Evolutionary Algorithm for Image Drawer

By: Nadav Lubman and Lior Haritonov

Image Drawer

תוכן עניינים:

- תיאור הבעיה----------3----
- מאפייני תמונה דיגיטלית---------4-6---
- השימוש באלגוריתם אבולוציוני-7-11--
- תוצאות----------12-15-
- ביבליוגרפיה----------16---

Image Drawer

תיאור הבעיה:

הבעיה:

יצירת תמונה רנדומלית וביצוע מניפולציות עלייה כדי שתהיה דומה לתמונת המקור בעזרת אלגוריתם אבולוציוני.

הנתונים שלרשותנו:

תמונת המקור: תמונה כלשהי שתשמש אותנו כיעד ליצירת התמונה החדשה.

אוכלוסיית תמונות ראשונית: אוסף של תמונות בגודל קבוע שנקבעו בצורה רנדומלית.

כל תמונה בגודל זהה לתמונת המקור הבנויה מאותה כמות פיקסלים של תמונת המקור אך צבעו של כל פיקסל בתמונה זו נבחר רנדומלית.

המטרה:

לבצע מניפולציות על אוכלוסיית התמונות הראשונית באמצעות אלגוריתם אבולוציוני על-מנת להגיע לתמונה דומה ככל הניתן לתמונת המקור.

Image Drawer

מאפייני התמונה:

מאפייני תמונה:

כל תמונה ממוחשבת מיוצגת על ידי מטריצה דו-ממדית של פיקסלים.

פיקסל הינו חידת מידע גרפית בסיסית במחשב.

כל תמונה בנויה ממאות אלפי עד עשרות מיליוני פיקסלים.

ככל שתמונה תכיל יותר פיקסלים, כלומר צפיפות הפיקסלים תהיה יותר גבוהה(מושג זה נקרא רזולוציית תמונה) כך התמונה תכיל יותר פרטים ותהיה חדה, טובה וקריאה יותר.

לדוגמה תמונה ברזולוציה 800 X 600 תכיל 480,000 פיקסלים.

מאפייני פיקסל:

כל פיקסל בתמונה מייצג צבע כלשהו.

קיימים מספר קידודים לייצוג צבע במחשב, אחד הנפוצים ביותר הינו קידוד RGB – שמייצג בראשי התיבות Red, Green, Blue.

הצבע על הצג נקבע על ידי שלוש העוצמות האלה.

כל פיקסל מיוצג על-ידי שלוש קואורדינטות כאשר כל קואורדינטה מייצגת את עוצמת הצבע – אדום, ירוק וכחול.

קביעת העוצמות של מרכיבי הצבע השונים נעשית על ידי השמת מספרים שלמים בין 0 ל 255 לכל אחד מן המשתנים.

Image Drawer

0 פירושו לא להאיר בכלל את הפיקסל, ו-255 מסמל הארה של הפיקסל במלוא העוצמה.

לדוגמא:

עבור הערך הבא של קידוד ה-RGB - (FF00FF):

שני השדות השמאליים ביותר FF – מייצגים את הצבע האדום בעוצמה מלאה בבסיס הקסדצימלי.

שני השדות הימניים ביותר FF – מייצגים את הצבע הכחול בעוצמה מלאה בבסיס הקסדצימלי.

ושני השדות האמצעיים 00 מייצגים את הצבע הירוק בעוצמה מינימלית בבסיס הקסדצימלי, כלומר הצבע הירוק אינו מאיר את הפיקסל כלל.

וכך צבע הפיקסל המתקבל על-ידי קידוד ה RGB (FF00FF) הינו ורוד-פוקסיה.

טווח הערכים הניתנים להצגה על-ידי קידוד הRGB הינו:

256 x 256 x 256 = 16777216, כלומר ניתן לייצג מעל ל16 מיליון גוונים שונים בעזרת קידוד זה.

דוגמה להצגת הצבעים הנפוצים ביותר:

Image Drawer

ייצוגם של צבעים נפוצים בקידוד RGB:

דוגמה	שם	שלשה הקסדצימלית	אדום	ירוק	כחול
	לבן (White)	FFFFFF	100%	100%	100%
	אדום (Red)	FF0000	100%	0%	0%
	צהוב (Yellow)	FFFF00	100%	100%	0%
	ורוד (פוקסיה) (Fuchsia)	FF00FF	100%	0%	100%
	כסף (Silver)	C0C0C0	75%	75%	75%
	אפור (Gray)	808080	50%	50%	50%
	בורדו (Maroon)	800000	50%	0%	0%
	זית (Olive)	808000	50%	50%	0%
	סגול (Purple)	800080	50%	0%	50%
	שחור (Black)	000000	0%	0%	0%
	ליים (Lime)	00FF00	0%	100%	0%
	ירוק (Green)	008000	0%	50%	0%
	אקווה/מים (Aqua)	00FFFF	0%	100%	100%
	ירוק-כחול (Teal)	008080	0%	50%	50%
	כחול (Blue)	0000FF	0%	0%	100%
	כחול כהה (Navy)	000080	0%	0%	50%

Image Drawer

השימוש באלגוריתם אבולוציוני:

Individual:

הצגנו כל אינדיבידואל בתור אובייקט הנקרא ImageIndividual שזהו אובייקט המכיל שני שדות:

- מערך דו מימדי של פיקסלים – image_array
- מרחק – dist (נרחיב עליו בהמשך)

fitness:

אנחנו משתמשים באובייקט הנקרא ImageEvaluator שתפקידו הוא להחזיר את השדה dist של אובייקט ה-ImageIndividual .

לפני שנרחיב על עוד על ה-fitness של אובייקט ה-ImageIndividual עליינו להסביר על חישוב מרחק של כל פיקסל לפיקסל אחר.

- מבנה הפיקסל הינו tuple של 3 איברים (R, G, B) כאשר R מייצג צבע אדום, G מייצג צבע ירוק וB מייצג צבע כחול.
  תווך הערכים של כל אחד מהם נע בין 0 ל-255 ומתאר את עוצמת ההארה של כל צבע כפי שתיארנו קודם לכן.

Image Drawer

- כאשר אנחנו יוצרים פיקסל חדש במיקום (i, j) אנחנו מחשבים את המרחק בין הפיקסל לבין הפיקסל במיקום ה-(i, j) בתמונת המקור.
- המרחק מחושב באמצעות הנוסחה הבאה:

  עבור שני פיקסלים בעלי ערכי ה RGB הבאים:

  $(r_1, g_1, b_1), (r_2, g_2, b_2)$ המרחק ביניהם יהיה:

$$dist = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2}$$

כאשר אנחנו יוצרים את המערך הדו-מימדי הזה עבור התמונה לכל פיקסל בתמונה יהיה שדה המייצג את מרחקו כפי שתיארנו מהפיקסל התואם לו(זה שממוקם באותם הקואורדינטות) בתמונת המקור.

כעת נגדיר את הdist של כל individual בתור המרחק הממוצע של כל הפיקסלים שלו.

כך שככל שהממוצע נמוך יותר כך הפיקסלים של התמונה דומים יותר לפיקסלים של תמונת המקור וכך מתקבלת תמונה שהיא דומה יותר לתמונת המקור, כלומר Higher is better = False.

Image Drawer

לצורכי אופטימיזציה אנו מחשבים את ה- dist של כל תמונה בעת היצירה שלה, כלומר בעת הרכבת הפיקסלים, מה שחוסך המון זמן חישוב משום שהפיקסלים אינם משתנים וה-dist שלהם כבר קיים. ולכן ה-Evaluator רק מחזיר את ה-dist ולא מחשב אותו, וכך חוסך שמן חישוב יקר.

<u>תהליך האבולוציה:</u>

אנחנו נרצה לבחור מכל האוכלוסייה הנוכחית את הפרטים אשר ירכיבו את הדור הבא (מלבד האוכלוסייה הראשונה שנוצרת באופן רנדומאלי).

נעשה זאת בעזרת ביצוע טורניר, בעצם ניקח x אינדיבידואלים מהאוכלוסייה ומתוכם נבחר את yהפרטים הטובים ביותר, כלומר עם ערך ה-dist הנמוך ביותר.

לאחר שקיבלנו את תת-האוכלוסייה שתרכיב את הדור הבא, נבחר מתוכה בצורה איטרטיבית בכל איטרציה שני פרטים בצורה רנדומאלית וניצור בעזרת שניהם אינדיבידואל חדש(נרחיב על כך בהמשך).

וכך נמשיך בתהליך זהה עד לאשר נקבל אוכלוסייה חדשה(דור חדש) בגודל זהה לגודל האוכלוסייה של הדור הקודם.

Image Drawer

וכך נמשיך בתהליך זהה עם האוכלוסייה החדשה ונמשיך ליצור דורות חדשים עד אשר נקבל פרט שערך ה-dist שלו מספיק נמוך עבורנו.

תנאי סיום – terminal checker:

נבחר תוצאה אופטימלית(0 – משמע שהתמונה זהה) וthreshold שיגדיר מה המרחק שאנו מסוגלים לספוג מהתוצאה האופטימלית.

<u>אופן יצירת פרט חדש באוכלוסייה:</u>

בהינתן של אינדיבידואלים הורים, עליינו לייצר מהם בן ואנו עושים זאת באופן הבא:

1. נאתחל מטריצה דו ממדית ריקה שתייצג את ערכי הפיקסלים של הבן.
2. נרוץ על המטריצה בצורה איטרטיבית ולכל תא (i, j) ניתן פיקסל מאחד ההורים באותו מיקום, כלומר נטיל מטבע וניקח את הפיקסל המתאים בהסתברות של 50% מכל הורה.

   יש לציין שהפיקסל כבר קיים וה-dist שלו ידוע ולכן אין לחשב אותו מחדש ולכן יותר קל לנו לחשב את ה-dist של הבן כבר בשלב זה.

Image Drawer

<u>מימוש הטורניר:</u>

תחילה השתמשנו באלגוריתם הטורניר הדיפולטיבי של EcKity, TournamentImageSelection.

ונתקלנו בבעיה לאחר ניסיונות רבים של שינוי ההיפר-פרמטרים(גודל האוכלוסייה, tournament size, מספר המנצחים בטורניר) בה הגענו לערך של dist קבוע עבור המקרה הטוב ביותר, הממוצע והגרוע ביותר.

כלומר, בשלב מסוים כל האוכלוסייה הייתה אותו הפרט מה שמנע מאיתנו להגיע לתוצאה מספקת.

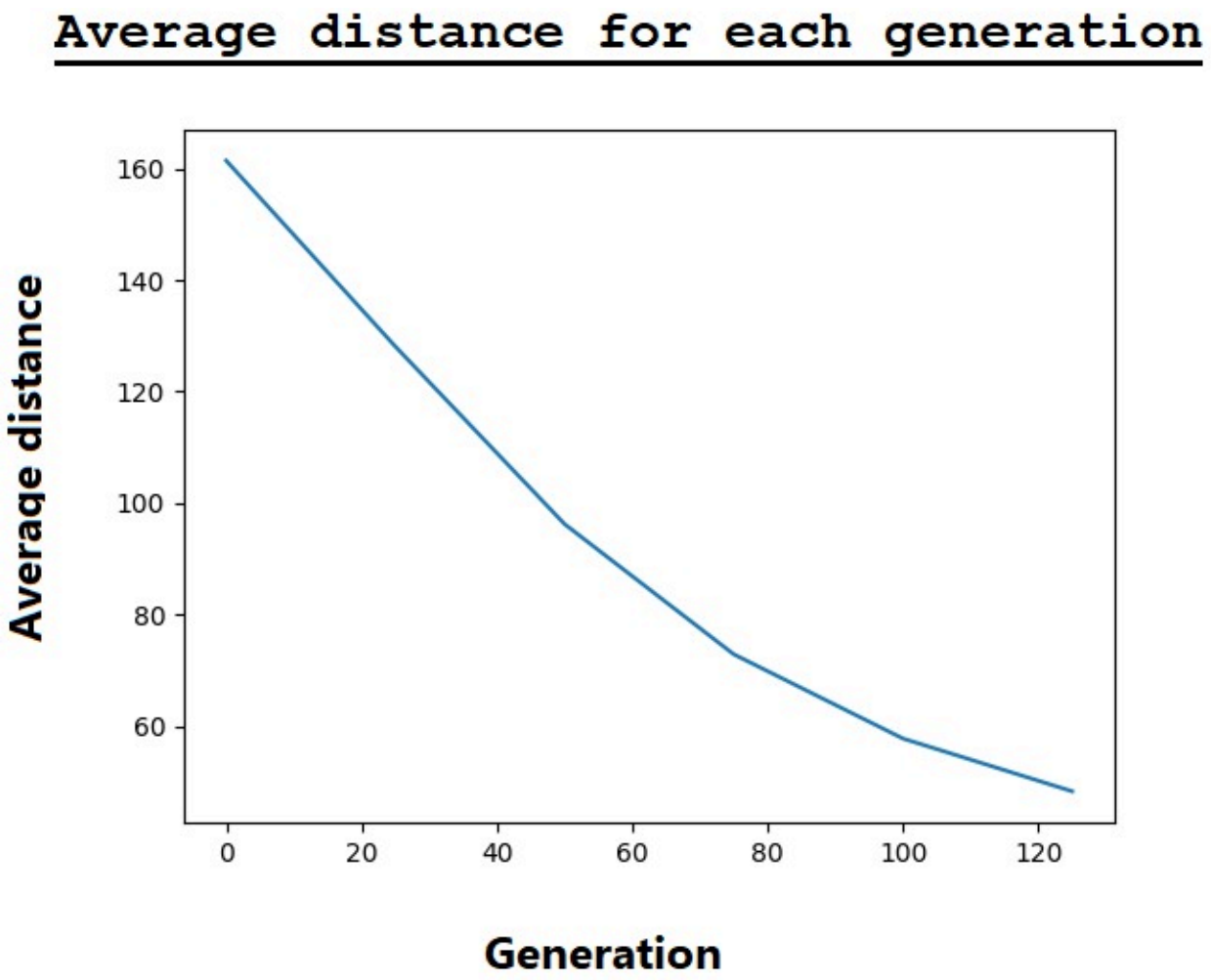
הבעיה בטורניר היא שפרטים עם fitness טוב מקבלים משקל גבוהה מכיוון שהם מופיעים מספר פעמים בתור התת-אוכלוסייה שתרכיב את הדור הבא.

שינינו זאת בכך שיצרנו טורניר חדש מסוג אחר שהוא זהה לטורניר הקודם מלבד ההגבלה שכל פרט יכול להופיע מספר מסוים של פעמים.

מה שאנחנו מקבלים מכך הוא שפרט שהוא יותר טוב מפרטים אחרים יופיע יותר פעמים בתת-אוכלוסייה שתרכיב את הדור הבא אך לא ישתלט עלייה מכיוון שישנה הגבלה על מספר הפעמים שהוא יכול להופיע בה.

Image Drawer

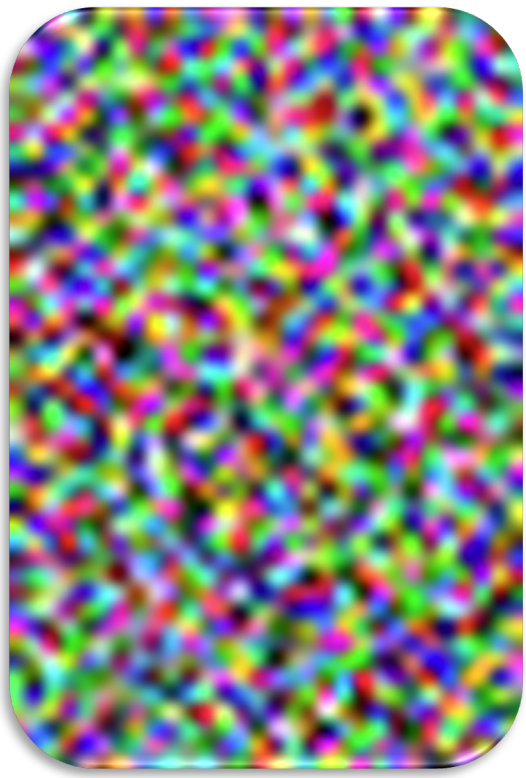
תוצאות:



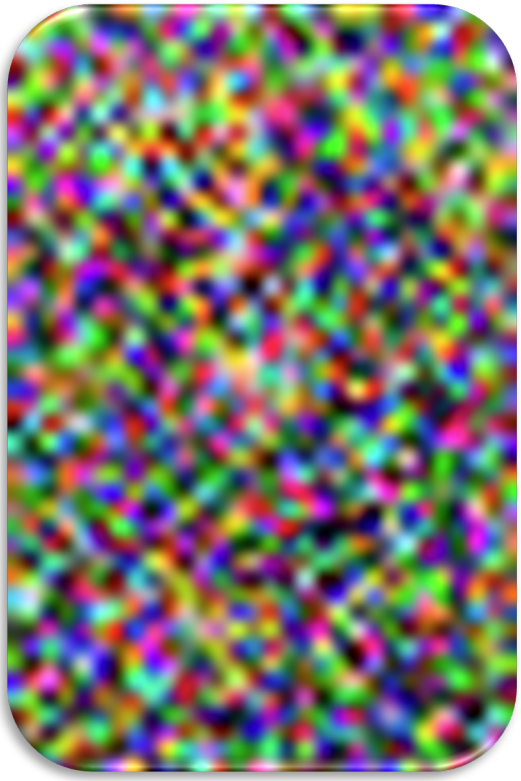
ניתן לראות כי האלגוריתם אכן משפר את רמת הfitness ככל שהדורות מתקדמים וניתן לראות כי ככל שעוברים יותר דורות שיפוע העקומה מתמתן.

כעת, נציג מדגם של תוצאות התמונה במהלך הדורות באינטרוולים של כ-25 דורות:

Image Drawer

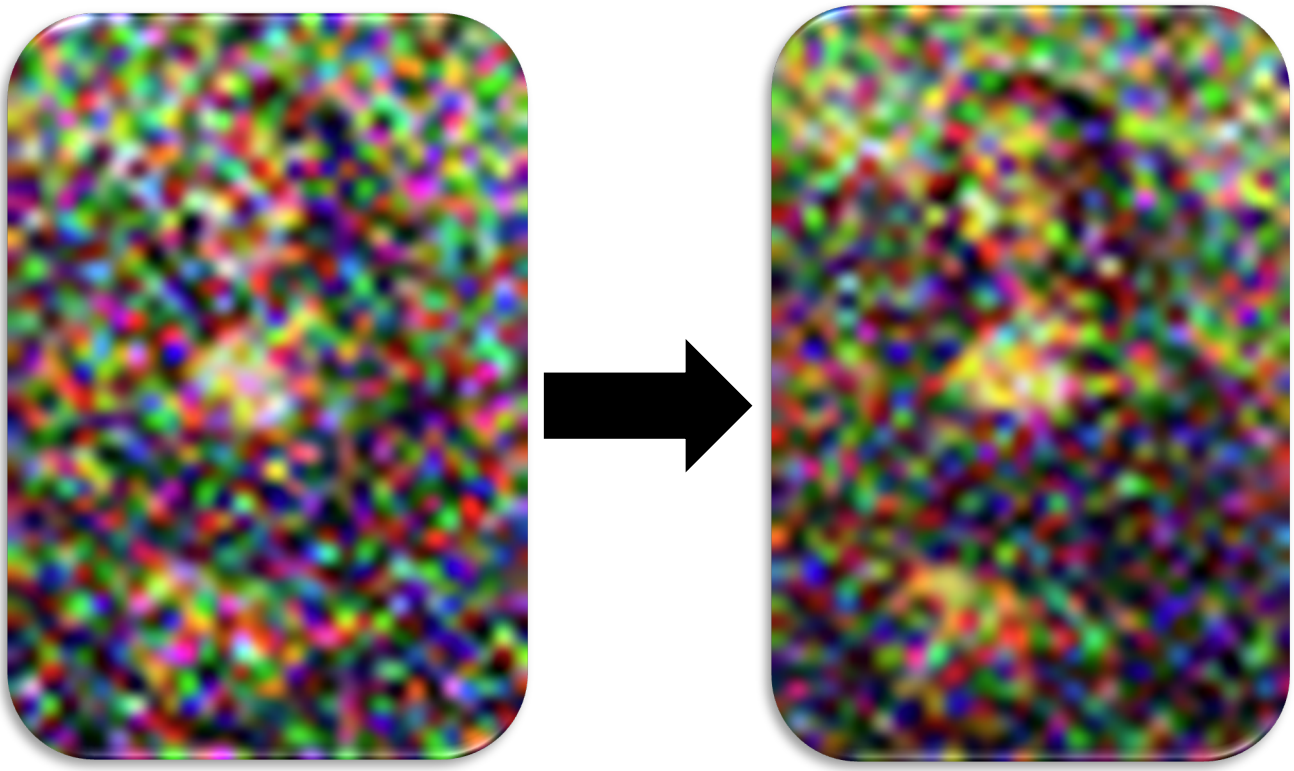


Generation: 0
Distance: 161.425



Generation: 25
Distance: 128.065

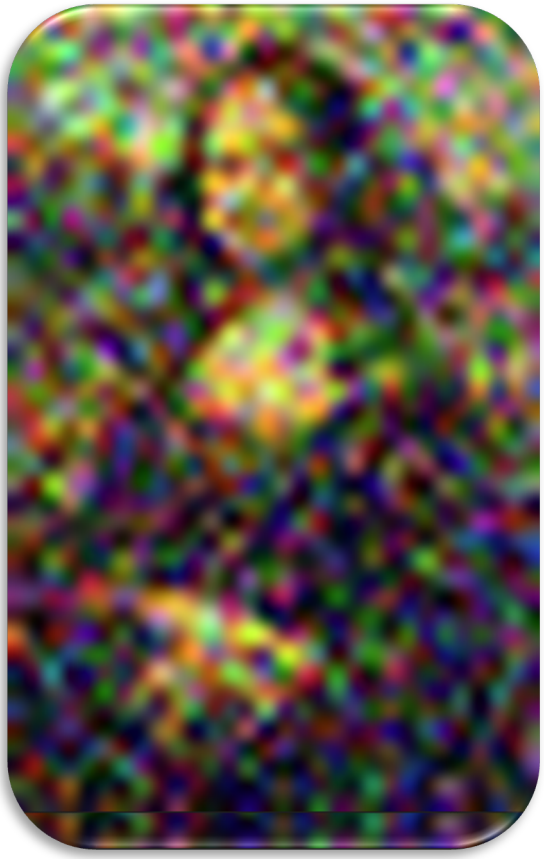
Image Drawer



Generation: 50
Distance: 96.212

Generation: 75
Distance: 72.850

Image Drawer



Generation: 100
Distance: 57.735

Generation: 125
Distance: 48.273

Image Drawer

ביבליוגרפיה:

- https://en.wikipedia.org/wiki/Digital_image
- https://en.wikipedia.org/wiki/RGB_color_model
- https://en.wikipedia.org/wiki/Evolutionary_algorithm
- https://github.com/EC-KitY/EC-KitY