7. Deep Generative Models

המודלים שהוצגו בפרקים הקודמים הינם מודלים דיסקרימנטיביים, קרי הם מוציאים פלט על בסיס מידע נתון, אך לא יכולים ליצור מידע חדש בעצמם. בניגוד אליהם ישנם מודלים גנרטיביים, שלא רק לומדים להכליל את הדאטא הנלמד גם עבור דוגמאות חדשות, אלא יכולים גם להבין את מה שהם ראו וליצור מידע חדש על בסיס הדוגמאות שנלמדו. ישנם שני סוגים עיקריים מודלים גנרטיביים – מודלים המוצאים באופן מפורש את פונקציית הפילוג של הדאטא הנתון ובעזרת הפילוג מייצרות דוגמאות חדשות, ומודלים שלא יודעים לחשב בפירוש את הפילוג אלא מייצרים דוגמאות חדשות בדרכים אחרות. בפרק זה נדון במודלים הפופולריים בתחום – VAE, GANs ו-Auto-regressive models (PixelCNN and PixelRNN).

יתרונות של VAE: קל לאימון, בהינתן x קל למצוא את z, וההתפלגות של z נתונה בצורה מפורשת.

יתרונות של GAN: התמונות יוצאות באיכות גבוהה, מתאים להרבה דומיינים.

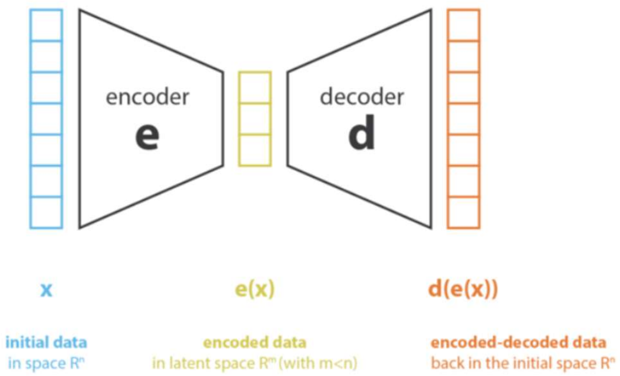
7.1 Variational AutoEncoder (VAE)

המודל הראשון הינו VAE, וכדי להבין אותו היטב יש להסביר קודם מהם Autoencoders, כיצד הוא עובד ומה החסרונות שלו.

7.1.1 Dimensionality Reduction

במקרים רבים, הדאטא אותו רוצים לנתח הוא בעל מימד גבוה, כלומר, לכל דגימה יש מספר רב של פיצ'רים, כאשר בדרך כלל לא כל הפיצ'רים משמעותיים באותה מידה עבור משימות שונות. לדוגמא – מחיר מניה של חברה מסוימת מושפע ממספר רב של גורמים, אך ככל הנראה גובה ההכנסות של החברה משפיע על מחיר המניה הרבה יותר מאשר הגיל הממוצע של העובדים. דוגמא נוספת – במשימת חיזוי גיל של אדם על פי הפנים שלו, לא כל הפיקסלים בתמונת הפנים יהיו בעלי אותה חשיבות לצורך החיזוי. כיוון שקשה לנתח דאטא ממימד גבוה ולבנות מודלים עבור דאטא כזה, הרבה פעמים מנסים להוריד את המימד של הדאטא תוך איבוד מינימלי של מידע. בתהליך הורדת המימד מנסים לקבל ייצוג חדש של הדאטא ממימד יותר נמוך, כאשר הייצוג הזה נבנה על יד הפרמטרים הכי משמעותיים, מתוך שאיפה שהוא ידמה כמה שיותר לדאטא המקורי. יש מגוון שיטות להורדת המימד של הדאטא כאשר הרעיון המשותף לכולם הוא לייצג את הדאטא במימד נמוך יותר (מימד זה נקרא גם המרחב הלטנטי), בו באים לידי ביטוי רק הפיצ'רים המשמעותיים יותר.

בחלק מהשיטות יש שלב נוסף בו לאחר בניית הייצוג במימד נמוך חוזרים למימד המקורי אך עם פחות פרמטרים. התהליך הדו-שלבי נראה כך: דאטא $x \in \mathbb{R}^n$ נכנס למערכת ועובר דרך encoder, ומתקבל $e(x) \in \mathbb{R}^m, (m < n)$. לאחר מכן התוצאה עוברת ב-decoder על מנת להחזיר את התוצאה למימד המקורי, ומתקבל $d\big(e(x)\big) \in \mathbb{R}^n$. אם לאחר התהליך מתקיים $x = d\big(e(x)\big)$ אז למעשה לא נאבד שום מידע בתהליך, ואם $x \neq d\big(e(x)\big)$ אז מידע מסוים אבד עקב הורדת המימד ולא היה ניתן לשחזר אותו בפענוח. האינטואיציה מאחורי המבנה הזה יחסית פשוטה: אם אנו מצליחים לשחזר את הקלט המקורי מהייצוג של במימד נמוך בדיוק טוב מספיק, כנראה שהייצוג במימד נמוך הצליח להפיק את הפיצ'רים המשמעותיים של הדאטא המקורי.



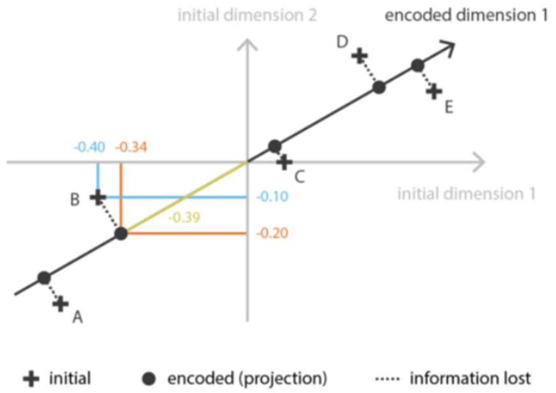
איור 7.1 ארכיטקטורת encoder ו-decoder.

המטרה העיקרית של השיטות להורדת מימד הינה לאמן את זוג ה-encoder-decoder השומרים על מקסימום מידע בעת הקידוד, וממילא מביאים למינימום של שגיאת שחזור בעת הפענוח. אם נסמן בהתאמה E ו-D את כל הזוגות של encoder-decoder האפשריים, ניתן לנסח את בעיית הורדת המימד באופן הבא:

$$(e^*, d^*) = \underset{(e,d)\,\in\, E\times D}{\arg\min}\ \epsilon\Big(x, d\big(e(x)\big)\Big)$$

כאשר $\epsilon\Big(x, d\big(e(x)\big)\Big)$ הוא המרחק בין הדאטא המקורי לבין הדאטא המשוחזר.

אחת השיטות השימושיות להורדת מימד שאפשר להסתכל עליה בצורה הזו היא Principal Components Analysis (PCA). בשיטה זו מטילים דאטא ממימד n למימד m על ידי מציאת בסיס אורתוגונלי במרחב ה-m מימדי בו המרחק האוקלידי בין הדאטא המקורי לדאטא בייצוג החדש הוא מינימלי.



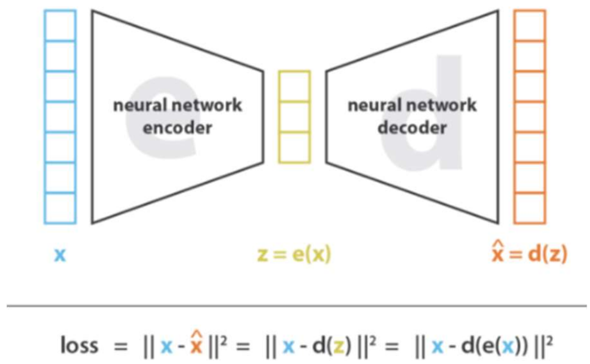
Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

איור 7.2 דוגמא להורדת מימד בשיטת PCA.

במונחים של encoder-decoder, ניתן להראות כי אלגוריתם PCA מחפש את ה-encoder שמבצע טרנספורמציה לינארית על הדאטא לבסיס אורתוגונלי במימד נמוך יותר, שיחד עם decoder מתאים יביא לשגיאה מינימלית במונחים של מרחק אוקלידי בין הייצוג המקורי לבין הייצוג החדש. ניתן להוכיח שה-encoder האופטימלי מכיל את הווקטורים העצמיים של מטריצת ה-covariance של הדאטא, וה-decoder הוא השחלוף של ה-encoder.

7.1.2 Autoencoders (AE)

ניתן לקחת את המבנה של ה-encoder-decoder ולהשתמש ברשת נוירונים עבור בניית הייצוג החדש ועבור השחזור. מבנה זה נקרא Autoencoder:

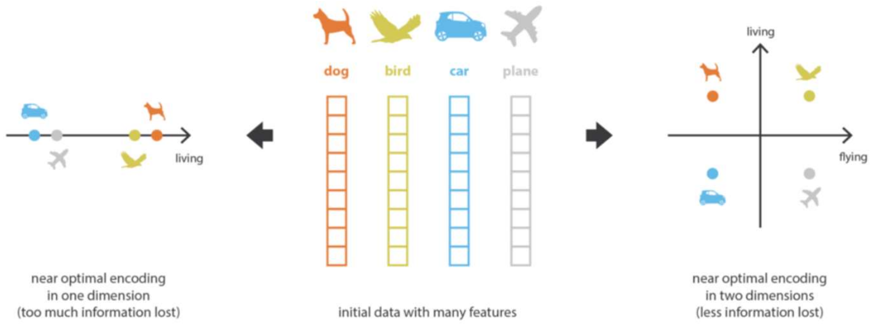


איור 7.3 Autoencoder – שימוש ברשתות נוירונים עבור הורדת המימד והשחזור.

באופן הזה, הארכיטקטורה יוצרת צוואר בקבוק לדאטא, שמבטיח שרק המאפיינים החשובים של הדאטא, שבאמצעותם ניתן לשחזר את הקלט המקורי בדיוק טוב, ישמשו לייצוג במרחב הלטנטי. במקרה הפשוט בו בכל רשת יש רק שכבה חבויה אחת והיא לא משתמשת בפונקציות אקטיבציה לא לינאריות, ניתן לראות כי ה-autoencoder יחפש טרנספורמציה לינארית של הדאטא באמצעותו ניתן לשחזרו באופן לינארי גם כן. בדומה ל-PCA, גם רשת כזו תחפש להוריד את המימד באמצעות טרנספורמציות לינאריות של הפיצ'רים המקוריים אך הייצוג במימד נמוך המופק על ידה לא יהיה בהכרח זהה לזה של PCA, כיוון שלהבדיל מ-PCA הפיצ'רים החדשים (לאחר הורדת מימד) עשויים לצאת לא אורתוגונליים.

כעת נניח שהרשתות הן עמוקות ומשתמשות באקטיבציות לא לינאריות. במקרה כזה, ככל שהארכיטקטורה מורכבת יותר, כך הרשת יכולה להוריד יותר מימדים תוך יכולת לבצע שחזור ללא איבוד מידע. באופן תיאורטי, אם ל-encoder ול-decoder יש מספיק דרגות חופש, ניתן להפחית כל מימד לחד-מימד ללא איבוד מידע. עם זאת, הפחתת מימד

דרסטית שכזו יכולה לגרום לדאטא המשוחזר לאבד את המבנה שלו. לכן יש חשיבות גדולה בבחירת מספר המימדים שבתהליך, כך שמצד אחד אכן יתבצע ניפוי של פרמטרים פחות משמעותיים ומצד שני המידע עדיין יהיה בעל משמעות. ניקח לדוגמא מערכת שמקבלת כלב, ציפור, מכונית ומטוס ומנסה למצוא את הפרמטרים העיקריים המבחינים ביניהם:



איור 7.4 דוגמא לשימוש ב-Autoencoder.

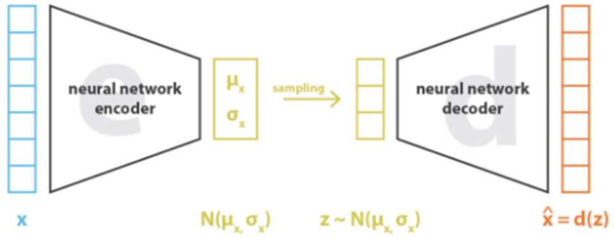
לפריטים אלו יש הרבה פיצ'רים, וקשה לבנות מודל שמבחין ביניהם על סמך כל הפיצ'רים. מעבר ברשת נוירונים יכול להביא לייצוג של כל הדוגמאות על קו ישר, כך שככל שפרט מסוים נמצא יותר ימינה, כך הוא יותר "חי". באופן הזה אמנם מתקבל ייצוג חד-מימדי, אבל הוא גורם לאיבוד המבנה של הדוגמאות ולא באמת ניתן להבין את ההפרדה ביניהן. לעומת זאת ניתן להוריד את המימד לדו-מימד ולהתייחס רק לפרמטרים "חי" ו"עף", וכך לקבל הבחנה יותר ברורה בין הדוגמאות, וכמובן שהפרדה זו היא הרבה יותר פשוטה מאשר הסתכלות על כל הפרמטרים של הדוגמאות. דוגמא זו מראה את החשיבות שיש בבחירת המימדים של ה-encoder.

7.1.3 Variational AutoEncoders (VAE)

ניתן לקחת את ה-AE ולהפוך אותו למודל גנרטיבי, כלומר מודל שמסוגל לייצר בעצמו דוגמאות חדשות שאכן מתפלגות כמו הפילוג של הדאטא המקור. אם מדובר בדומיין של תמונות למשל, אז נרצה שהמודל יהיה מסוגל לייצר תמונות שנראות אותנטיות. לשם כך יש לעמוד על המגבלות שלו וכיצד ניתן להתגבר עליהן. הרשתות של ה-AE מאומנות לייצג את הדאטא במימד נמוך שלוקח בחשבון את הפרמטרים העיקריים, ולאחר מכן לשחזר את התוצאה למימד המקורי, אך הן אינן מתייחסות לאופן בו הדאטא מיוצג במרחב הלטנטי. אם תוגרל נקודה כלשהו מהמרחב הלטנטי – קרוב לוודאי שהיא לא תהווה ייצוג לטנטי, כיוון שאמנם התקבל ייצוג אחר לדאטא, אך אין משמעות לייצוג הזה ללא השחזור. לכן אם היינו מכניסים את הנקודה שהוגרלה ל-decoder, סביר שהתוצאה לא הייתה דומה בכלל לדאטא המקורי. למשל אם אומן AE על סט של תמונות של כלבים ודוגמים וקטור מהמרחב הלטנטי שלו, הסיכוי לקבל תמונת כלב כלשהו לאחר השחזור של ה-decoder הינו אפסי.

כדי התמודד עם בעיה זו, ניתן להשתמש ב-Variational AutoEncoders (VAE). בשונה מ-AE שלוקח דאטא ובונה לו ייצוג ממימד נמוך, VAE מנסה להביא את המרחב הלטנטי להתפלג כרצוננו – התפלגות נורמלית עם תוחלת ושונות המתאימים לדאטא המקורי. לאחר מכן דוגמים וקטור מההתפלגות הנלמדת, ועליה מתבצע השחזור. באופן הזה, הלמידה דואגת לא רק להורדת המימד, אלא גם להתפלגות תחת המרחב הלטנטי. כאשר ההתפלגות טובה, ניתן בעזרתה גם ליצור דוגמאות חדשות, ובעצם מתקבל מודל גנרטיבי.

ב-VAE, ה-encoder מנסה לייצג את הדאטא המקורי באמצעות התפלגות נורמלית במימד נמוך יותר בעל תוחלת ומטריצת covariance: $z\sim p(z|x) = N(\mu_x, \sigma_x)$. התפלגות בעלת תוחלת ושונות דואגת לרגולריזציה של המרחב הלטנטי, כיוון שהתוחלת מפזרת את הדוגמאות סביב הממוצע ומזעור השונות דואג שלא יהיו במרחב החדש דוגמאות חסרות משמעות עקב שונות גדולה. באופן הזה המרחב הלטנטי מתאים לא רק לדוגמאות הנלמדות אלא יודע גם ליצור דוגמאות חדשות בעלות משמעות, כלומר דוגמאות שנראות דומות לדאטא המקורי.



איור 7.5 ארכיטקטורה של VAE.

לאחר שהוצג המבנה הכללי של VAE, ניתן לתאר את תהליך הלמידה, ולשם כך נפריד בשלב זה בין שני החלקים של ה-VAE. ה-encoder מקבל הרבה נקודות של דאטא מסוים, ומנסה ללמוד את התוחלת והשונות של התפלגות זו. מההתפלגות הנלמדת דוגמים נקודות חדשות ומעבירים ל-decoder, שאמור לבצע את הפעולה ההפוכה – לקחת דגימה מהתפלגות מסוימת ולייצר באמצעותה דוגמא חדשה הדומה לדאטא המקורי.

כעת נסמן את הווקטורים של המרחב הלטנטי ב-z, את הפרמטרים של ה-decoder ב-θ, ואת הפרמטרים של ה-encoder ב-λ. כדי למצוא את הפרמטרים האופטימליים של שתי הרשתות, נרצה להביא למקסימום את $p(\hat{x} = x_i; \theta)$, כלומר למקסם את הנראות המרבית של סט האימון תחת θ (-הסתברות שבמוצא ה-decoder נקבל $\hat{x}$ שיהווה שחזור טוב של הדאטא המקורי x_i). כיוון שפונקציית log מונוטונית, נוכל לקחת את לוג ההסתברות:

$$L(\theta) = \log \sum_i p(x; \theta)$$

אם נביא למקסימום את הביטוי הזה, נקבל את ה-θ האופטימלי. כיוון שלא ניתן לחשב במפורש את $p(x; \theta)$, יש להשתמש בקירוב. נניח וה-encoder הוא בעל התפלגות מסוימת $q(z_i|x; \lambda)$ (מה ההסתברות לקבל את z_i בהינתן x_i בכניסה). כעת ניתן לחלק ולהכפיל את $L(\theta)$ ב-$q(z_i|x; \lambda)$:

$$\log \sum_i p(x, z_i; \theta) = \log \sum_i q(z_i|x; \lambda) \frac{p(x, z_i; \theta)}{q(z_i|x; \lambda)} \geq \sum_i q(z_i|x; \lambda) \log \frac{p(x, z_i; \theta)}{q(z_i|x; \lambda)}$$

כאשר אי השוויון נובע מאי-שוויון ינסן, והביטוי שמימין לאי השיוויון נקרא $(ELBO(\theta, \lambda))$ Evidence Lower BOund. ניתן להוכיח שההפרש בין ה-ELBO לבין הערך שלפני הקירוב הוא המרחק בין שתי ההתפלגויות p, q, והוא נקרא Kullback–Leibler divergence ומסומן ב-$\mathcal{D}_{KL}$:

$$\log p(x; \theta) = ELBO(\theta, \lambda) + \mathcal{D}_{KL}(q(z|x; \lambda) \| p(z|x; \theta))$$

אם שתי ההתפלגויות שוות, אזי המרחק ביניהן הוא 0 ומתקבל שוויון: $\log p(x; \theta) = ELBO(\theta, \lambda)$. כזכור, מחפשים מקסימום לפונקציית המחיר $\log \sum_i p(x_i; \theta)$, וכעת בעזרת הקירוב ניתן לרשום:

$$L(\theta) = \log \sum_i p(x_i; \theta) \geq \sum_i ELBO(x_i, \theta, \lambda) = ELBO(\theta, \lambda)$$

$$L(\theta_{ML}) = \max_\theta L(\theta) \geq \max_\theta \max_\lambda ELBO(\theta, \lambda)$$

כעת ניתן בעזרת שיטת GD למצוא את האופטימום של הביטוי, וממנו לדעת את הפרמטרים האופטימליים של ה-encoder ול-decoder. נפתח יותר את ה-$ELBO(\theta, \lambda)$ עבור VAE, כאשר נשתמש במספר התפלגויות:

$p(x|z; \theta)$ – ההסתברות ש-decoder עם סט פרמטרים θ יוציא x בהינתן z.

$q(z_i|x; \lambda)$ –ההסתברות ש-encoder עם סט פרמטרים λ יוציא את z_i בהינתן x_i בכניסה

לפי הגדרה:

$$ELBO(\theta, \lambda) = \sum_i q(z_i|x; \lambda) \log p(x, z_i; \theta) - \sum_i q(z_i|x; \lambda) \log q(z_i|x; \lambda)$$

את הביטוי $\log p(x, z_i; \theta)$ ניתן לפתוח לפי בייס $p(x, z_i) = p(x|z_i) \cdot p(z_i)$:

$$= \sum_i q(z_i|x; \lambda)(\log p(x|z_i; \theta) + \log p(z_i; \theta)) - \sum_i q(z_i|x; \lambda) \log q(z_i|x; \lambda)$$

$$= \sum_i q(z_i|x; \lambda) \log p(x|z_i; \theta) - \sum_i q(z_i|x; \lambda)(\log q(z_i|x; \lambda) - \log p(z_i; \theta))$$

$$= \sum_i q(z_i|x; \lambda) \log p(x|z_i; \theta) - \sum_i q(z_i|x; \lambda) \frac{\log q(z_i|x; \lambda)}{\log p(z_i; \theta)}$$

הביטוי השני לפי הגדרה שווה ל-$\mathcal{D}_{KL}(q(z_i|x; \lambda) \| p(z_i; \theta))$, לכן מתקבל:

$$= \sum_i q(z_i|x;\lambda) \log p(x|z_i;\theta) - \mathcal{D}_{KL}(q(z_i|x;\lambda) \| p(z_i))$$

הביטוי הראשון הוא בדיוק התוחלת של $\log p(x|z_i;\theta)$. תחת ההנחה ש-z מתפלג נורמלית, ניתן לרשום:

$$= E_{q(z|x;\,\lambda)} \log N\big(x;\mu_\theta(z),\sigma_\theta(z)\big) - \mathcal{D}_{KL}\big(N\big(\mu_\lambda(x),\sigma_\lambda(x)\big) \| N(0,I)\big)$$

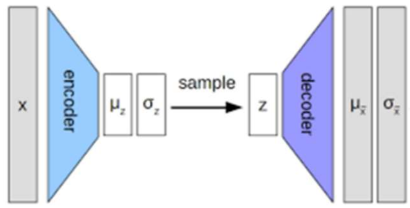
כדי לחשב את התוחלת ניתן פשוט לדגום דוגמאות מההתפלגות $z|x \sim N\big(\mu_\theta(x),\sigma_\theta(x)\big)$ ולקבל:

$$E_{q(z|x;\,\lambda)} \log N\big(x;\mu_\theta(z),\sigma_\theta(z)\big) \approx \log N\big(x;\mu_\theta(z),\sigma_\theta(z)\big)$$

ועבור הביטוי השני יש נוסחה סגורה:

$$\mathcal{D}_{KL}(N(\mu,\sigma^2) \| N(0,I)) = \frac{1}{2}(\mu^2 + \sigma^2 - \log \sigma^2)$$

כעת משיש בידינו נוסחה לחישוב פונקציית המחיר, נוכל לבצע את תהליך הלמידה.



$$x_t \to \mu_\lambda(x_t),\Sigma_\lambda(x_t) \to z_t \sim N\big(\mu_\lambda(x_t),\Sigma_\lambda(x_t)\big) \to \mu_\theta(z_t),\Sigma_\theta(z_t)$$

$$ELBO = \sum_t \log N\big(x_t;\mu_\theta(z_t),\Sigma_\theta(z_t)\big) - \mathcal{D}_{KL}\big(N\big(\mu_\lambda(x_t),\Sigma_\lambda(x_t)\big) || N(0,I)\big)$$

איור 7.6 תהליך הלמידה של VAE.

כאשר נתון סט דוגמאות x, ניתן להעביר כל דוגמא x_t ב-encoder ולקבל את $\mu_\lambda,\sigma_\lambda$. לאחר מכן דוגמים z מההתפלגות הנתונה, מעבירים אותו ב-decoder ומקבלים את μ_θ,σ_θ. לאחר התהליך ניתן להציב את הפרמטרים המתקבלים ב-ELBO ולחשב את ה-Loss. ניתן לשים לב שה-ELBO מורכב משני איברים – האיבר הראשון מחשב את היחס בין הדוגמא שבכניסה לבין ההתפלגות שמתקבלת במוצא, והאיבר השני מבצע רגולריזציה להתפלגות המתקבלת במרחב הלטנטי. הרגולריזציה גורמת לכך שההתפלגות במרחב הלטנטי תהיה קרובה להתפלגות נורמלית ויתקבלו והדגימות ממרחב זה יהיו קרובות לתוחלת 0 ושונות 1, ובכך נמנעים מ-overfitting. אם ההתפלגות במרחב הלטנטי אכן טובה, אז ניתן בעזרת ה-decoder ליצור דוגמאות חדשות, ובמובן הזה ה-VAE הוא מודל גנרטיבי.

הדגימה של z מההתפלגות במרחב הלטנטי יוצרת קושי בחישוב הגרדיאנט של ה-ELBO, לכן בדרך כלל מבצעים Reparameterization trick – דוגמים z_0 מהתפלגות נורמלית, ואז כדי לקבל את z משתמשים בפרמטרים של ה-encoder: $z = z_0\sigma_\lambda(x) + \mu_\lambda(x)$. בגישה הזו כל התהליך נהיה דטרמיניסטי – מגרילים z_0 מראש ואז רק נשאר לחשב באופן סכמתי את ה-forward-backward.

IWAE

VaDE – מחליפים את ההתפלגות של z בהתפלגות MoG.

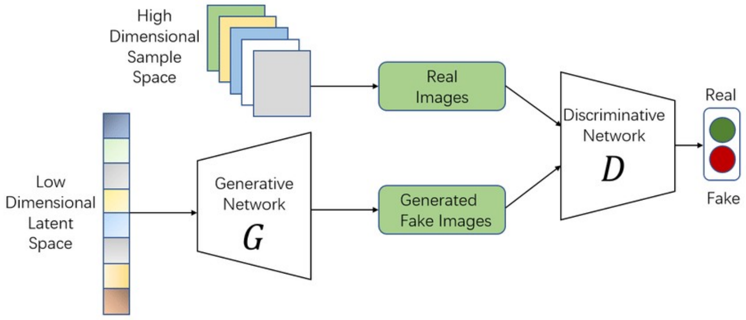
7.2 Generative Adversarial Networks (GANs)

מתודה אחרת של מודל גנרטיבי נקראת Generative Adversarial Networks או בקיצור GANs, ובשונה מ-VAE בגישה זו לא מוצאים התפלגות מפורשת של מרחב מסוים אלא יוצרים דאטא באופן אחר. הרעיון הוא לאמן שתי רשתות במקביל – רשת אחת שלומדת לייצר דאטא, ורשת שניה שלומדת להבחין בין דוגמא אמיתית לבין תמונה סינטטית. באופן הזה הרשת הראשונה היא למעשה מודל גנרטיבי, שלאחר שלב האימון היא מסוגלת לייצר דאטא סינטטי שלא ניתן להבין בינו לבין דאטא אמיתי.

7.2.1 Generator and Discriminator

כאמור, רשתות GAN מבוססות על שני אלמנטים מרכזיים – רשת שיוצרת דאטא (generator) ורשת שמכריעה האם הדאטא הזה סינטטי או אמיתי (discriminator), כאשר האימון נעשה על שתי הרשתות יחד. ה-discriminator עובר

תהליך אימון על דאטא אמיתי כדי לדעת להבחין בין דאטא אמיתי לבין דאטא סינטטי, וה-generator מייצר דוגמאות ומקבל פידבק מה-discriminator וכך לומד לייצר תמונות שנראות אמיתיות. נסמן את ה-generator ב-G ואת הdiscriminator ב-D, ונקבל את הסכמה הבאה:



איור 7.7 ארכיטקטורת GAN.

ה-discriminator הוא למעשה מסווג בינארי ($y = 1$ עבור תמונה אמיתית, ו-$y = 0$ עבור תמונה סינטטית), ונסמן ב-$D(x)$ את ההסתברות לקבל במוצא דוגמא אמיתית. כדי לאמן את ה-discriminator נרצה להביא למינימום את ה-cross entropy עבור $y = 1$ (=למצוא את ה-discriminator שטועה כמה שפחות בזיהוי דאטא אמיתי):

$$\min_D\{-y\log D(x) - (1-y)\log(1-D(x))\} = \min_D\{-y\log D(x)\}$$

באופן דומה נרצה לאמן את ה-generator כך שהדאטא שהוא מייצר יהיה כמה שיותר דומה לאמיתי, ולכן נרצה להביא למקסימום את ה-cross entropy של ה-generator כאשר $y = 0$ (=למצוא את ה-generator שהכי פחות מזייף, כלומר מייצר דאטא כמה שיותר אמיתי):

$$\max_G\left\{-(1-y)\log\left(1-D\big(G(z)\big)\right)\right\} = \max_G\left\{-\log\left(1-D\big(G(z)\big)\right)\right\}$$

אם מחברים את שני האילוצים האלה מקבלים את פונקציית המחיר של ה-GAN:

$$V(D,G) = \min_D\max_G -\mathbb{E}_{x\sim Data}\log D(x) - \mathbb{E}_{z\sim Noise}\log\left(1-D\big(G(z)\big)\right)$$

באופן שקול ניתן להפוך את האילוצים וביטול סימן המינוס:

$$V(D,G) = \min_G\max_D \mathbb{E}_{x\sim Data}\log D(x) + \mathbb{E}_{z\sim Noise}\log\left(1-D\big(G(z)\big)\right)$$

ה-discriminator מעוניין למקסם את פונקציית המחיר, כך ש-$D(x)$ יהיה כמה שיותר קרוב ל-1 ו-$D\big(G(z)\big)$ יהיה כמה שיותר קרוב ל-0. ה-generator לעומת זאת רוצה להביא למינימום את פונקציית המחיר, כך ש-$D\big(G(z)\big)$ יהיה כמה שיותר קרוב ל-1, כלומר ה-discriminator חושב ש-$G(z)$ הוא דאטא אמיתי.

כעת האימון נעשה באופן איטרטיבי, כאשר פעם אחת מקבעים את G ומאמנים את D, ופעם אחת מקבעים את D ומאמנים את G. אם מקבעים את G, אז למעשה מאמנים מסווג בינארי, כאשר מחפשים את האופטימום הבא:

$$\max_{\phi_d}\mathbb{E}_{x\sim Data}\log D_{\phi_d}(x) + \mathbb{E}_{z\sim Noise}\log\left(1-D_{\phi_d}\left(G_{\theta_g}(z)\right)\right)$$

אם מקבעים את D, אז האיבר הראשון של $V(D,G)$ קבוע, ונשאר רק לבדוק את הביטוי השני, שמחפש את ה-generator שמייצר הכי טוב דאטא שנראה אמיתי:

$$\min_{\theta_g}\mathbb{E}_{z\sim Noise}\log\left(1-D_{\phi_d}\left(G_{\theta_g}(z)\right)\right)$$

מציאת האופטימום נעשית בעזרת Gradient Descent/Gradient Ascent, במשך מספר מסוים של Epochs. דוגמים mini-batch בגודל m מהדאטא סט האמיתי $(x_1,\dots,x_m)$ ו-m דגימות מרעש נורמלי $(z_1,\dots,z_m)$. הנגזרת של פונקציית המחיר לפי ה-generator היא:

$$\nabla_\theta V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} \log\Big(1 - D_\phi\big(G_\theta(z_i)\big)\Big)$$

והנגזרת לפי ה-discriminator:

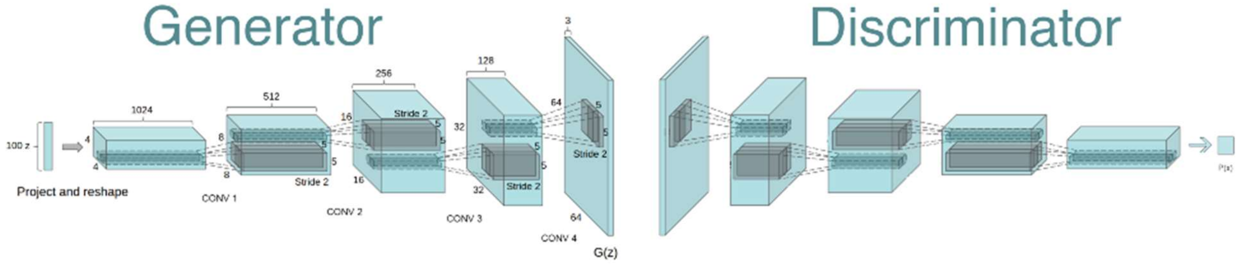
$$\nabla_\phi V(G_\theta, D_\phi) = \frac{1}{m}\nabla_\phi \sum_{i=1}^{m} \log D_\phi(x_i) + \log\Big(1 - D_\phi\big(G_\theta(z_i)\big)\Big)$$

נהוג לבצע מודיפיקציה קטנה על הביטוי של ה-generator. כיוון שבהתחלה הדאטא שמיוצר הוא גרוע, הביטוי $D\big(G(z)\big)$ שואף ל-0, וממילא גם הביטוי $\mathbb{E}_{z\sim Noise}\log\Big(1 - D\big(G(z)\big)\Big)$ שואף ל-0. עניין זה גורם לכך שהגרדיאנט גם יהיה מאוד קטן, ולכן כמעט ולא מתבצע שיפור ב-generator. לכן במקום לחפש מינימום על הביטוי $\mathbb{E}_{z\sim Noise}\log\Big(1 - D\big(G(z)\big)\Big)$ מחפשים מינימום לביטוי $-\mathbb{E}_{z\sim Noise}\log\Big(D\big(G(z)\big)\Big)$. הביטויים לא שווים לגמרי, אך הביטוי החדש עובד יותר טוב נומרית ומצליח לשפר את ה-generator יותר טוב.

לאחר שהוסבר המבנה הכללי של GAN, נעבור לסקור מספר ארכיטקטורות של מודלי GAN.

7.2.2 Deep Convolutional GAN (DCGAN)

כפי שהוסבר בפרק של רשתות קונבולוציה, עבור דאטא של תמונות יש יתרון גדול לרשתות קונבולוציה על פני רשתות FC. לכן היה טבעי לקחת רשתות קונבולוציה ולהשתמש בהן בתור generator ו-discriminator עבור דומיין של תמונות. ה-generator מקבל וקטור אקראי ומעביר אותו דרך רשת קונבולוציה על מנת ליצור תמונה, וה-discriminator מקבל תמונה ומעביר אותו דרך רשת קונבולוציה שעושה סיווג בינארי אם התמונה אמיתית או סינטטית.



איור 7.8 ארכיטקטורת DCGAN.

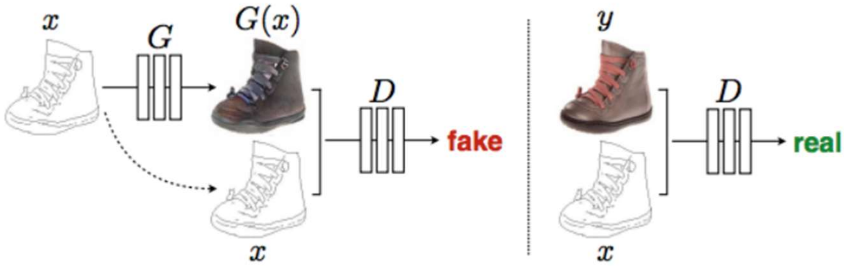
7.2.3 Pix2Pix

במקרה הפשוט z מוגרל מהתפלגות נורמלית, אך זה לא דבר מוכרח. שיטת Pix2Pix משתמשת בארכיטקטורה של GAN אך במקום לדגום את z, יוצרים סקיצה של תמונה בתור הקלט, וה-generator לומד להפוך את הסקיצה לתמונה אמיתית. ה-generator עצמו נשאר ללא שינוי ביחס למה שתואר קודם לכן, אך ה-discriminator כן משתנה – במקום לקבל תמונה ולבצע עליה סיווג בינארי, הוא מקבל זוג תמונות – את הסקיצה ואת התמונה הסינטטית, ועליו לקבוע האם התמונה הסינטטית היא אכן תמונה אמיתית של הסקיצה או לא. הווריאציה של ה-GAN משנה גם את פונקציית המחיר – כעת ה-generator צריך ללמוד שני דברים – גם ליצור תמונות טובות כך שה-discriminator יסווג אותן כאמיתיות, וגם למזער את ההפרש בין התמונה שנוצרת לבין תמונה אמיתית השייכת לסקיצה. אם נסמן תמונה אמיתית השייכת לסקיצה ב-y, נוכל לרשום את פונקציית המחיר כך:

$$V(D,G) = \min_G \max_D \mathbb{E}_{x,y}\Big(\log D(x,y) + \log\Big(1 - D\big(x, G(x)\big)\Big)\Big)$$

$$\min_{\theta_g} \mathbb{E}_{x,y}\Big(\log\Big(1 - D\big(x, G(x)\big)\Big) + \lambda\,\|G(x) - y\|\Big)$$

האיבר הראשון בפונקציית המחיר של G מתייחס לתשובה של ה-discriminator ביחס לתמונות שה-generator מייצר, והאיבר השני מתייחס להפרש בין התמונה הסינטטית לבין תמונה אמיתית השייכת לסקיצה של הקלט.

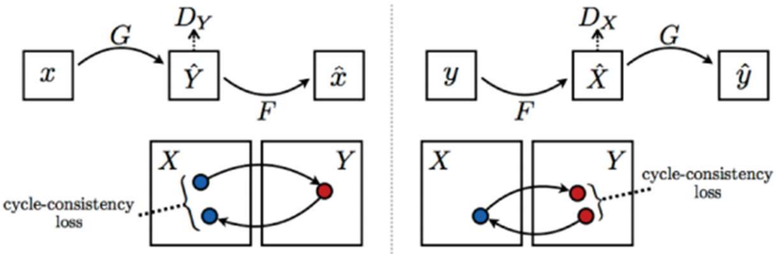


איור 7.9 ארכיטקטורת Pix2Pix - Image-to-Image Translation.

7.2.4 CycleGAN

ב-Pix2Pix הדאטא המקורי הגיע בזוגות – סקיצה ואיתה תמונה אמיתית. זוגות של תמונות זה לא דבר כל כך זמין, ולכן שיפרו את האלגוריתם כך שיוכל לקבל שתי תמונות x, y שאינן תואמות ולבצע השלכה מאחת לשנייה. הארכיטקטורה עבור המשימה הזו מורכבת משני generators – בהתחלה מכניסים את x ל-G שמנסה להפוך אותו ל-y, והפלט נכנס ל-F שמנסה לשחזר את המקור x. המוצא של ה-G נכנס לא רק ל-F אלא גם ל-discriminator D_y שמסווג האם התמונה שהתקבלה אמיתית או לא. ניתן לבצע את התהליך הזה באופן דואלי עבור y – מכניסים את y ל-F על מנת לקבל את x ואת המוצא מכניסים ל-discriminator D_x בכדי לבצע סיווג בינארי ול-G על מנת לנסות לשחזר את המקור. ה-generator השני בתהליך נועד לשפר את תהליך הלמידה – לאחר ש-x הופך ל-y דרך G, ניתן לקבל חזרה את x אם נעביר את y דרך F מתוך ציפייה לקבל $x \approx F(G(x))$. התהליך של השוואת הכניסה למוצא נקרא cycle-consistency, והוא מוסיף עוד איבר לפונקציית המחיר, שמטרתו למזער עד כמה שניתן את ההפרש בין התמונה המקורית לתמונה המשוחזרת:

$$V(D_x, D_y, G, F) = \mathcal{L}_{GAN}(G, D_y, x, y) + \mathcal{L}_{GAN}(F, D_x, x, y)$$
$$+\lambda\left(\mathbb{E}_x\|F(G(x)) - x\|_1 + \mathbb{E}_y\|G(F(y)) - y\|_1\right)$$



איור 7.10 ארכיטקטורת CycleGAN.

7.2.5 StyleGAN

לוקחים תמונה ומשנים את הסטייל שלה.

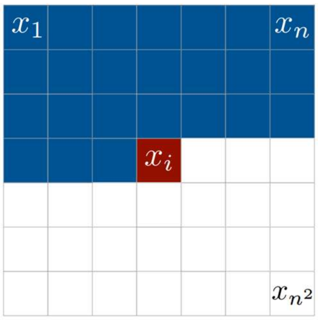
7.2.6 Wasserstein GAN

בד

7.3 Auto-Regressive Generative Models

משפחה נוספת של מודלים גנרטיביים נקראת Auto-Regressive Generative Models, ובדומה ל-VAE גם מודלים אלו מוצאים התפלגות מפורשת של מרחב מסוים ובעזרת התפלגות זו מייצרים דאטא חדש. עם זאת, בעוד VAE מוצא קירוב להתפלגות של המרחב הלטנטי, שיטות AR מנסות לחשב במדויק התפלגות מסוימת, וממנה לדגום ולייצר דאטא חדש.

תמונה x בגודל $n \times n$ היא למעשה רצף של n^2 פיקסלים. כאשר רוצים ליצור תמונה, ניתן ליצור כל פעם כל פיקסל באופן כזה שהוא יהיה תלוי בכל הפיקסלים שלפניו.



איור 7.11 תמונה כרצף של פיקסלים.

כל פיקסל הוא בעל התפלגות מותנית:

$$p(x_i|x_1 \dots x_{i-1})$$

כאשר כל פיקסל מורכב משלושה צבעים (RGB), לכן ההסתברות המדויקת היא:

$$p(x_{i,R}|\mathrm{x}_{<i})p(x_{i,G}|\mathrm{x}_{<i},x_{i,R})p(x_{i,B}|\mathrm{x}_{<i},x_{i,R},x_{i,G})$$

כל התמונה השלמה היא מכפלת ההסתברויות המותנות:

$$p(x)=\prod_{i=1}^{n^2}p(x_i)=\prod_{i=1}^{n^2}p(x_i|x_1 \dots x_{i-1})$$

הביטוי $p(x)$ הוא ההסתברות של דאטא מסוים לייצג תמונה אמיתית, לכן נרצה למקסם את הביטוי הזה כדי לקבל מודל שמייצג תמונות שנראות אותנטיות עד כמה שניתן.

7.3.1 PixelRNN

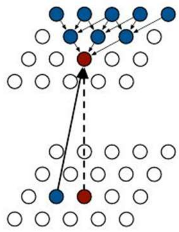
אפשרות אחת לחשב את $p(x)$ היא להשתמש ברכיבי זיכרון כמו LSTM עבור כל פיקסל. באופן טבעי היינו רוצים לקשר כל פיקסל לשכנים שלו:

$$\text{Hidden State }(i,j)=f\big(\text{Hidden State }(i-1,j),\text{Hidden State }(i,j-1)\big)$$

הבעיה בחישוב זה היא הזמן שלוקח לבצע אותו. כיוון שכל פיקסל דורש לדעת את הפיקסל שלפניו – לא ניתן לבצע אימון מקבילי לרכיבי ה-LSTM. כדי להתגבר על בעיה זו הוצעו כמה שיטות שנועדו לאפשר חישוב מקבילי.

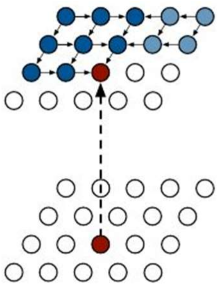
Row LSTM

במקום להשתמש במצב החבוי של הפיקסל הקודם, ניתן להשתמש רק בשורה שמעל הפיקסל אותו רוצים לחשב. שורה זו בעצמה מחושבת לפני כן על ידי השורה שמעליה, ובכך למעשה לכל פיקסל יש receptive field של משולש. בשיטה זו ניתן לחשב באופן מקבילי כל שורה בנפרד, אך יש לכך מחיר של איבוד הקשר בין פיקסלים באותה שורה (loss context).


איור Row LSTM 7.12 – כל פיקסל מחושב על ידי $k \geq 3$ פיקסלים בשורה שמעליו.

Diagonal BiLSTM

כדי לאפשר גם חישוב מקבילי וגם שמירה על קשר עם כל הפיקסלים, ניתן להשתמש ברכיבי זיכרון דו כיווניים. בכל שלב מחשבים את רכיבי הזיכרון משני הצדדים של כל שורה, וכך כל פיקסל מחושב גם בעזרת הפיקסל שלידו וגם על ידי זה שמעליו. באופן הזה ה-receptive field גדול יותר ואין loss context, אך החישוב יותר איטי מהשיטה הקודמת, כיוון שהשורות לא מחושבות בפעם אחת אלא כל פעם שני פיקסלים.



איור 7.13 Diagonal BLSTM – כל פיקסל מחושב על ידי $k \geq 3$ פיקסלים בשורה שמעליו.

כדי לשפר את השיטות שמשתמשות ברכיבי זיכרון ניתן להוסיף עוד שכבות, כמו למשל Residual blocks שעוזרים להאיץ את ההתכנסות ו-Masked convolutions כדי לנבא טוב את הצבעים השונים של כל פיקסל.