Neural Network For Images

22 באפריל 2023

$Ex2$
איתמר שכטר
315092759
$itamar.sh$

שאלות פרקטיות:

שאלה פרקטית **ראשונה** $Auto\ Encoding$:

בחרתי להשתמש בארכיטקטורה של 4 שכבות קונבולוציה עם קרנל בגודל שלוש בלי $padding$, בשכבה השנייה והרביעית הוספתי $stride = 2$ בשביל ליצור אפקט של $down\ sampeling$.
ה$decoder$ בהתאם עם $output\ padding = 1$ בשכבה השנייה והרביעית כדי שההגדלה תיתבצע למספרים זוגיים.

בסוף ה$encoder$ ותחילת ה$decoder$ הייתה שכבת fc מתאימה ביחס לגודל הקלט שהוא הפלט/קלט של שכבת הקונבולוציה/דיקונבלוציה ומימד ה$latent\ vector$.

בצורה כזאת הקלט מתחיל בגודל של 28 על 28 לפי הדאטא ובסוף ה$encoder$ המימדים הם 4 על 4 על כמות הערוצים של השכבה האחרונה.

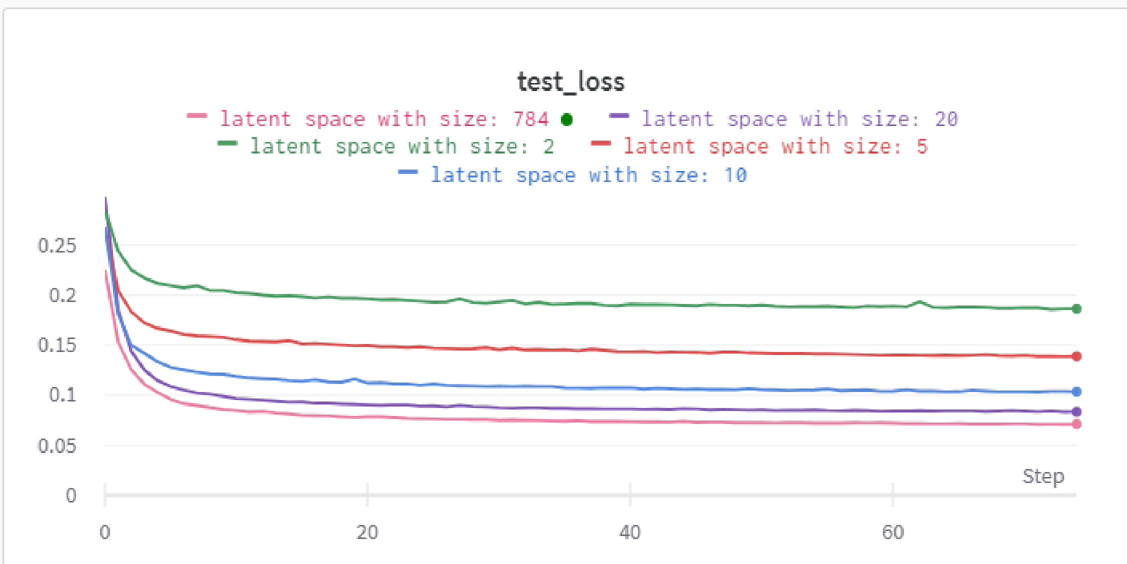
בנוסף השתמשתי ב$sigmoid$ בסוף ה$decoder$ ובפונקציית לוס BCE עם אופטימייזר $Adam$, בין היתר בשביל הרגולריזציה שהוא נותן.
כמות האפוקים בשלב הזה הייתה 3.

בחלק הראשון ־ בו רצינו לקבע את המשקולות ולשנות את ה$latent\ dimension$ בחרתי בכמות הערוצים הבאה:

בשכבת הקונבלוציה הראשונה (די קונבלוציה אחרונה) בחרתי 16 ערוצים.

ולפי הסדר בחרתי ב3 השכבות הבאות 32 64 128. ככה שנשמר הרעיון של הגדלת כמות הערוצים לקראת סיום הקונבולוציה. כמובן שהדיקונבלוציה בהתאם.

ערכתי בדיקה עם$latent\ dimension$ בגדלים 2 5 10 20 ו784.
אלה התוצאות שקיבלתי:

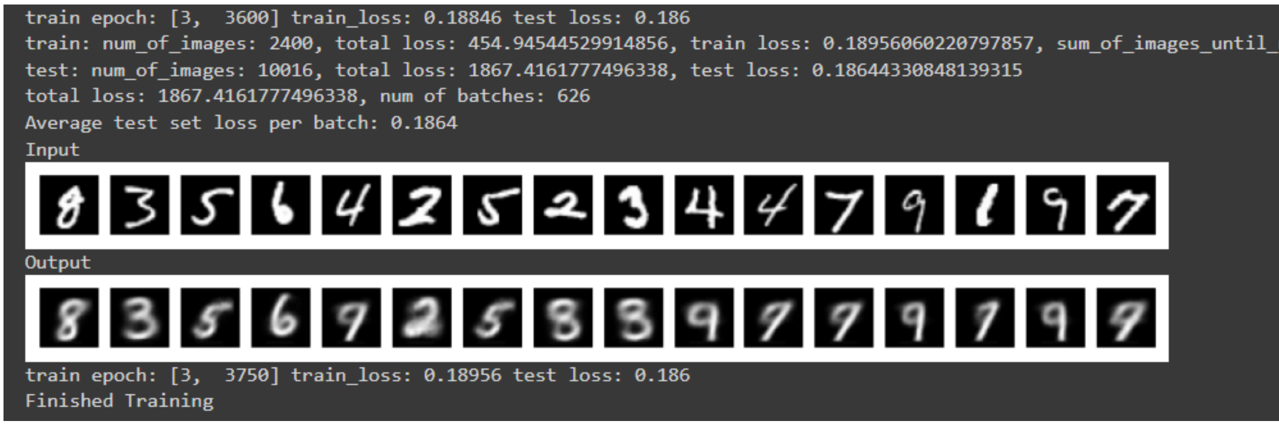


ניתן לראות בשלב ראשוני שיש קורולוציה מעולה בין כמה שאנחנו מכריחים את המודל לדחוס את המידע לבין הלוס שהוא מקבל. כלומר ככל שנדחוס את המידע יותר אנחנו נגיע ללוס גבוה יותר כי קשה לו יותר לשחזר תמונה מפחות מידע.

בנוסף נראה שהמודל לא מצליח ללמוד באופן מושלם ולהגיע ללוס 0 אפילו על ה$train$ אפילו כשאני נותן לו לדחוס את המידע לוקטור בגודל 784 שזה 28 בריבוע ככה שטכנית הוא יכל ללמוד וריאציה של פונקציית הזהות ולהגיע ללוס 0.

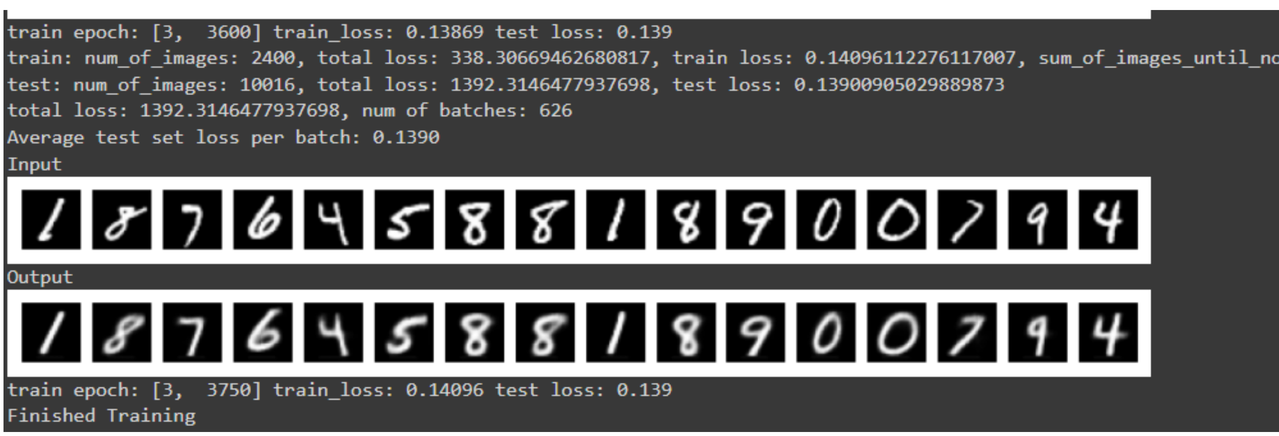
מה שכן אפשר לראות שהתוצאה הויזואלית באמת מגיעה לתוצאה מרשימה ככל שמגדילים את הוקטור ולתוצאה בינונית ואפילו גרועה ככל שמקטינים.

הנה התוצאות בצורה ויזואלית:
השורה הראשונה זה הקלט ולמטה זה הפלט של המודל -
$latent\ dimension = 2$:



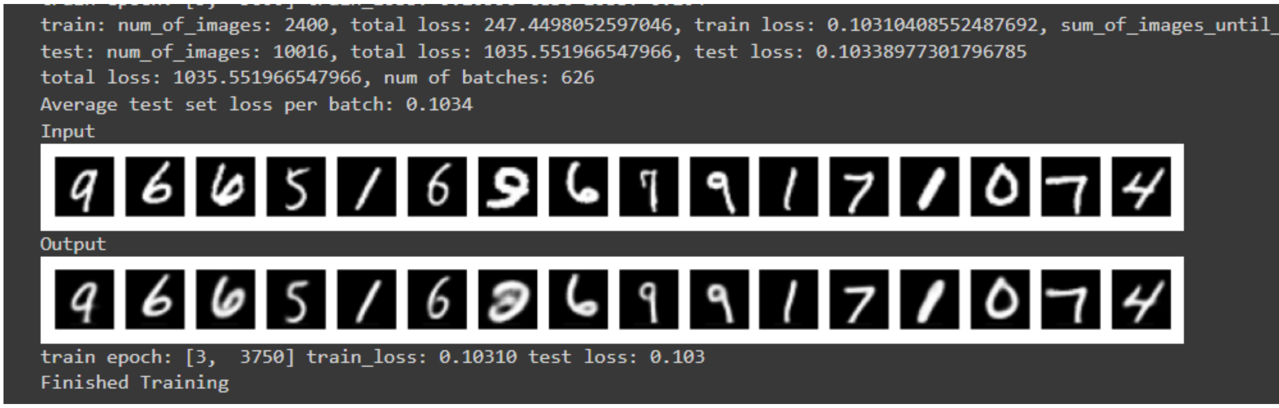
ניתן לראות פה ממש ספרות שגויות - הרבה פעמים 1 ו7 נהפכים ל9 והכל יחסית מטושטש.

$latent\ dimension = 5$:



כאן הספרות שוחזרו באופן יחסית מוצלח אך התמונה תמיד קצת מטושטשת למרות שניתן לזהות את הספרה הנכונה.

$latent\ dimension = 10$:

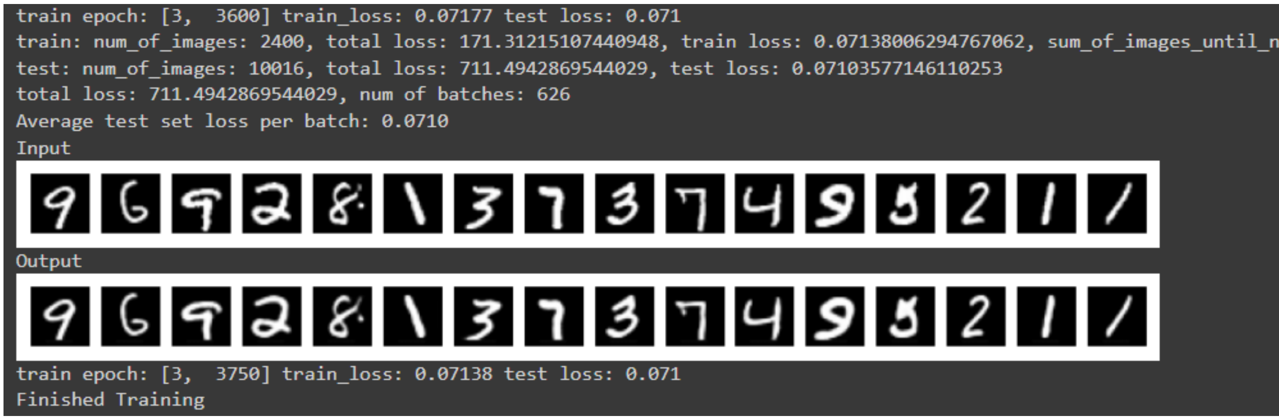


עדיין קצת מטושטש - איכות יחסית סבירה. יותר טוב מהמודל הקודם.

$latent\ dimension = 20$:

די מוצלח - בקושי מטושטש אפילו.

$latent\ dimension = 784$:



מאוד מדויק, חדי עין כן יזהו שינויים מינוריים.

לסיכום השלב הראשון - הצלחנו לשחזר כבר ממימד לא גדול של 10 ו20 שיחזור די מדויק של כל ספרה. ראינו גם שלא מגיעים בקלות לשיחזור אמיתי.

חלק שני של שאלה ראשונה

בשלב הזה קיבעתי את ה$latent\ dimension$ להיות 10 ושיניתי את כמות הערוצים ב4 השכבות ב$encoder$ וה$decoder$.

בחרתי להריץ 15 אפוקים הפעם כדי באמת לנסות ולבחון $overfit$ וההתנהגויות דומות.

הרצתי 6 מודלים עם כמות ערוצים שונה.

כאמור, יש 4 שכבות קונבולוציה, רציתי מצד אחד לאפשר בדיקות עם כמות פרמטרים קטנה - ולכן לא רציתי ששכבת הקונבולוציה האחרונה שמתחברת לשכבת הfc תיהיה עם הרבה ערוצים כי אז זה היה מכפיל את כמות הפרמטרים בשכבת הfc שהיא בעלת הרבה פרמטרים ללמידה בסופו של דבר.

מצד שני רציתי שתיהיה הגדלה של כמות הערוצים ככל שמתקדמים ברשת הקונבלוציה.

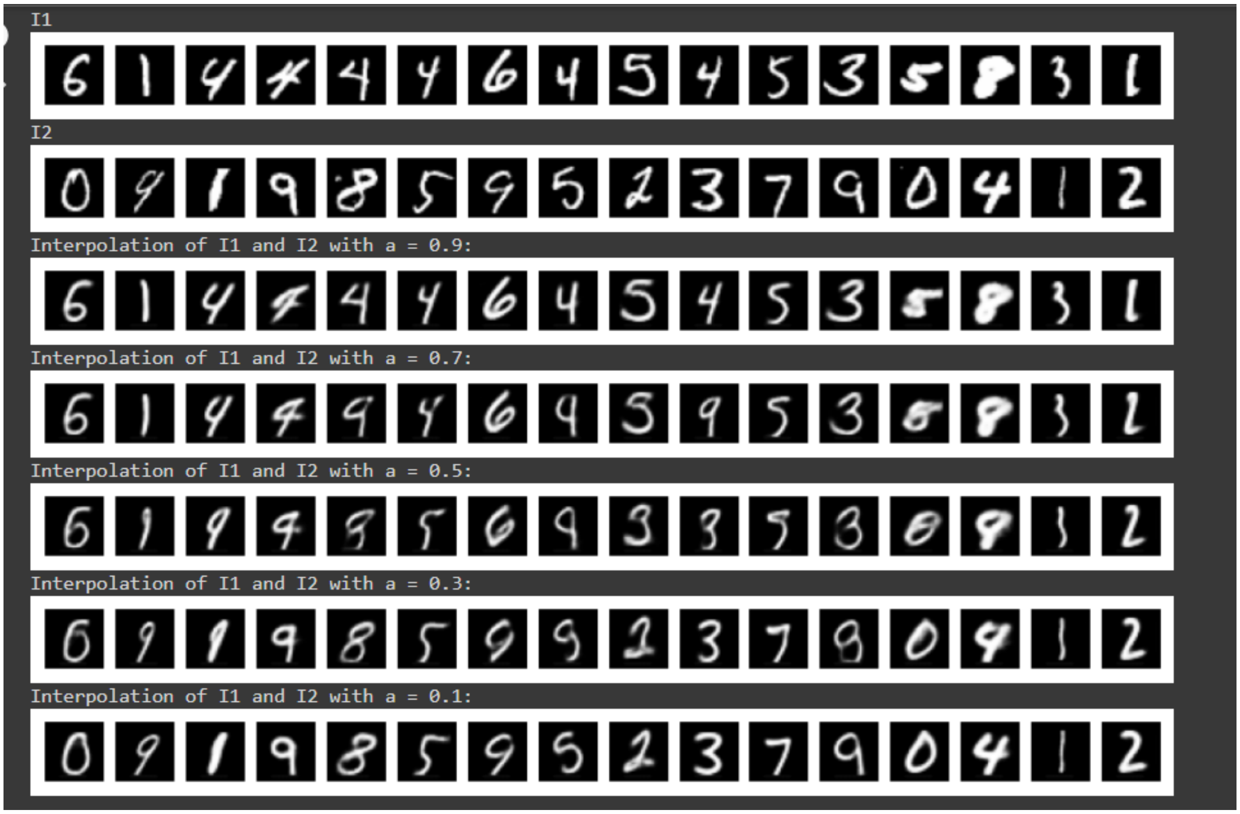
לכן החלטתי על 2 מספרים בכל רשת שייצגו את כמות הערוצים. המספר הראשון, הקטן ייצג את כמות הערוצים ב2 השכבות הראשונות והשני באחרונות.

לדוגמא הכי קטן היה עם 2 4 כלומר 2 ערוצים ב3 השכבות הראשונות, ו4 באחרונות.

והכי גדול היה עם 64 128 256 512 כלומר 64 בראשונה והאחרונה עם 512 ערוצים.

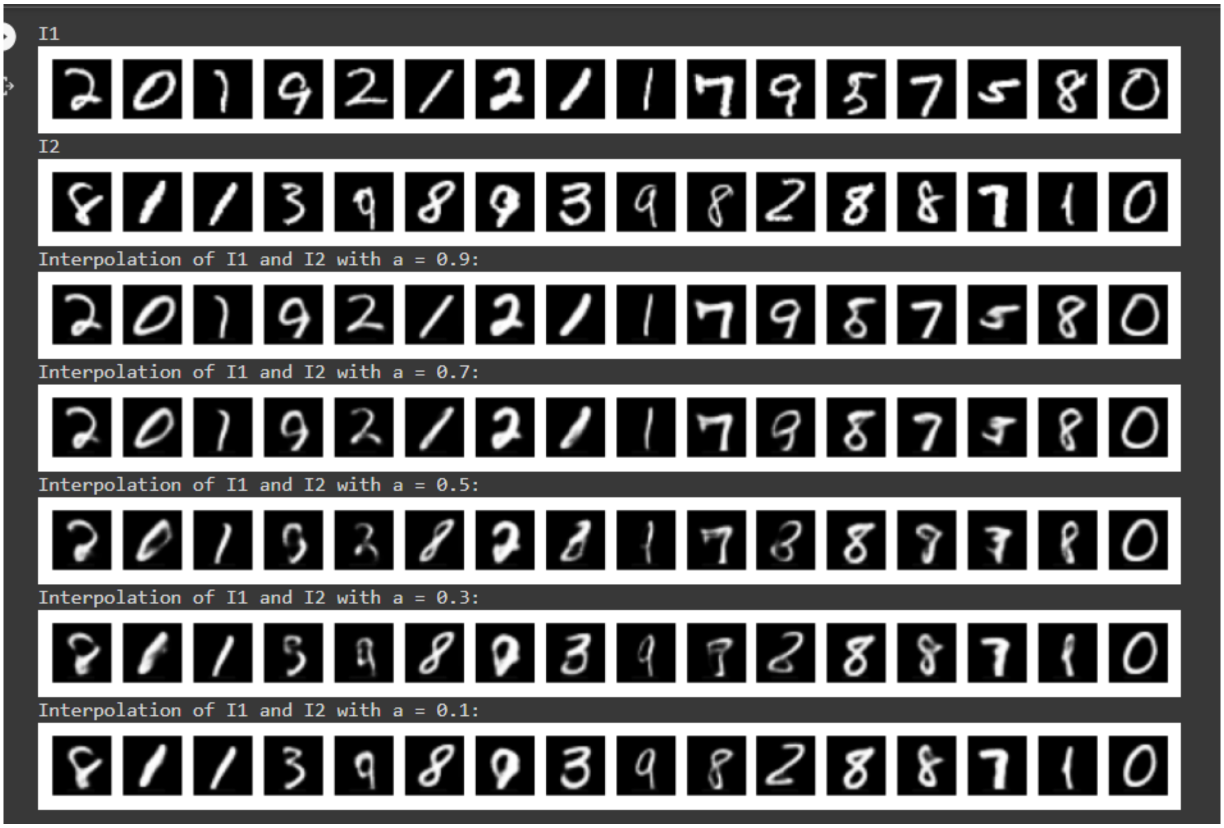
להערכתי הקנה מידה הראשונה יחסית קטן והאחרון יחסית גדול. ועם 15 אפוקים אנחנו אמורים להבחין בתוצאות מעניינות.

תוצאות אינטרפולציה ראשונה עם מימד של $latent\, space = 10$

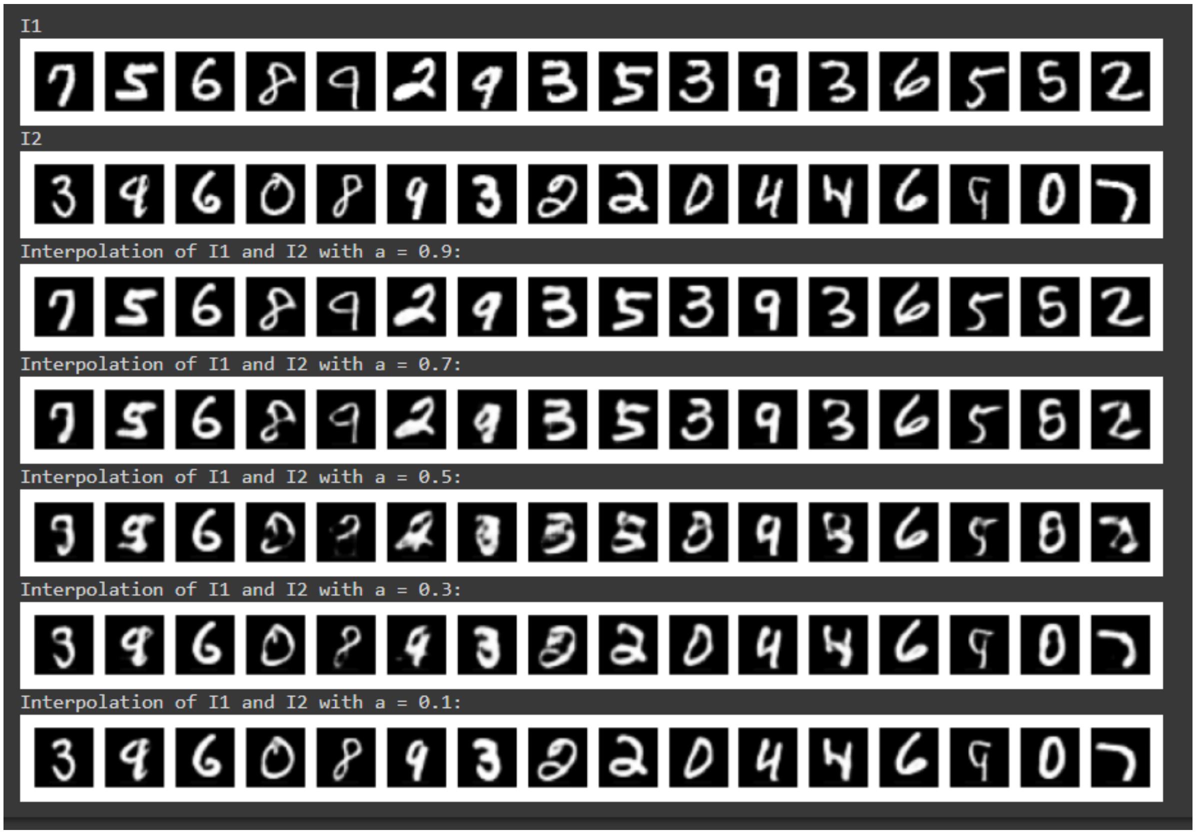


השורה הראשונה היא רצף של 16 תמונות מהטסט סט.
השורה השנייה היא גם רצף אחר של 16 תמונות מהטסט סט.
אנחנו נחבר בין השורות ונקבל 16 שילובים של ספרות שונות.
חמשת השורות הבאות אלו 5 אינטרפולציות עם α שנעה בין 0.9 ל0.1 וככה אפשר לראות את המיזוג שבהתחלה נוטה מאוד לטובת התמונות בשורה הראשונה ולאט לאט מתחלף לתמונה השנייה.

כעת נריץ את אותו קוד עם $latent\ space = 20$ ונראה מה קורה.



לצורך ההשוואה 0 הקצנתי והשתמשתי במודל עם $latent\ space = 728$.
אלו התוצאות:



אפשר לראות בעין שככל שה$latent\ space$ יותר גדולה ההתוצאה של האנטרפולציה לא נותנת תשובה שנראית כמו ספרה. כלומר האיכות קטנה ולכן עדיף מימד יותר נמוך.

בנוסף כשה$latent\ space$ נמוך האינטרפולציה נראית יותר מטושטשת ופחות חד משמעית וכשה$latent\ space$ גבוה המיזוג בין הספרות לא יוצר משהו מובן אבל כן משהו חד.

אני חושב שהסיבה לכך נובעת מהעובדה שהלוס הוא בעייה עם נקודות מינימום שמנסים להגיע אליהם - ככל שה$latent\ space$ יותר נמוך המימד קטן יותר ולכן המרחק בין נקודות המינימום יחסית קטן - מה שמאפשר מיזוג לינארי שהוא בעצמו בעל לוס נמוך. לעומת זאת כש$latent\ space$ גבוה הבעייה בעלת מימד גבוה יותר והמיזוג הלינארי יכול להתרחק בקלות רבה יותר - מה שיגרור תוצאה מה$decoder$ שלא קרובה לשום נקודת מינימום שהוא התכנס אליה.

שאלה 3 פרקטית:

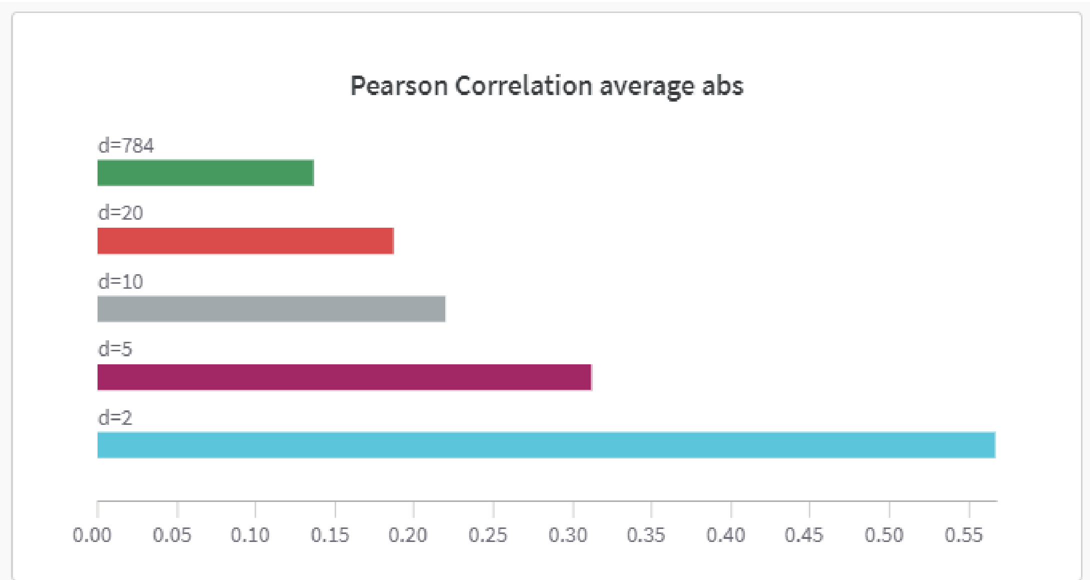
לקחתי את המודלים משאלה 1 והרצתי אותם על כל הטסט סט, כלומר 10 אלף תמונות.

כלומר כל מודל רץ על כל הטסט סט - לאחר מכן ביצעתי לכל מודל את הפעולות הבאות:

לקחתי את הוקטור שה$encoder$ פלט לכל תמונה וככה קיבלתי לכל קוואורדינטה ב$latent\ vector$ (כלומר בווקטור הייצוג של התמונה אחרי הדחיסה) ווקטור באורך 10 אלף שייצג את הפיצ'ר הנוכחי.

בשלב הזה לקחתי וביצעתי קורולציה בין כל ווקטורי הפיצ'רים ככה שקיבלתי מטריצה (סימטרית) עם הקורלציות בין הפיצ'רים, המטריצה בגודל הd על d כשהd הוא אורך ה$latent\ vector$.

למטריצה הזאת ביצעתי ערך מוחלט לכל הערכים בה וממוצע של כל הערכים בה וככה קיבלתי ערך יחיד שמייצג את כמות הקורלוציה לכל מודל.
אלו התוצאות שקיבלתי:



ניתן לראות באופן יחסית מובהק ירידה בקורולציה ככל שהd גדל.
ההגיון מאחורי זה הוא שכשווקטור הייצוג גדול יותר ניתן לשמור יותר פיצ׳רים שונים זה מזה ללא תלות אחד בשני.
כשהמידע דחוס לווקטור קטן יש יחסים יותר ברורים כי המידע חייב להכליל את כל מה שהוא ראה ולכן לכל תמונה חדשה שהוא מנסה ללמוד הוא ייאלץ להזיז מעט ממידע שעזר לייצג תמונה אחרת - ככה שמעט ייצוג יצטרך לכלול המון מידע ולכן כל קואורדינטה בפני עצמה תצטרך גם קואורדינטות אחרות כדי לסמן על תכונה ספציפית. לעומת זאת אם היה הרבה מקום לייצג את המידע כל תכונה ספציפית יכלה לקבל קוראורדינטה נפרדת או קרוב לכך.
לפעמים יכול להיות שהגדלה של מספר הפיצ׳רים כן יגרום לעלייה קטנה בקורלוציה, אני אישית לא ראיתי את זה - אולי כי לא בחרתי מספיק גדלי ווקטור לבדוק. הסיבה לעלייה כזאת יכולה להיות תלויה בקשר שנוצר בין כמה פיצ׳רים ספציפיים שמגדירים תכונות דומות שעדיין יותר חשובות משאר התכונות. כמו למשל 2 תכונות שמגדירות סוג של כלב כי מופיעים הרבה כלבים, למרות שכשהיה לנו ווקטור ייצוג קטן יותר לא יכולנו לתת על זה את הדעת כי יש דברים יותר חשובים מאיזה סוג של כלב.

שאלה 4 פרקטית:

בחרתי להישאר עם המודל המקורי - כשה$latent\ space = 10$ ויש 4 שכבות קונבלוציה עם 16,32,64,128 ערוצים בהתאם לכל שכבה עם fc בסוף ו$relu$ כאקטיבציה.
הרצתי 6 פעמים סה״כ, 3 עם שינוי המשקולות של ה$encoder$ ו3 בלי השינוי. על אימון שונה של תמונות.
כשמריצים 10 תמונות באימון:
כשמפרידים (מאמנים רק את הMLP וה$encoder$ נשאר אותו דבר):

```
test: num_of_images: 10001, total loss: 22900.12387943268, test loss: 2.2897834096023075
total loss: 22900.12387943268, num of batches: 10001
Average test set loss per batch: 2.2898
Test loss of MLP model without training the encoder togehter 2.2897834096023075
```

כשמחברים:

```
test: num_of_images: 10001, total loss: 23274.305893063545, test loss: 2.3271978695194027
total loss: 23274.305893063545, num of batches: 10001
Average test set loss per batch: 2.3272
Test loss of MLP model while training the encoder togehter 2.3271978695194027
```

התוצאות הן טובות יותר כשמאמנים מחדש רק את הMLP.
כשמריצים 50 תמונות באימון:
כשמפרידים:

```
test: num_of_images: 10001, total loss: 21672.67867076397, test loss: 2.167051161960201
total loss: 21672.67867076397, num of batches: 10001
Average test set loss per batch: 2.1671
Test loss of MLP model without training the encoder togehter 2.167051161960201
```

כשמחברים:

```
test: num_of_images: 10001, total loss: 22027.340943694115, test loss: 2.202513842985113
total loss: 22027.340943694115, num of batches: 10001
Average test set loss per batch: 2.2025
Test loss of MLP model while training the encoder togehter 2.202513842985113
```

שוב התוצאות הן טובות יותר כשמאמנים מחדש רק את הMLP.
כשמריצים 100 תמונות באימון:
כשמפרידים:

```
test: num_of_images: 10001, total loss: 21239.016813755035, test loss: 2.123689312444259
total loss: 21239.016813755035, num of batches: 10001
Average test set loss per batch: 2.1237
Test loss of MLP model without training the encoder togehter 2.123689312444259
```

כשמחברים:

```
test: num_of_images: 10001, total loss: 19976.841914892197, test loss: 1.9974844430449152
total loss: 19976.841914892197, num of batches: 10001
Average test set loss per batch: 1.9975
Test loss of MLP model while training the encoder togehter 1.9974844430449152
```

כעת התוצאות הן טובות יותר כשמאמנים מחדש את כל המודל כולל ה$encoder$ ולא רק את הMLP.

הסיבה לכך היא שיש קצת אימון אז ניתן יותר בקלות להתכנס לאימון טוב על כמות קטנה של פרמטרים 0 ולכן לאמן רק את הMLP עבד יותר טוב כשמנסים לאמן שוב על מעט תמונות. ככה הגראדיינטים לא ניסו להתפרש על מרחב ענק של פרמטרים והצליחו לבצע התכנסות של המודל הקטן מהר יותר.

לעומת זאת כשהגענו כבר ל100 תמונות כנראה כבר החלקשל המודל הקטן הצליח להתכנס אפילו על המודל המשולב ולכן מכיוון שהמודל המשולב מתאמן על כמות גדולה של פרמטרים הוא מצליח להגיע בסך הכל לפתרון יותר טוב.
במידה והיינו לוקחים מודל MLP קטן יותר אזי היה ניתן לראות את ההפרש בלימוד עם כמות תמונות קטנה אפילו יותר טוב.

שאלות תיאורטיות:
שאלה 1:
הראו שהרכבה של פונקציות לינאריות היא פונקציה לינארית.
הוכחה: יהי f, g פונקציות לינאריות. g ממפה איברים לגודל קלט שf מקבל.
מכיוון ששניהן פונקציות לינאריות הן מקיימות:

$$f(x_1 + y_1) = f(x_1) + f(y_1)$$

$$g(x_2 + y_2) = g(x_2) + g(y_2)$$

$$a \cdot f(x_1) = f(ax_1)$$

$$a \cdot g(x_2) = g(ax_2)$$

כשa הוא סקלר וx_i, y_i ווקטורים באותו גודל.
נוכיח ש$h = g(f)$ היא מקיימת את 2 התכונות הללו:

$$h(x_1 + y_1) = g(f(x_1 + y_1)) = g(f(x_1) + f(y_1)) = g(f(x_1)) + g(f(y_1))$$

לפי 2 תכונות הלינאריות של f וg.
נוכיח תכונה שנייה:

$$a \cdot h(x_1) = a \cdot g(f(x_1)) = g(a \cdot f(x_1)) = g(f(a \cdot x_1))$$

יכולנו בשיטה אחרת גם להשתמש בקשר בין מטריצה לפונציה לינארית ואז לבצע כפל מטריצות ולקבל מטריצה יחידה ואז להשתמש בטענה שלכל מטריצה יש פונקציה לינארית תואמת.

הראו שהרכבה של פונקציות אפיניות היא פונקציה אפינית.
פונקציה אפינית מוגדרת באמצעות הנוסחא $f = mx + b$
כשm יכולה להיות טנזור מכל סוג וx וb בהתאם למימדים שאנחנו מקבלים ופולטים.
התכונות של פונקציה כזאת היא לינאריות וקיום קבוע b.
קרי, יהי f פונקציה אפינית אזי f היא לינארית ומקיימת:

$$f(0) = b$$

כשb הוא קבוע כלשהו עם מימד כמו שהסברנו.
נוכיח:
יהי f, g פונקציות אפיניות - אזי הן לינאריות ולכן ההרגה שלהן לינארית כמו שהוכחנו.
נסמן אותן כך:

$$f = m_1 x + b_1$$

$$g = m_2 x + b_2$$

נוכיח כעת שיש להרכבה שלהן גם קבוע:

$$h(0) = g(f(0)) = g(b_1) = m_2 b_1 + b_2$$

ומכיוון שכל הרכיבים בצד ימין של המשוואה קבועים אזי קיבלנו קבוע.
מש"ל.

שאלה 2 תיאורטית:
שאלה: מהו תנאי העצירה של: $\theta^{n+1} = \theta^n - \alpha \nabla f_{\theta^n}(x)$?
תשובה: לשאלה הזאת יש כמה תשובות.

מכיוון שמדובר במודלי למידה, ולא בתשובה מתמטית, אזי כל אחת מהתשובות שאני אביא היא תשובה שאמורה להיות הגיונית ואפילו שילוב שלהן. בסוף מתכנתים את זה ורוצים שהמודל יעצור ללמוד כשהגענו לנקודת מינימום כלשהי של הלוס.

מתי נרצה להפסיק להתאמן? כשהאימון כבר לא יעיל וכל איטרציה נוספת לא מבצעת שינוי משמעותי.

נוכל להחליט לעצור על פי תנאי של $threshold$ שנבחר שאם הגראדיינט באיטרציה מסוימת קטן ממנו בערכו המוחלט אז נצא מהלולאה.

כמובן שנוכל לבחור תנאים שקשורים למגבלות החישוב שלנו - כמו מספר איטרציות מקסימלי או ערך לוס מסוים שכשירדנו ממנו זה מספיק לנו.

שאלה: השתמש במשפט טיילור רב מימדי מסדר שני כדי להגדיר תנאי שיאפשר לסווג נקודה סטיציונארית כמינימום או מקסימום לוקאלי.

תשובה: נרצה לבצע חישוב אחיד שייתן לנו תוצאה חד משמעית האם הנקודה הסציונארית היא מינימום או מקסימון לוקאלי, וגם יש אפשרות שהיא נקודת אוכף.

מכיוון שהנקודה סטציונארית אזי הנגזרת שלה מתאפסת ונרצה להסתכל על הנגזרת השנייה שלה.

מכיוון שאנחנו בעולם רק מימדי הנגזרת השנייה שלה היא ההסיאן שלה.

בעולם דו מימדי היינו רוצים לדעת האם בסביבה הקרובה של הנקודה הנגזרת השנייה חיובית או שלילית. בעולם של מטריצות זה מקביל לחישוב ערכים עצמיים.

אם כל הערכים העצמיים של מטריצת ההסיאן הם חיוביים אזי קיבלנו מינימום לוקאלי - הפונקציה בעצם קמורה בסביבה של הנקודה.

באופן שקול אם כל הערכים העצמיים שליליים אזי הפונקציה קעורה בסביבה ולכן יש לנו נקודת מקסימום.

אם חלק שליליים וחלק חיוביים אזי יש לנו אוכף.

במידה ולפחות ערך עצמי אחד הוא אפס אזי אי אפשר למצוא פתרון באמצעות השיטה הזאת כי אין לנו קמירות או קעירות חד משמעיים על הנקודה.

שאלה 3 תיאורטית:

נרצה לפתור בעייה של חיזוי זווית בין 0 ל360.

פתרון: נשתמש ב$CMAE$ שזה קיצור של $circular\ mean\ absolute\ error$.

נסמן y_{pred} בתור הפרדיקציה ואת y_{true} בתור הלייבל.

ונחשב:

1) נקבל כקלט y_{pred}, y_{true}

2) נגדיר אובייקט בשם $diff$ ונשים לו את הערך $|((y_{pred} - y_{true} + 180)\%360) - 180|$

ככה בעצם נקבל ערך שהוא סוג של מרחק בין הפרדיקציות, (הסבר על החישוב עכשיו:)

-אם הפרדיקציה גדולה מהלייבל:

-אם המרחק שלהם קטן מ180 אז נקבל ערך מתחת ל180 שייצג את המרחק שלנו במדוייק כי לא היה פעולת מודולו ואז התוספת והחיסור של ה180 הייתה על אותו הערך.

-אם המרחק שלהם גדול מ180 אזי הזווית קטנה יותר מהכיוון השני ולכן כשנוסיף 180 נקבל ערך גדול מ360 ואז פעולת המודולו תוריד אותנו למשלים ל180 של המרחק שלהם מהכיוון הקצר, מה שיגרור שפעולת מינוס ב180 וערך מוחלט תביא בדיוק את המרחק שלהם זה מזה.

נדגים, $y_{pred} = 350, y_{true} = 10$:

$$|((y_{pred} - y_{true} + 180)\%360) - 180| = |((350 - 10 + 180)\%360) - 180|$$

$$= |((340 + 180)\%360) - 180| = |((520)\%360) - 180| = |((160) - 180|$$

$$= |-20| = 20$$

-אם הפרדיקציה קטנה מהלייבל נקבל חישוב קצת הפוך. כי ההפרש תמיד יהיה שלילי.
-אם ההפרש קטן ממינוס 180 נקבל ערך חיובי קטן מ180 והמודולו לא ישנה כלום והמינוס 180 יחזיר לאותו הערך. (הערך מוחלט יהפוך את זה למרחק האמיתי ולא ישאיר מינוס)
-אם ההפרש גדול ממינוס 180 יהיה חישוב דומה של הזווית הנגדית כמו מקודם רק עם מינוס. (סוף הסבר על החישוב)
3) נחשב את הMSE של ה$diff$ ונחזיר כערך. כלומר ההעלאה בריבוע וממוצע.

שאלה 4 תיאורטית:
$(a$

$$\frac{\partial}{\partial x} f(x+y, 2x, z)$$

נגדיר פונקציה g שממפה את הקלט לf:

$$g(x,y,z) = (x+y, 2x, z).$$

ומכלל השרשרת:

$$\frac{\partial}{\partial x} f(g(x,y,z)) = \frac{\partial f}{\partial g} \cdot \frac{\partial}{\partial x} g(x,y,z)$$

נחשב את $\frac{\partial f}{\partial g}$:

$$\frac{\partial f}{\partial g} = \frac{\partial f}{\partial u} \cdot \frac{\partial u}{\partial g} + \frac{\partial f}{\partial v} \cdot \frac{\partial v}{\partial g} + \frac{\partial f}{\partial w} \cdot \frac{\partial w}{\partial g}$$

$$= \frac{\partial f}{\partial u}(1,0,0) + \frac{\partial f}{\partial v}(0,2,0) + \frac{\partial f}{\partial w}(0,0,1) = \left(\frac{\partial f}{\partial u}, 2\frac{\partial f}{\partial v}, \frac{\partial f}{\partial w}\right)$$

כשהגדרנו $u = x+y,\ v = 2x,\ w = z$.

נמשיך בחישוב:

$$\frac{\partial}{\partial x} g(x,y,z) = (1,2,0)$$

מבניית g.

$$\frac{\partial}{\partial x} f(x+y, 2x, z) = \frac{\partial f}{\partial u} + 2\frac{\partial f}{\partial v}$$

מהפעלת הנגזרת על כל רכיב.

$$\frac{\partial}{\partial x} f(x+y, 2x, z) = \frac{\partial f}{\partial (x+y)} + 2\frac{\partial f}{\partial (2x)}$$

מהצבה.

$f_1(f_2(...f_n(x)))$ (b

תשובה: נבצע באופן רפטטיבי את כלל השרשרת על הפונקציה f_i:

$$\frac{\partial}{\partial x} f_1(f_2(\dots f_n(x))) = \frac{\partial}{\partial u_1} f_1(u_1) \cdot \frac{\partial u_1}{\partial x}$$

כש־ $u_1 = f_2(\dots f_n(x))$.
ונחשב פנימה:

$$\frac{\partial u_1}{\partial x} = \frac{\partial}{\partial x} f_2(\dots f_n(x)) = \frac{\partial}{\partial u_2} f_2(u_2) \cdot \frac{\partial u_2}{\partial x}$$

כש ־ $u_2 = f_3(\dots f_n(x)..))$. נמשיך באופן גנרי עם $u_i = f_{i+1}(...f_n(x)..))$:

$$\frac{\partial}{\partial x} f_1(f_2(\dots f_n(x))) = \frac{\partial}{\partial u_1} f_1(u_1) \cdot \frac{\partial}{\partial u_2} f_2(u_2) \cdot \frac{\partial}{\partial u_3} f_3(u_3) \dots \frac{\partial}{\partial x} f_n(x)$$

(c

$$f_1(x, f_2(x, f_3(...f_{n-1}(x, f_n)))$$

תשובה: נבצע באופן רפטטיבי את כלל השרשרת על הפונקציה f_i:
הפעם כל פונקציה מקבלת 2 פרמטרים ולכן נצטרך לגזור פעמיים לכל f_i לפי כל פרמטר ולחבר את הנגזרות.
ולכן:

$$\frac{d}{dx} f_1(x, f_2(x, f_3(...f_{n-1}(x, f_n))) = \frac{\partial f_1}{\partial x} + \frac{\partial f_1}{\partial f_2} \cdot \frac{\partial f_2}{\partial x} = \frac{\partial f_1}{\partial x} + \frac{\partial f_1}{\partial f_2} \cdot \left(\frac{\partial f_2}{\partial x} + \frac{\partial f_2}{\partial f_3} \right)$$

$$= \frac{\partial f_1}{\partial x} + \frac{\partial f_1}{\partial f_2} \cdot \left(\frac{\partial f_2}{\partial x} + \frac{\partial f_2}{\partial f_3} \right)$$

וככה נמשיך לפתח פנימה את המשוואה כשכל $\frac{\partial f_i}{\partial x}$:

$$\frac{\partial f_i}{\partial x} = \left(\frac{\partial f_{i+1}}{\partial x} + \frac{\partial f_i}{\partial f_{i+1}} \right)$$

$(d$

$$f(x + g(x + h(x)))$$

נפרק כל ביטוי כפול סכום הנגזרות של הפונקציות הפנימיות:

$$\frac{df}{dx} = \frac{\partial}{\partial x} f(x + g(x + h(x)) \cdot (1 + \frac{\partial}{\partial x} g(x + h(x))) \cdot (1 + \frac{\partial}{\partial x} h(x))$$