

האוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

מימוש מודל CAN ליצירת אומנות ושילובו בגישת ה- DeepIE

פרויקט מתקדם זה הוגש כחלק מהדרישות לקבלת תואר
"מוסמך למדעים" M.Sc. במדעי המחשב
באוניברסיטה הפתוחה
המחלקה למתמטיקה ולמדעי המחשב

על-ידי
רותם יגיל

הפרויקט הוכן בהדרכתה של ד"ר מיריי אביגל

אפריל 2023

תוכן עניינים

3	רשימת איורים
3	רשימת טבלאות
4	תקציר
5	1. מבוא
6	2. Creative Adversarial Networks (CANs)
7	2.1. מימוש מודל CAN בפרויקט
7	2.1.1. מימוש הדיסקרימינטור
8	2.1.2. מימוש הגנרטור
12	2.2. אימון מודל CAN לאופטימיזציה של פונקציית הרווח/עלות – Cost Function
12	2.2.1. אימון הדיסקרימינטור
13	2.2.2. אימון הגנרטור
14	2.2.3. אלגוריתם האימון
17	2.2.4. בדיקה של מודל CAN תוך כדי האימון
20	2.2.5. הפקת יצירות האומנות הסינתטיות בפרויקט ע"י גנרטור EMA
21	3. Deep Interactive Evolution
21	3.1. מימוש גישת ה-DeepIE בפרויקט
21	3.1.1. מימוש פונקציית השחלוף האחיד
22	3.1.2. מימוש פונקציית המוטציה
22	3.1.3. מימוש פונקציית האבולוציה
27	4. ניתוח תוצאות מודל CAN ושילובו בגישת ה-DeepIE
28	4.1. סיכום והרחבה עתידית של הפרויקט
32	רשימת מקורות

רשימת איורים

7	איור 1 : מבנה מודל מסוג CAN
9	איור 2 : הגנרטור במודל DCGAN
10	איור 3 : מבנה שכבות הדיסקרימינטור
11	איור 4 : מבנה שכבות הגנרטור
15	איור 5 : אלגוריתם האימון במודל CAN
18	איור 6 : היצירות המתקבלות מהגנרטור במהלך האימון
20	איור 7 : מדדי המודל לאורך תהליך האימון והבדיקה
20	איור 8 : יצירה סינתטית המופקת מגנרטור EMA לעומת הגנרטור הקלאסי
23	איור 9 : שלבי גישת ה-DEEPIE
24	איור 10 : אלגוריתם ה-DEEPIE
24	איור 11 : דוגמה לפעולת שחלוף אחיד ומוטציה
25	איור 12 : דוגמה לתהליך אבולוציוני
29	איור 13 : יצירות סינתטיות מגנרטור EMA של מודל CAN לאחר האימון
29	איור 14 : יצירות סינתטיות לדוגמה שנוצרו באמצעות מודל CAN שב-[2]
31	איור 15 : דוגמאות למקרי קצה בתהליך האבולוציוני

רשימת טבלאות

16	טבלה 1 : מספר היצירות בכל סגנון עבור כל סט נתונים
----	---

תקציר

GAN (Generative Adversarial Network) הוא מודל גנרטיבי המורכב משתי רשתות נוירונים, דיסקרימינטור (מפלה) וגנרטור (מחולל), המתחרות זו בזו במשחק סכום אפס. מטרת התחרות היא לייצר נתונים סינתטיים חדשים, הדומים במאפיינם לנתונים שבסט האימון, כאשר מאז הוצג המודל בשנת 2014 נעשו בו שימושים רבים בתחומים כגון: אומנות, מדע, משחקי מחשב ועוד. בפרויקט זה אנו מממשים את מודל CAN (Creative Adversarial Network), מודל מבוסס GAN המשמש ליצירת אומנות סינתטית בסגנון מקורי ומשלבים את מודל CAN בגישת DeepIE (Deep Interactive Evolution), ליצירת אומנות מונחית משתמש.

1. מבוא

רשתות נוירונים ולמידה עמוקה מהווים את הבסיס למודל מסוג GAN, המורכב משתי רשתות נוירונים, רשת אחת גנרטיבית, המסוגלת לייצר דגימות סינתטיות חדשות של נתונים ורשת דיסקרימינטיבית, המשמשת לסיווג הדגימה הסינתטית המגיעה מהגנרטור כאמיתית או מזויפת. שתי הרשתות מתחרות ביניהן במשחק סכום אפס (רווח של רשת אחת הוא הפסד עבור הרשת השנייה ולהפך), כאשר מטרת התחרות היא לגרום לגנרטור להשתפר ביצירת הדגימות המזויפות עד אשר יצליח להערים על הדיסקרימינטור, שיסווג את הדגימות המזויפות כאמיתיות. כך נוכל להשתמש בגנרטור, לאחר שלמעשה ניצח בתחרות, כמייצר דגימות מזויפות הנראות כאמיתיות ובעלות מאפיינים דומים לסט הנתונים המקורי והאמיתי [1].

בפרויקט זה נשתמש בתכונה זו של GANs, שלאחר אימון באמצעות סט נתונים אמיתי יכול לייצר דגימות מזויפות, ליצירת אומנות סינתטית בסגנון מקורי בעזרת מודל CAN, המבוסס על הרעיון של GAN [2]. בחלק השני של הפרויקט נראה כיצד ניתן לשלב את מודל CAN בגישת ה-DeepIE, המציעה להשתמש בגנרטור שעבר אימון וברעיון של תהליך אבולוציוני באלגוריתם גנטי, ליצירת אומנות המושפעת מבחירותיו של המשתמש [3].

נציין כי פרויקט זה מממש את פרקים 3 ו-4 של העבודה המסכמת, כך שאת ההצגה המלאה של מודל CAN וגישת ה-DeepIE ניתן למצוא בפרקים אלה, בהתאמה. כמו כן ניתן למצוא את הרקע המלא הדרוש להבנת מודל GAN, בדמות למידת מכונה, רשתות נוירונים ולמידה עמוקה, בנוסף להצגת מודל GAN המלאה, בפרקים 1 ו-2 של העבודה המסכמת.

המסמך בנוי באופן הבא:

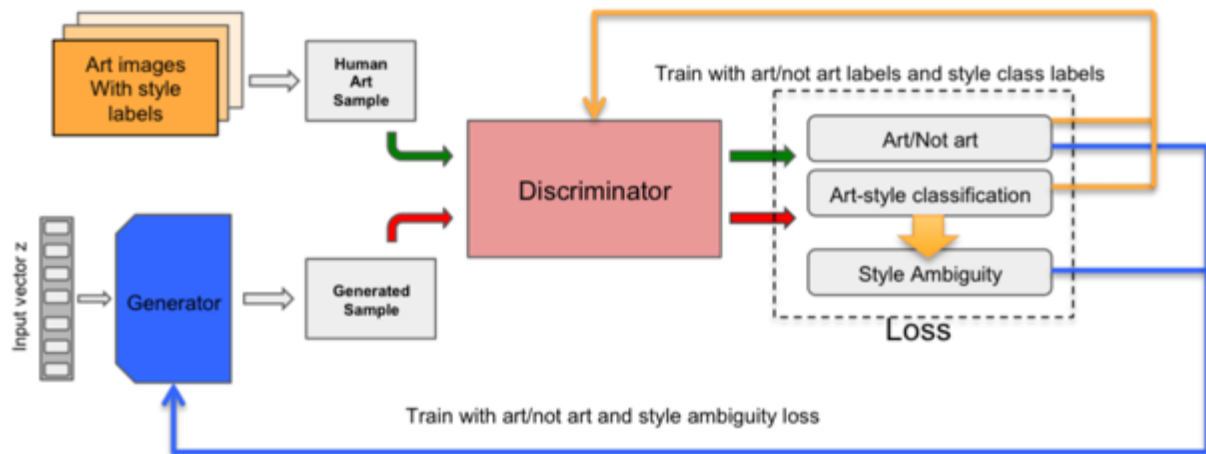
בפרק 2 נעסוק במימוש מודל CAN ונראה את היצירות שמודל שעבר אימון מסוגל להפיק.
בפרק 3 נעסוק במימוש רעיון ה-DeepIE ונראה יצירות אומנות סינתטיות שהתקבלו בעזרת הנחיית המשתמש, בעזרת שילוב של מודל CAN וגישת ה-DeepIE.
בפרק 4 ננתח את התוצאות, נסיק מהן מסקנות אודות יכולות מודל CAN וגישת ה-DeepIE, יחד עם היתרונות והחסרונות ונסכם את הפרויקט.

2. Creative Adversarial Networks (CANs)

בפרק זה נסקור יישום ליצירת אומנות באמצעות מודל רשת נוירונים מבוסס GANs. באמצעות מודל GAN קלאסי [1] וסט נתונים מתאים לאימון הדיסקרימינטור ומתן משוב לגנרטור, ניתן ליצור מידע סינתטי כלשהו מתוך רעש אקראי בכניסת הגנרטור. ניתן לנצל יכולת זו של מודל GAN על מנת ליצור אומנות, תוך שימוש בסט נתונים לאימון עם יצירות אומנות שונות. הגנרטור ברשת ינסה להערים על הדיסקרימינטור כאשר ילמד להפיק יצירות אומנות סינתטיות הדומות ליצירות מסט האימון, בעוד הדיסקרימינטור ינסה להבדיל בין היצירות האמיתיות מסט האימון ליצירות המזויפות המגיעות מהגנרטור. החיסרון במודל הקלאסי הוא שאיננו מעודד יצירות אצל הגנרטור, מכיוון שמטרת הגנרטור הינה להפיק יצירות סינתטיות מתוך התפלגות הנתונים האמיתיים, כך שבסופו של דבר נקבל יצירות סינתטיות הדומות למדי ליצירות האמיתיות. על מנת לעודד יצירות אצל הגנרטור נצטרך לשדרג את מודל ה-GAN הקלאסי.

באיור 1, ניתן לראות את מודל CAN, המבוסס על GAN ומעודד יצירות, אשר הוצג במאמר משנת 2017. הדיסקרימינטור במודל זה מתאמן באמצעות סט נתונים מתויג המחולק לסגנונות אומנות שונים, כאשר הגנרטור מתאמן בעזרת המשוב הקלאסי של סיווג היצירה הסינתטית לאמיתית או מזויפת ומשוב נוסף מהדיסקרימינטור אל הגנרטור הנקרא Style Ambiguity. משוב זה מעביר לגנרטור מידע עד כמה סגנון היצירה שהפיק מקורית, כלומר מה מידת הקושי של הדיסקרימינטור בסיווג היצירה לסגנון מסוים מהסגנונות שבסט הנתונים. מטרת הגנרטור כעת היא כפולה: מצד אחד להערים על הדיסקרימינטור שמדובר ביצירה אמיתית כמקודם ומצד שני לייצר יצירה בסגנון שונה מכל הסגנונות שבסט האימון, כך שהדיסקרימינטור אינו מצליח לשייך את היצירה הסינתטית לסגנון מסוים באופן מוחלט. שילוב זה מאפשר לגנרטור ללמוד להפיק יצירות אומנות אותנטיות, המדמות יצירות שצוירו ע"י אמנים אנושיים ובנוסף מעודד יצירות של הגנרטור, כאשר עליו ליצור יצירות בסגנון שונה מכל הסגנונות מהסט האמיתי.

כמובן שבמודל CAN מטרת הדיסקרימינטור גם כן משתנה, על מנת לספק לגנרטור את המשוב הנוסף. הדיסקרימינטור צריך כעת לסווג את היצירות המתקבלות לסגנון מסוים (Style Classification), בנוסף לסיווג הבינארי של יצירה אמיתית או מזויפת. במידה והדיסקרימינטור מקבל יצירה סינתטית מהגנרטור, בעלת מאפיינים השייכים לסגנון מסוים, כך שהוא מצליח לסווג את היצירה הסינתטית לאותו הסגנון, הגנרטור ייענש על כך. הגנרטור יאלץ להמשיך ולהתאמן עד אשר יצליח להערים על הדיסקרימינטור וליצור יצירות סינתטיות שיראו אמיתיות ובעלות מאפיינים שונים מכלל הסגנונות, כך שהדיסקרימינטור לא יצליח לסווג את היצירות לסגנון מסוים. איזון מושלם יהיה במקרה שבו ההסתברות לשיוך היצירות תהיה שווה עבור כל הסגנונות, כלומר הדיסקרימינטור יסווג לסגנון אקראי [2].



איור 1: מבנה מודל מסוג CAN [2]

הגנרטור מייצר דגימה סינתטית מתוך וקטור הרעש z ומתאמן בעזרת סיווג הדיסקרימינטור ליצירה אמיתית או מזויפת (כמו במודל GAN קלאסי), בתוספת מדד ה-Style Ambiguity המגיע מהדיסקרימינטור. הדיסקרימינטור מתאמן בסיווג היצירות לאמיתיות או מזויפות, בעזרת הדגימות המזויפות המגיעות מהגנרטור ובעזרת הדגימות האמיתיות המגיעות מסט הנתונים. בנוסף, מתאמן הדיסקרימינטור בסיווג סגנון היצירות בעזרת סט הנתונים המתויג לפי הסגנונות ומתוך כך מפיך משוב לגנרטור בנוגע לאי ודאות הסגנון, מדד ה-Style Ambiguity.

2.1. מימוש מודל CAN בפרויקט

על מנת לממש את מודל CAN בהתאם לארכיטקטורה שב-[2] (הפירוט המלא נמצא כאמור בפרק 3 של העבודה המסכמת) נעשה שימוש ב-TensorFlow, ספריית קוד פתוח לבנייה ואימון של רשתות נוירונים, שפותחה ע"י Google וב-Keras, ספריית קוד פתוח המשמשת כ-API ל-TensorFlow. הפיתוח נעשה בשפת Python 3 במחברת בסביבת Google Colab, המאפשרת לאמן את המודל בצורה יעילה ומהירה יותר בעזרת GPU. סט האימון וסט הוולידציה נלקחו מתוך WikiArt, כאשר יצירות האומנות מחולקות ל-27 סגנונות אומנות שונים.

2.1.1. מימוש הדיסקרימינטור

הדיסקרימינטור בפרויקט (איור 3) מומש במבנה הבא:

- השכבה הראשונה בדיסקרימינטור היא שכבת הקלט, שמקבלת יצירות אמיתיות מסט הנתונים ויצירות מזויפות מהגנרטור, ברזולוציה של 64×64 , בניגוד ל-[2] שבו נעשה שימוש ביצירות ברזולוציה של 256×256 . הסביבה לכך היא מגבלות חומרה, כלומר על מנת לאמן באמצעות יצירות ברזולוציה גבוהה ישנו צורך במשאבי חומרה נוספים כגון זיכרון RAM.
- שכבת הקונבולוציה הראשונה בדיסקרימינטור בעלת 32 פילטרים בגודל של 4×4 (kernel) עם stride בגודל 2×2 , כלומר ביצוע הקונבולוציה (הפעלת הפילטרים) בקפיצות של שני צעדים שתגרום לקבלת Feature Map קטנה פי שניים.
- בין שכבת הקונבולוציה הראשונה לשנייה נמצאת שכבת LeakyReLU ושכבת Dropout, למניעת overfitting.

- שכבות הקונבולוציה הבאות תהיינה בגודל 64, 128, 256, 512, בגודל 4x4 ו-stride בגודל 2x2, כאשר ביניהן תהיינה שכבות של Batch Normalization, לנרמול הנתונים התורם להתכנסות מהירה יותר ושכבות LeakyReLU ו-Dropout כמקודם.
- לאחר כל פעולות הקונבולוציה נקבל Feature Map בגודל 1x1x512 שמומרת לווקטור באורך 512 ומתפצל לשני סיווגים שונים: סיווג סגנון התמונה וסיווג אמיתי/מזויף.
- על מנת לקבל את סיווג אמיתי/מזויף, יציאת שכבת Dense (שבה קיים חיבור בין כל 512 הנוירונים ליציאה) מועברת דרך Sigmoid, בכדי לקבל הסתברות (מספר בין 0 ל-1) שמדובר ביצירה אמיתית.
- מצד שני, על מנת לקבל את סיווג הסגנון נעשה שימוש בשתי שכבות של Dense בגודל 1024 ו-512, עם שכבת LeakyReLU ביניהן ושכבת Dense בגודל 27, מס' הסגנונות השונים. 27 היציאות מועברות דרך פונקציית Softmax, על מנת לקבל הסתברות בין 0 ל-1 לכל סגנון, כאשר סכום 27 ההסתברויות הינו 1.

לסיכום, הדיסקרימינטור מקבל יצירות אומנות אמיתיות בסגנונות שונים מסט הנתונים ויצירות מזויפות מהגנרטור העוברות דרך 6 שכבות קונבולוציה עם Stride בגודל 2, המתבצעות באופן הבא:

$$32 \ 4x4 \ filters \rightarrow 64 \ 4x4 \ filters \rightarrow 128 \ 4x4 \ filters \rightarrow 256 \ 4x4 \ filters \\ \rightarrow 512 \ 4x4 \ filters \rightarrow 512 \ 4x4 \ filters$$

לכן יצירה המתקבלת כקלט לדיסקרימינטור, תניב פלט בגודל 4x4x512, כאשר שכבת Fully Connected תמיר פלט זה להסתברות $D_r(c|x)$ המציינת את ההסתברות שהתמונה אמיתית. בנוסף, 3 שכבות Fully Connected המקבלות את הפלט הנוסף בגודל 4x4x512 וממירות אותו לגודל 1024, 512 ולבסוף לפלט באורך 27, המציין את מספר הסגנונות הקיימים לאומנות. כל סגנון יקבל הסתברות $D_c(c_k|x)$ המציינת את הסיכוי של התמונה להשתייך לסגנון זה.

2.1.2. מימוש הגנרטור

- הגנרטור בפרויקט (איור 4), המבוסס על גנרטור מסוג DCGAN (איור 2), מומש באופן הבא:
- שכבת הקלט מקבלת וקטור באורך 100 המהווה דגימה מהמרחב הלטנטי, כלומר וקטור הנדגם אקראית מהתפלגות נורמלית.
 - שכבת הקלט מחוברת לשכבת Dense, כלומר חיבור מלא בין כל הנוירונים, בגודל 32,768 (4x4x2048).
 - השכבות הבאות הן שכבת Batch Normalization ושכבת LeakyReLU, כאשר הווקטור מסודר מחדש כטנזור של 4x4x2048.
 - סדרה של שכבות Conv2DTranspose עם kernel בגודל 4x4 ו-stride בגודל 2x2 מבצעות פעולה הפוכה לפעולת הקונבולוציה, כלומר בניגוד לפעולות הקונבולוציה שבדיסקרימינטור שהפיקו Feature Map קטנות פי שניים, נקבל הגדלה פי שניים לאחר כל שכבה. שכבת הקונבולוציה ההפוכה הראשונה הינה בעלת 1024 פילטרים, כאשר השכבות הבאות תהיינה קטנות פי שניים מהשכבה הקודמת, כלומר בעלות 512, 256, 128, 64 ולבסוף 3 פילטרים. כך מטנזור של

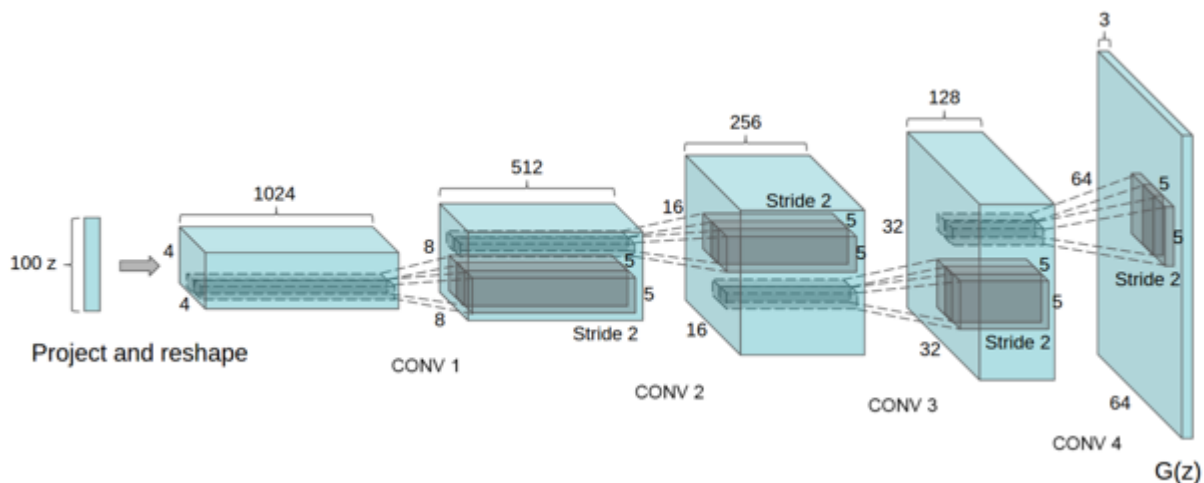
4x4x2048, שנוצר מווקטור הרעש האקראי, נקבל יצירה ברזולוציה 256x256x3 (תמונת RGB עם שלושה ערוצי צבע).

- בין שכבות הקונבולוציה ההפוכה, ישנן שכבות Batch Normalization ושכבות LeakyReLU כמקודם.
- יש לציין כי במהלך האימון, היצירה ברזולוציה 256x256 הוקטנה לתמונת 64x64 כקלט לדיסקרימינטור, באמצעות Average Pooling (חישוב ממוצע הפיקסלים בחלון של 4x4 והפיכתו לפיקסל עם הערך הממוצע), על מנת להקל על תהליך האימון בחומרה הנתונה כאמור לעיל.
- לבסוף נשתמש בפונקציית Tanh על מנת לקבל ערכי פיקסלים מנורמלים בין [-1,1], שיוצגו כיצירת האומנות הסינתטית שהתקבלה מהגנרטור.

לסיכום, וקטור הרעש בכניסת הגנרטור בעל 100 המימדים מסודר מחדש כטנזור 4 מימדי עם 2048 Feature Maps, כאשר סדרה של 4 פעולות קונבולוציה Fractionally-Strided מניבה יצירה סינתטית בגודל 256x256, כלומר היצירה מתקבלת באופן הבא :

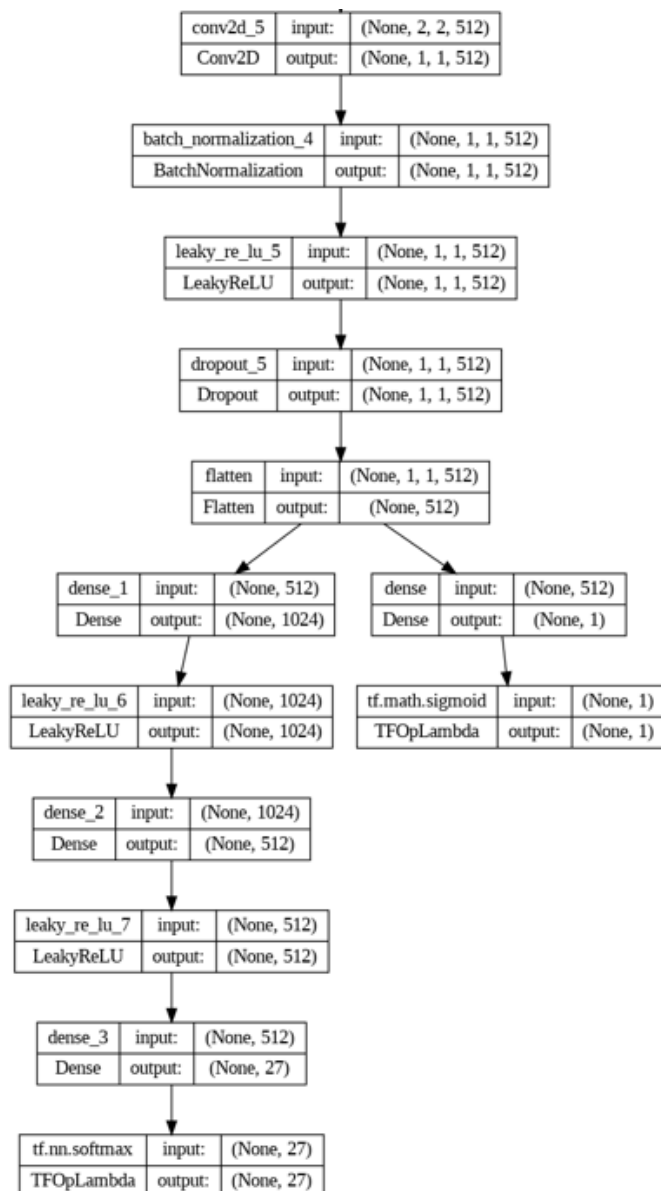
$$z \in \mathbb{R}^{100} \rightarrow 4 \times 4 \times 2048 \rightarrow 8 \times 8 \times 1024 \rightarrow 16 \times 16 \times 512 \rightarrow 32 \times 32 \times 256 \rightarrow 64 \times 64 \times 128 \\ \rightarrow 128 \times 128 \times 64 \rightarrow 256 \times 256 \times 3$$

בתהליך האימון כאמור, היצירה מומרת לרזולוציה נמוכה יותר של 64x64x3 באמצעות Average Pooling.



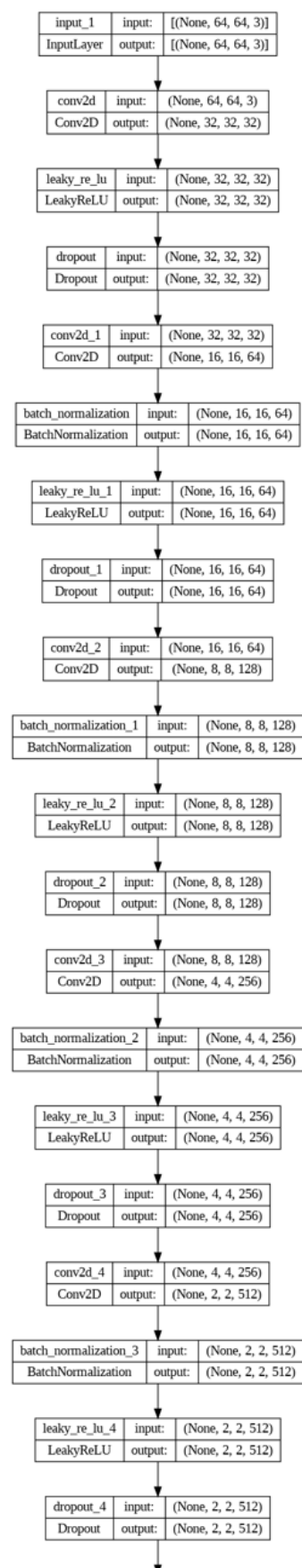
איור 2 : הגנרטור במודל DCGAN [7]

מווקטור הרעש z בעל 100 מימדים, הנדגם מתוך התפלגות אחידה, בכניסת הגנרטור ועד ליצירת הדגימה הסינתטית, מתבצעות פעולות קונבולוציה מסוג Fractionally-Strided שתפקידן לבצע מיפוי הווקטור ל-Feature Maps, עד אשר מתקבלת תמונה סינתטית $G(z)$ בגודל 64x64. כך מתבצעת פעולת Upsampling ללא שכבות Pooling, כאשר וקטור הרעש מעוצב מחדש כטנזור בעל 4 מימדים, ללא שכבת ה-Fully Connected. הדיסקרימינטור מקבל את התמונה הסינתטית ובאמצעות פעולות קונבולוציה עם Stride בגודל 2, מבצע פעולות Downsampling על מנת לסווג את התמונה הסינתטית.



איור 3 : מבנה שכבות הדיסקרימינטור

באיור שמימין ניתן לראות את שכבת הכניסה, המקבלת יצירה ברזולוציה של 64×64 , כאשר לאחר כל שכבת קונבולוציה נמצאת שכבת Batch Normalization, Leaky ReLU ו-Dropout. באיור שמלמעלה, לאחר שכבת הקונבולוציה השישית והאחרונה והשכבות הנלוות לה, שכבת ה-Flatten תייצר וקטור בגודל 512 המתפצל לשתי יציאות: האחת יציאת ה-Sigmoid להסתברות ליצירה אמיתית והשנייה יציאת Softmax, וקטור הסתברויות לשיוך היצירה לכל אחד מ-27 הסגנונות השונים.

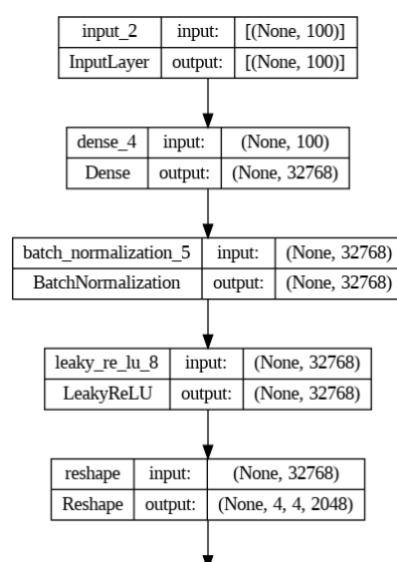




```

graph TD
    subgraph Block1 [ ]
        direction TB
        conv2d_transpose[conv2d_transpose]
        input1[input: (None, 4, 4, 2048)]
        Conv2DTranspose[Conv2DTranspose]
        output1[output: (None, 8, 8, 1024)]
        conv2d_transpose --- input1
        Conv2DTranspose --- output1
    end
    subgraph Block2 [ ]
        direction TB
        batch_normalization_6[batch_normalization_6]
        input2[input: (None, 8, 8, 1024)]
        BatchNormalization[BatchNormalization]
        output2[output: (None, 8, 8, 1024)]
        batch_normalization_6 --- input2
        BatchNormalization --- output2
    end
    subgraph Block3 [ ]
        direction TB
        leaky_re_lu_9[leaky_re_lu_9]
        input3[input: (None, 8, 8, 1024)]
        LeakyReLU[LeakyReLU]
        output3[output: (None, 8, 8, 1024)]
        leaky_re_lu_9 --- input3
        LeakyReLU --- output3
    end
    subgraph Block4 [ ]
        direction TB
        conv2d_transpose_1[conv2d_transpose_1]
        input4[input: (None, 8, 8, 1024)]
        Conv2DTranspose[Conv2DTranspose]
        output4[output: (None, 16, 16, 512)]
        conv2d_transpose_1 --- input4
        Conv2DTranspose --- output4
    end
    subgraph Block5 [ ]
        direction TB
        batch_normalization_7[batch_normalization_7]
        input5[input: (None, 16, 16, 512)]
        BatchNormalization[BatchNormalization]
        output5[output: (None, 16, 16, 512)]
        batch_normalization_7 --- input5
        BatchNormalization --- output5
    end
    subgraph Block6 [ ]
        direction TB
        leaky_re_lu_10[leaky_re_lu_10]
        input6[input: (None, 16, 16, 512)]
        LeakyReLU[LeakyReLU]
        output6[output: (None, 16, 16, 512)]
        leaky_re_lu_10 --- input6
        LeakyReLU --- output6
    end
    subgraph Block7 [ ]
        direction TB
        conv2d_transpose_2[conv2d_transpose_2]
        input7[input: (None, 16, 16, 512)]
        Conv2DTranspose[Conv2DTranspose]
        output7[output: (None, 32, 32, 256)]
        conv2d_transpose_2 --- input7
        Conv2DTranspose --- output7
    end
    subgraph Block8 [ ]
        direction TB
        batch_normalization_8[batch_normalization_8]
        input8[input: (None, 32, 32, 256)]
        BatchNormalization[BatchNormalization]
        output8[output: (None, 32, 32, 256)]
        batch_normalization_8 --- input8
        BatchNormalization --- output8
    end
    subgraph Block9 [ ]
        direction TB
        leaky_re_lu_11[leaky_re_lu_11]
        input9[input: (None, 32, 32, 256)]
        LeakyReLU[LeakyReLU]
        output9[output: (None, 32, 32, 256)]
        leaky_re_lu_11 --- input9
        LeakyReLU --- output9
    end
    subgraph Block10 [ ]
        direction TB
        conv2d_transpose_3[conv2d_transpose_3]
        input10[input: (None, 32, 32, 256)]
        Conv2DTranspose[Conv2DTranspose]
        output10[output: (None, 64, 64, 128)]
        conv2d_transpose_3 --- input10
        Conv2DTranspose --- output10
    end
    subgraph Block11 [ ]
        direction TB
        batch_normalization_9[batch_normalization_9]
        input11[input: (None, 64, 64, 128)]
        BatchNormalization[BatchNormalization]
        output11[output: (None, 64, 64, 128)]
        batch_normalization_9 --- input11
        BatchNormalization --- output11
    end
    subgraph Block12 [ ]
        direction TB
        leaky_re_lu_12[leaky_re_lu_12]
        input12[input: (None, 64, 64, 128)]
        LeakyReLU[LeakyReLU]
        output12[output: (None, 64, 64, 128)]
        leaky_re_lu_12 --- input12
        LeakyReLU --- output12
    end
    subgraph Block13 [ ]
        direction TB
        conv2d_transpose_4[conv2d_transpose_4]
        input13[input: (None, 64, 64, 128)]
        Conv2DTranspose[Conv2DTranspose]
        output13[output: (None, 128, 128, 64)]
        conv2d_transpose_4 --- input13
        Conv2DTranspose --- output13
    end
    subgraph Block14 [ ]
        direction TB
        batch_normalization_10[batch_normalization_10]
        input14[input: (None, 128, 128, 64)]
        BatchNormalization[BatchNormalization]
        output14[output: (None, 128, 128, 64)]
        batch_normalization_10 --- input14
        BatchNormalization --- output14
    end
    subgraph Block15 [ ]
        direction TB
        leaky_re_lu_13[leaky_re_lu_13]
        input15[input: (None, 128, 128, 64)]
        LeakyReLU[LeakyReLU]
        output15[output: (None, 128, 128, 64)]
        leaky_re_lu_13 --- input15
        LeakyReLU --- output15
    end

```



2.2. אימון מודל CAN לאופטימיזציה של פונקציית הרווח/עלות – Cost Function

פונקציית העלות במודל CAN שונה מהפונקציה במודל GAN המקורי [2]:

$$\min_G \max_D V(G, D) = \mathbb{E}_{x, \hat{c} \sim P_{data}} [\log D_r(x) + \log D_c(c = \hat{c}|x)] + \mathbb{E}_{z \sim P_z} \left[\log(1 - D_r(G(z))) - \sum_{k=1}^K \left(\frac{1}{K} \log(D_c(c_k|G(z))) + \left(1 - \frac{1}{K}\right) \log(1 - D_c(c_k|G(z))) \right) \right]$$

- z הוא וקטור רעש אקראי הנדגם מהמרחב הלטנטי P_z (התפלגות אחידה או גאוסיינית).
- $G(z)$ היצירה הסינתטית הנוצרת ע"י הגנרטור מוקטור הרעש.
- x יצירה אמיתית מהתפלגות סט הנתונים P_{data} .
- $\hat{c}$ תווית סגנון היצירה המתאימה ל- x .
- D_r היא פונקציית המיפוי לסיווג היצירות לאמיתיות או מזויפות.
- D_c היא פונקציית המיפוי לסיווג סגנון היצירה.
- K הוא מספר הסגנונות הקיימים בסט הנתונים.
- $D_c(c_k|G(z))$ ההסתברות שפונקציית המיפוי מנבאת עבור שיוך היצירה הסינתטית לסגנון c_k .
- $\min_G \max_D V(G, D) = \log D_r(x) + \log(1 - D_r(G(z)))$ מציינ את התחרות בין הגנרטור לבין הדיסקרימינטור, באופן דומה למודל ה-GAN המקורי. נזכיר כי הדיסקרימינטור מנסה למקסם את הסיווגים הנכונים, אמיתי לאמיתי ומזויף למזויף והגנרטור מנסה למזער את הסיווגים הנכונים של הדיסקרימינטור עבור הדגימות הסינתטיות.
- $\log D_c(c = \hat{c}|x)$ הינו ביטוי ה-Style Classification המציין את סיווגי הסגנון הנכונים של הדיסקרימינטור.
- $-\sum_{k=1}^K \left(\frac{1}{K} \log(D_c(c_k|G(z))) + \left(1 - \frac{1}{K}\right) \log(1 - D_c(c_k|G(z))) \right)$ ביטוי ה-Style Ambiguity שנובע מ-Cross Entropy עבור סיווג רב קטגורי של הסגנונות.

2.2.1. אימון הדיסקרימינטור

הדיסקרימינטור מתאמן על מנת להבדיל בין יצירות אמיתיות למזויפות, כמו במודל המקורי, ובנוסף מתאמן בצורה מפוקחת בעזרת סט נתונים מתויג עם תווית סגנון מתאימה לכל יצירה, על מנת לזהות את סגנון היצירה, מתוך 27 סגנונות שונים. המשמעות בתחרות מול הגנרטור הינה מקסום פונקציית העלות ע"י הביטוי:

$$\max_D V(D, G) = \mathbb{E}_{x, \hat{c} \sim P_{data}} [\log D_r(x)_{(1)} + \log D_c(c = \hat{c}|x)_{(2)}] + \mathbb{E}_{z \sim P_z} \left[\log(1 - D_r(G(z)))_{(3)} \right]$$

עבור היצירות האמיתיות, הדיסקרימינטור יתאמן לסווג כמה שיותר סיווגי "אמיתי" (ביטוי מס' 1), כמו כן למקסם את ביטוי ה-Style Classification (ביטוי מס' 2), ע"י דיוק בסיווגי הסגנון המתאים ליצירה.

מצד שני, הדיסקרימינטור יתאמן למקסם את ביטוי מס' 3 עבור היצירות הסינתטיות המגיעות מהגנרטור, ע"י דיוק ככל הניתן בסיווגי "זיוף" עבור היצירות המזויפות [2].

פונקציית ההפסד של הדיסקרימינטור בפרויקט מומשה בהתאם ל-[2] ובנויה משלושה מרכיבים: ערך ההפסד עבור יצירות אמיתיות, ערך ההפסד עבור יצירות מזויפות מהגנרטור וערך ההפסד עבור סיווג סגנון היצירה האמיתית. על מנת לחשב את שני המרכיבים הראשונים נעשה שימוש בממוצע ה-Sigmoid cross entropy, פונקציה לחישוב השגיאה בין ההסתברות שמדובר ביצירה אמיתית, שמנפיק הדיסקרימינטור ביציאתו עבור יצירה נתונה, לבין ההסתברות שאמורה להתקבל במקרה האידיאלי. עבור יצירות אמיתיות, ההסתברות האידיאלית לקבלה ביציאת הדיסקרימינטור הינה 1, כאשר עבור יצירות מזויפות ההסתברות האידיאלית היא 0. בפועל קשה לקבל ערכים מוחלטים אלו של 0 ו-1, לכן נבצע חישוב של cross entropy עם ערכים קרובים ל-1 כמו [0.8, 1.2] וערכים קרובים ל-0 כמו [0, 0.3]. המרכיב השלישי, הפסד סיווג היצירה מסט הנתונים, מחושב באמצעות ממוצע Softmax cross entropy לחישוב השגיאה בין חיזוי הסגנון שמתקבל מהדיסקרימינטור לבין הסגנון האמיתי. הסגנון האמיתי מקודד בסגנון one-hot, כלומר וקטור באורך מס' הסגנונות האפשרי (27 סגנונות במקרה של סט הנתונים WikiArt שעליו מתאמן מודל CAN), כאשר באינדקס המתאים בווקטור, שמסמן את הסגנון האמיתי של היצירה, מקבל ערך 1 ושאר הסגנונות 0. בפועל, חישוב השגיאה מתבצע בין וקטור הסגנון האמיתי באופן "מוחלק" ללא ערכי 0 ו-1 מוחלטים, כלומר באינדקס המתאים יהיה ערך קרוב מאוד ל-1 ובשאר האינדקסים יהיה ערך הקרוב ל-0.

2.2.2. אימון הגנרטור

הגנרטור מתאמן על מנת ליצור יצירות אומנות סינתטיות שנראות כמו יצירות אמיתיות ובנוסף מתאמן על מנת למקסם את מדד ה-Style Ambiguity, המשוב המגיע מהדיסקרימינטור. בתחרות מול הדיסקרימינטור, הגנרטור מנסה למזער את פונקציית העלות באמצעות הביטוי:

$$\min_G V(D, G) = \mathbb{E}_{z \sim P_z} \left[\log \left(1 - D_r(G(z)) \right)_{(1)} - \sum_{k=1}^K \left(\frac{1}{K} \log(D_c(c_k|G(z))) + \left(1 - \frac{1}{K} \right) \log \left(1 - D_c(c_k|G(z)) \right) \right)_{(2)} \right]$$

כאשר חלקו הראשון של הביטוי משמעותו מזעור פונקציית העלות ולהערים על הדיסקרימינטור בכך שסיווג כמה שפחות יצירות סינתטיות כמזויפות. החלק השני משמעותו מזעור פונקציית העלות בגרימת Cross Entropy גדול עבור הסתברות סגנון היצירה בהינתן יצירה מהגנרטור $p(c|G(z))$ עם התפלגות אחידה, על מנת למקסם את אי וודאות סגנון היצירה הסינתטית, Style Ambiguity. יש לציין כי בניגוד לדיסקרימינטור, הגנרטור אינו מתאמן בצורה מפורקת עם סט נתונים מתווג, כך שהוא לומד ליצור יצירות סינתטיות אמיתיות בסגנון חדש באמצעות המשוב המגיע מהדיסקרימינטור (האם היצירה סווגה כאמיתית/מזויפת וכמה גדולה אי וודאות הסגנון, מדד ה-Style Ambiguity) [2].

פונקציית ההפסד של הגנרטור בפרויקט כוללת את שני מרכיבים שהוצגו ב-[2] הנוגעים לגנרטור: האחד ערך השגיאה של הגנרטור בנוגע לסיווג יצירותיו הסינתטיות כאמיתיות ע"י הדיסקרימינטור, כאשר ערך השגיאה השני הינו בנוגע לאי ודאות סגנון היצירות הסינתטיות שמפיק הגנרטור, מדד

ה-Style Ambiguity. בכדי לחשב את שגיאת היצירות הסינתטיות, נשתמש בממוצע פונקציית ה-Sigmoid cross entropy בין ההסתברויות שמפיק הדיסקרימינטור עבור היצירות הסינתטיות לבין ההסתברות האידאלית. בניגוד לפונקציית ההפסד של הדיסקרימינטור, כעת מנקודת מבטו של הגנרטור, ההסתברות האידאלית עבור היצירות המזויפות הינה 1 ולא 0, כלומר שהגנרטור מצליח להערים על הדיסקרימינטור בכך שמסוגל יצירה סינתטית כאמיתית. המרכיב השני, אי ודאות הסגנון, מחושב באמצעות ממוצע פונקציית ה-Softmax cross entropy בין ההסתברויות שמפיק הדיסקרימינטור עבור סיווג סגנון היצירות הסינתטיות לבין ההסתברות האידאלית. כאמור, במוצא סיווג הסגנון של הדיסקרימינטור (פונקציית ה-Softmax) מתקבל סיווג סגנון עבור יצירה נתונה בצורת וקטור הסתברויות עבור כל סגנון אפשרי, כאשר הסיווג בפועל זהו ערך ה-argmax כלומר הסגנון שמתאים לאינדקס בווקטור שקיבל את ההסתברות הגבוהה ביותר. ההסתברות האידאלית במקרה של אי ודאות הסגנון תהיה אפוא וקטור שבו כל ההסתברויות שוות עבור כל הסגנונות (במקרה של 27 סגנונות, נרצה וקטור הסתברויות של $1/27$ עבור כל סגנון), כך שהדיסקרימינטור אינו מצליח לסווג את סגנון היצירה הסינתטית לסגנון קיים (למעשה הסיווג יהיה אקראי וכך נדע כי הסגנון שיצר הגנרטור הוא מקורי).

2.2.3 אלגוריתם האימון

אימון ובדיקת מודל CAN בפרויקט נעשה בשימוש בסט נתונים מתוך WikiArt בעל 81,002 יצירות המחולקות ל-27 סגנונות שונים (טבלה 1). סט הנתונים חולק לסט אימון וסט בדיקה ביחס של 70% ו-30% בהתאמה, כלומר סט האימון מכיל 56,702 יצירות, המשמשות את אלגוריתם האימון ואילו סט הבדיקה מכיל 24,300 יצירות, המשמשות להערכת ביצועי מודל ה-CAN. תהליך אימון מודל CAN בפרויקט דומה ל-[2] ומתבצע במשך 100 epochs בעזרת אלגוריתם Stochastic Gradient Descent מסוג mini-batch, כאשר גודל ה-batch הינו 32. המשמעות היא שסט האימון חולק ל-1771 קבוצות של 32 יצירות בכל קבוצה, חוץ מהקבוצה האחרונה שמכילה 30 יצירות, כאשר חישוב הגרדיאנטים ב-backpropagation ועדכון משקלי הדיסקרימינטור והגנרטור למזעור פונקציית השגיאה הנזכרות לעיל, מתבצעים לאחר מעבר על כל קבוצה עד אשר מתבצע מעבר על כל היצירות שבסט האימון, כלומר מסתיים epoch אחד. תהליך זה מתבצע 100 פעמים, עם קצב למידה של 0.0002 ועם שיפוע זליגה של פונקציית האקטיבציה מסוג LeakyReLU של 0.2 (כלומר במקרה שבו הנוירון כבוי, יעבור 20% מהגרדיאנט) ו-Adam Optimizer. סט הנתונים עובר נרמול של ערכי הפיקסלים, בכל ערוץ ב-RGB, מ-[0,255] ל-[-1,1].

בהתאם לאלגוריתם האימון שבאורך 5, אימון מודל CAN מתבצע באופן הבא:

1. הקלט x במודל הינו 32 יצירות מסט היצירות האמיתיות, עם תוויט סגנון היצירה $\hat{c}$ המתאימה לכל יצירה. הפרמטר S הוא מספר הקבוצות לחלוקת סט היצירות לאימון, 1772, בהתאם לגודל ה-batch ומספר היצירות הכולל בסט האימון.
2. הלולאה מתבצעת כאמור 1772 פעמים בכל epoch, במשך 100 epochs.
3. דגימה של 32 וקטורים מתוך המרחב הלטנטי (התפלגות נורמאלית) באורך 100.
4. העברת הווקטורים כקלט לגנרטור ליצירת 32 יצירות מזויפות, $\hat{x}$.
5. העברת קבוצת היצירות האמיתיות x כקלט לדיסקרימינטור וקבלת כפלט את ההסתברות ליצירה אמיתית עבור כל יצירה בקבוצה (וקטור באורך 32). חישוב השגיאה עבור היצירות האמיתיות מתבצע כ-Sigmoid Cross Entropy עם ההסתברויות האידאליות $[0.8, 1.2]$.

6. העברת קבוצת היצירות האמיתיות x , יחד עם תווית הסגנון לכל יצירה $\hat{c}$, וקבלת פלט סיווג היצירות לאחד מ-27 הסגנונות (מטריצה של 32 שורות ו-27 עמודות). חישוב השגיאה עבור סיווג היצירות לסגנונות השונים מתבצע כ-Softmax Cross Entropy עם התוויות האמיתיות מקודדות כ-one hot בצורה מוחלקת.
7. העברת קבוצת היצירות המזויפות $\hat{x}$ כקלט לדיסקרימינטור לקבלת פלט ההסתברויות ליצירה אמיתית עבור כל יצירה (וקטור באורך 32). חישוב השגיאה עבור היצירות המזויפות מתבצע כ-Sigmoid Cross Entropy עם ההסתברויות האידאליות $[0, 0.3]$ עבור הדיסקרימינטור והסתברות אידאלית של 1 עבור הגנרטור.
8. חישוב מדד ה-Style Ambiguity כ-Softmax Cross Entropy בין ההסתברויות שחזה הדיסקרימינטור עבור סגנון היצירות המזויפות $\hat{x}$ לבין וקטורי ההסתברות האידאלית באורך 27 עם הסתברות 1/27 עבור כל סגנון.
9. חישוב השגיאה הכוללת של הדיסקרימינטור, הכוללת את השגיאה על היצירות האמיתיות x , השגיאה על היצירות המזויפות $\hat{x}$ והשגיאה על סיווג היצירות האמיתיות x בהתאם לתוויות $\hat{c}$ בפועל.
10. עדכון משקלי הדיסקרימינטור בהתאם לגרדיאנטים.
11. חישוב השגיאה הכוללת של הגנרטור, הכוללת את השגיאה על היצירות המזויפות $\hat{x}$ והשגיאה שחושבה עבור מדד ה-Style Ambiguity.
12. עדכון משקלי הגנרטור בהתאם לגרדיאנטים.
13. איטרציה אחת מסתיימת והלולאה מתחילה משלב 3 שוב, עד אשר עוברים על כל 1772 הקבוצות ומסתיים epoch אחד. אלגוריתם האימון מתחיל מחדש ב-epoch הבא, עד אשר מסתיימים 100 epochs.

Algorithm 1 CAN training algorithm with step size α , using mini-batch SGD for simplicity.

```

1: Input: mini-batch images  $x$ , matching label  $\hat{c}$ , number of training batch steps  $S$ 
2: for  $n = 1$  to  $S$  do
3:    $z \sim \mathcal{N}(0, 1)^Z$  {Draw sample of random noise}
4:    $\hat{x} \leftarrow G(z)$  {Forward through generator}
5:    $s_D^r \leftarrow D_r(x)$  {real image, real/fake loss}
6:    $s_D^c \leftarrow D_c(\hat{c}|x)$  {real image, multi class loss}
7:    $s_G^f \leftarrow D_r(\hat{x})$  {fake image, real/fake loss}
8:    $s_G^c \leftarrow \sum_{k=1}^K \frac{1}{K} \log(p(c_k|\hat{x}) + (1 - \frac{1}{K})(\log(p(c_k|\hat{x})))$  {fake image Entropy loss}
9:    $\mathcal{L}_D \leftarrow \log(s_D^r) + \log(s_D^c) + \log(1 - s_G^f)$ 
10:   $D \leftarrow D - \alpha \partial \mathcal{L}_D / \partial D$  {Update discriminator}
11:   $\mathcal{L}_G \leftarrow \log(s_G^f) - s_G^c$ 
12:   $G \leftarrow G - \alpha \partial \mathcal{L}_G / \partial G$  {Update generator}
13: end for

```

איור 5: אלגוריתם האימון במודל CAN [2]

אלגוריתם האימון מבצע אימון לגנרטור בהפקה של יצירה סינתטית מתוך וקטור רעש אקראי. היצירה הסינתטית ויצירה אמיתית מסט הנתונים עוברות אל הדיסקרימינטור, שמתאמן בסיווג היצירות לאמיתיות או מזויפות. בנוסף, הדיסקרימינטור מתאמן בסיווג הסגנון המתאים בעזרת התמונות האמיתיות מסט הנתונים. עדכון משקלי הדיסקרימינטור מתבצע בעזרת השגיאה בסיווגים מדויקים לאמיתי או מזויף ולסגנון האומנות, Style Classification. עדכון משקלי הגנרטור מתבצע בעזרת

השגיאה בסיווגים של הדיסקרימינטור לאמיתי או מזויף (מנסה להשתפר בלהערים על הדיסקרימינטור) ומדד אי ודאות הסגנון,

Style Ambiguity, כאשר מנסה למקסם את אי הודאות ובכך ליצור יצירות בסגנון ייחודי.

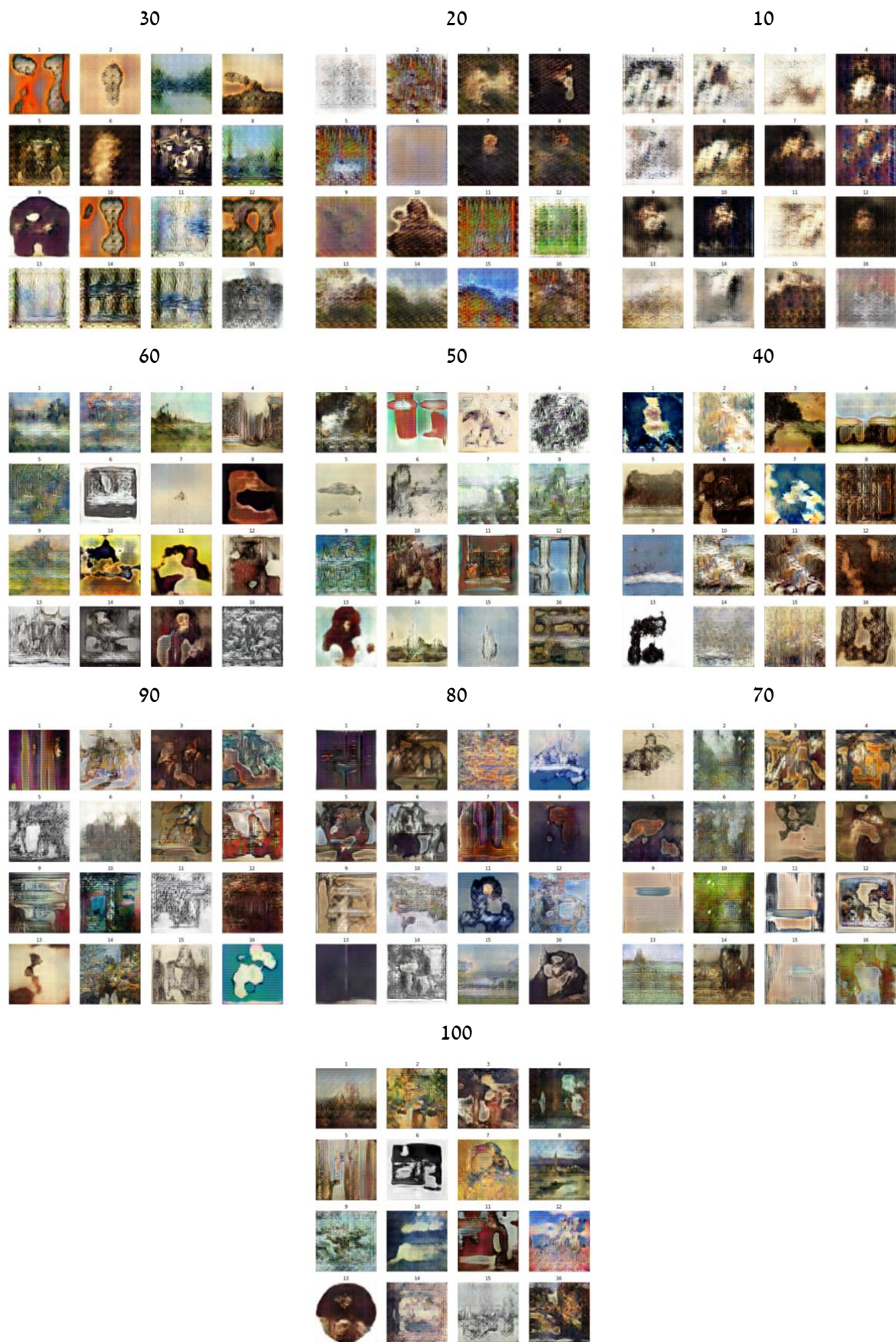
טבלה 1: מספר היצירות בכל סגנון עבור כל סט נתונים

Style	Train	Validation	Total
Abstract Expressionism	1906	874	2780
Action painting	70	28	98
Analytical Cubism	83	27	110
Art Nouveau Modern	3054	1267	4321
Baroque	2954	1270	4224
Color Field Painting	1134	481	1615
Contemporary Realism	356	125	481
Cubism	1554	681	2235
Early Renaissance	955	435	1390
Expressionism	4742	1992	6734
Fauvism	650	284	934
High Renaissance	917	425	1342
Impressionism	9096	3935	13031
Mannerism Late Renaissance	892	383	1275
Minimalism	935	402	1337
Naive Art Primitivism	1669	734	2403
New Realism	228	86	314
Northern Renaissance	1759	790	2549
Pointillism	363	150	513
Pop Art	1068	414	1482
Post Impressionism	4527	1906	6433
Realism	7488	3220	10708
Rococo	1494	593	2087
Romanticism	4896	2101	6997
Symbolism	3161	1367	4528
Synthetic Cubism	162	54	216
Ukiyo e	589	276	865
Total Images	56702	24300	81002

2.2.4. בדיקה של מודל CAN תוך כדי האימון

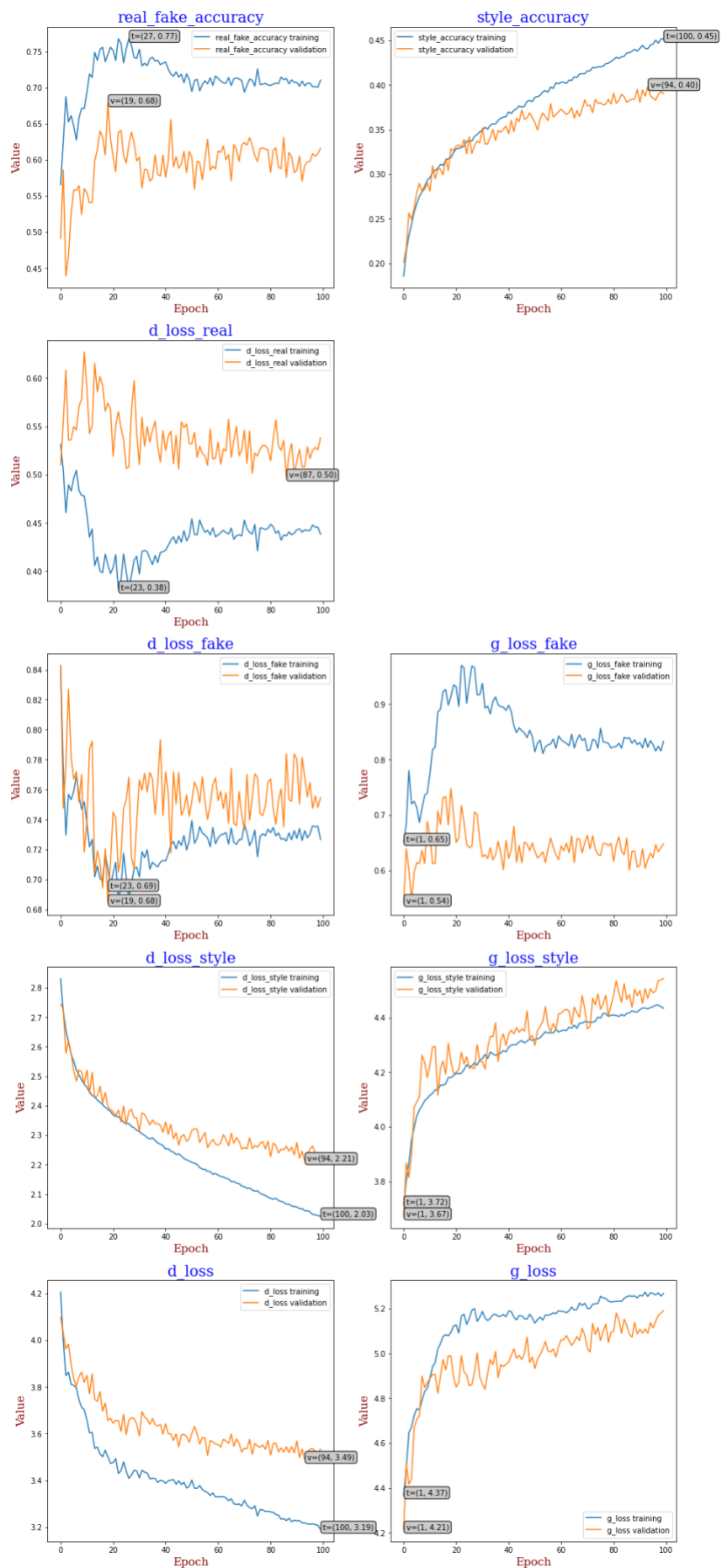
בניגוד ל-[2] שבו התבצעו סקרים הנזכרים לעיל על מנת להעריך את ביצועי המודל, בפרויקט לאחר סיום כל epoch מתבצעת בדיקה של ביצועי הדיסקרימינטור והגנרטור באמצעות סט הבדיקה שמכיל יצירות חדשות שאינן משתתפות בתהליך האימון, בכדי להבטיח שביצועי המודל טובים גם עבור סט האימון וגם עבור סט הבדיקה וכך למנוע מצב של overfitting. בנוסף, לאחר כל epoch מוצגות יצירות שמופקות מהגנרטור על מנת לעקוב אחר התקדמות האימון ושיפור איכות היצירות (איור 6). המדדים שנבדקים במהלך האימון והבדיקה הינם:

- שגיאת הדיסקרימינטור (d_loss): סיווג היצירות האמיתיות כאמיתיות (d_loss_real), סיווג היצירות המזויפות כמזויפות (d_loss_fake) וסיווג סגנון היצירה כהלכה (d_loss_style).
 - שגיאת הגנרטור (g_loss): סיווג היצירות המזויפות כאמיתיות ע"י הדיסקרימינטור (g_loss_fake) וסיווג סגנון היצירה המזויפת בצורה אקראית ולא וודאית (g_loss_style).
 - דיוק סיווג הדיסקרימינטור: סיווג היצירות כהלכה לאמיתיות ומזויפות ($real_fake_accuracy$) וסיווג סגנון היצירות נכונה ($style_accuracy$).
- תוצאות המדדים נמצאות באיור 7, כאשר ניתן לראות כי דיוק הדיסקרימינטור עבור סיווג היצירות האמיתיות והמזויפות מתייצב סביב ה-70% בסט האימון וסביב ה-60% בסט הבדיקה, כמצופה ממודל מסוג GAN. דיוק הדיסקרימינטור בסיווג היצירות ל-27 הסגנונות הולך וגדל עם האימון, כלומר אכן מתרחש תהליך של למידה, אם כי קיים חשש של Overfitting במקרה של המשך האימון עקב הפער ההולך וגדל בתוצאות הסיווג עבור סט האימון והבדיקה. ערך פונקציית ההפסד של הדיסקרימינטור עבור סיווג היצירות האמיתיות והמזויפות הולך ומתייצב עם תהליך האימון, כך גם עבור פונקציית ההפסד של הגנרטור עבור סיווג היצירות המזויפות כאמיתיות ע"י הדיסקרימינטור. ערך פונקציית ההפסד עבור סגנון היצירות הולך וקטן עבור הדיסקרימינטור (מצליח ללמוד כהלכה את הסיווג הנכון לסגנון עם האימון, אם כי גם כאן ניתן לראות סכנת Overfitting) והולך וגדל עבור הגנרטור, אם כי היינו מצפים לירידה בשלב מסוים כאשר הגנרטור מצליח לשפר את מדד ה-Style Ambiguity עם התקדמות האימון. לבסוף ניתן לראות את ערך פונקציית ההפסד הכוללת של הדיסקרימינטור והגנרטור שהולכת ומשתפרת עבור הדיסקרימינטור ודועכת עבור הגנרטור, בעיקר בגלל ערך פונקציית ההפסד של הסגנון.



איור 6 : היצירות המתקבלות מהגרנטור במהלך האימון

ניתן לראות כי איכות היצירות המתקבלות מגרנטור ה-EMA הולכות ומשתפרות עם האימון (לפי מספר ה-epoch), כאשר הן כוללות צבעים ואלמנטים רבים יותר והשוני בין היצירות גדול יותר.



שורה ראשונה - דיוק סיווגי הדיסקרימינטור : מצד שמאל סיווג היצירות לאמיתי ומזויף, מצד ימין סיווג היצירות לסגנון המתאים.

שורה שנייה - ערך פונקציית ההפסד של הדיסקרימינטור עבור סיווג היצירות האמיתיות כהלכה.

שורה שלישית - ערך פונקציית ההפסד עבור היצירות המזויפות : מצד שמאל ערך הדיסקרימינטור לסיווג נכונה, מצד ימין ערך הגנרטור להצלחה בהפקת יצירות מזויפות המסווגות כאמיתיות.

שורה רביעית - ערך פונקציית ההפסד עבור סגנון היצירה : מצד שמאל ערך הדיסקרימינטור לסיווג נכונה, מצד ימין ערך הגנרטור להצלחה בהפקת יצירות מזויפות מקוריות (Style Ambiguity).

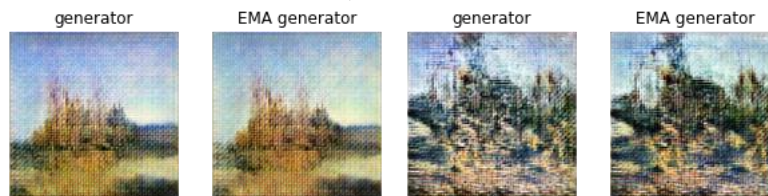
שורה חמישית - ערכי פונקציית ההפסד הכוללת : מצד שמאל עבור הדיסקרימינטור, מצד ימין עבור הגנרטור. בכל גרף, ניתן לראות באיזה epoch נמצאת נקודות האופטימום, באימון ובבדיקה.

2.2.5 הפקת יצירות האומנות הסינתטיות בפרויקט ע"י גנרטור EMA

במהלך ולאחר תהליך האימון, ניתן להשתמש בגנרטור עם משקלי הנוירונים הנוכחיים על מנת להפיק יצירות אומנות מתוך וקטורי רעש אקראיים, הנדגמים מהתפלגות נורמאלית. כלומר בהינתן למשל 10 וקטורים שונים, נוכל לקבל 10 יצירות אומנות סינתטיות שונות, כאשר בסוף האימון היצירות נראות כאמיתיות ובעלות סגנון מקורי. על מנת לשפר את איכות היצירות המתקבלות מהגנרטור, נעשה שימוש בפרויקט בטכניקת EMA או בשמה המלא Exponential Moving Average. טכניקת EMA מנסה למנוע מצב של שינוי דרמטי במשקלי הגנרטור במהלך תהליך האימון וכך למנוע מצב של התכנסות במינימום מקומי של פונקציית ההפסד. הטכניקה ממומשת בפרויקט באמצעות גנרטור נוסף, הנקרא EMA generator, שבעת כל עדכון משקלי הגנרטור הקלאסי (בכל batch) לאחר חישוב הגרדיאנטים, מעדכן את משקליו באמצעות ממוצע משוקלל של המשקלים הקודמים שלו ושל משקלי הגנרטור. המשמעות היא שמשקלי ה-EMA generator אינם מתעדכנים באופן מלא מתוך הגרדיאנטים כמו בגנרטור הקלאסי, אלא מתעדכנים רק בחלקם מתוך משקלי הגנרטור המעודכנים. בפרויקט נעשה שימוש בממוצע משוקלל עם מתן משקל β של 99% למשקלי ה-EMA generator ו-1% למשקלי הגנרטור החדשים. כך ניתן להבטיח שינויים קלים במשקלי הגנרטור מסוג EMA בכל עדכון של המשקלים בעת מעבר על היצירות שבכל batch. למשל, אם משקל נוירון מסוים בגנרטור ה-EMA מה-batch הקודם הוא $\theta_{EMA}^{(t-1)}$ והמשקל החדש מה-batch הנוכחי שקיבל הנוירון בגנרטור הקלאסי הוא $\theta^{(t)}$, אזי במקום למחוק את $\theta_{EMA}^{(t-1)}$ ולשים במקומו את $\theta^{(t)}$, המשקל החדש יהיה ממוצע משוקלל של המשקל הקודם $\theta_{EMA}^{(t-1)}$ והמשקל החדש $\theta^{(t)}$ של הגנרטור הקלאסי (עם $\beta = 0.99$):

$$\theta_{EMA}^{(t)} = \beta \theta_{EMA}^{(t-1)} + (1 - \beta) \theta^{(t)} = 0.99 \theta_{EMA}^{(t-1)} + 0.01 \theta^{(t)}$$

[10]. באיור 8 מוצגת דוגמה להבדל באופי היצירות המתקבלות משני הגנרטורים.



איור 8 : יצירה סינתטית המופקת מגנרטור EMA לעומת הגנרטור הקלאסי

ניתן לראות כי היצירות זהות במבנה הכללי, מכיוון שהופקו מאותו וקטור אקראי בכניסת הגנרטורים, כלומר הבדל משקלי הנוירונים בשני הגנרטורים יהיה קטן מספיק על מנת לספק יצירה דומה מאותו הווקטור. למרות זאת בפרטים העדינים יותר, נראה כי גנרטור ה-EMA מפיק תמונה חדה יותר כך שהשימוש בטכניקה זו עשוי לשפר את היצירות הסינתטיות.

3. Deep Interactive Evolution

בפרק זה נציג גישה בשם Deep Interactive Evolution (DeepIE), המשלבת בין מודל מבוסס GAN שעבר תהליך של אימון, לבין Interactive Evolutionary Computation (IEC), שיטה לשימוש באלגוריתם גנטי בשילוב עם בחירות של המשתמש האנושי, המשמשות כפונקציית ערך הכשירות. בגישה זו הגנרטור של מודל ה-GAN מקבל וקטור מהמרחב הלטנטי ומפיק דגימות סינתטיות שנראות אמיתיות מתוך התפלגות הנתונים בסט האימון הרצוי, כאשר המשתמש יכול לבחור את הדגימות שתמשכנה בתהליך האבולוציה כרצונו. יתרונה של גישה זו בכך שהיא מאפשרת שליטה נרחבת יותר של המשתמש על הדגימות הסינתטיות שהגנרטור מפיק, כאשר ללא בחירות המשתמש יפיק דגימות אקראיות מתוך התפלגות הנתונים בסט שעליו התאמן [3]. בהמשך הפרק נראה כיצד ניתן להשתמש בגישה זו למטרה של יצירת אומנות סינתטית בסגנון מקורי באמצעות CAN שעבר אימון, אלגוריתם גנטי ובחירות המשתמש. את הרקע המלא אודות אלגוריתם גנטי, שיטת ה-IEC וגישת ה-DeepIE ניתן למצוא בפרק 4 של העבודה המסכמת.

3.1. מימוש גישת ה-DeepIE בפרויקט

גישת ה-DeepIE בפרויקט בהתאם ל-[3] (איור 9), כאשר בשלב הראשון מוצגות למשתמש 16 יצירות סינתטיות לבחירה. הווקטורים שמהם נוצרו היצירות הנבחרות ממשיכים באבולוציה, כאשר באמצעות פעולות שחלוף אחיד נוצרים וקטורים חדשים, בעלי שילוב תכונות של שתי יצירות ובנוסף נדגמים שני וקטורים חדשים לגמרי. בשלב הבא כל הווקטורים עוברים מוטציה בהסתברות של 50%, כאשר וקטורי הרעש לביצוע המוטציה נדגמים מהתפלגות נורמאלית עם סטיית תקן הנקבעת ע"י המשתמש, לשליטה בעוצמת הרעש (המוטציה). בסוף התהליך, הווקטורים מוזנים מחדש לגנרטור שמפיק מהם יצירות חדשות להצגה למשתמש וחוזר חלילה. על מנת לממש את אלגוריתם DeepIE שבאיור 10, ראשית יש לאמן מודל מבוסס GAN שיוכל להפיק דגימות סינתטיות להצגה למשתמש. בפרויקט נעשה שימוש כאמור במודל CAN שעבר אימון מבעוד מועד, על מנת להציג למשתמש יצירות לבחירה. לאחר האימון של מודל CAN נשמרים משקלי הניורונים כך שניתן לשחזרם בכל עת, ללא צורך באימון מחדש. לכן בשלב הראשון בפרויקט, התואם לשלב הראשון באלגוריתם DeepIE, נבנות שכבות מודל CAN, כאשר המשקלים משוחזרים מהזיכרון וכך נחסך זמן יקר. לאחר מכן מתבצעת דגימה של 16 וקטורים אקראיים באורך 100 מהמרחב הלטנטי, כלומר מהתפלגות הנורמאלית (תואם לשלב השלישי). על מנת לבצע את התהליך האבולוציוני שבשלב 4-18, יש לממש את פונקציית השחלוף האחיד (Uniform) ואת פונקציית המוטציה (Mutate).

3.1.1. מימוש פונקציית השחלוף האחיד

הפונקציה שמומשה בפרויקט בהתאם לשלבים 19-22 באלגוריתם ה-DeepIE, מקבלת כקלט את אוכלוסיית הפריטים הנבחרים, כלומר במקרה של מודל CAN הפונקציה מקבלת את הווקטורים באורך 100 שמופיעו ע"י הגנרטור ליצירות האומנות שנבחרו ע"י המשתמש. מטרת פונקציית השחלוף האחיד היא ליצור וקטור חדש משילוב אקראי של איברי שני וקטורים מאוכלוסיית הפריטים הנבחרים, כלומר למעשה ליצור יצירת אומנות חדשה מתכונות הנבחרות אקראית משתי יצירות:

- בהתאם לשלבים 20-21, הפונקציה דוגמת אקראית שני וקטורים (מצב שבו וקטור אחד נבחר פעמיים גם כן אפשרי).

- בהתאם לשלב 22, מתבצעת הטלת מטבע הוגן 100 פעמים, עבור כל אינדקס i בווקטור החדש שנוצר, לקביעה האם האיבר באינדקס הנוכחי יילקח מהווקטור הראשון או השני.
- לבסוף הפונקציה מחזירה את הווקטור החדש שנוצר כאמור בחלקו מהווקטור הראשון וחלקו מהווקטור השני, בהתאם להטלות המטבע:

$$child_i = RandomChoise(parent_i^A, parent_i^B)$$

באיור 11 ניתן לראות דוגמה לפעולת שחלוף אחיד והשפעתה על תכונות יצירות האומנות הסינתטיות המופקות מהגנרטור, כאשר מתוך שני וקטורים (a ו-b) נוצר וקטור שלישי (child) בעזרת פעולת השחלוף.

3.1.2 מימוש פונקציית המוטציה

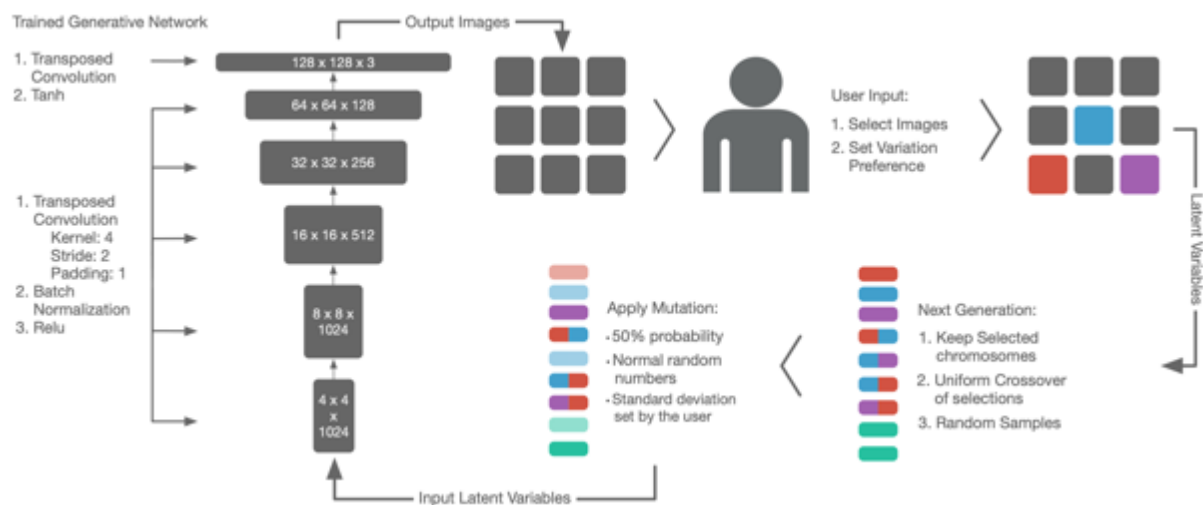
- הפונקציה שמומשה בהתאם לשלבים 23-25 באלגוריתם ה-DeepIE, מקבלת כקלט וקטור יחיד שמייצג יצירה מסוימת וערך סטיית התקן של ההתפלגות הנורמאלית, שנקבע ע"י המשתמש:
- בהתאם לשלב 24, מתבצעת הטלת מטבע הוגן לקביעה האם תתבצע מוטציה, כלומר הסיכוי שתתבצע מוטציה בווקטור המתקבל כקלט הוא 50%.
 - בהתאם לשלב 25, נדגם וקטור רעש מהתפלגות נורמאלית, בעלת סטיית התקן שנקבעה.
 - במידה והטלת המטבע קבעה כי אכן תתבצע מוטציה, מבצעים חיבור בין וקטור הרעש ובין הווקטור שבקלט לקבלת וקטור "מורעש". ביצוע המוטציה בווקטור הקלט שמייצג יצירת אומנות מתבצע למעשה ע"י הוספת ערך לכל איבר, כך שתכונות היצירה משתנות בהתאם לרעש. סטיית התקן קובעת את עוצמת המוטציה, כאשר סטיית התקן נמצאת בסקלה שבין 0 ל-1. במקרה של ערך סטיית תקן של 1, נקבל התפלגות נורמאלית סטנדרטית, כך שעוצמת הרעש תהיה זהה לעוצמת האיברים שבווקטור הקלט, כלומר מוטציה עם רעש בעוצמה מקסימאלית. במקרה שבו סטיית התקן הינה 0, לא תתבצע מוטציה כלל ועבור ערכי סטיית התקן שבין 0 ל-1, נקבל תוספת של רעש לווקטור הקלט בעוצמה פחותה יותר, בהתאם לערך שנקבע.
- באיור 11 ניתן לראות דוגמה לווקטורים המייצגים את היצירות שעוברים מוטציה, בהתאם לסטיית התקן של ההתפלגות הנורמאלית שממנה נדגמים וקטורי הרעש. בדוגמה ניתן להבחין בהשפעה של סטיית התקן על עוצמת המוטציה וכפועל יוצא את ההשפעה על היצירות.

3.1.3 מימוש פונקציית האבולוציה

- הפונקציה מומשה בהתאם ללולאה שבשלבים 4-18 באלגוריתם ה-DeepIE ומשמשת לביצוע התהליך האבולוציוני, כאשר כל איטרציה נחשבת כדור אחד. הפונקציה מקבלת כקלט את 16 הווקטורים באורך 100 שנדגמו מהמרחב הלטנטי (אוכלוסיית הפריטים), את האינדקסים של היצירות שנבחרו ע"י המשתמש ואת ערך סטיית התקן של ההתפלגות הנורמאלית שממנה נדגם וקטור הרעש לביצוע המוטציה:
- 16 הווקטורים שנדגמו בשלב 3 מוזנים לגנרטור על מנת שימפה אותם ל-16 יצירות אומנות המוצגות למשתמש לבחירה (שלבים 5-7).
 - לאחר שהמשתמש סיים לבחור את היצירות המועדפות עליו, הווקטורים שמופו ליצירות הללו נשמרים כאוכלוסיית הפריטים הנוכחית בתהליך האבולוציוני (שלבים 8-9).
 - מספר פעולות השחלוף האחיד להפקת וקטורים חדשים שימופו ליצירות אומנות מתוך תכונות היצירות שנבחרו, מחושב כהפרש בין 16 היצירות המוצגות לבין מספר היצירות הנבחרות, פחות 2

יצירות חדשות הנוצרות בכל דור (שלבים 11-13). במקרה של ערך שלילי, לא תתבצענה פעולות שחלוף כלל. למשל במקרה שהמשתמש בוחר 5 יצירות, תתבצענה $9 = 16 - 5 - 2$ פעולות שחלוף בהתאם למימוש שהוצג בסעיף 3.1.1. לאחר כל פעולת שחלוף, הווקטור המתקבל עובר מוטציה בהתאם למימוש מסעיף 3.1.2 וערך סטיית התקן שקבע המשתמש.

- שני וקטורים חדשים נדגמים מההתפלגות הנורמאלית הסטנדרטית, כאשר במידה והמשתמש בחר 15 יצירות מועדפות, ידגם רק וקטור חדש ואילו אם בוחר המשתמש את כל 16 היצירות, לא ידגמו וקטורים חדשים כלל (שלבים 14-15).
 - הווקטורים של היצירות הנבחרות עוברים גם כן תהליך של מוטציה בהתאם לסטיית התקן (שלב 16).
 - לבסוף הפונקציה מחזירה את אוכלוסיית הפריטים החדשה, הכוללת 16 וקטורים חדשים שימופו ליצירות אומנות להצגה למשתמש. תחילה תוצגנה היצירות הנבחרות לאחר ביצוע מוטציה, לאחר מכן תוצגנה היצירות שנוצרו מפעולות השחלוף והמוטציה והיצירות האחרונות תהיינה היצירות החדשות לגמרי, שנוצרו מהווקטורים החדשים שנדגמו (שלב 17).
- התהליך האבולוציוני חוזר על עצמו, באמצעות קריאה מחדש לפונקציית האבולוציה כאשר המשתמש יכול לבחור מחדש יצירות מועדפות, לצד שינוי ערך סטיית התקן המשפיע על עוצמת המוטציה, עד אשר המשתמש בוחר לסיים את התהליך. באיור 12 ניתן לראות דוגמה לתהליך אבולוציוני ולאופן שבו נוצר דור חדש מהדור שלפניו, בהתאם ליצירות הנבחרות ולשינוי ערך סטיית התקן ע"י המשתמש.



איור 9: שלבי גישת ה-DeepIE [3]

ראשית, מודל GAN שעבר אימון יוצר אוכלוסייה ראשונית מתוך וקטורי ייצוג אקראיים. שנית, המשתמש בוחר את הדגימות (או התכונות שבדגימות) האהובות עליו. לאחר מכן, לצד הווקטורים של היצירות הנבחרות, האלגוריתם הגנטי יוצר וקטורי ייצוג חדשים מתוך הווקטורים של הדגימות שנבחרו, באמצעות פעולות שחלוף אחיד (Uniform Crossover) ובנוסף יוצר וקטורים אקראיים חדשים להצגת תכונות חדשות. לבסוף האלגוריתם מבצע מוטציות בהסתברות של 50%, בהתאם לסטיית התקן שנבחרה והווקטורים מוזנים שוב אל הגנרטור ליצירת דגימות סינתטיות המוצגות למשתמש לבחירה.

Algorithm 2 Deep Interactive Evolution.Defaults $m, n \leftarrow 20, \mu \leftarrow 0, \sigma^2 \leftarrow 1, p \leftarrow .5, foreign \leftarrow 2$

```

1:  $G_\theta \leftarrow \text{trainGAN}(\text{data})$  ▷ we use WGAN-GP [8]
2:  $\text{interface} \leftarrow \text{Interface}()$ 
3:  $Z \leftarrow m$  by  $n$  matrix where  $z_{i,j} \sim \mathcal{N}(\mu, \sigma^2)$  ▷ population size  $m$  with  $n$  latent variables each
4: repeat
5:    $\text{images} \leftarrow G_\theta(Z)$ 
6:    $\text{interface.display}(\text{images})$ 
7:   wait until  $\text{interface.buttonNextPressed}()$  is True
8:    $\text{indices} \leftarrow \text{interface.getSelectedImages}()$ 
9:    $\text{selection} \leftarrow Z_{\text{indices}}$ 
10:   $s^2 \leftarrow \text{interface.getMutationParameter}()$ 
11:   $\Delta \leftarrow m - \text{length}(\text{selection})$ 
12:   $x \leftarrow \max(0, \Delta - \text{foreign})$ 
13:   $\text{cross} \leftarrow x$  by  $n$  matrix where  $\text{cross}_i \sim \text{mutate}(\text{uniform}(\text{selection}), s^2)$ 
14:   $x \leftarrow \max(\text{foreign}, \Delta)$ 
15:   $\text{new} \leftarrow x$  by  $n$  matrix where  $\text{new}_{i,j} \sim \mathcal{N}(\mu, \sigma^2)$ 
16:   $\text{selection} \leftarrow \text{apply mutate with } s^2 \text{ to each } \text{selection}_i$ 
17:   $Z \leftarrow \text{selection} + \text{cross} + \text{new}$ 
18: until user stops
19: function UNIFORM(population)
20:    $a \leftarrow$  random individual from population
21:    $b \leftarrow$  random individual from population
22:    $\text{mask} \leftarrow$  vector of length  $n$  where  $\text{mask}_i \sim \text{Bernoulli}(.5)$ 
23:   return  $\text{mask} \cdot a + (1 - \text{mask}) \cdot b$ 
24: function MUTATE(individual, std)
25:    $\text{mutation} \leftarrow \text{Bernoulli}(p)$ 
26:    $\text{noise} \leftarrow$  vector of length  $n$  where  $\text{noise}_i \sim \mathcal{N}(\mu, \text{std})$ 
27:   return  $\text{individual} + \text{mutation} \cdot \text{noise}$ 

```

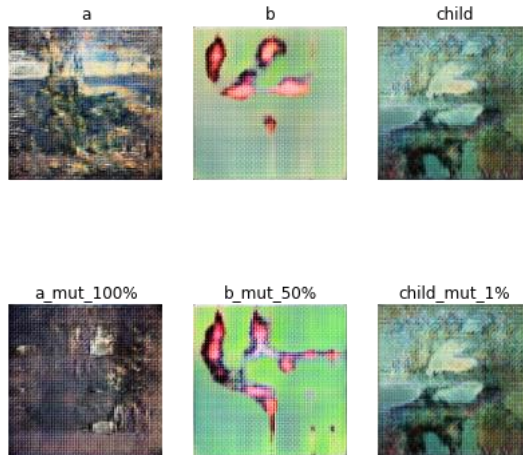
איור 10 : אלגוריתם ה-DeepIE [3]

בשלב הראשון, מבצעים אימון למודל ה-GAN ובשלב השלישי יוצרים את מטריצת הווקטורים האקראיים מהמרחב הלטנטי. בלולאה משורה 4 ועד שורה 18 מתבצע תהליך ה-DeepIE (יצירת דגימות סינתטיות והצגתן למשתמש, בחירת המשתמש, ביצוע פעולות שחלוף ומוטציות, עדכון מטריצת הווקטורים המייצגים והזנת המטריצה לגנרטור ליצירת דגימות בהתאם על מנת להציגן שוב למשתמש). הפונקציה בשורות 19 עד 22 משמשת לביצוע פעולות שחלוף אחיד (Uniform Crossover) בין שני הורים רנדומאליים ואילו הפונקציה בשורות 23 ועד 25 משמשת לביצוע מוטציות באמצעות הוספת וקטור המוטציה מוכפל ברעש, בהתאם לסטיית התקן, אל וקטור הייצוג של הפרט.

```

a: [-0.61287558  0.1828482  -0.39099622 -0.6599564  0.11333589 -0.46548042]
b: [-0.54374671  0.27914506  0.2213918  -0.26553509 -0.06302827 -1.45901167]
child: [-0.61287558  0.27914506 -0.39099622 -0.6599564  -0.06302827 -0.46548042]
a_mut_100%: [ 0.27541311  1.92385569 -2.74441807 -0.96721756 -1.56629817 -0.55750231]
b_mut_50%: [-0.81870585  0.72932135 -0.2840508  -0.21211767 -0.05159081 -1.24806359]
child mut 1%: [-0.60659885  0.26550671 -0.38396076 -0.64398466 -0.05924117 -0.46698117]

```



איור 11 : דוגמה לפעולת שחלוף אחיד ומוטציה

נתבונן בשתי יצירות, a ו-b, שמחן נוצרת יצירה חדשה (child) באמצעות פעולת שחלוף אחיד. למעלה ניתן לראות את ששת האיברים הראשונים בוקטור הייצוג באורך 100 עבור כל יצירה, כאשר עבור כל איבר ביצירה child מתבצעת הטלת מטבע לקביעה האם האיבר באינדקס הנוכחי יילקח מהורה a או מהורה b. האיברים באינדקס הראשון, השלישי, הרביעי והשישי ביצירה child הוגרלו מהורה a ואילו האיברים באינדקס השני והחמישי הוגרלו מהורה b. התהליך חוזר על עצמו עד אשר נבנה וקטור ייצוג חדש משני ההורים באורך 100, שמוזן לגנרטור להפקת היצירה החדשה.

בנוסף, באיור ניתן לראות שלוש דרגות לביצוע מוטציה: הוספת 100% מהרעש לווקטור הייצוג, הוספת 50% מהרעש ולבסוף, הוספת 1% אחד בלבד מהרעש. ביצירה a מתבצעת מוטציה בעוצמה מקסימאלית עם 100% מהרעש, כך שווקטור הייצוג באיור למעלה משתנה לבלי הכר (a לעומת a_mut_100%), לכן היצירה לאחר המוטציה גם כן משתנה לגמרי. ביצירה b מתבצעת מוטציה עם 50% מהרעש, כך שווקטור הייצוג משתנה קלות וכך גם היצירה (b לעומת b_mut_50%). לבסוף עבור יצירת ה-child שנוצרה מפעולת השחלוף אחיד מתבצעת מוטציה עם 1% אחד בלבד מהרעש, כך שווקטור הייצוג כמעט ואינו משתנה וגם היצירה נותרת כמעט ללא שינוי בעיני המתבונן (child לעומת child_mut_1%).



איור 12: דוגמה לתהליך אבולוציוני

בשורה הראשונה משמאל מוצגות למשתמש 16 יצירות סינתטיות שהופקו מהגנרטור, מתוך וקטורים אקראיים שנדגמו מההתפלגות הנורמאלית. המשתמש בוחר את יצירות מס' 5, 9 ו-14 (מסומנות באדום), יחד עם סטיית תקן של 1% לביצוע המוטציה (כלומר 1% מהרעש מתווסף במקרה שמתרחשת מוטציה). מימין ניתן לראות את תוצאת תהליך האבולוציה, כאשר שלוש היצירות הראשונות הן שלוש היצירות שנבחרו ע"י המשתמש (מסומנות באדום), שתי היצירות האחרונות הן היצירות שנוצרו מווקטורים חדשים הנדגמים אקראית מהמרחב הלטנטי (מסומנות בירוק) ושאר היצירות שביניהן הן היצירות שנוצרו מפעולות שחלוף אחיד מהיצירות הנבחרות (מסומנות בכחול). היצירות כאמור עוברות מוטציה בהסתברות של 50%, אך כיוון מסטיית התקן הינה אחוז אחד בלבד, למעשה לא מתרחשת מוטציה הנראת לעין.

בשורה השנייה משמאל מוצג הדור הקודם של היצירות למשתמש, כאשר המשתמש בוחר את היצירות מס' 4 ו-16 (באדום). מימין ניתן לראות את תוצאות האבולוציה, כאשר שתי היצירות הראשונות הן הנבחרות (באדום), שתי היצירות האחרונות הן החדשות (בירוק) והיצירות ביניהן נוצרו מפעולות השחלוף (בכחול). בדור זה סטיית התקן למוטציה הינה 50%, כלומר במקרה שבו מתרחשת מוטציה אזי 50% מהרעש נוסף לווקטור הייצוג, לכן ניתן לראות שינוי משמעותי יותר בחלק מהיצירות (למשל יצירה 16 משמאל שהופכת ליצירה 2 מימין).

בשורה השלישית משמאל מוצגות היצירות מהדור האחרון, כאשר המשתמש בוחר את יצירות 1, 3, 7 ו-14 (באדום), עם סטיית תקן מקסימאלית של 100%. מימין ניתן לראות את ארבע היצירות הנבחרות (באדום), שתי יצירות חדשות (בירוק) וביניהן היצירות שנוצרו מפעולות שחלוף מהיצירות הנבחרות (בכחול). בדור זה כאשר מתרחשת מוטציה, היצירות משתנות בצורה משמעותית ביותר מכיוון שווקטור הרעש כולו נוסף לווקטורי הייצוג.

4. ניתוח תוצאות מודל CAN ושילובו בגישת ה-DeepIE

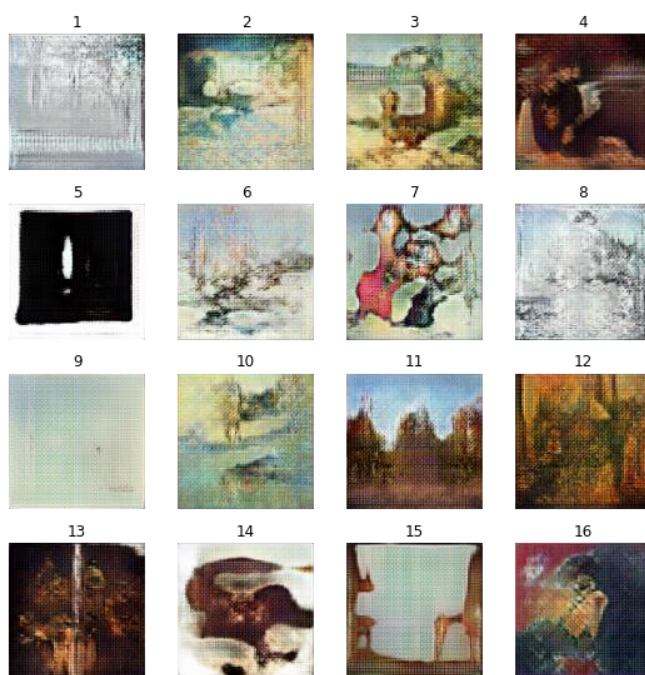
לפי התבוננות באיכות היצירות הסינתטיות שמפיק הגנרטור של מודל CAN בכל epoch (איור 6), ניתן לראות כי אכן מתקיים תהליך של למידה במהלך האימון, כאשר הגנרטור הולך ומשתפר בזכות השיפור במשוב שמתקבל מהדיסקרימינטור, אודות הקרבה ליצירות האמיתיות והסגנון המקורי (Style Ambiguity). בנוסף, ניתן לראות כי הפקת היצירות באמצעות גנרטור EMA משפר את איכות היצירות לעומת היצירות המתקבלות מהגנרטור הקלאסי, ללא חישוב הממוצע המשוקלל של המשקלים (איור 8). כאשר מתבוננים ביצירות סינתטיות שהופקו מגנרטור EMA שבפרויקט (איור 13) ומשווים אותן מראייה סובייקטיבית ליצירות של מודל CAN מ-2 [2] (איור 14), ניתן להסיק כי ביצועי מודל CAN שבפרויקט מתקרבים באיכותם למודל שהוצג ב-2 [2]. היתרון המרכזי של היצירות שהתקבלו ממודל CAN ב-2 [2] הוא ברזולוציה הגבוהה יותר של היצירות שבאמצעותן מתאמן המודל במאמר, בעזרת החומרה החזקה יותר שהייתה זמינה לחוקרים. נזכיר כי בפרויקט עקב מגבלות חומרה, מודל CAN מתאמן באמצעות יצירות אמיתיות ברזולוציה של 64×64 ומנסה להפיק יצירות סינתטיות ברזולוציה גבוהה יותר של 256×256 ואילו ב-2 [2] האימון מתבצע על יצירות ברזולוציה של 256×256 , כך שהגנרטור מסוגל להפיק יצירות סינתטיות ברזולוציה של 256×256 ללא צורך בביצוע שנמוך הרזולוציה ל- 64×64 בעת האימון. על מנת להבין את משמעות שילוב מודל CAN בגישת DeepIE מעבר לדוגמה לתהליך האבולוציוני שבאיור 12, נתבונן בכמה מקרי קצה באיור 15: (שורה 1) במקרה שבו המשתמש בוחר את כל היצירות המוצגות לו להמשך באבולוציה, יחד עם סטיית תקן של 0 שתבטיח שלא תתקיים מוטציה בווקטורים, הדור הבא באבולוציה יהיה זהה למעשה לדור הקודם. הסיבה לכך הינה שכל 16 היצירות ממשיכות לדור הבא, כאשר גודל האוכלוסייה המקסימלי הינו 16 ואין מקום פנוי להוספת וקטורים באמצעות פעולות שחלוף או דגימה של שני וקטורים חדשים, כמו כן היצירות נשארות זהות כיוון שהווקטורים לא עוברים מוטציה (נציין כי גם במקרה הנגדי שבו המשתמש אינו בוחר אף יצירה, התהליך האבולוציוני לא מתרחש ולכן היצירות אינן משתנות). (שורה 2) במקרה שבו המשתמש בוחר את כל היצירות, חוץ מהיצירה האחרונה (ללא הגבלת הכלליות), נקבל 15 יצירות זהות, ללא פעולות שחלוף ורק היצירה האחרונה תהיה חדשה. (שורה 3) מקרה דומה יהיה בחירה של 14 יצירות ראשונות, כך שישנו מקום לשתי יצירות חדשות ללא פעולות שחלוף. (שורה 4) בנוסף נתבונן במקרה שבו המשתמש בוחר 13 יצירות ומטה, כך שישנו מקום גם לשתי יצירות חדשות לגמרי וגם ליצירה אחת לפחות הנוצרת מתוך תכונות היצירות הנבחרות, באמצעות פעולות שחלוף. (שורה 5) לבסוף נתבונן במקרה מעניין נוסף שבו המשתמש בוחר יצירה אחת בלבד, כאשר ניתן לראות כי חוץ משתי היצירות האחרונות שהן היצירות הנוצרות מדגימת וקטורים חדשים לגמרי, כל היצירות האחרות זהות ליצירה הנבחרת. הסיבה לכך היא שהיצירה הראשונה הינה היצירה הנבחרת, כאשר ישנו מקום ליצירת 13 יצירות חדשות הנוצרות מתוך היצירה הנבחרת באמצעות פעולות שחלוף, המבצעות בכל המקרים שחלוף של וקטור המייצג את היצירה הנבחרת עם עצמו (כי כאמור נבחרה יצירה אחת בלבד לביצוע שחלוף). (שורה 6) על מנת להבחין בהשפעת פעולת המוטציה, נבחר את כל היצירות להמשך האבולוציה, יחד עם סטיית תקן מקסימאלית של 1 ונראה כי למרות שכל היצירות ממשיכות לדור הבא באבולוציה, ללא פעולות שחלוף ודגימת וקטורים חדשים, חלק מהיצירות שהגרלת המטבע קבעה כי תבצע פעולת מוטציה בווקטורים המייצגים אותן, משתנות בתכונותיהן (צבע וצורה).

4.1. סיכום והרחבה עתידית של הפרויקט

בפרק 3 ראינו כי שילוב מודל CAN יחד עם גישת ה-DeepIE מאפשר שליטה מסוימת על מאפייני היצירות הסינתטיות, כאשר הגנרטור מפיק יצירות הנראות כאמיתיות ובסגנון מקורי ככל הניתן, בעוד שהמשתמש יכול לאחר מכן לבחור יצירות או מאפיינים האהובים עליו להמשיך התהליך האבולוציוני. מצד שני, שליטה זו מוגבלת שכן היצירות הסינתטיות מהגנרטור הינן אקראיות והתהליך האבולוציוני בגישת ה-DeepIE מבוסס על יצירות אקראיות אלו. כלומר למשתמש אין שליטה על מאפייני היצירות מתחילת התהליך, אלא רק לאחר הפקתן ע"י הגנרטור. בנוסף, יתכן כי היצירות שמתקבלות כתוצאה מהתהליך האבולוציוני לא תהיינה בהתאם להעדפתו של המשתמש ולכן יאלץ להתחיל את התהליך מחדש. במצב אחר יתכן כי המשתמש יצליח להפיק כתוצאה מבחירותיו בתהליך האבולוציוני את היצירות כראות עיניו, אך התהליך עלול להיות ארוך ומייגע ויגרום כאמור לתופעת עייפות המשתמש.

ניתן לומר כי מודל GAN, שהוצג לראשונה ב-[1] בשנת 2014, היה פורץ דרך כאשר תרם רבות להתפתחות מודלים בתחום הבינה המלאכותית היוצרת (Generative AI). פרויקט זה מבוסס בעיקר על [2] ו-[3] משנת 2017 ו-2018 בהתאמה, כאשר הרעיונות שהוצגו במודלים הללו נחשבו לחדשים בתחום של יצירת אומנות סינתטית בעזרת מודל GAN, שהיווה את הבסיס למאמרים הללו. עם זאת, ניתן להרחיב פרויקט זה עם חידושים נוספים מהשנים האחרונות כאשר חלה התקדמות משמעותית בתחום בינה מלאכותית יוצרת בכלל ובפרט בתחום האומנות. החידוש המרכזי הינו שליטה על הנתונים הסינתטיים אותם מייצר המודל באמצעות טקסט המוזן ע"י המשתמש. יישום פופולרי בשם ChatGPT בתחום ה-NLP, עיבוד שפה טבעית, של חברת OpenAI הושק בשנת 2022 ומאפשר ליצור מגוון רחב של טקסט, באמצעות טקסט המתאר מה ברצון המשתמש לקבל (תשובות לשאלות, מאמרים, קטעי קוד בשפות תכנות, תרגום לשפות ועוד) [11]. באופן דומה, ניתן להשתמש בטקסט על מנת להנחות מודלים מחוללי תמונות אודות המאפיינים הרצויים ביצירות הסינתטיות. למעשה הנחייה באמצעות טקסט היא הדרך הפופולרית כיום לחולל תמונות או יצירות סינתטיות באמצעות מודלים של למידה עמוקה. כך המשתמש יכול לתאר באמצעות כמה מילים את אשר הוא מעוניין לקבל ביצירה, למשל "שדה ירוק בשקיעה" וכך לקבל מספר אפשרויות לבחירה של יצירות המכילות את האובייקטים שבטקסט. בפרק 7 בעבודה המסכמת הנלוות לפרויקט הוצג מודל מסוג זה מבוסס GAN, מודל GauGAN2 משנת 2021 [6], המאפשר המרה של טקסט למפת סגמנטציה המתארת את הסצנה והאובייקטים שבטקסט, כאשר מפת הסגמנטציה מוזנת למודל GauGAN הבסיסי [4,5], שממיר את מפת הסגמנטציה ליצירת אומנות בפועל. למרות היכולות המרשימות של מודלים מבוססי GAN, במאמר של חוקרים מחברת OpenAI משנה 2021, נטען כי ניתן להשיג תוצאות טובות יותר בהפקת תמונות או יצירות סינתטיות באמצעות מודלים מסוג Diffusion [12]. מודל Diffusion מבוסס על רעיון של שימוש ברעש להפקת התמונות או היצירות הסינתטיות, בדומה ל-GAN, אם כי במקום גנרטור שמתאמן במיפוי וקטור הרעש לכדי היצירה, המודל מתאמן בשחזור הדרגתי של תמונת רעש לכדי יצירה לאחר הוספה הדרגתית של רעש ליצירה נתונה (כלומר מתאמן בהפיכה של תמונת רעש ליצירה). בשנת 2022 הוצג מודל Stable Diffusion בקוד פתוח המבוסס על הרעיון הנ"ל, כאשר מאפשר להמיר טקסט לכדי יצירה מתוך תמונת רעש ומודל CLIP המנחה את ביטול הרעש לכדי תיאור האלמנטים שבטקסט [13]. יישום פופולרי בשם Midjourney ככל הנראה משתמש ברעיון ה-Stable Diffusion עם אימון באמצעות מספר רב של תמונות, על מנת להמיר טקסט לכדי תמונות או יצירות אומנות סינתטיות המתארות את הטקסט [14]. בנוסף, באותה השנה הוצגו שני מודלים ליצירת תמונות או יצירות אומנות מתוך טקסט נתון, גם כן אינם מבוססי GAN: מודל DALL-E 2 של חברת OpenAI ומודל Imagen של Google. מודל DALL-E 2 מבוסס על מודל CLIP שאומן על מאות מילוני

תמונות המשויכות לטקסט המתאר אותן ומסוגל לנבא את מידת השיוך של הטקסט לתמונה (בדומה ל-Stable Diffusion). מודל DALL-E 2 מסוגל להמיר טקסט נתון לייצוג כווקטור, להמיר את ווקטור הטקסט לווקטור תמונה מתאים ולהפיק תמונה מתוך ווקטור ייצוג התמונה, כך שהטקסט ההתחלתי כקלט למודל מתקבל כתמונה, או יצירת אומנות, כפלט [8]. מודל Imagen מחולל את התמונות או יצירות האומנות באמצעות מקודד הטקסט של Google בשם T5, שמקבל טקסט כקלט ומקודד אותו לווקטור המייצג את הטקסט. הווקטור מועבר לרכיב אחר הממיר אותו לתמונה סינתטית [9].

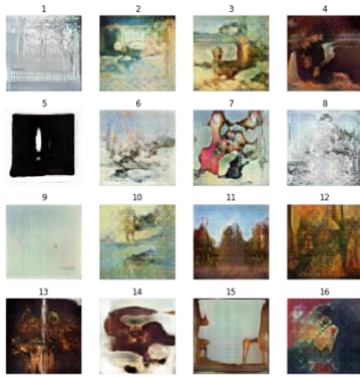


איור 13 : יצירות סינתטיות מגנרטור EMA של מודל CAN לאחר האימון

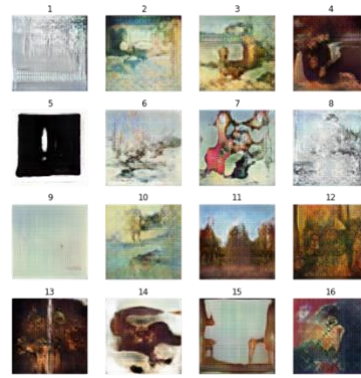


איור 14 : יצירות סינתטיות לדוגמה שנוצרו באמצעות מודל CAN שב- [2]

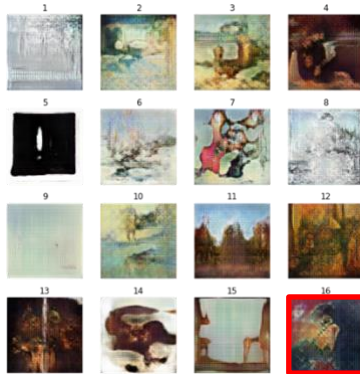
Before:



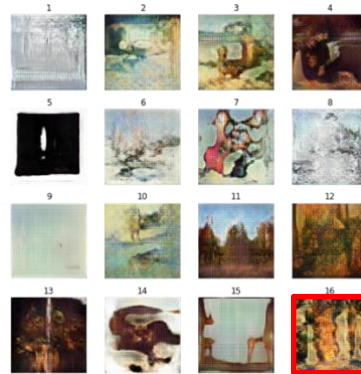
After:



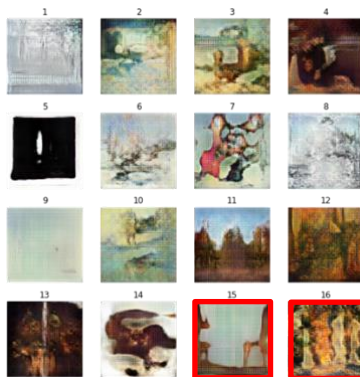
Before:



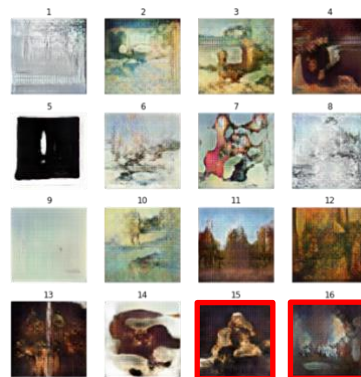
After:



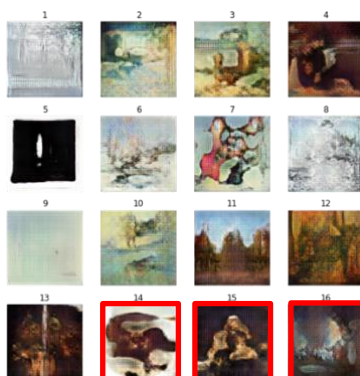
Before:



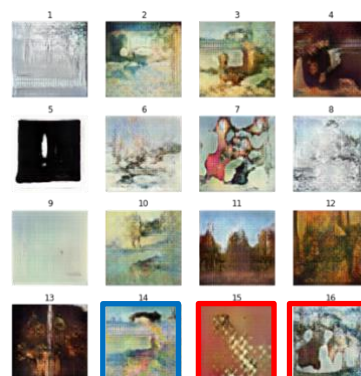
After:

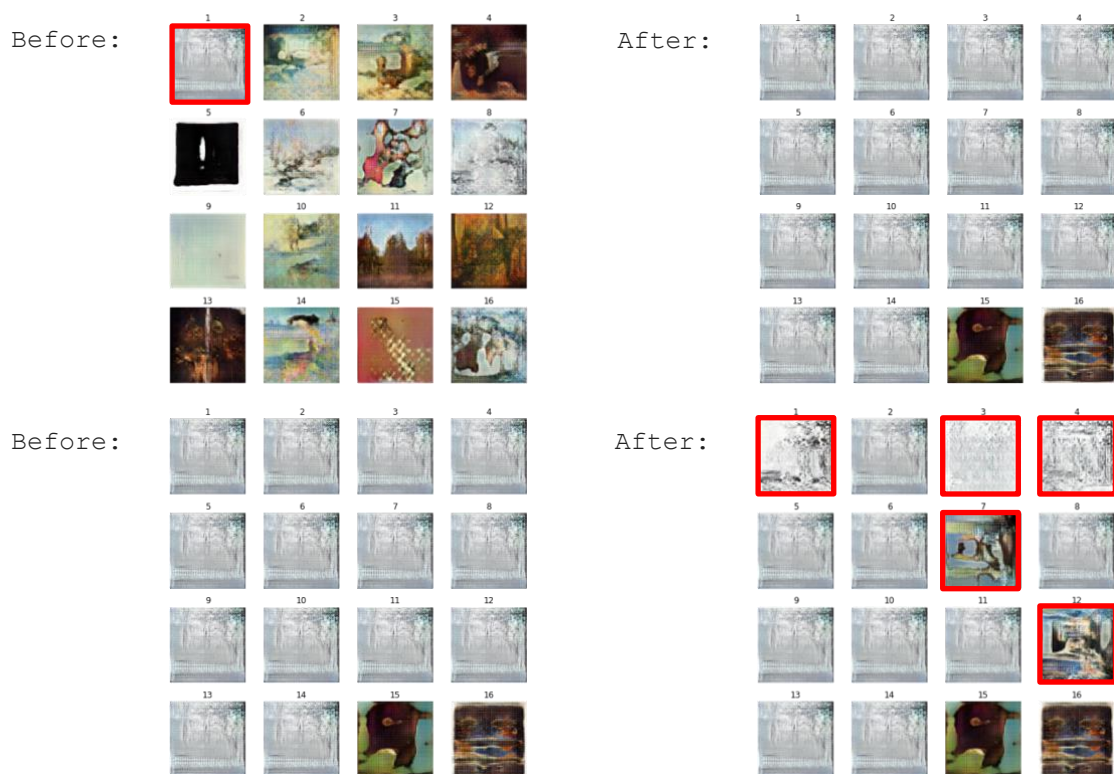


Before:



After:





איור 15 : דוגמאות למקרי קצה בתהליך האבולוציוני

שורה 1 (סטיית תקן 0) : לפני – המשתמש בוחר את כל היצירות (או לחילופין לא בוחר אף יצירה). אחרי – התהליך האבולוציוני אינו משנה דבר ביצירות.

שורה 2 (סטיית תקן 0) : לפני – המשתמש בוחר את כל היצירות, להוציא את היצירה האחרונה. אחרי – התהליך האבולוציוני אינו משנה דבר, פרט ליצירה החדשה האחרונה שנוצרת מווקטור חדש (באדום).

שורה 3 (סטיית תקן 0) : לפני – המשתמש בוחר את כל היצירות, להוציא את שתי היצירות האחרונות. אחרי – התהליך האבולוציוני אינו משנה דבר, פרט לשתי היצירות החדשות האחרונות הנוצרות משני וקטורים חדשים (באדום).

שורה 4 (סטיית תקן 0) : לפני – המשתמש בוחר את כל היצירות, להוציא את שלוש היצירות האחרונות. אחרי – התהליך האבולוציוני אינו משנה דבר, פרט לשלוש היצירות האחרונות, שתיים חדשות לגמרי (באדום) ואחת שנוצרת מפעולת השחלוף (בכחול).

שורה 5 (סטיית תקן 0) : לפני – המשתמש בוחר את היצירה הראשונה בלבד. אחרי – התהליך האבולוציוני יוצר 13 יצירות מפעולות שחלוף של יצירה אחת בלבד, כלומר זהות, יחד עם היצירה הנבחרת עצמה שלמעשה זהה ליצירות שנוצרו מהשחלוף ושתי יצירות חדשות לגמרי.

שורה 6 (סטיית תקן 1) : לפני – המשתמש בוחר את כל היצירות. אחרי – התהליך האבולוציוני משאיר את כל היצירות כפי שהן, כלומר ללא יצירות הנוצרות מפעולות שחלוף ויצירות חדשות מווקטורים חדשים, אך הפעם סטיית התקן לביצוע מוטציה הינה מקסימאלית כך שיצירות שהוגרלו לביצוע מוטציה משתנות בצורה משמעותית (באדום).

- [1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.
- [2] Elgammal, A., Liu, B., Elhoseiny, M. and Mazzone, M., 2017. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv: 1706.07068*.
- [3] Bontrager, P., Lin, W., Togelius, J. and Risi, S., 2018, April. Deep interactive evolution. In *International Conference on Computational Intelligence in Music, Sound, Art and Design* (pp. 267-282). Springer, Cham.
- [4] Park, T., Liu, M.Y., Wang, T.C. and Zhu, J.Y., 2019. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2337-2346).
- [5] Salian, I., 2019. Stroke of Genius: GauGAN Turns Doodles into Stunning, Photorealistic Landscapes. NVIDIA. <https://blogs.nvidia.com/blog/2019/03/18/gaugan-photorealistic-landscapes-nvidia-research>.
- [6] Salian, I., 2021. 'Paint Me a Picture': NVIDIA Research Shows GauGAN AI Art Demo Now Responds to Words. NVIDIA. <https://blogs.nvidia.com/blog/2021/11/22/gaugan2-ai-art-demo>.
- [7] Radford, A., Metz, L. and Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv: 1511.06434*.
- [8] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C. and Chen, M., 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv: 2204.06125*.
- [9] Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S.K.S., Ayan, B.K., Mahdavi, S.S., Lopes, R.G. and Salimans, T., 2022. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. *arXiv preprint arXiv: 2205.11487*.
- [10] Yaz, Y., Foo, C.S., Winkler, S., Yap, K.H., Piliouras, G. and Chandrasekhar, V., 2018, September. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*.
- [11] <https://en.wikipedia.org/wiki/ChatGPT>.
- [12] Dhariwal, P. and Nichol, A., 2021. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34, pp.8780-8794.
- [13] Rombach, R., Blattmann, A., Lorenz, D., Esser, P. and Ommer, B., 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 10684-10695).
- [14] <https://en.wikipedia.org/wiki/Midjourney>.

Abstract

GAN (Generative Adversarial Network) is a generative model consisting of two neural networks, a discriminator and a generator, competing against each other in a zero-sum game. The purpose of the competition is to produce new synthetic data, similar in characteristics to the data in the training set. Since the model was introduced in 2014, it has been used many times in fields such as: art, science, computer games and more. The goal of the project is to implement a GAN-based model called CAN (Creative Adversarial Network) used to create art and combine it with the DeepIE approach, to create user-directed art.

Table of Contents

Figures List	3
Tables List	3
Abstract	4
1. Introduction	5
2. Creative Adversarial Networks (CANs)	6
2.1. Implementation of CAN Model	7
2.1.1. Discriminator's Implementation	7
2.1.2. Generator's Implementation	8
2.2. Training CAN Model to Optimize the Cost Function	12
2.2.1. Training the Discriminator	12
2.2.2. Training the Generator	13
2.2.3. Training Algorithm	14
2.2.4. CAN Model Validation	17
2.2.5. EMA Generator Synthetic Art	20
3. Deep Interactive Evolution	21
3.1. Implementation of DeepIE Approach	21
3.1.1. Uniform Crossover Function Implementation	21
3.1.2. Mutation Function Implementation	22
3.1.3. Evolution Function Implementation	22
4. Analyzing CAN Model Results and DeepIE Integration	27
4.1. Summary and Project Possible Expansion	28
References	32

The Open University of Israel
Department of Mathematics and Computer Science

Implementation of Art Creating CAN Model with DeepIE Approach Integration

Advanced project submitted as partial fulfillment of the requirements
towards an M.Sc. degree in Computer Science
The Open University of Israel
Department of Mathematics and Computer Science

By
Rotem Yagil

Prepared under the supervision of Dr. Mireille Avigal

April 2023