Practical Part – Bar Rousso I.D. 203765698

1. Auto-Encoding

A) <u>Comparing the SAME encoder/decoder architecture, while changing the latent space dimension</u>

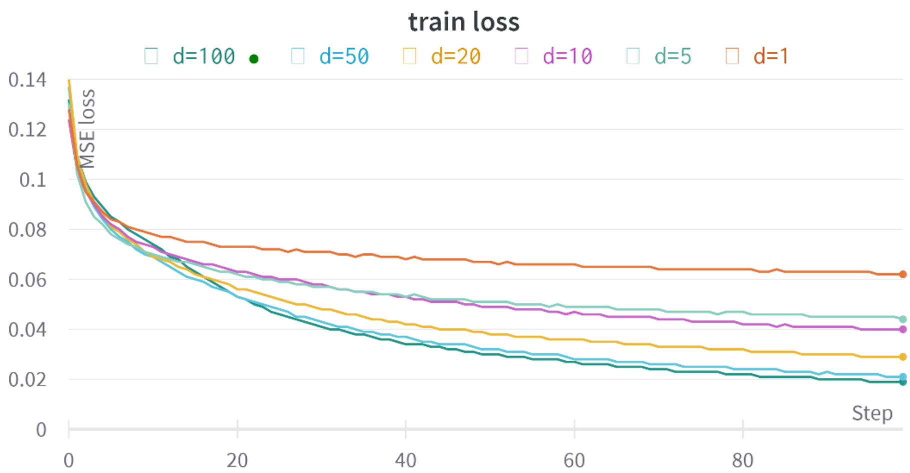
Encoder network structure: (bais included)

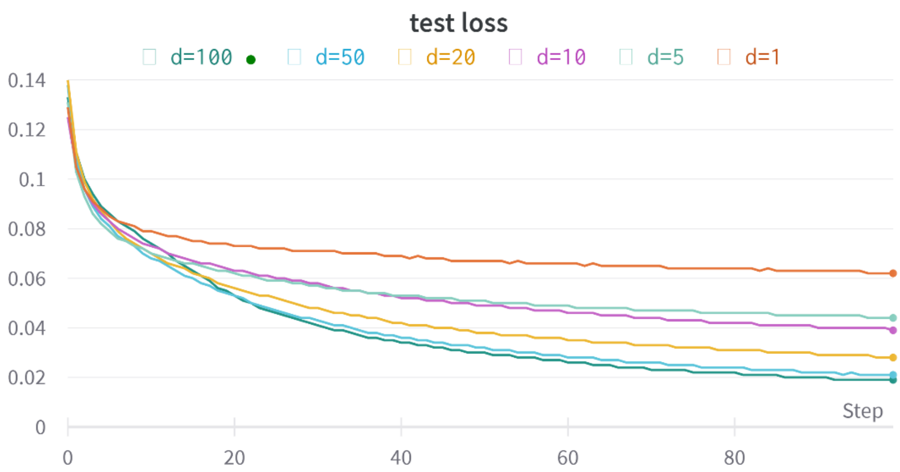
- Convolution layer: 1 input channel, 18 output channels, 5X5 kernel size, stride = 2
- 2D Batch norm
- RELU
- Convolution layer: 18 input channels, 36 output channels, 3X3 kernel size, stride = 2
- 2D Batch norm
- RELU
- Fully connected layer: 900 input feathers, *d* output features
- RELU

Encoder network structure: (bais included)

- Fully connected layer: *d* input features, 900 output features
- RELU
- Transpose convolution layer: 36 input channels, 18 output channels, 3X3 kernel size, stride = 2, out padding = 1
- 2D Batch norm
- RELU
- Transpose convolution layer: 18 input channels, 1 output channel, 5X5 kernel size, stride = 2, out padding = 1
- Sigmoid

Results:





Latent dimension	Number of parameters	Train loss	Test loss
1	15482	0.062	0.062
5	22686	0.044	0.044
10	31691	0.04	0.038
20	49701	0.029	0.028
50	103731	0.021	0.021
100	193781	0.019	0.019

Conclusions:

We can see that as we increase the latent space dimension, we get **lower** train and test losses. This can be explained as bigger latent space dimension enable each encoded vector to captures more details of its original image.

B) <u>Comparing encoder/decoder architectures with different number of layers, while FIXING the latent space dimension</u>

<u>One layer architecture:</u>

Encoder network structure: (bais included)

- Convolution layer: 1 input channel, 18 output channels, 5X5 kernel size, stride = 4
- 2D Batch norm
- RELU
- Fully connected layer: 648 input feathers, *d* output features
- RELU

Encoder network structure: (bais included)

- Fully connected layer: *d* input features, 648 output features
- Transpose convolution layer: 18 input channels, 1 output channel, 5X5 kernel size, stride = 4, out_padding = 3
- Sigmoid

Total number of parameters: 27543

<u>Two layers architecture:</u> The same architecture described in the first section

<u>Three layers architecture:</u>

Encoder network structure: (bais included)

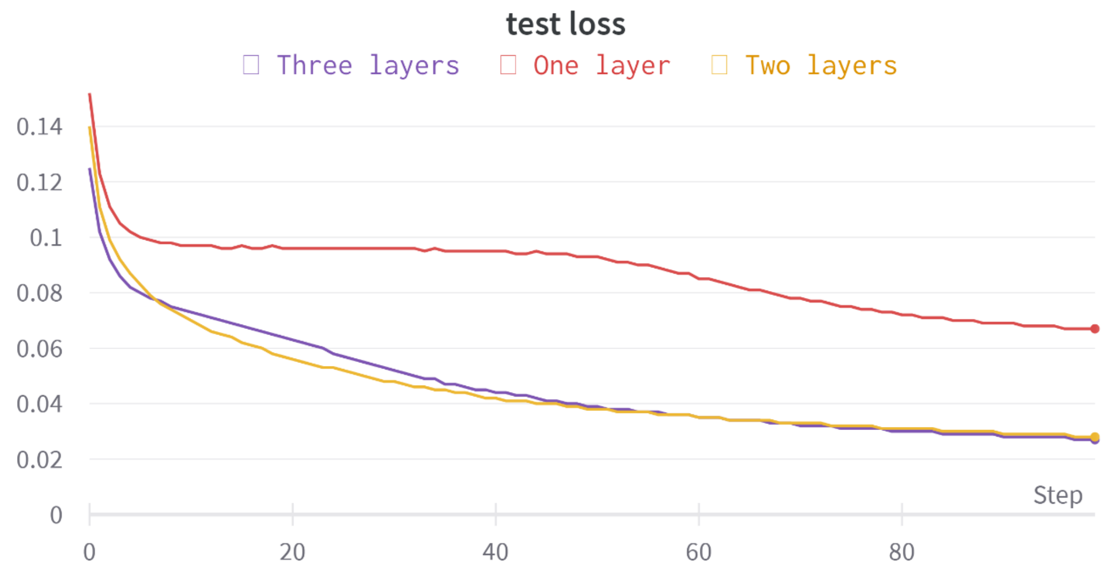
- Convolution layer: 1 input channel, 9 output channels, 3X3 kernel size, stride = 2, padding = 1
- 2D Batch norm
- RELU
- Convolution layer: 9 input channels, 18 output channels, 3X3 kernel size, stride = 2, padding = 1
- 2D Batch norm
- RELU
- Convolution layer: 18 input channels, 36 output channels, 3X3 kernel size, stride = 2, padding = 1
- 2D Batch norm
- RELU
- Fully connected layer: 324 input feathers, *d* output features
- RELU

Encoder network structure: (bais included)

- Fully connected layer: *d* input features, 324 output features
- RELU
- Transpose convolution layer: 36 input channels, 18 output channels, 3X3 kernel size, stride = 2, out_padding = 1
- 2D Batch norm
- RELU
- Transpose convolution layer: 18 input channels, 9 output channels, 3X3 kernel size, stride = 2, padding = 1, out_padding = 1
- 2D Batch norm
- RELU
- Transpose convolution layer: 9 input channels, 1 output channel, 3X3 kernel size, stride = 2, padding = 1, out_padding = 1
- Sigmoid

Total number of parameters: 28317

Results:



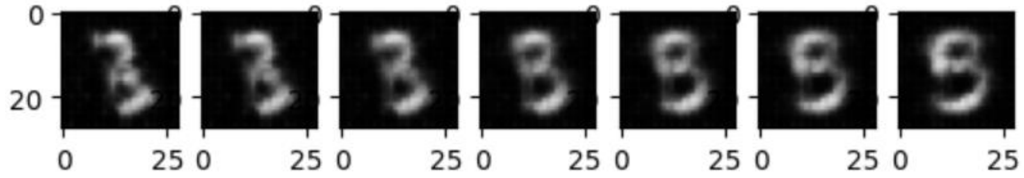
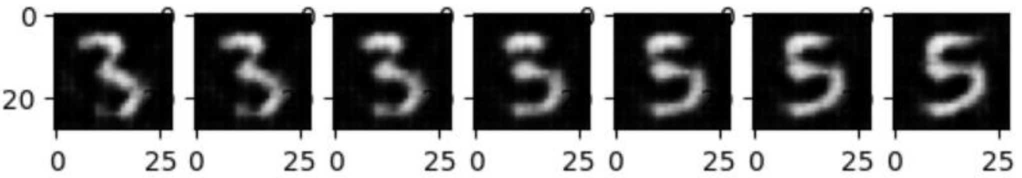
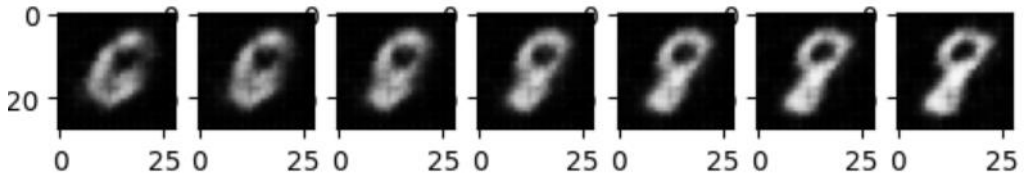
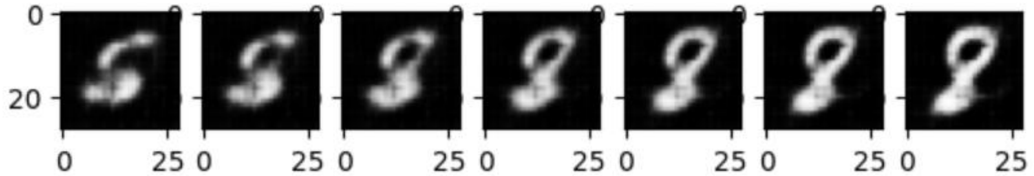
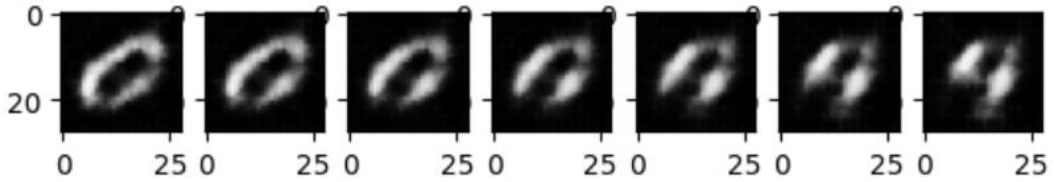
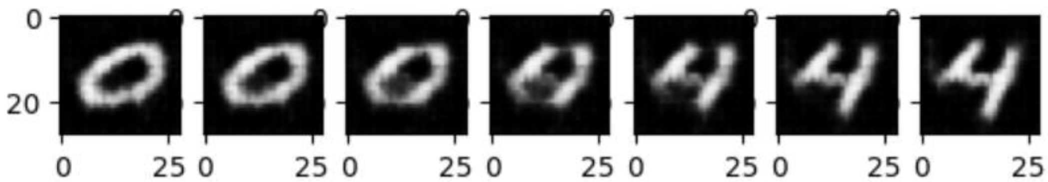
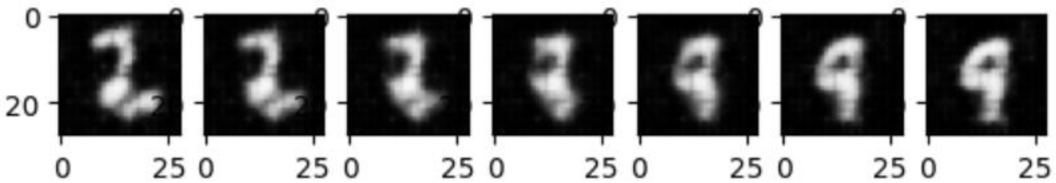
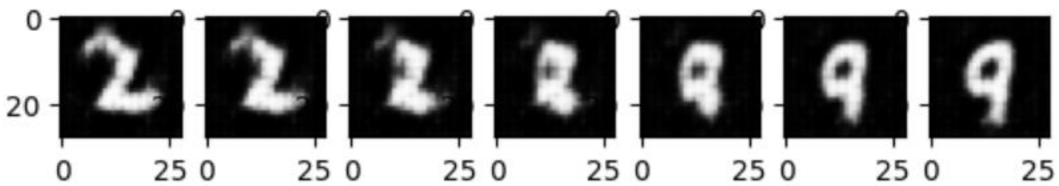
Conclusions:

We can see that as we add more convolutions layers, we got a better module in terms of:

(1) Lower final train and test losses results
(2) Faster convergence to a module with loss train and test losses
(3) Using less parameters for better performance

2. Interpolation

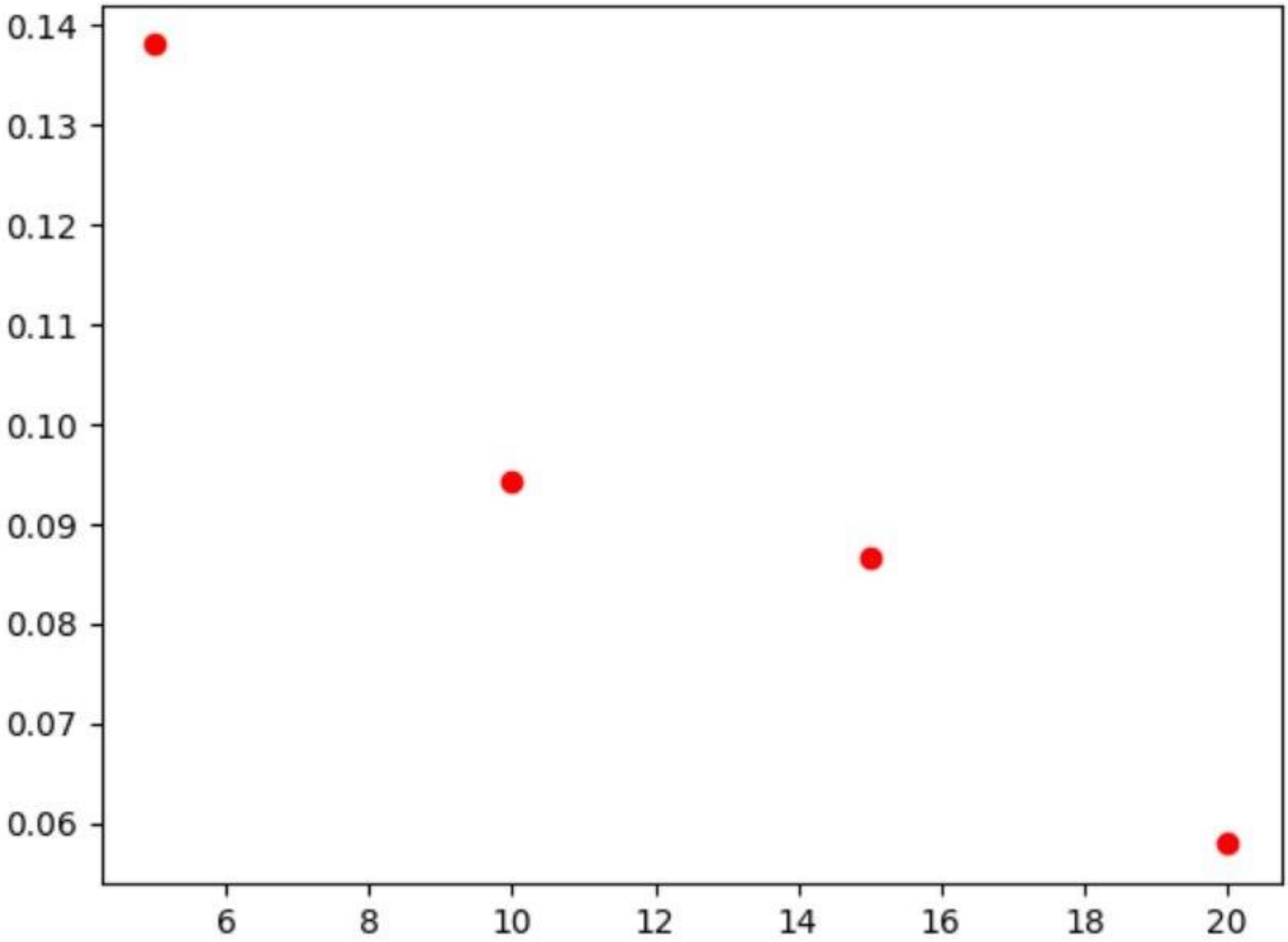
Comparing results generated from different latent space dimensions 20 VS 100:

Interpolation between	latent space dimension	Results
3 -> 5	20	
	100	
5 -> 8	20	
	100	
0 -> 4	20	
	100	
2 -> 9	20	
	100	

Conclusions: We can see that the bigger latent space dimension we use, the decoder produce more "sharper" images that look more realistic. This is because a bigger latent space dimension enables us to encode more details of the image.

3. **Decorrelation**

Plot of the MEAN Pearson correlation between all couples of coordinates in the latent space (in absolute values) as a function of the latent space dimension:



Conclusions:

As we can see in the plot, as the we increase the latent space dimension, we get a smaller correlation between different coordinates. This is because a larger latent space dimension enables us to better represent the images by capturing **more variations** of the image, resulting in more coordinates with **wicker correlation**.

Note: Some pairs of the Pearson correlation resulted in NAN. To overcome this, I replaced them with zeros to not influence the mean.

4. **Transfer Learning**

Trained Encoder network structure: (bais included, latent space dimension = 20)

- Convolution layer: 1 input channel, 9 output channels, 3X3 kernel size, stride = 2, padding = 1
- 2D Batch norm
- RELU
- Convolution layer: 9 input channels, 18 output channels, 3X3 kernel size, stride = 2, padding = 1
- 2D Batch norm
- RELU
- Convolution layer: 18 input channels, 36 output channels, 3X3 kernel size, stride = 2, padding = 1
- 2D Batch norm
- RELU
- Fully connected layer: 324 input feathers, 20 output features
- RELU

MLP network structure: (bais included)

- Fully connected layer: 20 input feathers, 50 output features
- RELU
- Fully connected layer: 50 input feathers, 100 output features
- RELU
- Fully connected layer: 100 input feathers, 10 output features
- SoftMax

The **Classification** module concrete both encoder and MLP, and was trained with respect to Cross Entropy loss function.

The classification model was trained according to two scenarios:

1. Only MLP wights were updated
2. Both MLP and Encoder wights were updated

Results:

Scenario 1: `train loss=2.303, test loss=2.304`

Scenario 2: `train loss=2.300, test loss=2.300`

Comment:

Unfortunately, I got losses values that is not make sense. I didn’t succeed to debug the problem.

I know that the second scenario supposed to have better results, since we allow to more parameters to be updated.

Also, as I mentioned above, the bigger the latent space dimension, the better the latent vectors as they capturing more details of the input images.

Thus, the MLP module will do a better job in classify the vectors correctly as they contain more data.

חלק תיאורטי - מגיש דר חסון 203765698

1) א) צ"ל: הרכבה של פונקציות לינאריות היא פונקציה לינארית

הוכחה: תהיינה $g: A\to B$ $f: B\to C$ שתי פונקציות לינאריות כאשר A,B,C מרחבים וקטוריים מעל שדה R.

נראה כי גם $f\circ g: A\to C$ היא העתקה לינארית

(I) אדטיביות: יהיו $a_1,a_2\in A$ אז

$$f\circ g(a_1+a_2) \underset{\text{אדטיביות של } g}{=} f(g(a_1)+g(a_2)) \underset{\text{אדטיביות של } f}{=} f\circ g(a_1)+f\circ g(a_2)$$

(II) הומוגניות: יהי $a\in A$ ו- $\alpha\in R$.

$$f\circ g(\alpha\cdot a) \underset{\text{הומוגניות של } g}{=} f(\alpha\cdot g(a)) \underset{\text{הומוגניות של } f}{=} \alpha\, f\circ g(a)$$

לכן מהגדרה נובע כי $f\circ g$ פונקציה לינארית

ב) צ"ל: הרכבה של פונקציות אפיניות היא פונקציה אפינית.

הוכחה: יהיו $g: A\to B$, $f: B\to C$ שתי פונקציות אפיניות כאשר A,B,C מרחבים אפיניים מעל שדה R

לפי הגדרה: קיימים קבועים $b^*\in B$, $c^*\in C$ והעתקות לינאריות $M_g: A\to B$, $M_f: B\to C$ כך ש:

לכל $a\in A$ $g(a)=M_g(a)+b^*$

לכל $b\in B$ $f(b)=M_f(b)+c^*$

נראה כי לכל $a\in A$:

$$f\circ g(a) = f(M_g(a)+b^*) = M_f(M_g(a)+b^*)+c^* \underset{\text{אדטיביות של } M_f}{=} M_f\circ M_g(a)+M_f(b^*)+c^*$$

מכיוון ש $M_f\circ M_g$ פונקציה לינארית (מסעיף קודם)

וכי $M_f(b^*)+c^*$ קבוע ב C נובע כי $f\circ g$ אפינית - מש"ל

2) ס) עבור ה אופטימיזציה הבאה $\theta^{n+1}=\theta^{n}-\alpha\nabla f_{\theta^n}(x)$

מה התנאי עצירה?

<u>תשובה</u>: נשים לב תמיד, כי התהליך הדיסקרטי עלול אמור להוביל אותנו למינימום מקומי של f_{θ^n}. ואם f_{θ^n} קמורה פונקציה קמורה אז זהו מינימום גלובלי

התנאי עצירה הם היפר פרמטר של האלגוריתם שלנו ויכולים להיות מוגדר מ:

- מספר מקסימלי של איטרציות
- $\left|f_{\theta^n}(\theta^n)-f_{\theta^n}(\theta^{n-1})\right|<\epsilon$ עבור ϵ שנקבע מראש.

את התנאים עצמם בד"כ קובעים בהתאם למשאבים שאנחנו ברשותינו ולרמת הדיוק אליה אנו שואפים להגיע.

ב) לפי הקירוב מסדר שני, עבור סביבה x_0 בה פונקציה f גזירה פעמיים:

$$f(x)=f(x_0)+\nabla f(x_0)\cdot(x-x_0)+(x-x_0)^T\cdot H(x_0)\cdot(x-x_0)+o(\|x-x_0\|^3)$$

כאשר $H(x_0)$ זה מטריצת ההסיאן של f בנק' x_0.

בעזרת הקירוב לעיל, נמצא נסח את התנאים לכך נק' <u>סטציונרית</u> x_0 היא מינימום / מקסימום מקומי של f.

I) נטען כי אם $H(x_0)$ מטריצה מוגדרת <u>חיובית</u> לחלוטין אזי x_0 היא נק' <u>מינימום</u> לוקאלית של f.

אכן, מכיוון ש $H(x_0)$ מוגדרת חיובית-לחלוטין, אז <u>בפרט</u> לכל $x\neq x_0$ מתקיים: $(x-x_0)^T\cdot H(x_0)\cdot(x-x_0)>0$

מכאן נובע כי קיים ε מספיק קטן כך שלכל

$x_0 \neq x \in (x_0-\varepsilon, x_0+\varepsilon)$ מתקיים כי

$$(x-x_0)^T H(x_0)(x-x_0) + o(\|x-x_0\|^3) \geq 0$$

לכן

$$f(x) = f(x_0) + \underbrace{Df(x_0)(x-x_0)}_{=0} + \underbrace{(x-x_0)^T \cdot H(x_0) \cdot (x-x_0) + o(\|x-x_0\|^3)}_{\geq 0} \geq$$

$$\geq f(x_0)$$

נובע כי x_0 נקודת מינימום לוקאלי של f.

(II) אם $H(x_0)$ מטריצה מוגדרת שלילית לחלוטין אז

x_0 נקודת מקסימום לוקאלי של f.

אכן, מכיוון ש-$H(x_0)$ מוגדרת שלילית לחלוטין, אז לכל $x \neq x_0$

$$(x-x_0)^T H(x_0) \cdot (x-x_0) < 0$$

בדומה למקודם, נובע כי קיים ε מספיק קטן כך שלכל

$x_0 \neq x \in (x_0-\varepsilon, x_0+\varepsilon)$ מתקיים כי

$$(x-x_0)^T H(x_0)(x-x_0) + o(\|x-x_0\|^3) \leq 0$$

לכן

$$f(x) = f(x_0) + \underbrace{Df(x_0)(x-x_0)}_{=0} + \underbrace{(x-x_0)^T \cdot H(x_0) \cdot (x-x_0) + o(\|x-x_0\|^3)}_{\leq 0} \leq$$

$$\leq f(x_0)$$

נובע כי x_0 נקודת מקסימום לוקאלי של f.

Scanned with CamScanner

3) נניח כי הם נערכת לתחום כווני מסויים (0°-360°)
נשאיר הגדרת loss רק נשנה את המעגליות של הכווני
כך אותה $Loss(2,360) = loss(0,2)$

Pseudo-code:

```
def loss(pred_deg, true_deg):

    rad_pred_deg = 2π·(pred_deg/180)
    rad_true_deg = 2π·(true_deg/180)

    while rad_pred_deg < 0:
        rad_pred_deg += 2π

    while rad_true_deg < 0:
        rad_true_deg += 2π

    difference = |rad_true_deg - rad_pred_deg|
    result = min{difference, 2π - difference}
    return result
```

4) נגדיר את הפונקציה הבאה:

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

$\boxed{\text{מניח שהתמונה של } f \text{ היא } \mathbb{R}:}$

נגדיר את הפונקציה $g:\mathbb{R}^3\to\mathbb{R}^3$ ע"י $g(x,y,z)=(x+y,2x,z)$

אז למעשה אנו מחפשים לחשב את $\frac{\partial}{\partial x} f\circ g(x,y,z)$

מתקיים לפי כלל השרשרת כי

$$D_{f\circ g}(x,y,z) = D_f(g(x,y,z))\cdot D_g(x,y,z) =$$

$$= D_f(x+y,2x,z)\cdot\begin{pmatrix}1,&1,&0\\2,&0,&0\\0,&0,&1\end{pmatrix} = \nabla_f(x+y,2x,z)^T\begin{pmatrix}1,&1,&0\\2,&0,&0\\0,&0,&1\end{pmatrix}$$

לכן נקבל כי:

$$\frac{\partial}{\partial x} f(x+y,2x,z) \underset{\substack{\uparrow\\ \text{כניסה ראשונה}}}{=} \Big(D_{f\circ g}(x,y,z)\Big)_1 = \nabla_f(x+y,2x,z)\cdot\begin{pmatrix}1\\2\\0\end{pmatrix}$$

b) $f_1(f_2(\dots f_n(x)))$

$\boxed{\text{מניח כי לכל } 1\le i\le n \quad f_i:\mathbb{R}^n\to\mathbb{R}^n}$

לפי כלל השרשרת:

$$D_{f_1\circ\dots\circ f_n}(x) = D_{f_1\circ\dots\circ f_{n-1}}(f_n(x))\cdot D_{f_n}(x) =$$

$$= D_{f_1\circ\dots\circ f_{n-2}}(f_{n-1}\circ f_n(x))\cdot D_{f_{n-1}}(f_n(x))\cdot D_{f_n}(x) =$$

$$= D_{f_1\circ\dots\circ f_{n-3}}(f_{n-2}\circ f_{n-1}\circ f_n(x))\cdot D_{f_{n-2}}(f_{n-1}\circ f_n(x))\cdot D_{f_{n-1}}(f_n(x))\cdot D_{f_n}(x) =$$

$$= \dots = D_{f_1}(f_2\circ\dots\circ f_n(x))\cdot D_{f_2}(f_3\circ\dots\circ f_n(x))\cdot\dots\cdot D_{f_n}(x) =$$

$$= \prod_{i=1}^{n} D_{f_i}(f_{i+1}\circ\dots\circ f_n(x))$$

c) $f_1(x, f_2(x, f_3(\dots f_{n-1}(x, f_n(x))))$

נניח כי $f_n : \mathbb{R} \to \mathbb{R}$

ולכל $1 \le i \le n-1$ $f_i : \mathbb{R}^2 \to \mathbb{R}$

נגדיר $g_n(x) = (x, f_n(x))$:

$$f_{n-1}(x, f_n(x)) = f_{n-1} \circ g_n(x)$$

לכן:

$$\left(f_{n-1} \circ g_n(x)\right)' = D_{f_{n-1}}(g_n(x)) \cdot D_{g_n}(x) =$$

$$= \nabla f_{n-1}(x, f_n(x))^T \cdot \begin{pmatrix} 1 \\ f_n'(x) \end{pmatrix}$$

נגדיר $g_{n-1}(x) = (x, f_{n-1} \circ g_n(x))$:

$$f_{n-2}(x, f_{n-1}(x, f_n(x))) = f_{n-2} \circ g_{n-1}(x)$$

לכן:

$$\left(f_{n-2} \circ g_{n-1}(x)\right)' = D_{f_{n-2}}(g_{n-1}(x)) \cdot D_{g_{n-1}}(x) =$$

$$= \nabla f_{n-2}(x, f_{n-1} \circ g_n(x))^T \cdot \begin{pmatrix} 1 \\ (f_{n-1} \circ g_n(x))' \end{pmatrix}$$

נגדיר $g_{n-2}(x) = (x, f_{n-2} \circ g_{n-1}(x))$

$$f_{n-3}(x, f_{n-2}(x, f_{n-1}(x, f_n(x)))) = f_{n-3} \circ g_{n-2}(x)$$

לכן

$$\left(f_{n-3} \circ g_{n-2}(x)\right)' = D_{f_{n-3}}(g_{n-2}(x)) \cdot D_{g_{n-2}}(x) =$$

$$= \nabla f_{n-3}(x, f_{n-2} \circ g_{n-1}(x))^T \cdot \begin{pmatrix} 1 \\ (f_{n-2} \circ g_{n-1}(x))' \end{pmatrix}$$

נמשיך באופן איטרטיבי ולבסוף עבור

נקבל כי $g_2(x) = (x, f_2 \circ g_3(x))$

$$f_1(x, f_2(x, f_3(\dots f_{n-1}(x, f_n(x)))) = f_1 \circ g_2(x)$$

לכן

$$(f_1 \circ g_2(x))' = D_{f_1}(g_2(x)) \cdot D_{g_2}(x) =$$

$$= \nabla_{f_1}(x, f_2 \circ g_3(x))^T \cdot \begin{pmatrix} 1 \\ (f_2 \circ g_3(x))' \end{pmatrix} //$$

עבור כל $3 \le i \le n$:

$$f_{i-1} \circ g_i(x) = \begin{cases} f_{i-1}(x, f_i \circ g_{i+1}(x)), & \text{if } i \le n-1 \\ f_{n-1}(x, f_n(x)), & \text{if } i = n \end{cases}$$

$$(f_{i-1} \circ g_i(x))' = \begin{cases} \nabla_{f_1}(x, f_i \circ g_{i+1}(x))^T \cdot \begin{pmatrix} 1 \\ (f_i \circ g_{i+1}(x))' \end{pmatrix}, & \text{if } i \le n-1 \\ \nabla_{f_{n-1}}(x, f_n(x))^T \cdot \begin{pmatrix} 1 \\ f_n'(x) \end{pmatrix}, & \text{if } i = n \end{cases}$$

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

אני מניח כי $f,g,h:\mathbb{R}^n\to\mathbb{R}^n$

נסמן $k_1,k_2:\mathbb{R}^n\to\mathbb{R}^n$ כך:

$k_1(x)=x+h(x)$

$k_2(x)=x+g\circ k_1(x)$

אז:

$$D_{f\circ k_2}(x)=D_f(k_2(x))\cdot D_{k_2}(x)=$$

$$=D_f(x+g\circ k_1(x))\cdot\left(I_n+D_{g\circ k_1}(x)\right)=$$

$$=D_f(x+g(x+h(x)))\cdot\left(I_n+D_g(k_1(x))\cdot D_{k_1}(x)\right)=$$

$$=D_f(x+g(x+h(x)))\cdot\left(I_n+D_g(x+h(x))\cdot(I_n+D_h(x))\right)=$$

$$=D_f(x+g(x+h(x)))\cdot\left(I_n+D_g(x+h(x))+D_g(x+h(x))\cdot D_h(x)\right)//$$

אם $n=1$ נקבל כי

$$(f\circ k_2(x))'=f'(x+g(x+h(x)))\cdot(1+g'(x+h(x))+g'(x+h(x))\cdot h'(x))//$$