learning process, at each step, the parameters ϕ, μ, σ are updated based on their corresponding adaptation coefficients. We can view this as a learning rate used for updating the parameters by giving a weighted sum between step j and j+1.

4.5.2.2 Mixture Density Network

Even though both Mixture Density Networks (MDNs) and Gaussian Mixture Models approximate distributions using weighted sums of Gaussians, they differ in the way they do It and in the way they are trained.

In general, MDNs use a linear combination of kernel functions [19] and the pdf takes the following form

$$p(y|x) = \sum_i^K \varphi_i(x) f_i(y|x) \tag{4.5.7}$$

The MDN is a supervised learning model that computes $\varphi_i(x), \mu_i(x), \sigma_i(x)$ given an input x. For simplification purposes, kernel functions of Gaussian form are used, which makes the pdf of the MDN take a similar form to the one of the GMM.

The training process aims to minimize the negative log likelihood (NLL). The error is given by

$$E = \sum^T E_t \tag{4.5.8}$$

where E_T is the NLL of the training sample x_t

$$E_t = -\ln\Big(\sum_i^K \varphi_i(x_t) N\big(y\big|\mu_i(x_t), \sigma_i(x_t)\big)\Big) \tag{4.5.9}$$

4.6 Baseline Models, Challenges and Mitigation

4.6.1 Variational Autoencoders

Variational Autoencoders (VAEs) are, as their name infers, closely related to the traditional autoencoders as they share a similar basic architecture. They each have an Encoder, that compresses the input into a latent representation, as well as a Decoder that attempts to reconstruct the original input, given the latent representation. However, diving into detail, the differences between these models give VAEs exceptional generative powers as opposed to the traditional autoencoders.

Autoencoders are part of deterministic models, as they encode their input into a single discrete latent variable. VAEs, on the other hand, output continuous probability distribution over the latent space. This behavior puts VAEs into the category of probabilistic models and gives them strong generative powers as they are able to manipulate data inside the latent space and diversify using the continuous probabilities.

First, let's consider a generative model that wants to generate handwritten digits from 0 to 9. The output of the model must surely be one of the digits and not just any random handwritten symbol that resembles the training data. To solve this, the model first chooses a random sample of the available ones, using a latent variable, and then generates the desired output. So, the results would be a joint distribution of the observed variables x from the dataset, and the latent variables z. Latent variables are not observable and are not part of the dataset. For the model to be representative of our dataset, there needs to be a latent variable matched by every sample of the training data, so that in some way a similar result to any instance can be generated.

Initially, we view the observed variable x as a sample from the unknown real data distribution. VAEs attempt to approximate the real distribution $p(x)$ by modeling $p_\theta(x)$ with parameters θ. The goal is to optimize θ so that $p_\theta(x)$ approaches the datapoints of the dataset. As latent variables come in play, the goal is to model the joint distribution of both z and x

$$p_\theta(x) = \int p_\theta(x, z)\, dz \tag{4.6.1.1}$$

where $p_\theta(x, z)$ can be expressed as $p_\theta(z)p_\theta(x|z)$

The intractability of $p_\theta(x)$ is related to the intractability of the posterior $p_\theta(z|x)$. So, making $p_\theta(z|x)$ tractable can lead to a good approximation of $p_\theta(x)$.

To approximate the posterior inference and make the solution tractable, the inference model $q_\varphi(z|x)$ is introduced. This model represents the Encoder of the VAE and provides the latent space. The most common choice is to use a Gaussian encoder and model q_φ as

$$q_\varphi(z|x) = N\big(z; \mu, diag(\sigma)\big) \tag{4.6.1.2}$$

The training objective of the variational autoencoder is the maximization of the variational lower bound or evidence lower bound (ELBO) that derives from the following:

$$\log p_\theta(x) = E\left[\log p_\theta(x)\right] = E\left[\log\left(\frac{p_\theta(x,z)}{p_\theta(z|x)}\right)\right]$$

$$= E\left[\log\left(\frac{p_\theta(x,z)}{q_\varphi(z|x)}\right)\right] + E\left[\log\left(\frac{q_\varphi(z|x)}{p_\theta(z|x)}\right)\right] \tag{4.6.1.3}$$

The second term represents the Kullback-Leibler divergence between $q_\varphi(z|x)$ and $p_\theta(z|x)$. It basically shows how similar two distributions are. The smaller the value, the more similar the distributions are. This accounts for the training of the encoder as its goal is the approximation of the posterior p_θ.

The first term is the variational lower bound ($L_{\theta,\varphi}(x)$) and can also be expressed using the same KL divergence:

$$L_{\theta,\varphi}(x) = \log p_\theta(x) - D_{KL}(q_\varphi(z|x)\big|\big|p_\theta(z|x)\big) \tag{4.6.1.4}$$

So, KL divergence here represents the gap between $L_{\theta,\varphi}(x)$ and $\log p_\theta(x)$.

Simplifying the objective, in practice, when training VAEs, instead of maximizing $logp_\theta(x)$, we minimize the negative log likelihood. Assuming p_θ follows gaussian distribution $p_\theta(x)\sim N(x|\mu(x), \sigma(x))$, we can compute the negative log likelihood in closed form as

$$L_{NLL} = \frac{1}{2}\sum\left[\log(\sigma(x)^2) + \frac{\big(y-\mu(x)\big)^2}{\sigma(x)^2}\right] \tag{4.6.1.5}$$

plus constants that can be ignored.

The second term can also be simplified since, in practice, the objective of the encoder is to create a latent space that follows the normal distribution with mean 0 and variance 1, meaning $p(z)\sim N(0, I)$. We get

$$D_{KL}(q(z|x)||\, p(z)) = \frac{1}{2}\sum(\sigma(x)^2 + \mu(x)^2 - 1 - \log\sigma(x)^2) \qquad (4.6.1.6)$$

The implementation of a Variational Autoencoder usually consists of three neural networks. The most common form that $q_\varphi(z|x)$ takes is $N\ (z|\ \mu, \sigma)$, so if we think of q(z) as $N(\mu(x), \sigma(x))$, where μ(x) represents the mean and σ(x) represents the variance. Two neural networks are used to find the best q_φ from this given family of functions. To be more precise, they are not entirely separate networks as they have a common first part. The third neural network comes in play for the decoder as it tries to optimize $p_\theta(x|z)$ using a function f(z) for the mean of the distribution while considering a fixed value for the variance.

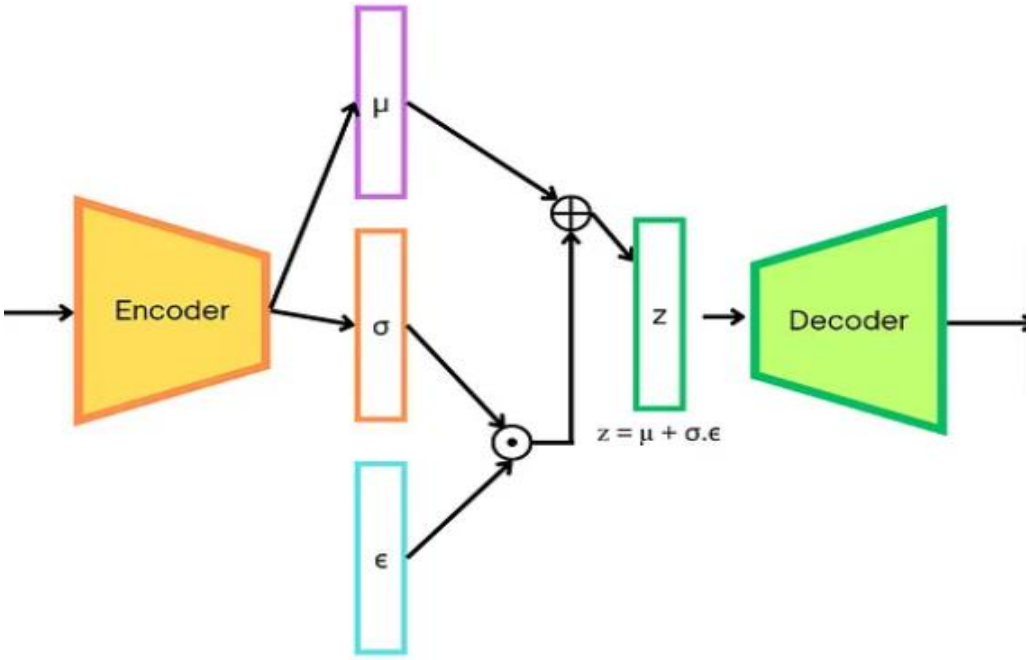


Figure 18: VAE architecture

VAE models use back propagation and update their weights in order to achieve training. To complete back propagation through the entire length of the network, we need to calculate the error and its derivatives on every single node using the chain rule. The encoder outputs the calculated mean and variance for the corresponding input. These two values are to be used to create an input for the decoder. However, sampling from *N(μ, σ)* introduces non differentiability to the network. This breaks the gradient flow and back propagation can no longer be completed for the nodes that come before the sampling process. Putting it more simply, the decoder sees nothing but a random value that has come from an unknown distribution while all the other layers know that their inputs have come from explicit multiplications and sums.

To overcome this objective, Variational Autoencoders implement the so called Reparameterization Trick. The whole point of this method is to keep the sampling, hence non differentiable operation, out of the parts of the network where back propagation is needed. This is achieved by introducing a new noise variable $\epsilon \sim N(0,1)$. The input of the decoder is then calculated as

$$z = \mu + \sigma * \epsilon \tag{4.6.1.7}$$

In this way, the sampling process does not break the gradient flow as we can now propagate through μ and σ. We can think of ϵ as a weight for σ that the network does not need to learn [2], [20].

4.6.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) were first introduced by Ian J. Goodfellow and his research team in 2014 [1] as a pioneering approach in the spectrum generative AI models. They have since received much acknowledgement due to their prolific performance, especially in image generation. Many task-specific GANs have been published with some examples being: medGAN tailored for generating multi-labeled discrete patient records, textGAN for natural language text generation, discoGAN, cycleGAN and many more.

In general, a GAN consists of two multilayer perceptrons, the Generator and the Discriminator. Consider a set x of training data. The generator takes an input of random noise $z \sim p_z(z)$. Then, maps the input into the data space x as $G(z; \theta_g)$, where θ_g are the parameters of the generator. After that, the outputs of the generator are passed into the discriminator along with some training samples from x. The discriminator is basically a classifier that learns the probability of the input class being real or fake. Its output is defined as $D(x; \theta_d)$ and represents the probability of an input x coming from the data rather than the generator's output, where θ_d are the parameters of the multilayer perceptron that comprises the discriminator.

The training process resembles a minimax two player game where the discriminator's goal is to maximize the probability of classifying generated instances and samples from the training set correctly, whereas the generator's goal is to minimize the discriminator's

accuracy by minimizing $log(1 - D(G(z))$. The value function of this problem would be:

$$\min_G \max_D V(D,G) = E_x[\log D(x)] + E_z\left[\log\left(1 - D\left(G(z)\right)\right)\right] \tag{4.6.2.1}$$

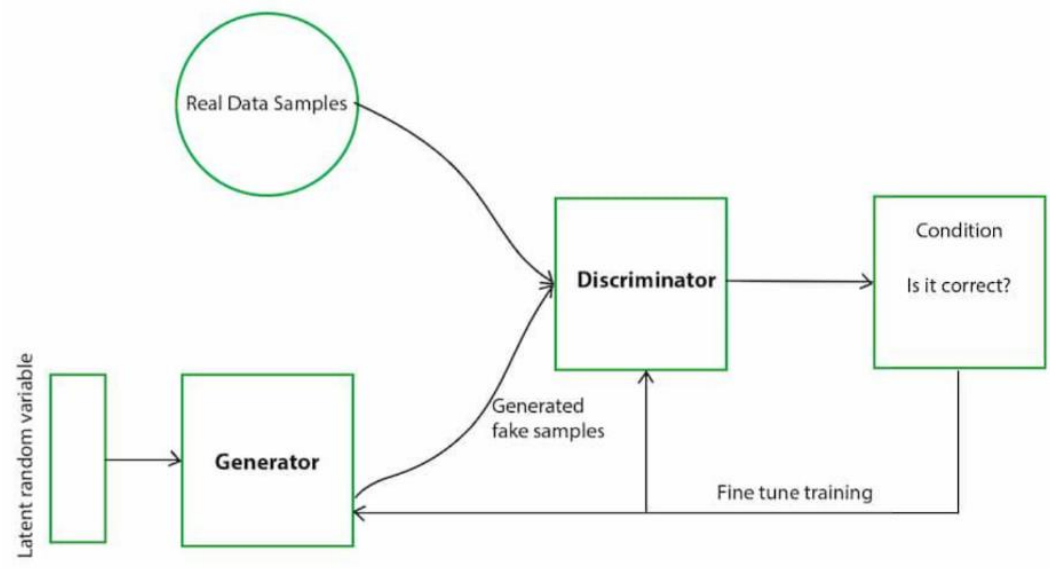


Figure 19: GAN architecture

It needs to be noted that training is an iterative process interchanging between updating the generator G and the discriminator D. Either way, the cost function usually remains the same. The main difference in the two subprocesses is that in the first case, only the output data of the generator is passed as input into the discriminator. On the contrary, when updating the discriminator, both generated outputs and values of the training data are passed into the discriminator.

In fact, in the early stages of training, the generator produces outputs poor enough so that the discriminator has high accuracy. This can result in the generator training at a slow rate. To avoid that, G can be trained to maximize $\log(D(G(x))$ in the beginning, which provides stronger gradients and a steeper approach towards the desired performance.

Another problem that must be dealt with is the Helvetica scenario. This can occur when the generator is overtrained in comparison to the discriminator. If the discriminator is not frequently updated, the generator will avoid approaching certain instances of the data distribution that cannot trick the discriminator efficiently enough. The gradients obtained by the same discriminator repeatedly will collapse many noise values to a limited number of the training data and eventually, the model will not have enough diversity to replicate the distribution of the data.

Last but not least, what needs to be taken into consideration in the early stages of development is that the generator and the discriminator must initially be at a similar level. If, for example, the discriminator was a good classifier from the beginning, then, almost none of the generator's

output would be classified as part of the training data, hence the error becomes minimal, and the generator can no longer “learn” [1].

4.6.3 Diffusion Models

Apart from the most used deep generative models, GANs and VAEs, Diffusion Models are also a state-of-the-art approach for most of the generative related tasks such as image synthesis, audio generation and language processing [21]. Diffusion Models are part of the class of likelihood-based models as are the Variational Autoencoders. Likelihood-based models explicitly define a likelihood function to estimate the probability distribution of observed data. Using a likelihood function, unlike implicit generative models such as GANs, helps Diffusion Models avoid mode collapse and training instabilities; two phenomena that have been major issues for data synthesis throughout the years. However, they can be more resource and time demanding.

Diffusion Models have been inspired by non-equilibrium thermodynamics and especially as their name infers, diffusion [22]. In physics, diffusion is described as the process in which a group of molecules moves from a region of higher concentration to a region of lower concentration.

These models follow a two-step approach, consisting of a forward process and a backward/reverse process. In the forward process, a sample from the data space is corrupted by a forward Markov process that adds Gaussian noise based on a variance schedule $\beta_1, \dots, \beta_T$. This process transforms the data point into a latent variable which is basically pure noise [23].

$$q(x_{1:T}, |x_0) = \prod_{1}^{T} q(x_t|x_{t-1}) \tag{4.6.3.1}$$

$$q(x_t|x_{t-1}) = N\left(x_t; \sqrt{1-\beta_t}\, x_t, \beta_t I\right) \tag{4.6.3.2}$$

Here, $q(x_t)$ denotes the distribution of latent variable x_t in the forward process. The forward process converts the unknown high dimensional distribution of the input data into a known one as $p(x_T) = N(x_T; 0, I)$. Setting $1 - b_t = a_t$ and $\bar{a}_t = \Pi_{s=1}^{t} a_s$, we can see that each step corrupts x_{t-1} to $x_t = \sqrt{a_t} x_{t-1} + \sqrt{1-a_t}\epsilon$ where ϵ is random Gaussian noise sampled from $N(0, I)$.

The latent variable then learns how to be denoised back into the real data distribution by the reverse process, creating new synthetic data [21].

$$p_\theta(x_{0:T}) = p(x_T) \prod_{1}^{T} p_\theta(x_{t-1}|x_t) \tag{4.6.3.3}$$

$$p_\theta(x_{t-1}|x_t) = N\left(x_{t-1}\,; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right) \tag{4.6.3.4}$$

$p_\theta(x_{t-1}|x_t)$ represents the estimated distribution of the latent variable x_{t-1} and is approximated using a neural network with parameters θ. The simplified process is depicted as a Markov chain in the following image

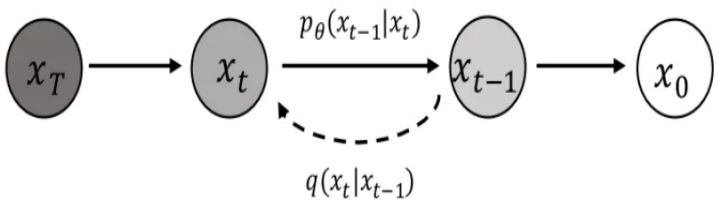


Figure 20: Reverse and forward diffusion depicted as a Markov Chain

Training is performed by optimizing the variational upper bound on negative log-likelihood:

$$L = E_q\big[\, D_{KL}(q(x_T|x_0)\big|\big|p(x_T)\big) + \sum_{t>1} D_{KL}(q(x_{t-1}|x_t, x_0)\big|\big|p_\theta(x_{t-1}|x_t)\big) - \log\big(p_\theta(x_0|x_1)\big)\,\big] \tag{4.6.3.5}$$

where $q(x_{t-1}|x_t, x_0) = N\big(x_{t-1};\ \widehat{\mu_t}(x_t, x_0), \widehat{\beta_t} I\big)$

4.7 Difficulties in tabular data generation

Generative Neural Networks are mostly designed to generate images. Some have shown excellent performance in quality as well as in diversity. However, being able to produce high quality images does not guarantee good performance in other types of applications. More importantly, in some cases, generative models cannot even produce the type of data needed. In tabular data, the attributes often have many different types of values, and especially the categorical ones are very hard to handle. Also, the combinations across values can become vast or some values can be represented only by a tiny part of the dataset. Handling different types of data, high dimensionality, sparsity and dataset imbalances can be very challenging for tabular data generation.

4.7.1 Non-Differentiability of Discrete Data

Neural Networks are designed to work with continuous values and utilize backpropagation as a gradient based optimization method. Consider the generation of an image where each pixel is

represented by a value between 0 and 1. The network computes the error and the gradients of the loss function with respect to the weights of every layer. This can then be used to adjust the weights in a way that next time, the error by the loss function will be smaller. This makes it quite easy to understand why gradient descent can tune continuous values to possibly reduce the error after every update of the parameters.

What makes it hard for categorical and discrete values is that they may not have any type of continuity and relation between them, thus not enabling the model to approach a better solution in certainty as the gradients cannot show if changes in weights will lower the error through the loss function. For instance, in a tabular dataset, someone can come across columns that represent the type of a vehicle, the city, the blood type of a person and many other discrete-value attributes. These values are distinct categories or classes without any gradient between them.

4.7.2 Mode Collapse

Mode collapse is a concerning challenge when creating and training generative models. We come across this term mostly when dealing with GANs, however many other models suffer from what seems to be a quite usual problem. A very important part of a model is the loss function on which it performs its training. Yet, a generative model cannot count on the loss function alone. In many cases, extra care is needed to prevent a model from generating only a certain subset of possible outputs only to minimize the loss function. In GANs, for example, the generator fools the discriminator by learning to produce real-looking outputs but with no diversity. Even VAEs, which are less prone to suffer from this phenomenon, poor handling of the latent space and weak encoder can create really low diversity.

The problem described above is exacerbated when dealing with tabular data instead of images. Having discrete categories in certain attributes of the dataset makes it easier for the model to collapse into limited number of those and makes it even harder to recover from it. As opposed to distinct categories, we can define some categories in continuous data but there will be minor variations among them. The absence of these minor variations in discrete tabular data makes it easier for the model to fall into producing repetitive outputs.

4.8 Introduction to Embeddings

Word Embeddings are a foundational technique in natural language processing (NLP). They transform words into dense and continuous vector representations. Word embeddings have many advantages over one-hot encoding. One-hot encoding is just a different form of the same

representation, which is viewing words as atomic units. Word Embeddings capture semantic relationships by mapping similar words closer together in the space they are represented.

Since extracting word embeddings requires some computation, one can think that creating them and using them to train a new model increases the complexity and resources of the whole process. At first glance, two different models are created but, on the other hand, word embeddings can reduce the complexity of our final model significantly compared to simple one-hot encoding. Apart from the fact that word embeddings are learnable representations of words and don't treat them as individual units, they have the advantage of reducing the dimensionality of the data.

When using one-hot encoding, the resulting vectors are high-dimensional and sparse. For a vocabulary of V words, each word is represented by a V-dimensional vector with just one element equal to 1 while the rest are zero. Embeddings map words into any dimensionality we desire, even into a single scalar. However, reducing the dimensionality reduces the amount of information and relationships in the new space. Using smaller vectors keeps the size of the model parameters from increasing, hence keeping memory usage and complexity low.

To illustrate and better understand how word embeddings work, consider the words "king", "queen", "man" and "woman". Encoding these words using a traditional one-hot encoding method, each word would be represented by a binary vector with all values equal to 0, except one. The position of the 1 in the vector determines each word and differentializes it from all the others. However, there is no information about how words relate to each other. For example:

- "king" → [0, 0, 1, 0, 0, …]
- "queen" → [1, 0, 0, 0, 0, …]

These representations do not capture any semantic meaning and they are orthogonal, meaning any multiplication between them results to 0. In contrast, word embeddings represent each word as a dense, low-dimensional vector of real numbers, capturing meaningful relationships and information across the vocabulary. The goal is to transform words that appear in similar contexts into similar vector representations that inherit the analogies and determine the strength of the similarity.

For example, after training an embedding model on a corpus, we could obtain the following vectors:

- "king" → [0.65, 0.47, 0.23, 0.10]
- "queen" → [0.63, 0.50, 0.20, 0.21]

- “man” → [0.70, 0.43, 0.10, 0.10]
- “woman” → [0.72, 0.49, 0.12, 0.21]

These vectors hold semantic relationships between the words representing the differences and analogies of each pair. One famous property of word embeddings is that with vector arithmetic, we can manipulate, preserve and represent certain characteristics. For example, if king is to a queen what woman is to a man, then $king - queen \approx man - woman$. Semantically, the two subtractions represent the difference in sex. This property can be applied on many occasions and forms naturally from the training process of the embeddings.

One of the most influential and powerful approaches to generating word embeddings is Word2Vec [24], introduced in 2013. It uses shallow neural networks to learn representations based on the context of the sentences that a word is part of. There are to architectural versions of Word2Vec: Continuous Bag of Words (CBOW), which aims to predict each word based on its surrounding context, and Skip-Gram, which aims to represent surrounding words given a target word.

Chapter 5

Methodology

5.1 Baseline Models

5.1.1 Conditional Tabular GAN

The CTGAN [25] is a GAN-based architecture model with optimizations for generating tabular data. Even though there are no substantial changes in the discriminator, the generator is carefully adapted to fulfill the requirements of a well performing generative model for mixed type tabular data. The optimizations applied are related to the preprocessing of the learning data, the way the inputs are passed to the generator and the inputs of the discriminator at each step.

Most of the models that aim to handle discrete and categorical data transform them using one-hot encoding and the CTGAN does not deviate from that. However, for continuous attributes, it uses a so-called mode-specific normalization. Mode-specific normalization is based on Gaussian Mixture Models that were mentioned earlier. Each column of the dataset is processed individually with a variational Gaussian mixture model (VGM) trained on its data. The VGM creates a Gaussian mixture with an estimated number of modes. Each numerical value is transformed into a one-hot vector and a scalar. The one-hot vector represents one of the modes of the mixture and the scalar represents the value within the selected mode. The mode is sampled based on the probability of the current value being represented by it. The scalar is explicitly calculated based on the mode.

The CTGAN incorporates conditional generation in order to handle class imbalances. In this way, it is trained sufficiently on underrepresented classes by conditioning on the categorical attributes and it also achieves balanced generation with the same method.

$$P(row) = \sum P_G(row|D_{i^*} = k^*)P(D_{i^*} = k^*) \tag{5.1.1}$$

where k* is the value from the i_{th} discrete column D_i^* that the model attempts to match with the generated samples.

To handle the conditioning, the conditional vector is introduced. m_i is associated to the i_{th} discrete column one-hot vector d_i. The condition $D_{i^*} = k^*$ can be represented by the mask vectors in the following way

$$m_{i^*}^{k^*} = 1 \; and \; m_i^k = 0 \tag{5.1.2}$$

The conditional vector cond is the concatenation of the m_i vectors. It has the length of the number of all categorical attributes and only the value of the categorical attribute, that we want to condition on, is equal to 1.

Even though the network is set to condition on certain attribute values, there is nothing to encourage it to actually output rows that belong to the defined category. A term must be added in the loss function that penalizes it if the output does not produce the same value at the corresponding conditional categorical attribute. To achieve that, the cross entropy between m_{i^*} and d_{i^*}, averaged over all instances of the batch, is introduced to the loss function.

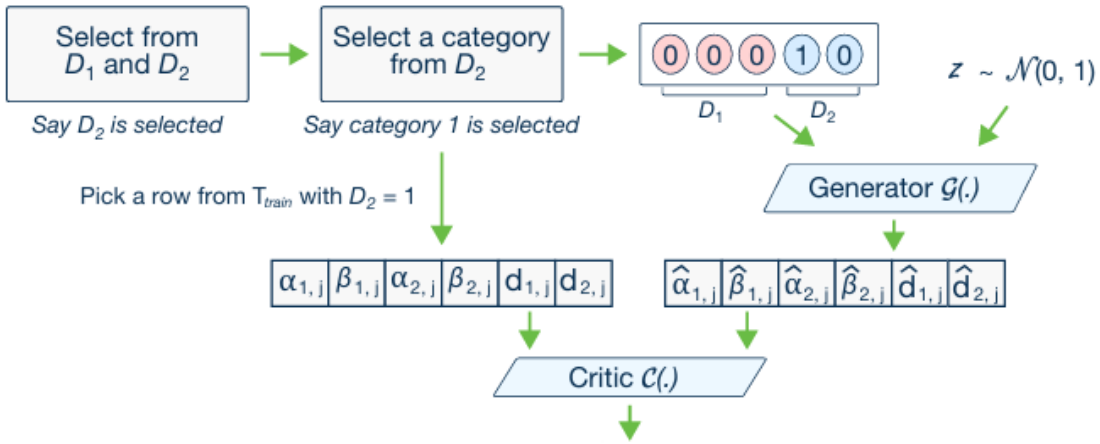


Figure 21: CTGAN architecture and data flow of an example with 2 categorical and 2 numerical attributes

In addition, the CTGAN is implemented using the optimizations of Wasserstein GAN (WGAN) [26]. The Wasserstein Gan aims to improve the diversity of the GAN as well as prevent mode collapse. It adds a term to the loss function that encourages the network to produce outputs closer to the distribution of the real data. The term is the Wasserstein distance, or else the Earth Mover's Distance. Intuitively, this metric shows how much 'mass' needs to be moved in order for two distributions to become the same.

To better detect and prevent mode collapse, the PacGAN algorithm is also used. With pacGAN the discriminator accepts input in packs, enabling it to review groups of generator outputs together. In this way, it detects more generic errors and lack of diversity as it assesses the relationships between the samples of the pack.

5.1.2 Tabular VAE

The tabular VAE (tVAE) [25] is a Variational Autoencoder optimized for mixed type tabular data. The main architecture does not differ from the vanilla VAE. Similarly to the CTGAN it uses a detailed preprocessing method which enables it to take advantage of the VAE properties and apply them on categorical and discrete data. Apart from that, it only adapts its loss function in order to compute it since the output has mixed types.

The mode-specific normalization described previously is applied on the training data of the tabular VAE too. Two neural networks are used: one for the decoder, which models $p_\theta(z|x)$, and one for the encoder, which models $q_\varphi(x|z)$. The network is trained on the evidence lower bound (ELBO). Its objective includes minimizing the negative log likelihood of $p_\theta(x|z)$ which, for one-hot encoded vectors, becomes the cross entropy.

5.1.3 TabDDPM

TabDDPM [27] is a generative mode based on Diffusion Models. It is an approach that aims to take advantage of the properties of the denoising diffusion probabilistic model (DDPM) on the discrete space. This mixed data type model showcases great performance even in comparison to CTGAN, TVAE and other common high performing models like SMOTE.

First, we analyze the base of TabDDPM, the DDPM [28]. The denoising diffusion probabilistic model comes from the vanilla diffusion model with some simplifications. General diffusion models use forward process variances $\beta_t$which are learnable. The DDPM uses fixed values instead of learnable $\beta_t$which results in L_T, the first term of L, not having any learnable parameters, thus being ignored in training.

Except for the L_T not taking any part in the training process, p_θ is simplified as well as $\Sigma_\Theta(x_t, t)$ becomes $\sigma_t^2 I$. As a result, we have $p_\theta(x_{t-1}|x_t) = N(x_{t-1};\ \mu_\theta(x_t, t),\ \sigma_t^2 I)$ and having constant variance on both p_θ and forward process posteriors $q(x_{t-1}|x_t, x_0)$ the term training objective, deriving from the term L_{t-1} term, becomes

$$L_{t-1} = E_q\left[\frac{1}{2\sigma_t^2}\left|\left|\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t)\right|\right|^2\right] + C \tag{5.1.3}$$

However, with further parametrization, instead of learning μ_θ to predict the mean $\tilde{\mu}_t$ of the normal distribution from which the noise is sampled, the objective becomes minimizing the sum of mean-squared errors between $\epsilon_\theta(x_t, t)$ and ϵ.

$$L_t^{gaussian} = E_{x_0,\epsilon,t}\ \|\ \epsilon\ -\ \epsilon_\theta(x_t, t)\ \|_2^2 \tag{5.1.4}$$

The TabDDPM handles categorical and discrete attributes using the Multinomial Diffusion [29]. The Multinomial Diffusion Model is an extension of diffusion models designed to generate discrete data, particularly categorical sequences like text, tabular data, and other structured information. Unlike traditional diffusion models that operate in continuous space (e.g., images with Gaussian noise), multinomial diffusion adapts the process to discrete variables.

Given the real data at space $\{0,1\}^K$, each categorical data x_t is a one-hot encoded categorical variable with K values. In the forward process, each one-hot vector is corrupted by replacing it with another vector from the data space with probability β_t. The new category is sampled uniformly

$$q(x_t|x_{t-1}) = C\left(x_t\middle|(1-\beta_t)x_{t-1} + \frac{\beta_t}{K}\right) \tag{5.1.5}$$

Here $C(x\ |y)$ denotes a categorical distribution with probability parameters y. Considering x_t is a one-hot vector, we end up with K-1 probability parameters of value and one equal to which corresponds to the position of 1 in the vector x_{t-1}. Through the Markov process we can acquire

$$q(x_t|x_0) = C\left(x_t\middle|\bar{a}_t x_0 + \frac{1-\bar{a}_t}{K}\right) \tag{5.1.6}$$

where $a_t = 1-\beta_t$ and $\bar{a_t} = \Pi_{T=1}^{t}\alpha_T$. The categorical posterior can be computed as

$$q\ (x_{t-1}|x_t, x_0) =\ C(x_{t-1}|\frac{\pi}{\sum\pi_K}) \tag{5.1.7}$$

Where $\pi = \left[a_t x_t + \frac{1-a_t}{K}\right] \cdot [\bar{a}_{t-1}x_0 + \frac{1-a_{t-1}}{K}]$. In the reverse process the model is trying to learn $p(x_{t-1}|x_t)$ which is parametrized as $q(x_{t-1}|x_t, \widehat{x_0}(x_t,t))$ and $\widehat{x_0} = \mu(x_t,t)$ is predicted by a neural network. Setting $\theta(x_t,x_0) = \frac{\pi}{\sum\pi_K}$ we get

$$L_t = KL(\ C(x_{t-1}|\theta(x_t,x_0)|\big|C(x_{t-1}|\theta(x_t,\widehat{x_0})\big) \tag{5.1.8}$$

5.1.3.1 Model Overview

TabDDPM handles categorical and binary features in a different way to numerical ones. The multinomial diffusion is used to model categorical and binary features. Each categorical feature x_{cat_i} with K_i categories is transformed into a one hot vector $x_{cat_i}^{ohe} \in \{0,1\}^{K_i}$ and for each category, a different forward diffusion process is applied. The numerical features are modeled using Gaussian diffusion.

For the reverse diffusion, a multi-layer neural network is used. Its input and output dimensions are the same and are equal to $N_{num} + \sum K_i$, which is the size of the post processed input data sample. The training process tries to minimize the sum of the mean-squared error of the Gaussian diffusion ($L_t^{gaussian}$) and the KL divergences $L_{t,i}$ of the multinomial diffusion processes. Since every categorical feature is handled by a separate forward process, we add the KL divergences for all categorical features C and divide them by C:

$$L_t^{TabDDPM} = L_t^{gaussian} + \frac{\sum^C L_{t,i}}{C} \tag{5.1.9}$$

5.2 Embedding Model Implementation

Word embeddings are used for transforming words of a sentence into a set of learnable vectors. Our subject is studying synthetic data coming from tabular datasets. So, for this purpose, we can view our dataset as a text corpus and each row of the dataset as a sentence of this corpus. Similarly to a set of words creating a coherent sentence, the values of the attributes of a consistent dataset are set in such a way, creating a meaningful row of information. The only differences are that tabular datasets are of fixed number of columns, and the attributes of each row follow the same sequence. These two characteristics make our objective even easier.

5.2.1 Skip-Gram Model

The implementation of the word embeddings used for the purposes of the following models is inspired by the Continuous Skip-Gram model [24]. Skip-Gram model enables efficient learning of word representations while keeping the computational complexity low. Faster training is achieved through removing the non-linear hidden layer used by other similar networks. A neural network is trained given a center word and a context word each time. Its objective is maximizing the classification of the context word based on the center word from the same sentence, or the same row in our occasion. The original Skip-Gram model samples context words close to the target more frequently, since words close within a sentence are more likely to be related. On the contrary, there is no such relation with the row of a dataset because the attributes can be in a random order, so context word sampling must be random or include all the attributes other of the row.

The goal is to train a network to maximize the likelihood of a target value, given a context value from the same row. In essence, though, only the embedding matrix is extracted from the network. Each categorical value of the dataset is mapped to a unique embedding vector from the

embedding matrix E. Given a center word, its corresponding embedding vector e_c is acquired from E. The model calculates the probability of the context given the center, $p(x_{con}|x_t)$, using softmax. The input of the softmax are the values resulting from the dot product of e_c with every vector of the output representation matrix Theta. Theta matrix is the second set of parameters of the network and has the same size as E: a set of embedding dimensions for each word of the vocabulary.

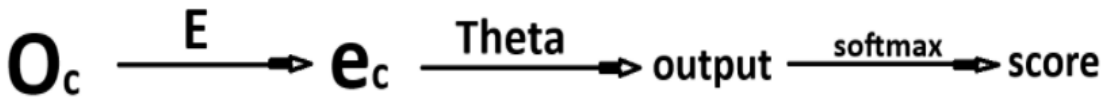


Figure 22: high-level representation of skip-gram model

Oc is the one-hot vector of the center word that maps it to the embedding space. e_c is multiplied with Theta matrix and the output is a vector of scalars, one for each word of the vocabulary. The score is calculated in the following way

$$score = \frac{e^{\theta_{con}^T * e_c}}{\sum_{j=1}^{V} e^{\theta_j^T * e_c}} \quad (5.2.1)$$

θ_{con} is the vector of the output representation matrix Theta that corresponds to the context word. This formula is impractical because of its complexity. To get the score of a context word we have to compute the exponentials for all the other words of the vocabulary.

5.2.2 Negative Sampling

To mitigate this problem, Negative Sampling [30] is introduced. In negative sampling, complex calculations are avoided by turning the softmax part of the network into a classifying problem. For a pair of center-context words w_c, w_{con} of the same sentence, or row in the occasion of a tabular dataset, a few 'negative' values are sampled from the rest of the dataset. 'Negative' samples are random words of the vocabulary, different than w_{con}. New input data is constructed for the new model: the current center word paired with the context word with class label 1 and also paired with the negative samples with class labels 0.

In this way, the last part of the Skip-Gram model is replaced by a simple sigmoid function, and the task of the new model becomes the following: with an input of two words, a center and a random one, calculate the probability of the random being a context word given the first.

To create embeddings representative of the words, however, the sampling of the negative words must be taken care of. Supposing the sampling is done uniformly among the vocabulary, it would not reflect the distribution of the data since some words are observed more than others. On the other hand, sampling based on observed frequency would not be very efficient either, as values that don't hold much substantial information can be present in the dataset at very high frequency. For the forementioned reasons, the sampling is done from the following pdf

$$p(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_j f(w_j)^{\frac{3}{4}}} \tag{5.2.2}$$

This is the unigram distribution raised to the $\frac{3}{4}$ power and gives something in between the uniform sampling and the frequency sampling. Note that $f(w_i)$ in the equation above represents the observed frequency of word w_i.

5.2.3 Custom Embedding Model

To learn meaningful embeddings from categorical attributes of tabular data, a custom embedding model was developed, heavily inspired by the skip-gram model with negative sampling. Unlike traditional sequential data, tabular data lacks spatial ordering, so the model aims to capture co-occurrence patterns across columns. Specifically, it forms positive training pairs by sampling values from randomly chosen column pairs within each row. These pairs act analogously to the word-context positive pairs of the negative sampling technique in NLP.

In order to further reduce time and computational complexity without missing important information on the embeddings, this custom embedding model has the distinctive feature of not excluding the positive samples from the candidate pool of negative samples. When selecting the negative samples, all tokens remain eligible for sampling, no matter if the current negative sample matches the positive sample with respect to which it was selected. While this can introduce more overlapping between negative and possible samples, it is deemed as subtle for damaging the fidelity of the embeddings. Moreover, it can produce more robust and generalizable results.

Negative sampling is performed, as previously discussed, using the unigram distribution raised to the $\frac{3}{4}$ power. The computation of this distribution is not done dynamically on every batch; it is calculated once in the beginning for the entire dataset and then used at each step.

In addition, the model avoids imposing an artificial context window. This would be irrelevant and damaging for tabular data embeddings as there is no spatial ordering; the order of the values is determined by the random order of the columns of the dataset. In place of the concept of the

context window, a percentage-wise positive sample selection is introduced. Apart from reducing complexity, it also acts as a generalization technique. Let us consider the context value in a step of negative sampling. Instead of selecting all the rest of the values in the row to complete $N_d - 1$ positive samples (N_d being the number of attributes), only a percentage of them is used. This percentage can be viewed as the equivalent of the context window, only it is not spatially applied.

5.3 Proposed Model Architectures

5.3.1 eGAN

The embedding GAN model adapts the classic Generative Adversarial Network (GAN) to operate on learned embeddings of categorical variables rather than raw one-hot or ordinal encoded data and consists of two parts. The first part is the Custom Embedding Model described in 4.3.2, and the second part is the adapted GAN.

The categorical columns of the dataset are used to train the Custom Embedding Model. After training, we extract the embedding matrix E that contains the vector representations of all observed categorical values. Each categorical value is mapped to its corresponding continuous dense vector of fixed size.

After that, the initial dataset of mixed data types is transformed into a dataset of purely numerical values through this simple transformation process. The categorical values are transformed into vector representations, and each row is then flattened to take the form of a single datatype record. After the transformation, the dataset's initial dimensions have been altered. If each row of the initial dataset consists of N_d number of categorical values and N_c number of continuous values, the new dataset, instead of $N_{total} = N_d + N_c$ number of values, now has $N'_{total} = N_d * n_f + N_c$ values, where n_f is the number of features of each values in the embedding space.

The GAN part of the model is a variation that operates on the flattened records. The implementation generally follows the vanilla version of GAN. The Generator takes random noise as input and outputs a vector of the same shape as the real embedded data, in our case N'_{total}. The Discriminator receives these vectors and distinguishes between real and synthetic embedded samples.

After training and generation, the synthetic vectors' columns corresponding to the flattened embedded values of categorical variables are mapped back to actual categories. This reverse transformation process is done in the following way. Consider a n_f sized vector v that represents the categorical column c. Vector v is compared to all the embedded values of column c. For each embedded value of c, the cosine similarity between it and the vector v is calculated. The

categorical value, whose embedding vector has the largest cosine similarity with v, is used to replace v in the final dataset. Numerical values are not transformed throughout the process, only normalized.

5.3.2 eVAE

Equivalently to eGAN, this embedding VAE model is introduced as an alternative way of generating synthetic data. It extends the Variational Autoencoder (VAE) for tabular data by operating on the same embedding-augmented input space as eGAN.

The categorical values of the dataset are passed through the Embedding Model and transformed into vectors. The new dataset is created, where each row has a size of $N'_{total} = N_d * n_f + N_c$. The vectors are flattened, and each scalar is handled independently. The flattened dataset is used to train a VAE model that handles numerical values without any specifications for categorical columns. The encoder maps the input into a latent space defined by a mean vector μ and a standard deviation vector σ. Using the reparameterization trick, a latent vector is sampled and passed to the decoder. The decoder learns to reconstruct the original input vector.

The model is trained to minimize the standard VAE loss consisting of two parts: the reconstruction loss, simplified into the negative log-likelihood of the decoders output, and the KL divergence, which regularizes the learned latent space to follow a standard Gaussian Distribution.

As with eGAN, after generation, the decoder's outputs corresponding to categorical variables are mapped back to discrete values using similarity against the embedding matrix corresponding to the initial column of the vector.

5.3.3 eDDPM

The eDDPM incorporates the Denoising Diffusion Probabilistic Model (DDPM) [28] into the embedding-based generative framework, leveraging the strengths of likelihood-based training and noise scheduling.

As in eGAN and eVAE, the Custom Embedding Model learns to map the input categorical values into continuous dense vector representations. Every record, meaning numerical attributes with embedding vectors, is normalized and flattened into a single vector.

The forward process adds Gaussian noise to each vector progressively over multiple steps. The amount of noise at each timesteps is determined by a fixed variance schedule and the model emphasizes more in training on later timesteps, as they introduce more noise and are harder to learn. The reverse process is executed by a neural network, which learns to denoise a noisy input

step-by-step to create a clean synthetic data sample. At each step, the denoiser aims to predict the noise hypothetically introduced at the specific step.

After training, synthetic samples are generated by starting from random gaussian noise and iteratively applying the learned denoising steps. Each time, the model predicts the noise of the step and subtracts it from the sample, thus progressing into a clean sample. As with other models, the resulting vectors are decoded into categorical values using cosine similarity with the corresponding values of the embedding matrix.

Chapter 6

EVALUATION

6.1 Method Introduction

To assess the performance and quality of the synthetic data generated by the models described in the previous sections, a series of evaluation experiments were conducted, using the Adult Income [31] and the Mushroom [32]dataset from the UCI Machine Learning Repository. This dataset is often used as a benchmark for tabular data tasks due to its data quality and number of instances. It is an ideal dataset for the purpose of this experiment as it contains a sufficient number of attributes of mixed data types, both numerical and categorical.

The UCI Machine Learning Repository is one of the most widely recognized and utilized resources in the machine learning community. It was established by UCI PhD student David Aha in 1987 and it has since become an essential tool for research thanks to many contributors and librarians as well as to the funding support from the National Science Foundation.

After evaluating the generative models on the Adult Income dataset, its numerical attributes were removed, the models were retrained, and their performance was evaluated again to see how they manage without the numerical attributes of the dataset. Since without the numerical attributes the Adult Income dataset does not pose a hard task for the models due to the low complexity and cardinality of the data, the models were also assessed on the Mushroom dataset [32], also from the UCI Machine Learning Repository. The Mushroom dataset comprises of more categorical attributes of higher cardinality, making it a challenge to model.

6.2 Datasets Description

6.2.1 Adult Income

The Adult dataset [31] consists of 48,842 instances and 14 attributes. Its main usage is for classification tasks since its label ‘income’ is a binary attribute of values ‘<=50K’ and ‘>50K’ that

determines whether the person of the current instance has an income larger than $50,000 a year, or not.

- Classes: <=50K, >50K
- Instances: 48842
- Features: 14
- Categorical Features: workclass, education, marital-status, occupation, relationship, race, sex, native-country, income
- Numerical Features: age, fnlwgt, education-num, capital-gain, capital-loss, hours-per-week

For the purpose of these experiments, the instances with null values were dropped from the dataset. The attributes ‘capital-gain’, ‘capital-loss’ were not used and the final number of rows dropped down to 32,561.

age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	hours-per-week	native-country	income
39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	40	United-States	>50K

Figure 23: 10 samples of the ‘Adult’ dataset. The samples were taken after the null values and the columns ‘capital-gain’ and ‘capital-loss’ were dropped

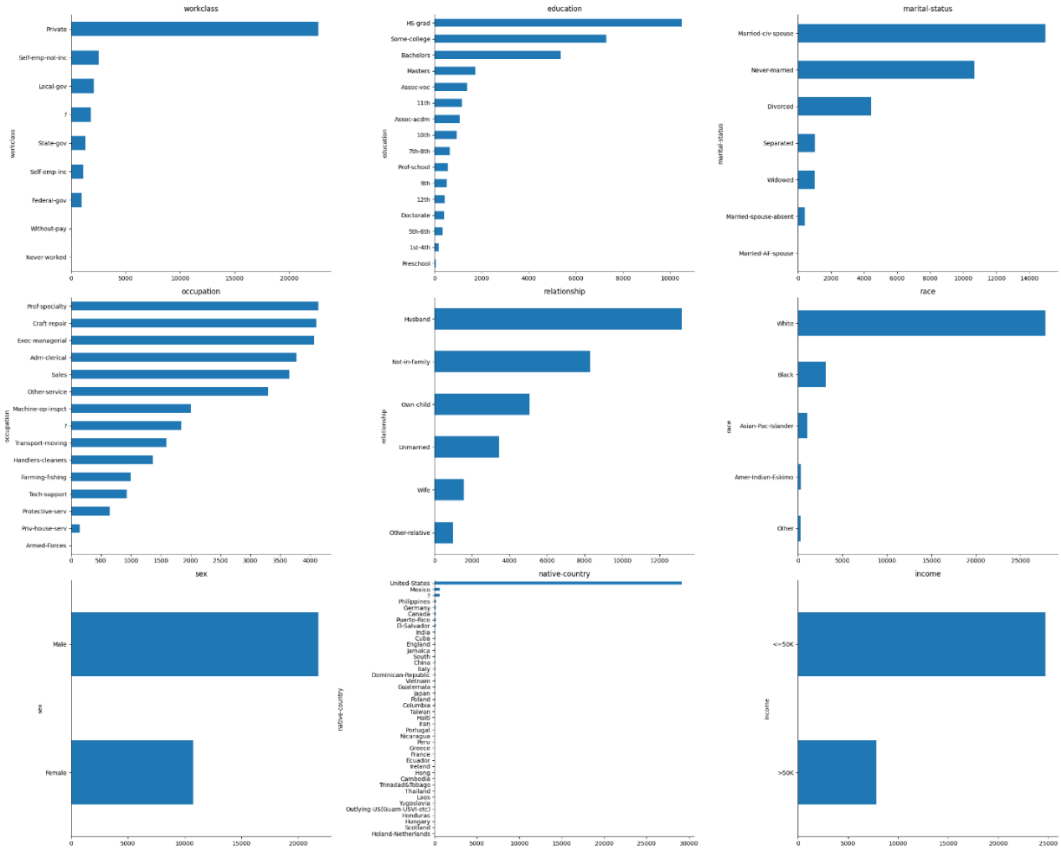


Figure 24: Bar charts of the frequencies of the categorical attribute values of the 'Adult' dataset

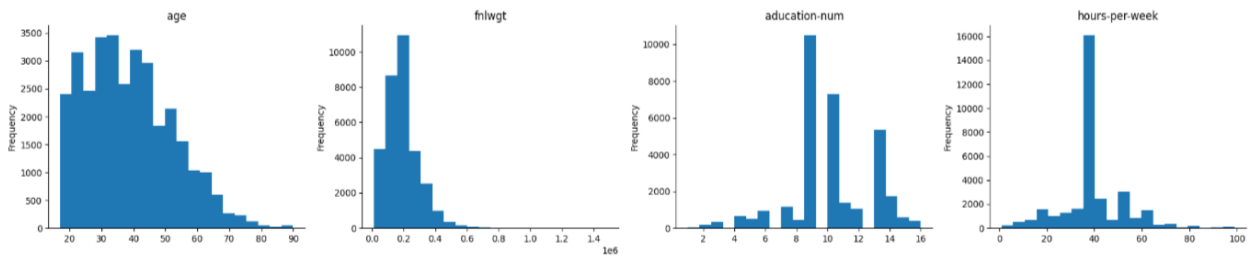


Figure 25: Histograms of the frequencies of the numerical attribute values of the 'Adult' dataset

6.2.2 Mushroom

The Mushroom dataset [32] consists of 8,124 rows and includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. It is mainly used for classification task and is equipped with a label 'class' with two possible values: e for edible and p for poisonous

- Classes: e, p
- Instances: 8,124

- Features: 23
- Categorical Features: class, cap-shape, cap-surface, cap-color, bruises, odor, gill-attachment, gill-spacing, gill-size, gill-color, stalk-shape, stalk-root, stalk-surface-above-ring, stalk-surface-below-ring, stalk-color-above-ring, stalk-color-below-ring, veil-type, veil-color, ring-number, ring-type, spore-print-color, population, habitat.
- Numerical Features: -

	Number of values
class	2
cap-shape	6
cap-surface	4
cap-color	10
bruises	2
odor	9
gill-attachment	2
gill-spacing	2
gill-size	2
gill-color	12
stalk-shape	2
stalk-root	5
stalk-surface-above-ring	4
stalk-surface-below-ring	4
stalk-color-above-ring	9
stalk-color-below-ring	9
veil-type	1
veil-color	4
ring-number	3
ring-type	5
spore-print-color	9
population	6
habitat	7

Figure 26: The attributes of the Mushroom dataset with their corresponding number of values

6.3 Training Procedure

Handling such delicate generative process matters requires effective configuration of each model. To determine the most effective configuration, several training runs were conducted with varying hyperparameters. The final parameter settings were selected based on many factors. Training stability and convergence behavior were regarded as major factors taken into account with respect to the overall synthetic data quality based on specific metrics. While the original papers provided a valuable starting point for model hyperparameters, adjustments were made to account for the characteristics of the specific dataset used.

Evaluating generative models during training is inherently challenging as their objective is different from the ordinary machine learning task. Instead of optimizing the predictive accuracy or loss based on given labels, they aim to learn the distribution of the training data. This means that, apart from the loss of each epoch, in order to draw insight on the progress and performance of the model, it would be best to generate new synthetic samples each time, and evaluate the samples based on selected metrics against real data.

For this purpose, each time the model needed validation, 1,000 samples were generated using the model trained up to the current epoch and insightful metrics evaluated its performance. The average Wasserstein Distance of categorical attributes and numerical attributes was measured compared to the real dataset as well as the L2 distance of the Correlation Matrix of the Numerical values of the 1,000 synthetic samples compared to the same matrix of the real dataset.

For better performance and speed, when using the embedded datasets to train eGAN, eVAE and eDDPM, the corresponding embedding vectors were not transformed back into categorical values and were instead compared to the embedded values of the real dataset.

6.4 Custom Embedding Model Specifications

The Embedding Model is the embedding model designed to learn low-dimensional representations of categorical variables for the custom models eGAN, eVAE and eDDPM. Each unique categorical value was mapped to a 5-dimensional embedding vector and the model was trained to predict co-occurrence between values from different columns. During training, at each step, 30% of all possible column pairs were randomly sampled to generate positive pairs. For each positive target, 3 to 5 negative samples were drawn based on the smooth unigram distribution. The model trained in the objective of minimizing the binary cross-entropy loss over both positive and negative pairs. Training was performed in batches of size 4,000 to utilize GPU parallelism and improve convergence speed.

6.5 Quality Report

One of the tools used to evaluate the fidelity and utility of the synthetic data generated by each model, SDMetrics [33] was employed. SDMetrics is an open-source Python library developed by the SDV (Synthetic Data Vault) team. SDMetrics provides a comprehensive framework for evaluating the quality of synthetic data by comparing them to real data through a variety of

statistical and distributional metrics. It also provides useful visualization tools for easier and demonstrative understanding of the results.

One of the key features of SDMetrics is the Quality Report. It offers a high-level overview of how well synthetic data replicates the properties of the real dataset. These are metrics that evaluate, in essence, the fidelity of the data. The Quality Report includes metrics regarding column-wise distributions as well as pair-wise correlations.

6.5.1 Column Shape

The first Quality Report property is the Column Shape. The shape of a column describes its overall distribution. It computes a score of the similarity of each column of the real dataset with its corresponding column of the synthetic dataset. For numerical columns, the KSComplement metric is used as opposed to the boolean and categorical attributes for which the TVComplement metric extracted the score.

The KSComplement computes the similarity of a real column compared to a synthetic column as far as column shape is concerned. It uses the Kolmodorov-Smirnov statistic, a nonparametric test that quantifies the distance between the empirical cumulative distribution functions (CDFs) of two datasets. To compute it, the framework converts the numerical distributions of the two columns currently under investigation into their CDFs and calculates the maximum difference between them. Mathematically, given the two CDFs of the distributions, $F_1 \; and \; F_2$, the KS statistic is defined as

$$D = \sup_x |F_1(x) - F_2(x)| \tag{6.5.1.1}$$

The TVComplement computes the similarity between two categorical columns via the Total Variation Distance (TDV). To get the TVD, this test computes the frequency of each category value and represents them as probabilities. It then compares the differences in probabilities given the following formula

$$\delta(R, S) = \frac{1}{2} \sum_W |R_w - S_w| \tag{6.5.1.2}$$

where w represents every possible category in a column W. R and S are the observed frequencies from the real and synthetic data respectively. The framework returns higher score for higher similarity so it outputs $1 - \delta(R, S)$.

6.5.2 Column Pair Trends

Quality Report of SDMetrics provides a second important evaluation property called Column Pair Trends. It examines the generated synthetic data attributes as pairs and produces scores and insight on the relationship between each pair. It complements the Column Shape property creating a more well-rounded evaluation approach. Single column examination is inevitably important as it shows how the generative model understands individual columns. It can be viewed as the first layer of data fidelity. However, the performance of a model should be also tested on its comprehension of column relationships. Keeping good individual column distributions does not guarantee in any case truthful individual records as the joint distributions of attributes are ignored.

The Column Pair Trends property applies different methodologies based on the data type of each attribute. It captures correlation between numerical columns using Correlation Similarity and, respectively, it uses ContingencySimilarity to review the correlation between two categorical or boolean attributes. In order to process numerical columns in relation to categorical ones, it discretizes the numerical values into bins and uses them as categorical values, computing ContingencySimilarity between the two attributes.

The CorrelationSimilarity is nothing but a normalized correlation difference. The framework computes the correlation of the two attributes under examination on both the real and synthetic datasets. It then normalizes and returns their score in the following way

$$score = 1 - \frac{|S_{A,B} - R_{A,B}|}{2} \tag{6.5.2.1}$$

For the ContingencySimilarity the framework first computes the normalized contingency table for the real and synthetic data, given two columns A and B. The contingency table contains the observed instances of each value pair between the two columns. After that, it computes a score of difference between the two tables, the one from synthetic and the one from real data. The score is computed using the Total Variation Distance (TVD), also used for the TVComplement score of the Column Shape property, leading to the following score

$$score = 1 - \frac{1}{2}\sum_{A}\sum_{B}\left|S_{a,b} - R_{a,b}\right| \tag{6.5.2.2}$$

where a represents all the possible values of attribute A and b the ones of attribute B.

6.6 Privacy Evaluation

6.6.1 Distance to Closest Record

The Distance to Closest Record (DCR) [33] technique is used to identify how identical the records of the synthetic dataset are to those of the real dataset. DCR is a simple privacy evaluation technique that measures how far each record of the synthetic data is from any real record. DCR equal to 0 means real information leakage by the synthetic record, while higher values indicate stronger privacy prevention. It is important to note that this technique is not a completely truthful way to detect data copying of the generative model.

The score calculated for the performance of the model, based on its synthetic data, is the mean DCR of every record with respect to every instance of the real dataset. The Gower's Distance [34] is used to calculate the score of similarity between two rows of the datasets. For each feature f of the features F and between the records I and j, the Gowers distance is computed separately for numerical and categorical columns.

For numerical columns

$$GDn_{i,j} = \sum_{F} 1 - \frac{\left|x_{if} - x_{jf}\right|}{R_f} \tag{6.6.1}$$

where x_{if} represents the value of feature f of the record I and R_f represents the range of the values of the feature.

For categorical columns $GDc_{i,j} = 1$ if $x_{if} = x_{jf}$ or $GDc_{i,j} = 0$ if $x_{if} \neq x_{jf}$

This method iterates through all the records of the real data for every record of the synthetic data to compute the Gower's Distance for each pair of records. It then keeps the minimum score for every synthetic record and averages all these scores to finally compute the mean DCR.

6.6.2 Exact Matches

Another simple, yet informative, evaluation method on the privacy prevention of synthetic data is the exact match count of the records between the real and the synthetic data. A high number of exact matches may indicate high privacy risk as the model could be memorizing and leaking real data records rather than generating new samples. Ideally, a generative model should preserve a good balance between utility and privacy, resembling real data without any real information leakage.

Chapter 7

Results

In the following pages, the evaluation results of the synthetic data generated by each model are presented. Alongside, the hyperparameters and architectural specifics are displayed. These factors were determined after following the methodology described in the previous sector. Each model was fine-tuned through a combination of parameter exploration and guidance from the original papers presenting the models, at least for the existing ones. The resulting datasets are analyzed based on multiple metrics, also mentioned in the previous sector, assessing the similarity between them and the real dataset. Emphasis was given to the fidelity of the generated distributions, the correlation between attributes that each model presented and the diversity of the samples.

This chapter presents the evaluation of the performance of all models on different use cases. First, models are trained on the mixed type 'Adult' dataset, and their generated data are compared with the metrics presented in the previous chapter. After evaluating the models on the mixed-type dataset, the numerical attributes are removed, and the models are trained again and generate new synthetic data. A similar evaluation is conducted again to observe the performance of the models on categorical-only datasets. Also, the results are compared to the ones of the full dataset in order to study how the models perform on the same dataset if all numerical attributes are removed. However, to test the models on a more challenging categorical-only tabular dataset, the models were trained again with minor tuning on the 'Mushroom' dataset that includes more rows with higher cardinality.

7.1 Model Tuning

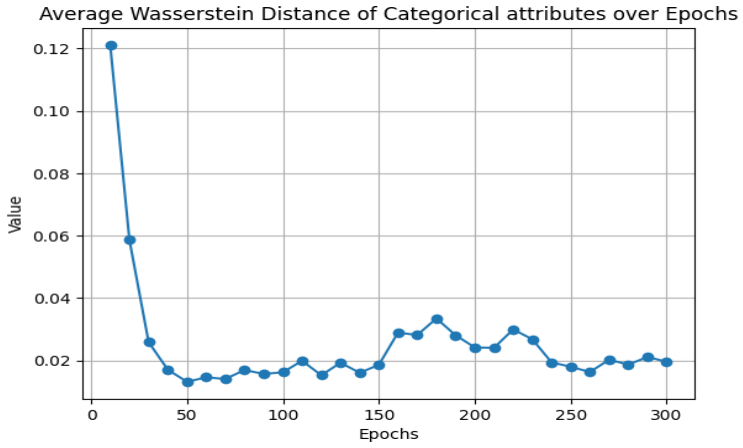
7.1.1 CTGAN

Parameter	Value
generator learning rate	10^{-5}
discriminator learning rate	10^{-5}
discriminator dropout	0.5
batch size	4000

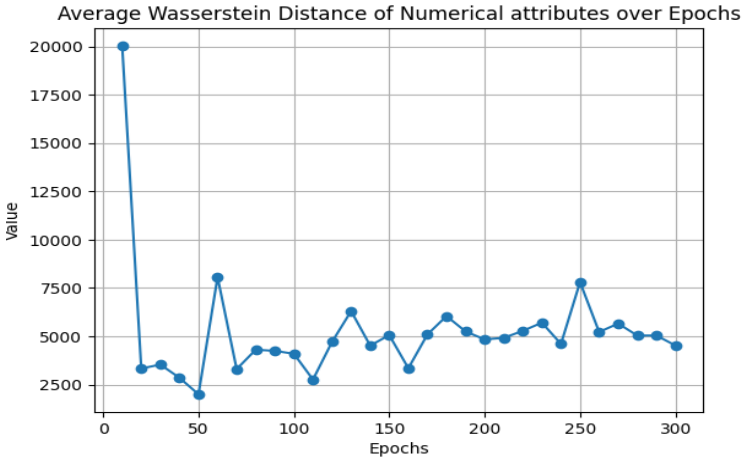
pac size	10
gradient penalty lambda	10

Table 4: CTGAN model parameters

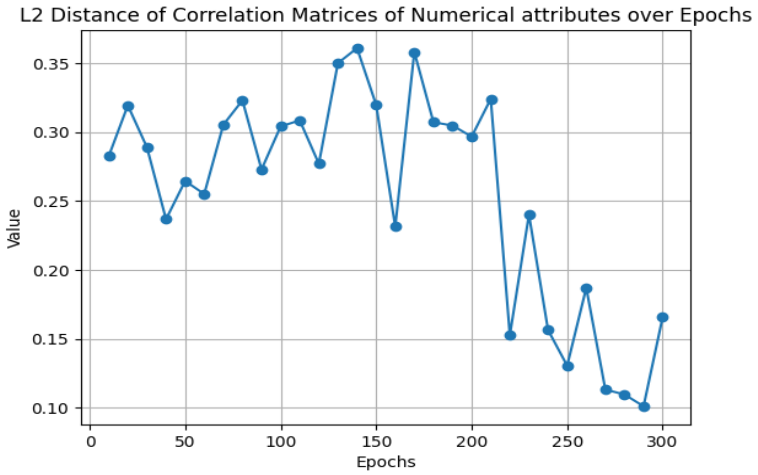
The Generator of the CTGAN Model consists of 4 layers in each one of which 512, 1024, 1024 and 512 nodes were added respectively compared to their previous layer, while the discriminator required a smaller architecture of two layers with 512 nodes each. The training process interchanged between Generator and Discriminator training with the same learning rate. However, the Discriminator completed two steps of training at each epoch.



a



b



c

Figure 27: CTGAN training progress based on average Wasserstein Distance between Numerical (b) and Categorical (a) columns and Euclidean Distance of Correlation Matrix of Numerical columns (c).

The model was trained on a total of 300 epochs. Every 10 epochs 1,000 samples were generated and the Wasserstein distance between them and the real data was measured as well as the Euclidean distance between the Correlation Matrices. During training, while the measured Wasserstein distances quickly reduced to satisfactory values, the model required some more epochs to also approach the correlations between the attributes better. GAN training is hard to stabilize, however, using Wasserstein GAN with Gradient Penalty, Batch Normalization in the Generator, Dropout in the Discriminator and PacGAN architecture led to better results.

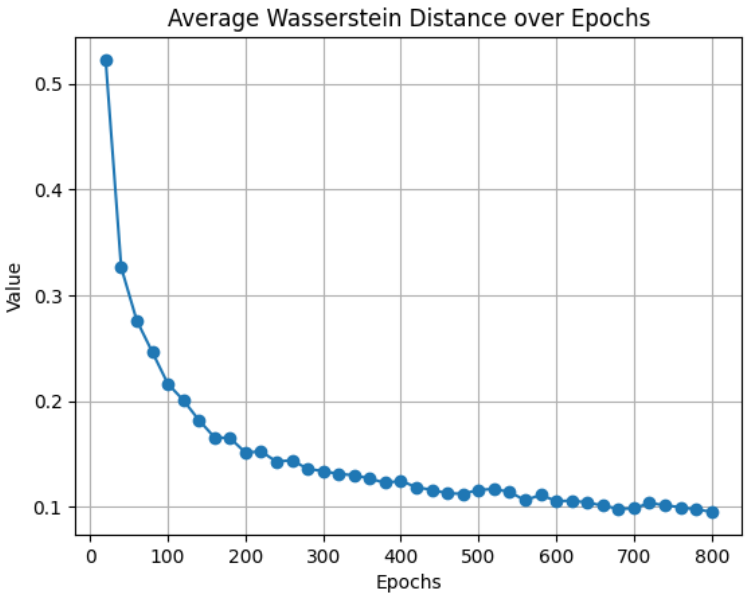
7.1.2 eGAN

Parameter	Value
generator learning rate	10^{-4}
discriminator learning rate	$4 * 10^{-4}$
discriminator dropout	0.3
batch size	2000
gradient penalty lambda	2

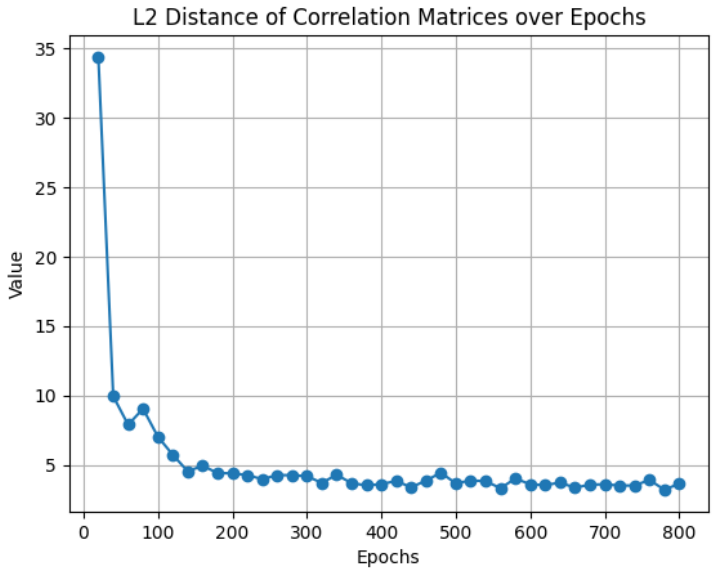
Table 5: eGAN model parameters

The eGAN model's Generator consists of 5 layers with 256, 512, 1024, 512 and 256 nodes each, whereas the Discriminator has 3 layers of size 512. As opposed to CTGAN, the discriminator is

trained 1 step at each epoch. However, to account for the less training, a larger learning rate is used compared to the Generator.



a



b

Figure 28: eGAN training progress based on average Wasserstein Distance between all columns (a) and Euclidean Distance of Correlation Matrix of all columns (b).

The eGAN model was trained on 800 epochs. After every 20 epochs, 1,000 samples were generated. However, these embedded samples were not transformed back to their corresponding

categorical values. That contributed to calculating the average Wasserstein distance over all columns (both numerical and embedded categorical) as well as including the categorical attributes (in their embedded form) in the calculation of the Correlation Matrix and the Euclidean distance between it and the one of the embedded original dataset.

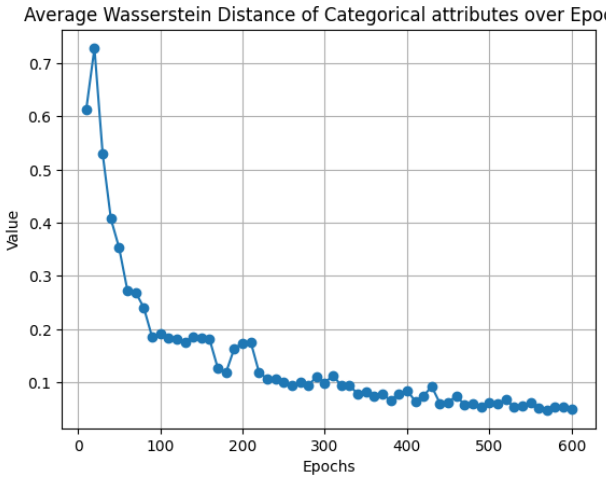
In contrast to the CTGAN model, eGAN captures the relationships between the attributes much faster. It is evident that the plot has a steep fall early on in the second graph depicting the L2 Distance of the two Correlation Matrices. Even though it seems like it reaches a bottleneck, further training was needed to reduce the average Wasserstein Distance as shown in the first graph. It was also observed during tuning that smaller leaning rate kept the Wasserstein Distance undesirably high. Larger learning rate posed an early bottleneck in the correlation distance, also producing untruthful results.

7.1.3 TVAE

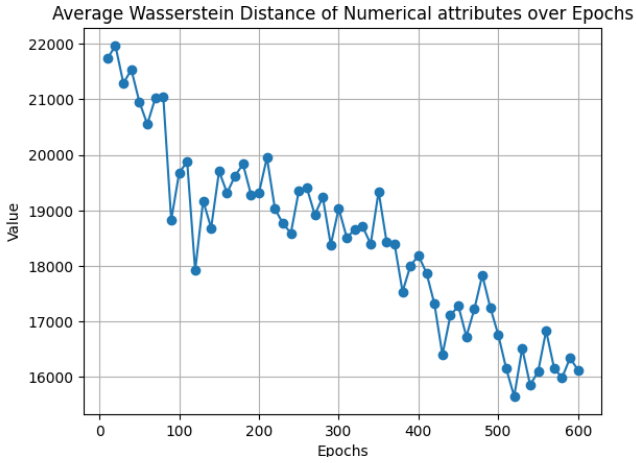
Parameter	Value
learning rate	10^{-3}
batch size	4000
loss factor	2
sigma clamp	$[0.01, 1.0]$
embedding dimension	128

Table 6: TVAE model parameters

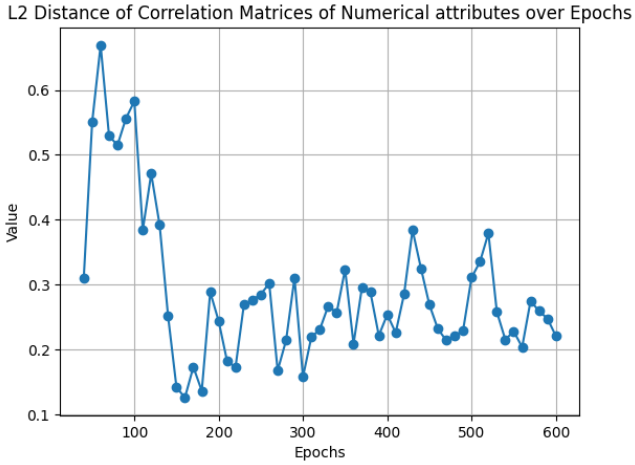
The TVAE model consists of an encoder and decoder. They both have 3 layers of sizes 1024, 512 and 256, however, the encoder acting as a compressing component uses them in the given order in contrast to the decoder, which acts as e decompressing component and uses them in an inverted order. The sigmas' values output by the decoder are clipped each time between the values 0.01 and 1. The loss factor represents the weight assigned to the negative log-likelihood compared to the KL-Divergence term of the loss.



a



b



c

Figure 29: TVAE training progress based on average Wasserstein Distance between Numerical (b) and Categorical (a) columns and Euclidean Distance of Correlation Matrix of Numerical columns (c).

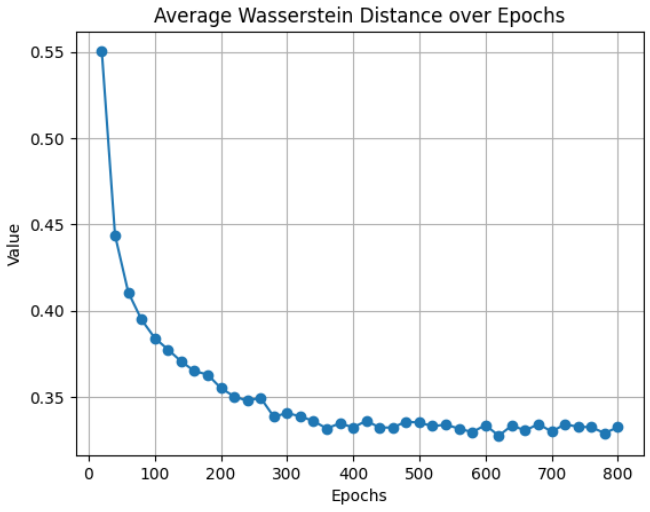
The TVAE model required 600 to converge based on our metrics. A tradeoff between Correlation and Wasserstein Distances was selected during tuning the parameters of the model. Lower loss factors showed worse performance on individual attributes, something that the Wasserstein Distance indicated, as opposed to higher loss factors, which focused more on individual attributes without really providing better performance than the final selected value.

7.1.4 eVAE

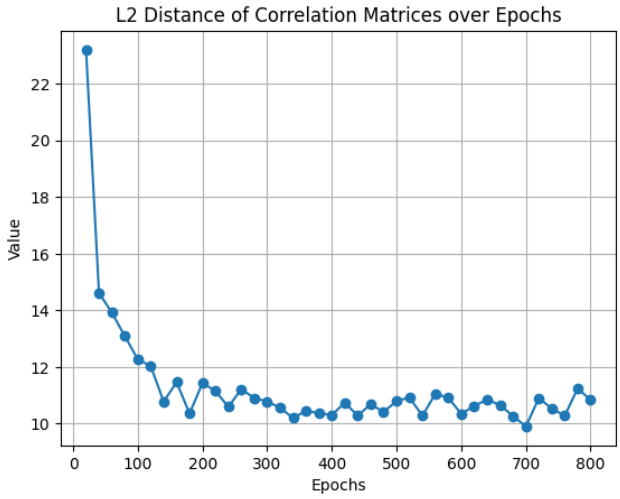
Parameter	Value
learning rate	10^{-3}
batch size	2000
loss factor	3
sigma clamp	$[0.01, 1.0]$
embedding dimension	128
encoder dropout	0.2

Table 7: eVAE model parameters

The eVAE model was heavily based on TVAE as far as coding and tuning is concerned. A different batch size and loss factor is used, while the eVAE also integrated dropout with a rate of 0.2. Regarding model dimensions and number of parameters, the decoder and encoder components also remained identical. The main difference is that the weight introduced by the loss factor was applied on the KL-Divergence term, instead of the negative log-likelihood one.



a



b

Figure 30: eVAE training progress based on average Wasserstein Distance between all columns (a) and Euclidean Distance of Correlation Matrix of all columns (b).

At this part, the eVAE model seems closer to the eGAN model than its blueprint, the TVAE. It was also trained on 800 epochs, however, convergence is present earlier in the training process. It showed increased stability compared to the TVAE and its graphs resemble more the ones of eGAN, even though the latter seems to be managing better in reducing the metrics.

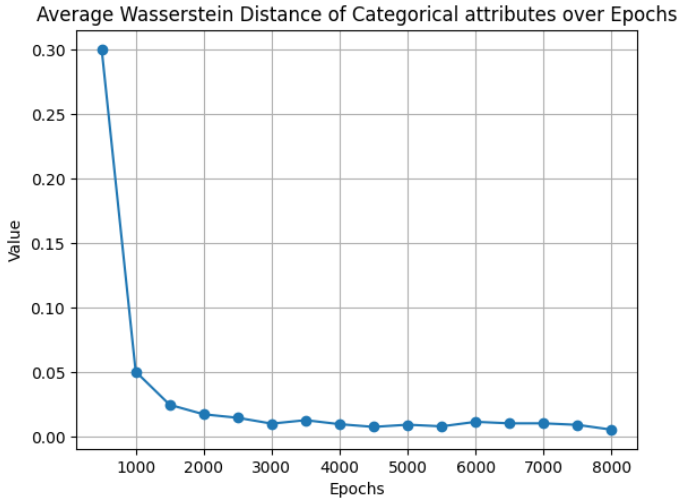
7.1.5 TabDDPM

Parameter	Value
learning rate	10^{-3}
batch size	2000
T	400
beta scheduler	cosine

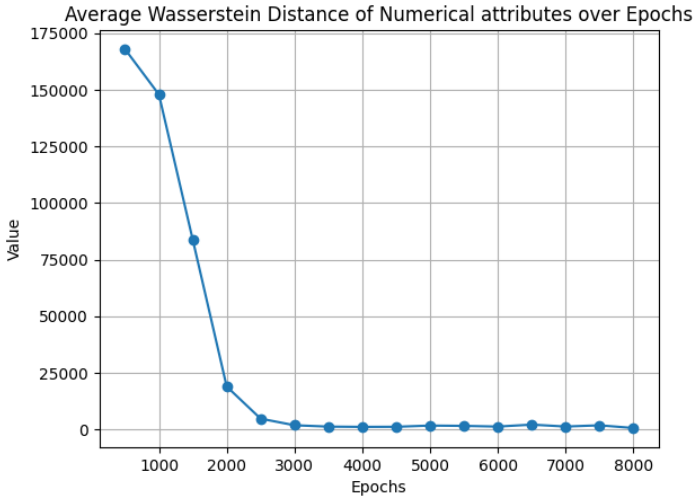
Table 8: TabDDPM model parameters

The TabDDPM model showed faster training in terms of time per epoch, though slower as far as information gained per epoch is concerned. So, using 8,000 epochs, which is 10 times or more compared to the other models did not affect the fairness of evaluation of the models. Also, using a small number of timesteps T in the diffusion process led to poor and unstable training as each timestep introduced large noise. More timesteps made learning slower but, even though they enhanced the stability, they compromised some detail of the end result.

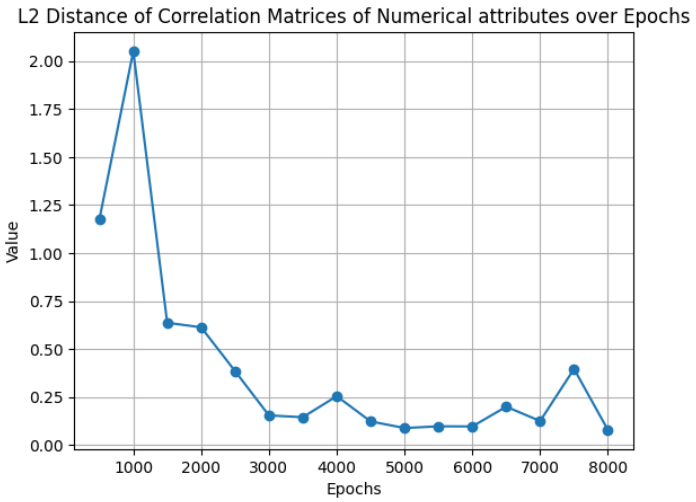
This DDPM model consists of 4 layers of sizes 512, 1024, 1024 and 256 and uses a cosine beta scheduler. The beta parameters determine how much noise is injected in every step of the forward diffusion process. Using a cosine scheduler will lower the noise on early steps, making it easier for the model to capture the structure of the data. With a gradually increasing noise the model learns smoother denoising and converges in earlier steps.



a



b



c

Figure 31: TVAE training progress based on average Wasserstein Distance between Numerical (b) and Categorical (a) columns and Euclidean Distance of Correlation Matrix of Numerical columns (c).

1,000 samples were generated every 500 epochs. Based on the graphs, the model has almost converged on very satisfying metric already at the 4000th epoch. Even so, having the model taken little time to reach this state, the remaining epochs contributed to decreasing the Wasserstein distance between Numerical columns a bit more.

It is also important to note that since the adult dataset is used for classification tasks, the TabDDPM model used the "income" attribute as its y label and was trained based on this classification objective.

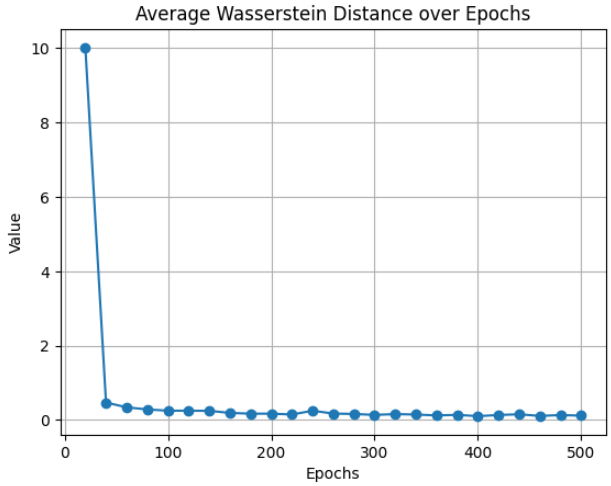
7.1.6 eDDPM

Parameter	Value
learning rate	$5 * 10^{-5}$
batch size	1000
T	1000
beta scheduler	linear

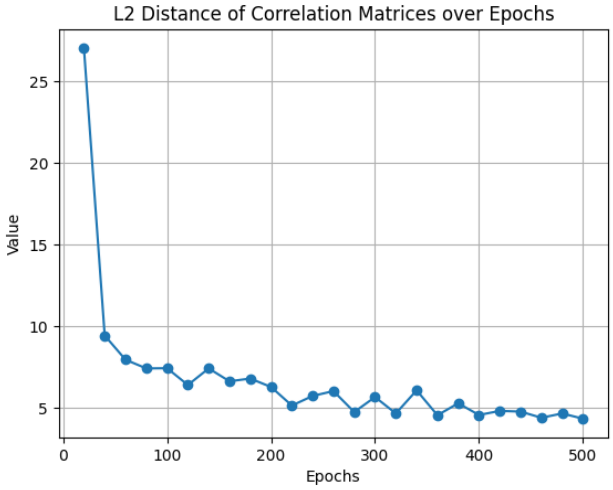
Table 9: eDDPM model parameters

The eDDPM model was trained in 500 epochs using a linear beta scheduler with values between 10^{-5} and 0.01. During training the denoiser of the model, the timestep selected for each training instance was not selected uniformly. A linear series of weights was attached on the timesteps giving more emphasis on later ones. Later timesteps were sampled with an increased probability

compared to earlier ones. This works as earlier timesteps are easier for the model to learn, so it was encouraged to pay attention to the objective it struggles more to learn. Cosine scheduler did not work well for this DDPM model and using the linear scheduler without the weighted selection made the model output many random and outlier values.



a



b

Figure 32: eVAE training progress based on average Wasserstein Distance between all columns (a) and Euclidean Distance of Correlation Matrix of all columns (b).

Metric for 1,000 samples were computed every 20 epochs. It is evident that already in the 40th epoch, the eDDPM model has reduced significantly the average Wasserstein Distance between the attributes of the real and synthetic data. We can detect very little progress, around this metric at least, for the remaining of the training. Nonetheless, further training was deemed useful as the model managed to improve on correlating the attributes of the dataset. We can see steady progress until it converges around the 500th epoch.

7.2 Metrics Results

7.2.1 Quality Report

7.2.1.1 Adult Dataset

Synthetic Data Model	Column Shapes Score
eGAN	0.95
eVAE	0.86
eDDPM	0.93
CTGAN	0.93
TVAE	0.90
TabDDPM	0.98

Table 10: Quality Report average Column Shape scores for the whole mixed type 'Adult' dataset

First, we measure the performance of the models on the whole mixed type 'Adult' dataset. The results in Table 7 show the average Column Shapes Scores produced by the SDMetrics Quality Report for each generated synthetic dataset of the corresponding model. This metric evaluates how well each model captures the distribution of each individual column based on the real dataset, as explained in previous sectors.

Among the used models, TabDDPM seems to have achieved the highest score (0.98), suggesting that its generated column-wise distributions are almost indistinguishable from the ones of real data. Coming at a close second is the eGAN model (0.95), which is also closely followed by CTGAN and eDDPM, each with the same score (0.93). The VAE based models stayed a bit behind with the eVAE (0.86) having a slightly worse score than its counterpart, TVAE (0.90).

These results indicate that diffusion-based models and GAN architecture can effectively capture marginal distributions, no matter whether they aim to train on embedded datasets, as eGAN and eDDPM do, or they process the different data types in a more distinctive and different way with each other, like CTGAN and TabDDPM.

The custom models (eGAN, eVAE, eDDPM) were designed as the simplified embedded data variants of their counterparts (CTGAN, TVAE and TabDDPM, respectively). The results indicate that, interestingly, their performance is directly comparable to the models they emulate. For instance, eDDPM and eVAE, stayed really close to their counterparts, trailing only by 5% and 4% in Column Shape Scores, whereas the eGAN model outperformed its counterpart by 2%. These findings suggest that even with architectural or training simplifications, such as embedding-based

input representations, the core models maintain strong performance on column-wise distribution matching.

Figure 33: Quality Report Scores of every model for each column of the dataset. Dark blue bars represent numerical attributes and light blue represent categorical attributes

Studying the Column Shape Scores individually for each column, it is evident that all models perform better in categorical columns than numerical ones. However, this depends on the emphasis given on the two different data types. In all synthetic data, the column “hour-per-week” appears to have the worst, or at least one of the worst, scores. This numerical column is quite unique for the dataset due to its distribution, as it makes it hard to approach.

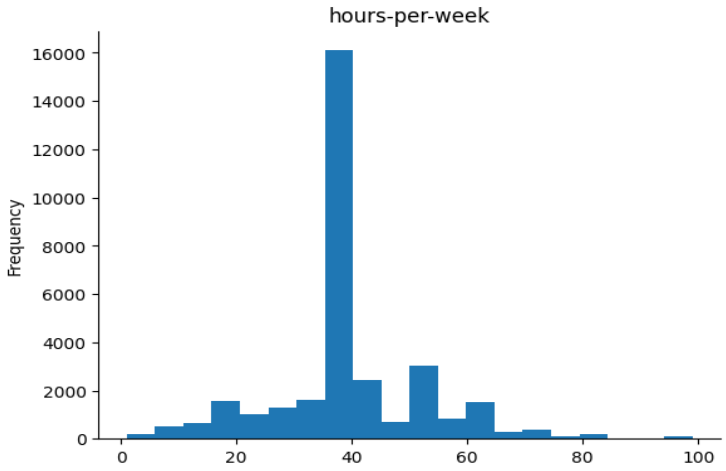


Figure 34: Distribution of values of the numerical column "hours-pe-week" of the real dataset

The distribution of the column "hours-per-week" has a spike at value 40 and the rest of the values are represented at around a 7 times smaller rate. The custom models, using the embedded dataset as input struggle to approach this distribution due to their simplified architecture. As opposed to their counterparts, eGAN and eVAE do not use mode-specific-normalization. This technique is based on Gaussian Mixture Models and simplifies the objective for the final model making it easier for CTGAN and TVAE to capture such extreme distributions while the embedded models produce smoother results.

7.2.1.2 Categorical-only Adult Dataset

Synthetic Data Model	Column Shapes Score
eGAN	0.98
eVAE	0.90
eDDPM	0.98
CTGAN	0.93
TVAE	0.85
TabDDPM	0.99

Table 11: Column Shapes scores for the categorical-only 'Adult' dataset

After cleaning the dataset of all numerical columns, retraining the models and generating new synthetic datasets of categorical-only attributes, the Column Shapes scores of the above table are obtained. Again, the TabDDPM (0.99) has performed the best among all models. However, the key takeaway from these results is the increase of the scores of the embedding-based models.

All three models have demonstrated stronger results with eGAN (0.98) and eDDPM (0.98) showing a 0.03 increase and trail only by 0.01 from the TabDDPM model. eGAN (0.93) did not have any downfall in its performance. TVAE (0.85) performed worse by 5% than in the mixed-type dataset while at the same time, eVAE (0.90) increased its score by 4%.

7.2.1.3 Mushroom Dataset

Synthetic Data Model	Column Shapes Score
eGAN	0.94
eVAE	0.95
eDDPM	0.95
CTGAN	0.94
TVAE	0.96
TabDDPM	0.98

Table 12: Column Shapes scores for the 'Mushroom' dataset

Evaluating the synthetic 'Mushroom' datasets generated by both baseline and proposed models, TabDDPM (0.98) scored the highest Columns Shapes Score again. Notably, all the other models performed extremely good as well, indicating really strong performance of the embedding-based models even in approaching the distributions of many attributes in a wide categorical domain.

7.2.2 Column Pair Trends

7.2.2.1 Adult Dataset

Synthetic Data Model	Column Pair Trends Score
eGAN	0.90
eVAE	0.78
eDDPM	0.88
CTGAN	0.85
TVAE	0.85
TabDDPM	0.97

Table 13: Average Column Pair Trends scores for the whole mixed type 'Adult' dataset

The results shown in the table above were the scores drawn from a custom implementation of the Column Pair Trends property of SDMetrics [33]. In fact, it does not have any customizations

from what was described previously. It is a straightforward adaptation of the given code on its corresponding github repository.

The Column Pair Trends Scores reported above measure how well each model captures the relationships between pairs of columns in the synthetic dataset, compared to the original data. Among all models, TabDDPM shows again excellent results with the highest score (0.97), this time significantly more than the second best, which is again eGAN (0.90). TabDDPM seems to be able to handle high-dimesional dependencies really well especially compared to the rest of the models.

All other models demonstrate good performance. eDDPM (0.88) had great results and along with TabDDPM they had the best combined score, making DDPM models a very strong candidate for modeling complex tabular data. eGAN (0.90) and CTGAN (0.85) show relatively strong results and similar performance suggesting that GAN architecture is again well suited for preserving and depicting feature interactions. The two VAE models combined, eVAE (0.78) and TVAE (0.85), performed worse, indicating minor limitations in capturing relationships between features, even with increased latent space dimensionality.

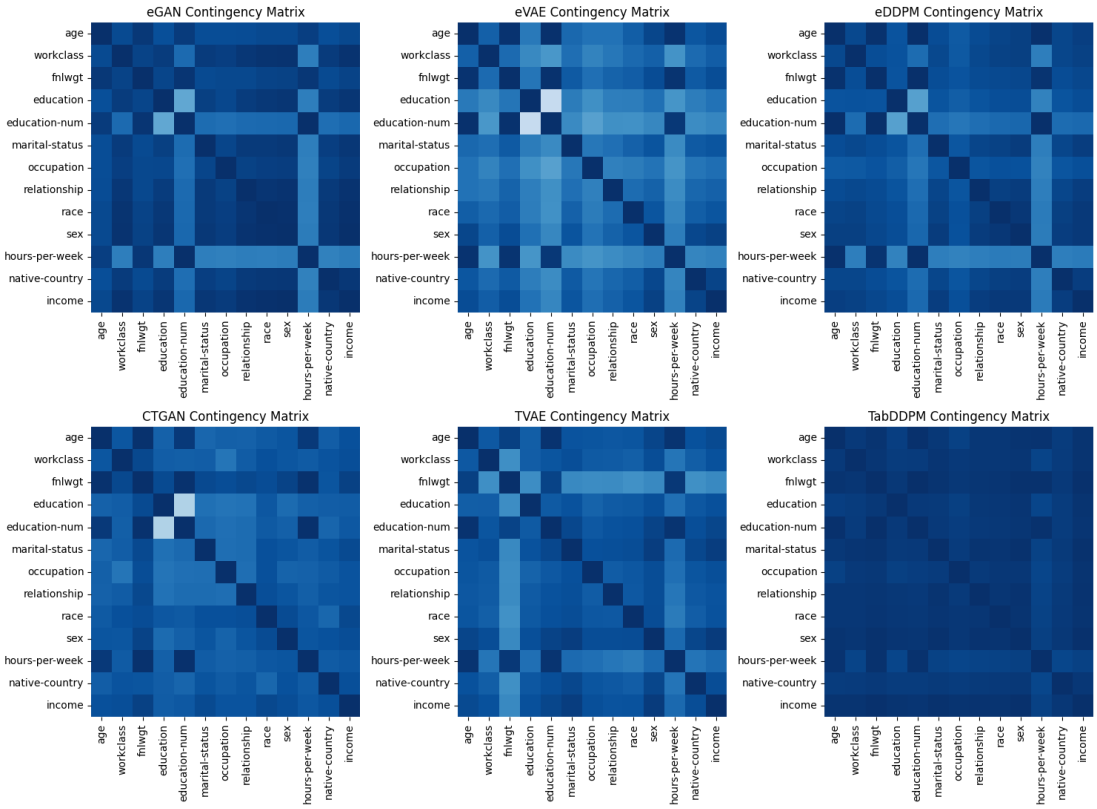


Figure 35: Column Pair Trends matrices for every synthetic dataset compared to the real dataset as calculated by Quality Report for the full mixed-type 'Adult' dataset

The above figures depict how well each synthetic model captures the relationships between different attributes. Each square represents the contingency or correlation similarity score of the attributes it crosses. The darker the color, the higher the score.

It is really important to note that the eGAN model, with the second highest score overall of 0.90, shows excellent performance in capturing relationships between pairs of categorical or pairs of numerical attributes. The only pairs negatively affecting its overall score are pairs between a categorical and a numerical attribute. This is also observed at a smaller amount on the rest of the custom models (eVAE and eDDPM). The eDDPM model also shows a similar behavior with only 2 numerical attributes though, 'fnlwgt' and 'hours-per-week'. However, this pattern, seen in all custom embedding models, is the main factor limiting their performance and suggests the possibility of weakness in the embedding procedure, the handling of embedded values by the models or the transformation of embedded vectors back to categorical values.

An important factor that needs to be considered is the truthfulness of the evaluation metric. The score between a numerical and a categorical attribute is computed as the contingency score of the two columns, after discretizing the numerical values into a chosen number of bins (10 in our case). As demonstrated by Figure 25, numerical attributes can have extreme distributions around specific values. As models only approach this distribution, the bin in which the highly observed values fall each time affects the resulting score significantly. For that reason, even with some certain values of smaller bin size, the score was higher. Analogously, we could observe lower scores with certain larger bin sizes. Both of these observations are quite contradictory to the logic that the lower bin size classifies the values more favorably.

7.2.2.2 Categorical-only Adult Dataset

Synthetic Data Model	Column Pair Trends Score
eGAN	0.95
eVAE	0.85
eDDPM	0.96
CTGAN	0.86
TVAE	0.77
TabDDPM	0.97

Table 14: Average Column Pair Trends scores for the categorical-only 'Adult' dataset

TabDDPM remains the top performing model with a score of 0.97. However, all embedding-based models show significant improvement in performance as opposed to CTGAN and TVAE.

Also, eGAN and eDDPM are insignificantly trailing the TabDDPM making them strong competitors.

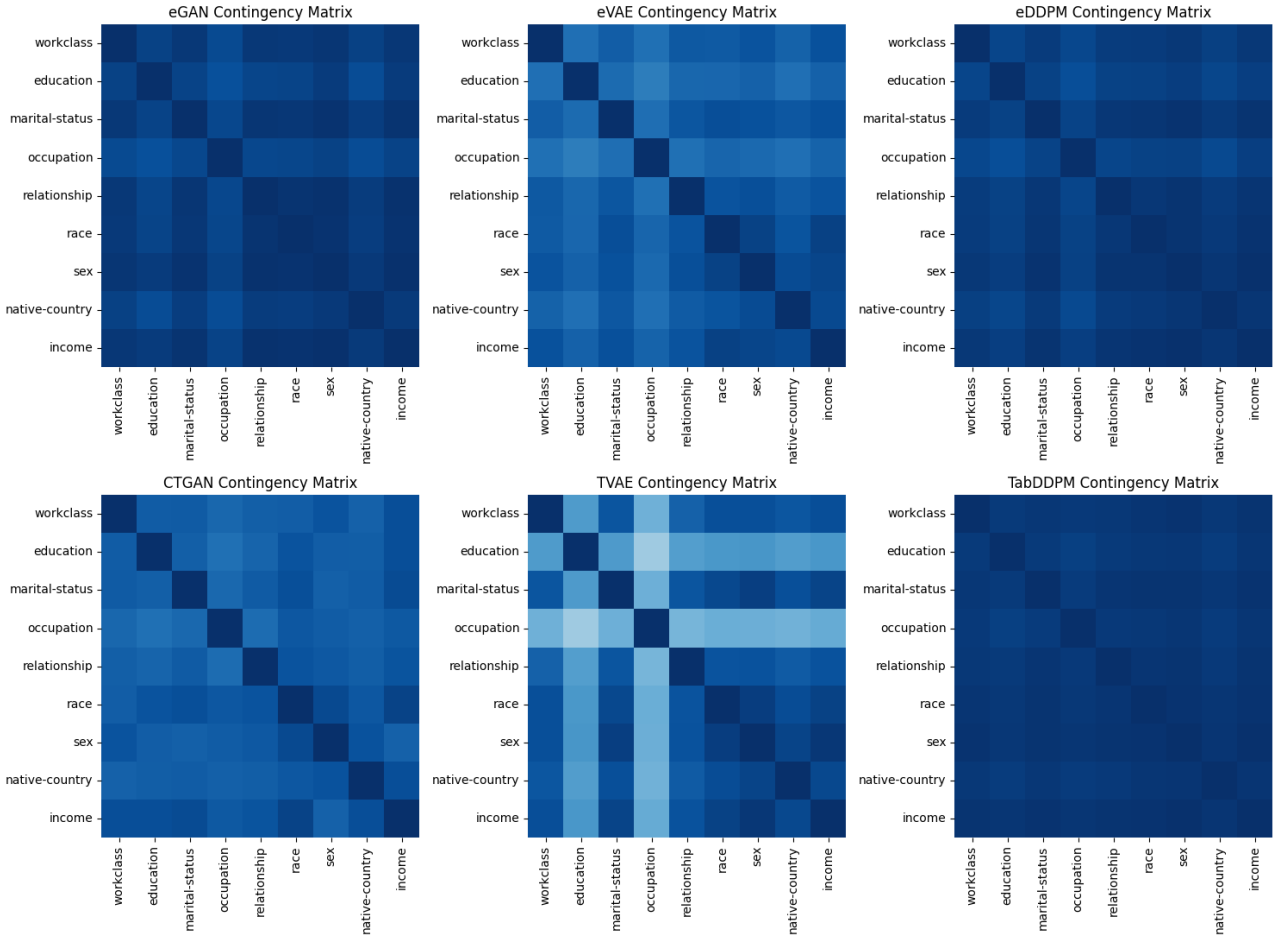


Figure 36: Column Pair Trends matrices for every synthetic dataset compared to the real dataset as calculated by Quality Report for the categorical-only 'Adult' dataset

It is clear that all models, apart from eGAN struggle a bit to capture the relationships related to "education", "occupation" and "native-country". Every model shows a relative balance across the Column Pair Trends scores, except for the TVAE, which has greatly struggled to perform in the first two of three forementioned attributes, making its average score the lowest among all models in this particular case.

7.2.2.3 Mushroom Dataset

Synthetic Data Model	Column Pair Trends Score
eGAN	0.90
eVAE	0.91
eDDPM	0.92

CTGAN	0.88
TVAE	0.92
TabDDPM	0.96

Table 15: Average Column Pair Trends scores for the 'Mushroomt' dataset

We see TabDDPM performing best out of all the models again with a score of 0.96. Even though completing the objective on this dataset, which is more complex and has less instances, is much harder than modeling the categorical-only ‘Adult’ dataset, our proposed models show great performance, absolutely comparable to the baseline models.

As we can see in the following table, depicting the Contingency Similarity Matrices between real and synthetic data, all models struggled a little with the attributes of the highest cardinality, in comparison to the other attributes. Especially the VAE-based models that show lower scores for ‘gill-color’, ‘cap-color’ and ‘odor’. At the same attributes seem to have struggled to some degree the rest of the models as well.

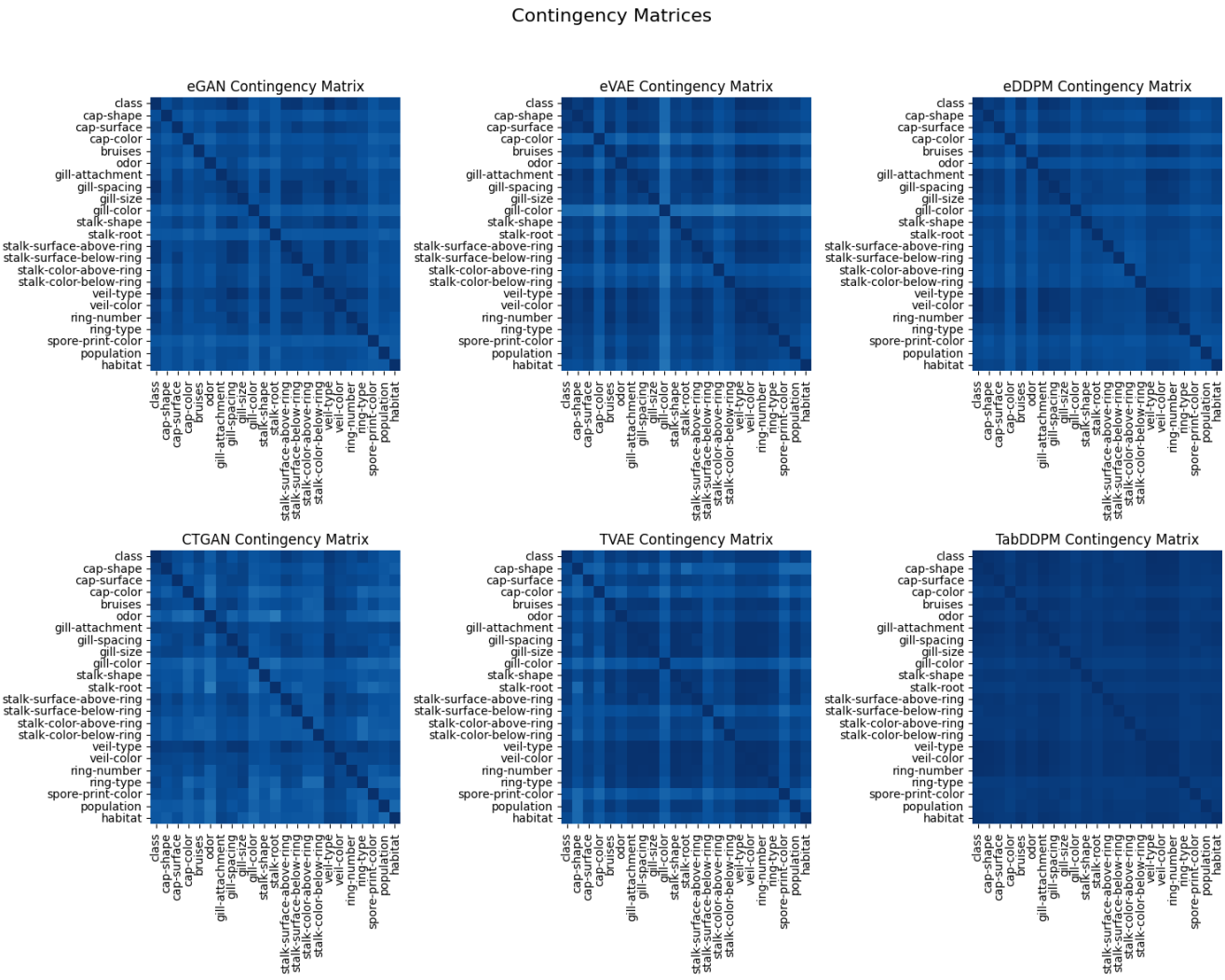


Figure 37: Contingency matrices for every synthetic dataset compared to the real dataset as calculated by Quality Report for the 'Mushroom' dataset

7.2.3 Privacy Results

7.2.3.1 Mean DCR

The mean Distance to Closest Record (DCR) as described before, cannot alone show much information about the privacy prevention level of a model. However, assessing the results, TVAE generates the lowest DCR around 0.014. It is followed by TabDDPM, eDDPM and eGAN which have similar mean DCRs around 0.03. eVAE comes after at an increased 0.06 value while CTGAN comes last with a significantly high score around 0.085.

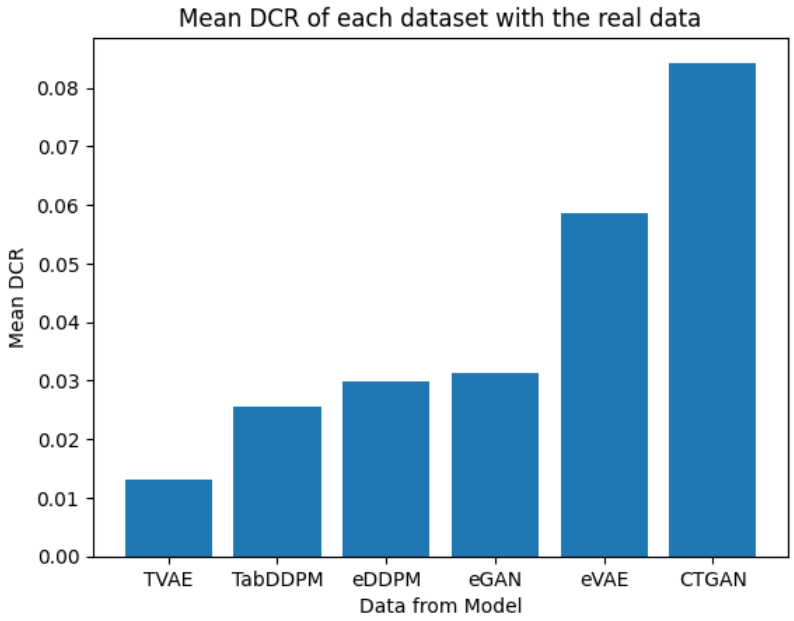


Figure 38: Histogram of the mean DCR values of each dataset in ascending order

High DCR indicates that the records of the synthetic dataset deviate from individual real dataset records, hence protective possible sensitive information and preserving privacy. Of course, it is only a privacy indicator as a random untrained model would score extremely high DCR due to its irrelevant and uncoordinated individual instance generation capacity. For models capturing excellently the distributional characteristics of real data it is really difficult to generate data with high DCR.

The lowest the mean DCR a model can produce, the hardest it is to keep the exact matches low. Generating records close to the original ones, meaning low DCR, makes it more likely for some of them to end up being exact matches of some of the real data records. That is the case for TVAE which has showcased the lowest DCR and highest exact match count. This makes a model a bad choice for privacy preservation.

7.2.3.2 Exact Matches

The lowest the mean DCR a model can produce, the hardest it is to keep the exact matches low. Generating records close to the original ones, meaning low DCR, makes it more likely for some of them to end up being exact matches of some of the real data records. That is the case for TVAE which has showcased the lowest DCR and highest exact match count. This makes a model a bad choice for privacy preservation.

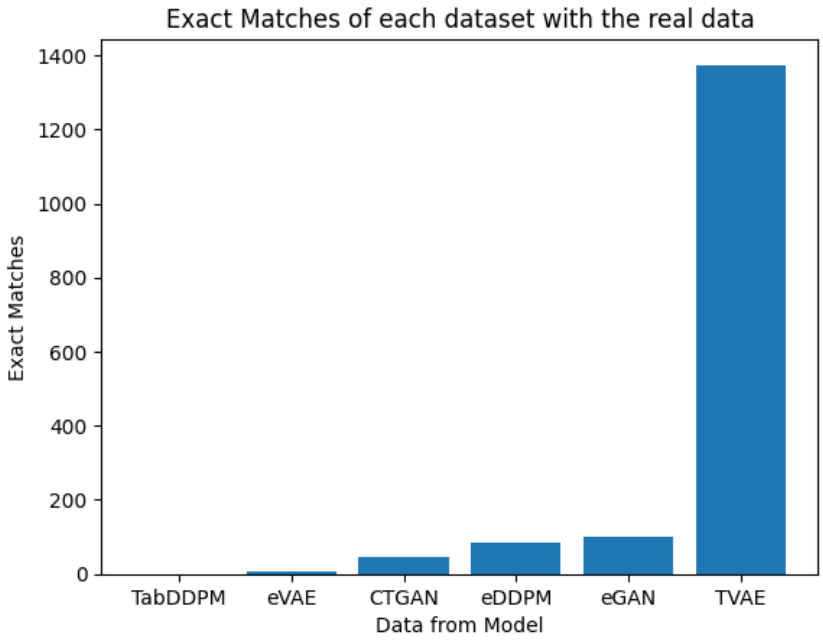


Figure 39: Histogram of the exact matches of records of each dataset in ascending order

It is important to note that for the measurements of the exact matches, the attribute 'fnlwgt' was excluded, since its values are large and diverse, hence producing an exact match would be highly unlikely.

The rest of the models seem to perform satisfactory given the combination of DCR and match counts, and especially the TabDDPM, that sampled 0 exact matches to the original dataset, while keeping the mean DCR at a pleasing high value of a little lower than 0.03. The same can be said for eDDPM and eGAN which generated around only 100 exact matches while maintaining a relatively high DCR. CTGAN also performs really well in these privacy preserving metrics. It showed a really high mean DCR and really low exact match count. Such results can also be present in a poorly performing model. However, this is not the case here as CTGAN demonstrated a good performance also on distributional metrics.

Chapter 8

Conclusions

In this thesis, we explored the landscape of modern generative models and their adaptations on mixed-type tabular data. Specifically, the models that were studied are Variational Autoencoders (VAEs), Generative Adversarial Networks (GANs) and Diffusion Models. Apart from the high performance widely known and appreciated adaptations of these models, namely CTGAN, TVAE and TabDDPM, a custom variation using word embeddings was introduced for each one. This work scrutinizes both theoretical foundations and practical use of the models. Architectural design, implementation details empirical evaluation and results are all presented in detail, leading to conclusions and thoughts on further research.

8.1 Summary of Contributions

We began by presenting a comprehensive understanding of Artificial Intelligence and Machine Learning escalating to the more complex concepts of our advanced generative frameworks of VAEs, GANs and Diffusion Models. We also analyzed their respective mathematical foundations that base the architecture and learning objective of each model.

One of the central challenges addressed by all of these frameworks was the generation of tabular data, which imposes difficulties due to the presence of heterogenous data types (e.g., continuous, categorical, binary). Modeling discrete values with imbalanced distributions and avoiding mode collapse are the most common obstacles such models face.

The custom models (eGAN, eVAE, eDDPM) aim to take advantage of the inherent characteristics of the vanilla models in a more raw and straightforward way. Creating word embeddings and transforming the original data into one catholic data type, the numerical, enables us to use approaches of these models closer to the original ones. By learning dense continuous representations for categorical values, we enabled gradient-based training over formerly discrete inputs.

All models were evaluated on the UCI Adult Income [31] and Mushroom [32] datasets using metrics that included Wasserstein distance, L2 distance between correlation matrices, various metrics from the Quality Report of SDMetrics [33] and, also, privacy heuristics like mean Distance to Closest Record (DCR) and exact record match count.

8.2 Summary of results

The results showed that TabDDPM outperformed all other models in preserving both individual column distributions and feature correlations while maintaining an excellent privacy preserving score with moderate mean DCR and no exact matches. Its embedding architecture counterpart (eDDPM) showed great results in approaching individual column distributions and maintaining privacy of the original data. However, it lacked excellency in copying the relationships between attributes, even though it performed great at that part too.

The GAN based models also showed good overall performance with our eGAN being the second-best overall performer. It maintained a slightly better score in both Column Shape and Column Pair Trends than CTGAN, showing better synthetic data fidelity. On the other hand, CTGAN was arguably the best privacy preserving model, having the highest mean DCR while keeping the exact matches below average.

Both VAE-architecture based models did not manage to keep their performances within distance of the other models', when tested on the Adult dataset. Their average scores in the two Quality Report metrics were the lowest. The embedding-based VAE model was outperformed by TVAE in these tests. Nonetheless, TVAE showed by far the worst performance in the privacy metrics by scoring the lowest mean DCR while generating the most exact record copies, with almost 10 times more than the second worst. Meanwhile, eVAE showed great privacy prevention performance with the second highest mean DCR and the second lowest exact match count, making it arguably the best performing model based on the combination of these metrics.

A common bottleneck in performance for all three custom embedding variation models was their limited ability in capturing the correlation between a numerical and a categorical attribute. As far as datasets with only categorical attributes are concerned, our proposed models showed great performance overall, comparable to these of the baseline models. Especially when tested on the Mushroom dataset, which is more complex and has more attributes of higher cardinality, even eVAE showed really promising results.

8.3 Future Work

- Construct enhanced numerical column modeling through:

    - More inclusive embeddings, with the model involving numerical attributes too, instead of constructing the embedding matrix by training only on categorical columns
    - More complex approach of the numerical attributes from the generative models, integrating multimodal modeling like Gaussian Mixtures, like CTGAN and TVAE do, or Mixture Density Models.
- Achieve dimensionality reduction through column-specific embeddings. Learning embeddings where only values of the same column are mapped on the same latent space can reduce dimensionality and still keep important semantic information of the values.
- Attempt increased diversity and privacy prevention with softer reverse transformation from embedded vectors back to categorical values.
- Investigate transfer learning approaches where generative models trained on one domain or dataset can be adapted to another domain with limited data.
- Develop methods to quantify uncertainty and instance-wise error. This would detect out-of-distribution samples as well as outliers, and provide more confidence in data quality assessment

Chapter 9

Bibliography

[1] I. J. Goodfellow *et al.*, “Generative Adversarial Networks,” Jun. 2014.

[2] D. P. Kingma and M. Welling, “An Introduction to Variational Autoencoders,” Dec. 2019, doi: 10.1561/2200000056.

[3] E. Poslavskaya and A. Korolev, “Encoding categorical data: Is there yet anything ‘hotter’ than one-hot encoding?,” Dec. 2023.

[4] M. Haenlein and A. Kaplan, “A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence,” *Calif Manage Rev*, vol. 61, no. 4, Aug. 2019, doi: 10.1177/0008125619864925.

[5] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP,” Jun. 2019.

[6] R. Raina, A. Madhavan, and A. Y. Ng, “Large-scale deep unsupervised learning using graphics processors,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, USA: ACM, Jun. 2009, pp. 873–880. doi: 10.1145/1553374.1553486.

[7] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science (1979)*, vol. 349, no. 6245, pp. 255–260, Jul. 2015, doi: 10.1126/science.aaa8415.

[8] P. Lee, “SYNTHETIC DATA AND THE FUTURE OF AI”, doi: 10.1093/oi/authority.20110803095426960.

[9] W. Wei and L. Liu, “Trustworthy Distributed AI Systems: Robustness, Privacy, and Governance,” Feb. 2024.

[10] A. ’Williams, M. ’Miceli, and T. ’Gebru, “The exploited labor behind artificial intelligence,” *Noema*, Oct. 13, 2022.

[11] S. Bharati, M. R. H. Mondal, P. Podder, and V. B. S. Prasath, “Federated learning: Applications, challenges and future directions,” Jun. 2022, doi: 10.3233/HIS-220006.

[12] L. SWEENEY, “k-ANONYMITY: A MODEL FOR PROTECTING PRIVACY,” *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10, no. 05, pp. 557–570, Oct. 2002, doi: 10.1142/S0218488502001648.

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic Minority Over-sampling Technique,” *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[14] N. ’Manchev, “Enhancing model accuracy with SMOTE oversampling techniques,” *Domino.ai*, May 19, 2022.

[15] F. Lucini, “The Real Deal About Synthetic Data,” *MIT Sloan Management Review*, Oct. 20, 2021.

[16] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990, doi: 10.1109/5.58337.

[17] M. T. Almuqati, F. Sidi, S. N. M. Rum, M. Zolkepli, and I. Ishak, “Challenges in Supervised and Unsupervised Learning: A Comprehensive Overview,” *Int J Adv Sci Eng Inf Technol*, vol. 14, no. 4, pp. 1449–1455, Aug. 2024, doi: 10.18517/ijaseit.14.4.20191.

[18] D. Reynolds, “Gaussian Mixture Models,” in *Encyclopedia of Biometrics*, Boston, MA: Springer US, 2009, pp. 659–663. doi: 10.1007/978-0-387-73003-5_196.

[19] C. M. Bishop, “Mixture Density Networks,” 1994, Accessed: May 19, 2025. [Online]. Available: http://www.ncrg.aston.ac.uk/

[20] C. Doersch, “Tutorial on Variational Autoencoders,” Jan. 2021.

[21] J. Austin, D. D. Johnson, J. Ho, D. Tarlow, and R. van den Berg, “Structured Denoising Diffusion Models in Discrete State-Spaces,” Feb. 2023.

[22] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, “Deep Unsupervised Learning using Nonequilibrium Thermodynamics,” Nov. 2015.

[23] M. Chen, S. Mei, J. Fan, and M. Wang, “An Overview of Diffusion Models: Applications, Guided Generation, Statistical Rates and Optimization,” Apr. 2024.

[24] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space,” Sep. 2013.

[25] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, “Modeling Tabular data using Conditional GAN,” Oct. 2019.

[26] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein GAN,” Dec. 2017.

[27] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, “TabDDPM: Modelling Tabular Data with Diffusion Models,” Oct. 2024, doi: 10.5555/3618408.3619133.

[28] J. Ho, A. Jain, and P. Abbeel, “Denoising Diffusion Probabilistic Models,” Dec. 2020.

[29] E. Hoogeboom, D. Nielsen, P. Jaini, P. Forré, and M. Welling, “Argmax Flows and Multinomial Diffusion: Learning Categorical Distributions,” Oct. 2021.

[30] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed Representations of Words and Phrases and their Compositionality,” Oct. 2013.

[31] “B. Becker and R. Kohavi. ‘Adult,’ UCI Machine Learning Repository, 1996. [Online]. Available: https://doi.org/10.24432/C5XW20.”

[32] “‘Mushroom,’ UCI Machine Learning Repository, 1981. [Online]. Available: https://doi.org/10.24432/C5959T.”

[33] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, “Data Synthesis based on Generative Adversarial Networks,” Jul. 2018, doi: 10.14778/3231751.3231757.

[34] M. D’Orazio, “Distances with mixed type variables some modified Gower’s coefficients,” Jan. 2021.

Appendices

Appendix A

```python
def train(self):
    num_samples = len(discrete_df)
    counter = 0

    for epoch in range(self.num_epochs):
        total_loss = 0

        for i in range(0, num_samples, self.batch_size):
            counter+=1
            batch_df = discrete_df.iloc[i:i+self.batch_size]
            if len(batch_df) == 0:
                continue

            training_pairs = self.get_training_pairs(self.pairs_pc, batch_df)
            targets = training_pairs[:, 1]
            negative_samples = self.get_negative_samples(targets, self.num_negatives)

            tp_exp = training_pairs[:, 0].repeat_interleave(negative_samples.shape[1])
            negative_samples = negative_samples.flatten()
            negative_samples = torch.stack([tp_exp, negative_samples], dim=1)

            self.optimizer.zero_grad()

            pos_scores = self.forward(training_pairs[:,0], training_pairs[:,1], batched=True)
            pos_loss = self.criterion(pos_scores, torch.ones_like(pos_scores))

            neg_scores = self.forward(negative_samples[:,0], negative_samples[:,1], batched=True)
            neg_loss = self.criterion(neg_scores, torch.zeros_like(neg_scores))

            loss = (pos_loss + neg_loss)
            loss.backward()
            self.optimizer.step()

            total_loss += loss.item()
```

Appendix A: Training loop of the proposed Embedding Model.
Available: https://github.com/stavroskout/Synthetic-Tabular-Data

Appendix B

```python
def get_training_pairs(self, pairs_pc, total_batch):
    """
    Sample positive training pairs (center-target) from random column pairs.

    Args:
        pairs_pc (float): Percentage of column pairs to sample.
        total_batch (pd.DataFrame): Current batch of data.

    Returns:
        Tensor: Training pairs of word indices with shape (num_pairs, 2).
    """
    num_pairs = int(len(self.column_pairs)*pairs_pc)
    pairs_set = random.sample(self.column_pairs, k=num_pairs)
    training_pairs = []
    for c, t in pairs_set:
      centers = total_batch[c].map(self.word_to_idx)
      targets = total_batch[t].map(self.word_to_idx)
      pairs = torch.stack([
          torch.tensor(centers.values, device=self.device),
          torch.tensor(targets.values, device=self.device)
      ], dim=1)
      training_pairs.append(pairs)

    # Concatenate all pairs from all column combinations
    training_pairs = torch.cat(training_pairs, dim=0)
    return training_pairs


def get_negative_samples(self, positive_idxs, num_negatives):
    """
    Generate negative samples for each positive target index.

    Args:
        positive_idxs (Tensor): Indices of positive target words.
        num_negatives (int): Number of negatives to sample per positive.

    Returns:
        Tensor: Negative samples with shape (batch_size, num_negatives).
    """

    neg_samples = []
    numbers, probabilities = zip(*self.f_matrix)
    for _ in range(num_negatives):
        r = random.choices(numbers, weights=probabilities, k=len(positive_idxs))
        neg_samples.append(r)
    return torch.tensor(neg_samples, device = self.device).T
```

Appendix B: Core helper functions of the proposed Embedding Model class.
Available: https://github.com/stavroskout/Synthetic-Tabular-Data