(67103) רשתות נוירונים לתמונות | תרגיל 3

שם: רונאל חרדים | ת"ז:208917641

חלק I

פרקטי:

ניסיתי להתקין את $CalebA$ אך לצערי לא הצלחתי, לכן עבדתי במהלך התרגיל עם $MNIST$.

שאלה 1

הגדרתי את הרשת באופן הבא:

```python
# Define the Generator network
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()

        self.convTranspose1 = nn.ConvTranspose2d(100, 64 * 4, 3, 2, 0,
                                                 bias=False).to(device)
        self.batchNorm1 = nn.BatchNorm2d(64 * 4).to(device)
        self.convTranspose2 = nn.ConvTranspose2d(64 * 4, 64 * 2, 3, 2, 0,
                                                 bias=False).to(device)
        self.batchNorm2 = nn.BatchNorm2d(64 * 2).to(device)
        self.convTranspose3 = nn.ConvTranspose2d(64 * 2, 64, 3, 2, 0,
                                                 bias=False).to(device)
        self.batchNorm3 = nn.BatchNorm2d(64).to(device)
        self.convTranspose4 = nn.ConvTranspose2d(64, 1, 3, 2, 2, 1, bias=False).to(device)
        self.relu = nn.ReLU().to(device)
        self.sigmoid = nn.Sigmoid().to(device)
```

```python
# Define the Discriminator network
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()

        self.conv1 = nn.Conv2d(1, 28, 4, 2, 1, bias=False).to(device)
        self.conv2 = nn.Conv2d(28, 28 * 2, 4, 2, 1, bias=False).to(device)
        self.batchNorm2 = nn.BatchNorm2d(28 * 2).to(device)
        self.conv3 = nn.Conv2d(28 * 2, 28 * 4, 4, 2, 1, bias=False).to(device)
        self.batchNorm3 = nn.BatchNorm2d(28 * 4).to(device)
        self.conv4 = nn.Conv2d(28 * 4, 1, 4, 2, 1, bias=False).to(device)
        self.leaky_relu = nn.LeakyReLU(0.2, inplace=True).to(device)
        self.sigmoid = nn.Sigmoid().to(device)
```

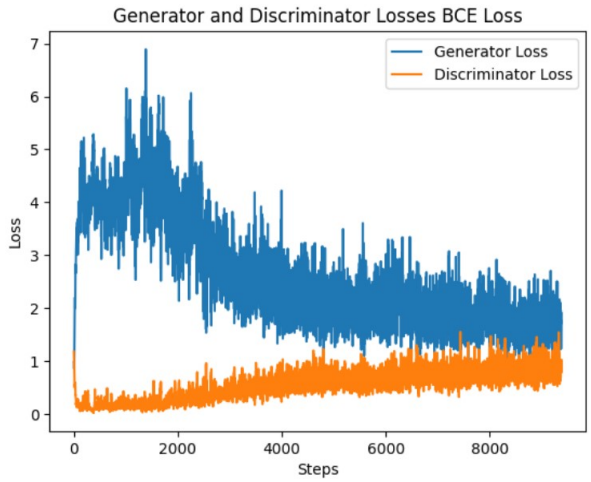
הפרמטרים שהגדרתי לרשת הם:

```python
latent_dim_ = 100
batch_size_ = 128
num_epochs_ = 20
lr = 0.0002
```

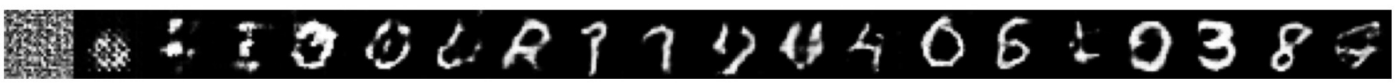
הגרפים שקחבלתי עבור ה $Loss$־ים השונים:

(א)

הגרף עם $BCE\ Loss$:

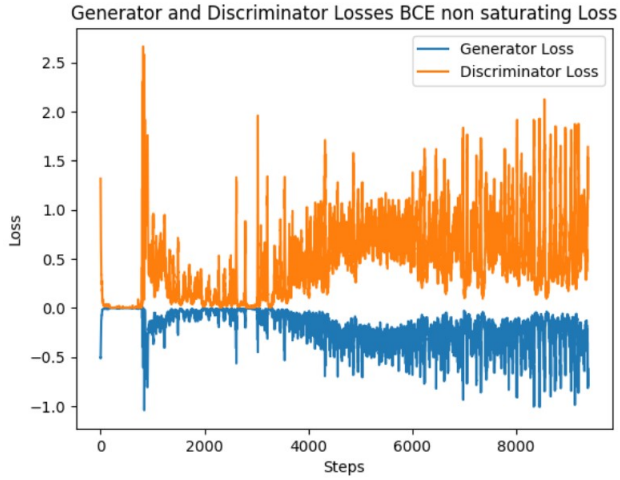


התמונות שהגנרטור ייצר בהינתן latent vector כתלות ב $epoch$:

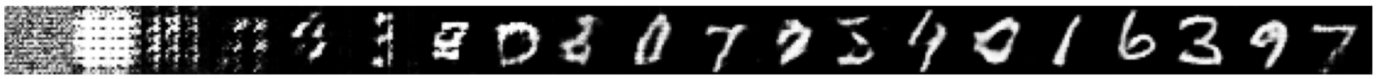


(ב)

הגרף עבור BCE-non saturating Loss:

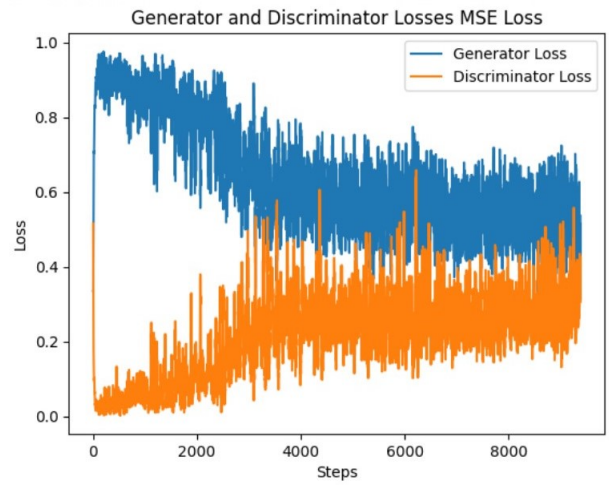


התמונות שהגנרטור ייצר בהינתן latent vector כתלות ב $epoch$:

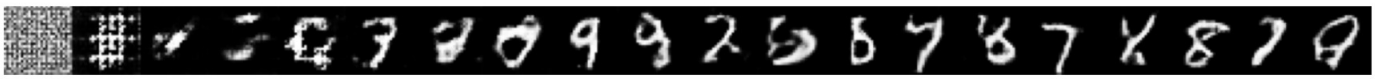


(ג)

הגרף עבור $MSE\ Loss$**:**



התמונות שהגנרטור ייצר בהינתן latent vector כתלות ב $epoch$**:**



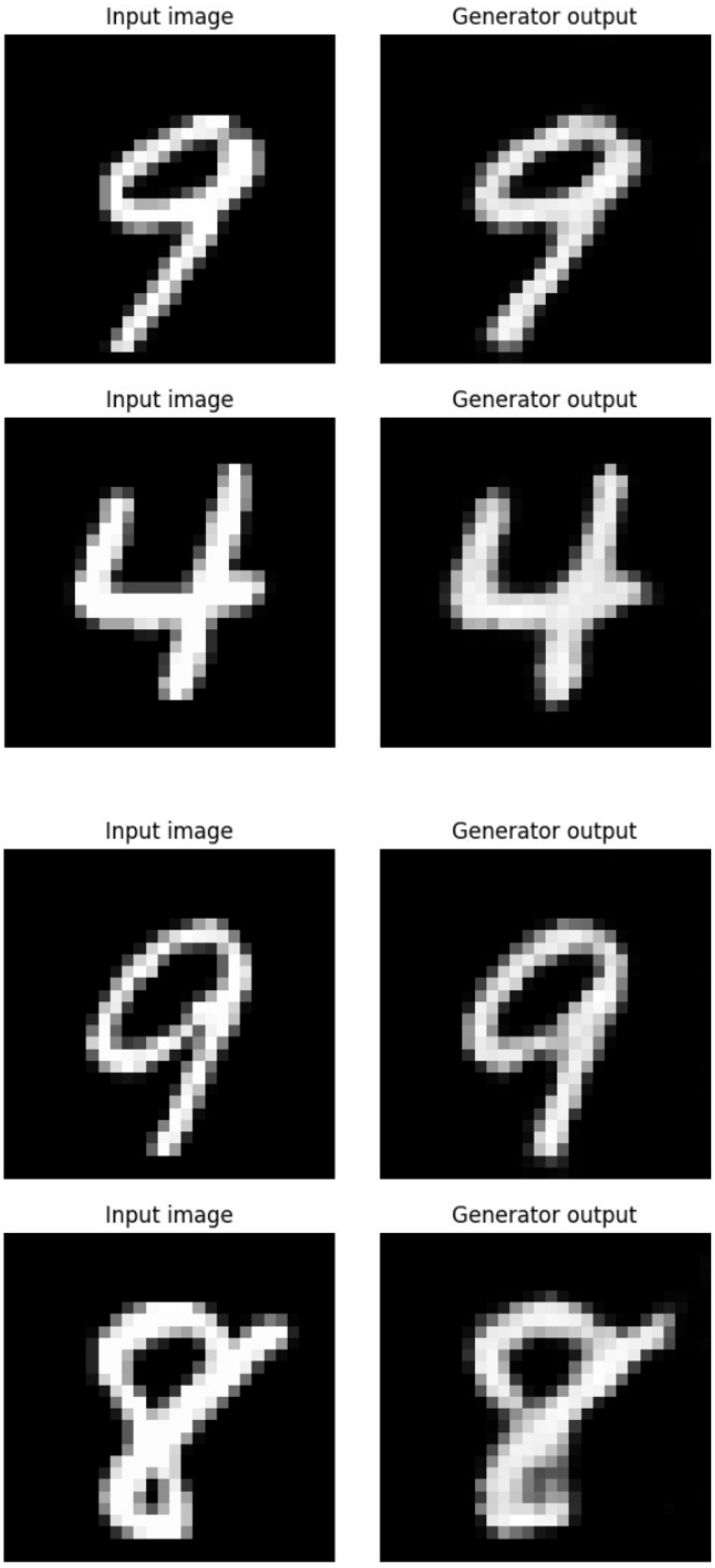
הסבר:

כפי שניתן לראות ה־ $non-saturating$ ו ה־ $least-squares$ עם התוצאות הכי טובות, אך ה BCE הרגיל פחות יציב ולא כל כך משתפר (ניתן לראות כי ה $Loss$ גבוה), וזה בגלל בעיית הסיטורציה.

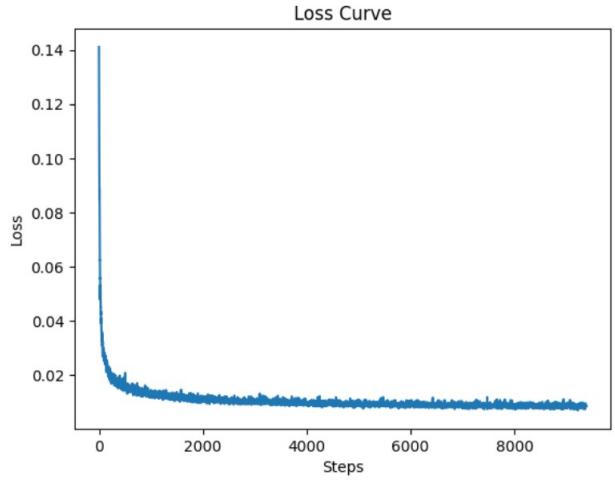
שאלה 2

בשאלה השתמשתי ברשת $encoder$ שתמפה את התמונות ל $latent\ vector$ עם מימד $d = 100$, אימנתי אותה במקביל לרשת ה GAN שכבר אומנה. ניביתי לעשות אופטימיזציה על z לבד, ללא רשת $encoder$ ולא הצלחתי. התוצאות שהתקבלו עם z שנבחר עם רשת $encoder$ היו טובות מאד.

להלן חלק מהתוצאו שקיבלתי:


Input image
Generator output
Input image
Generator output
Input image
Generator output
Input image
Generator output

גרף ה $Loss$ שהתקבל הוא:



הסבר:

אנחנו רואים כי המודל הצליח לשחזר את התמונות בצורה טובה מאד, למעט פיקסלים בודדים בצדדים. הסיבה לכך היא כי הרשת הצליחה ללמוד טוב מאד את המיפוי של התמונה ל $latent\ vector$ בדומה לרשתות $encoder - decoder$. את הפרטים הקטני והפחות חשובים היא לא הצליחה ללמוד, כי רמת הדיוק של הרשת והפרטים שנשמרים בווקטור לא גבוהה עד כדי כך.

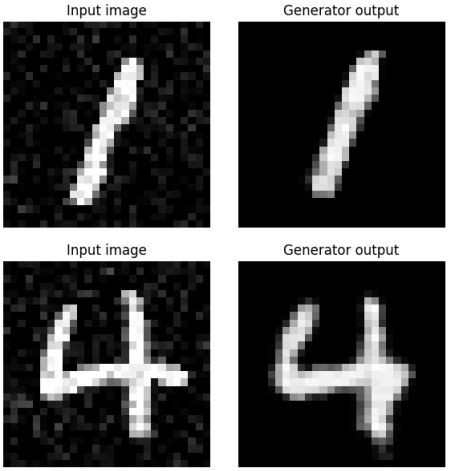
שאלה 3

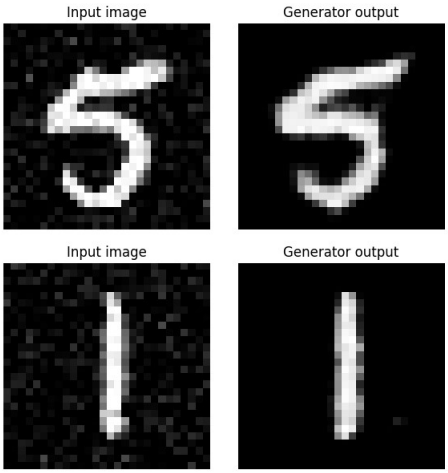
השתמשתי באותו סגנון רשת כמו בשאלה 2, אימנתי את רשת ה GAN וביצעתי אופטימיזציה על ה latent vector.

(א)

השתמשתי בנורמת $L2$ עם $MSELoss$.

עבור תמונות עם רעש, קיבלתי את התוצאות הבאות:



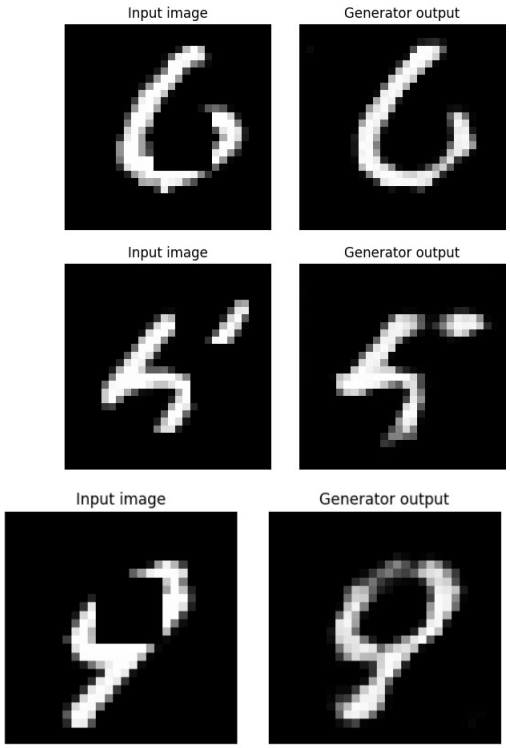


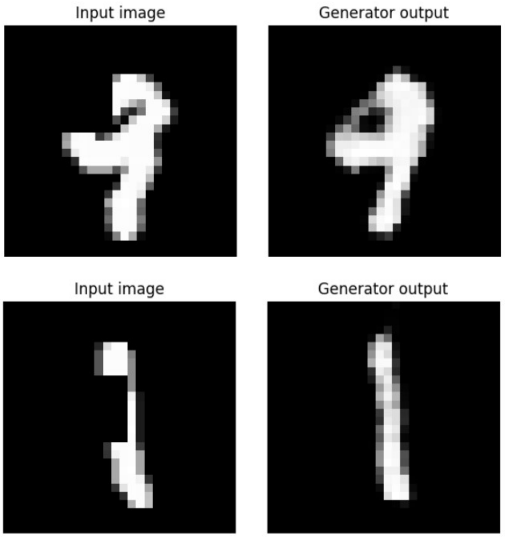
ניתן לראות כי הרעש נוקה מהתמונות באופן יעיל למדיי.

(ב)

במשימה זו השתמשתי בנורמת $L1$, הרשת הצליחה לשחזר את התמונות אך לא את כולן. ההצלחה של הרשת הייתה חלקית והביצועים לא היו טובים כמו בחלק של הרעש.

חלק מהתוצאות שקיבלתי:





חלק II

תיאורתי:

שאלה 1

נראה כיצד ניתן לחשב את הפונקציה $p(x)$ ב $GLOW$.
תהי רשת נוירונים $M = h_k \circ \ldots \circ h_1$, והתפלגות z.
נניח כי $x = M_\theta(z)$ אזי:

$$P(x) = \left|\frac{\nabla M^{-1}(x)}{\nabla x}\right| P_z\left(M^{-1}(x)\right)$$

בנוסף:

$$\log P_\theta(x) = \log P(z) + \sum_j \left|det\left(\frac{dh_{j-1}}{dh_j}\right)\right|$$

באופן כזה נוכל לדגום וקטור מההתפלגות zולחשב את הדטרמיננטה שלו, המטריצה היא מטריצה משולשית וכמו שראינו בהרצאה, ניתן לחשב פעולה זו ביעילות על מטריצות משולשיות. כך נוכל לחשב בקלות את $p(x)$.

הסבר מדוע לא ניתן לבצע זאת עם GAN או GLO:
הדרך היחידה לחשב את $p(x)$ היא לעבור על כל הוקטורים שנמצאים ב $letent\ space$ ולהסיק מי מהם נשלח לתמונה x.
סיבוכיות החישוב גדולה מאד ולכן זה לא מעשי למצוא את $p(x)$ באופן שכזה.

שאלה 2

לפי הנוסחה של רשת GAN מתקיים:

$$\min_G \max_D V(D,G) = \min_G \max_D E_{p_{\text{data}}}\ [\log(D(x)] + E_{p_z}[\log(1 - D(G(z))]$$

אם D כבר התכנס, אזי:

$$\frac{\partial V}{\partial G} = \frac{1}{m}\sum_{i=1}^{m} \frac{\log e * \left(-\frac{\partial D(G(z))}{\partial G(z)}\right)}{1 - D(G(z))}$$

וגם $D\left(G\left(z\right)\right) \Rightarrow 0$, ולכן נקבל כי:

$$\frac{\partial V}{\partial G} = \frac{1}{m}\sum_{i=1}^{m} \frac{\log e * \left(-\frac{\partial D(G(z))}{\partial G(z)}\right)}{1 - D(G(z))}$$

שאלה 3

(א)

נתון לנו דיסקרימנטור D, ואנחנו מחפשים גנרטור G שימזער את $V(D,G) \Rightarrow 0$, כלומר יצליח "לעבוד" על D כדי שיחשוב שהתמונות ש G מייצר הן אמיתיות.

(ב)

בהנחה שהבעיה הפנימית נפתרה, כלור מצאנו גנרטור G כמו שרצינו. כעת, יש לנו בעיה נוספת - אנחנו רוצים למצוא דיסקרימנטור D שיצליח "לעלות" על G. כלומר, נרצה למקסם את היכולות של D לזהות איזה תמונה היא המזוייפת. זוהי בעיית קלאסיפיקציה, ולכן גם רשת עם יכולות מוגבלות תצליח לפתור אותה.

(ג)

הבעיה היא - אם הדיסקרימנטור D שמצאנו הוא גרוע, כלומר הוא לא מזהה טוב איזו תמונה היא המזוייפת. הגנרטור G לא ישאף להשתפר, משום שגם עם תמונות גרועות הוא מצליח "לעבוד" על D. לכן גם G לא יהיה טוב, ולא ייצר לנו תמונות טובות.