Practical part – Bar Rousso 203765698

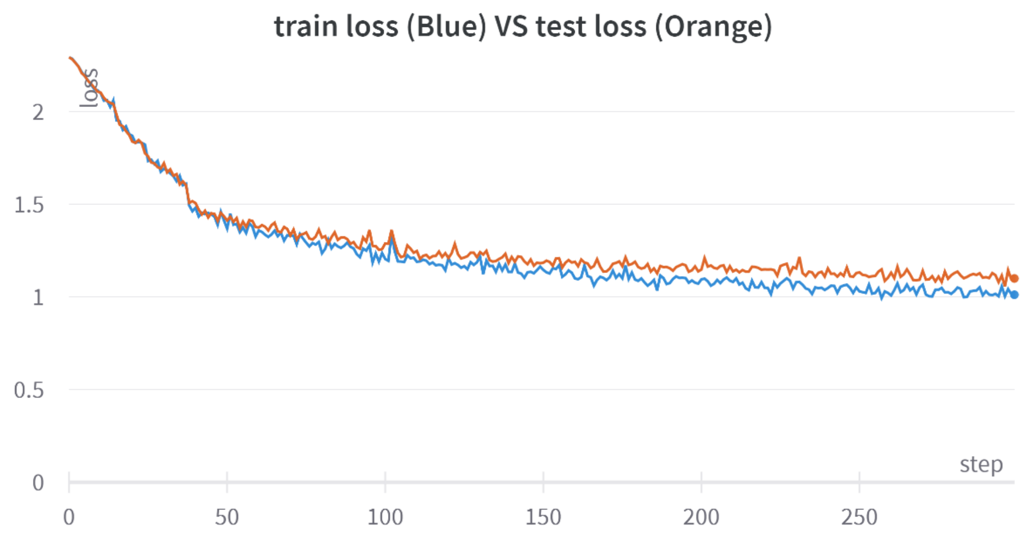
Q1:

Base model (what I started with):

network structure: (bais included)

- Convolution layer: 3 input channels, 6 output channels, 5x5 kernel size
- RELU
- Max pooling: 2x2 Window size
- Convolution layer: 6 input channels, 16 output channels, 5x5 kernel size
- RELU
- Max pooling: 2x2 Window size
- Fully connected layer: 400 input features, 10 output features

Total learnable parameters:
$(5 \cdot 5 \cdot 3 + 1) \cdot 6 + (5 \cdot 5 \cdot 6 + 1) \cdot 16 + \ (16 \cdot 5 \cdot 5 + 1) \cdot 10 = 6882$
Learning rate: 0.001
batch size: 32
Number of epochs: 10



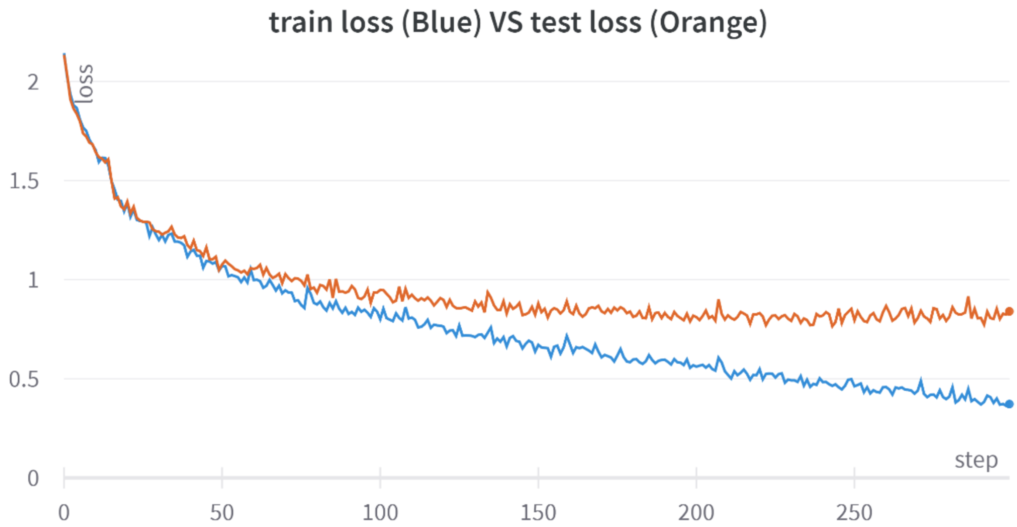
Comment: In this part I only first use the original network from the tutorial and observed the results.

Overfitting model:

network structure: (bais included)

- Convolution layer: 3 input channels, 75 output channels, 5x5 kernel size
- RELU
- Max pooling: 2x2 Window size
- Convolution layer: 75 input channels, 150 output channels, 5x5 kernel size
- RELU
- Max pooling: 2x2 Window size
- Fully connected layer: 3750 input features, 10 output features

Total learnable parameters:
$(5 \cdot 5 \cdot 3 + 1) \cdot 75 + (5 \cdot 5 \cdot 75 + 1) \cdot 150 + \ (150 \cdot 5 \cdot 5 + 1) \cdot 10 = 324{,}610$
Learning rate: 0.001
batch size: 32
Number of epochs: 20



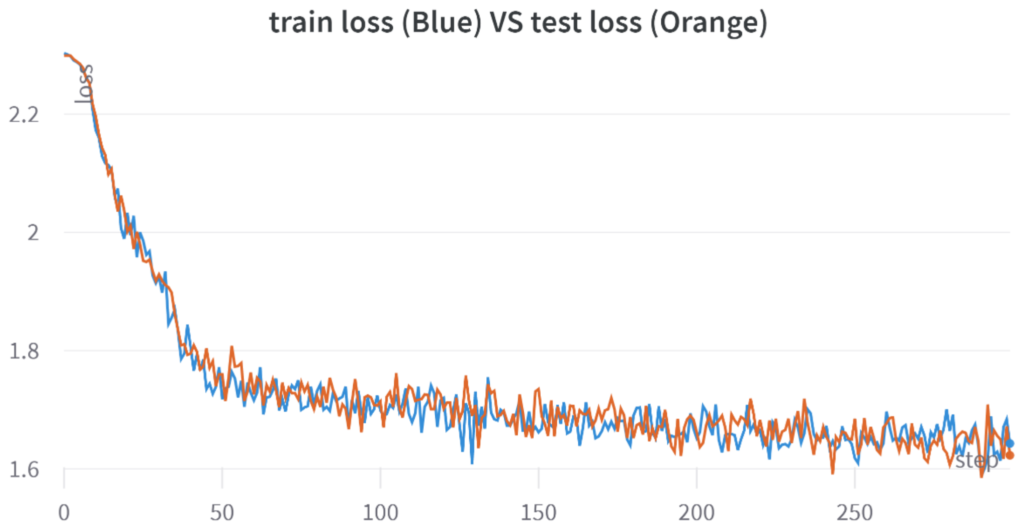
Comment: In order to achieve overfitted model I did:
- Increased the number of learnable parameters to create more complex hypnosis class.
- Increase the number of epochs to make the model "memorize" the training set, by repeatedly going through the training set and keep adjust the model to better match the training results.

Underfitting model:

network structure: (bais included)

- Convolution layer: 3 input channels, 2 output channels, 3x3 kernel size
- RELU
- Max pooling: 2x2 Window size
- Convolution layer: 2 input channels, 2 output channels, 3x3 kernel size
- RELU
- Max pooling: 2x2 Window size
- Fully connected layer: 72 input features, 10 output features

Total learnable parameters:
$(3 \cdot 3 \cdot 3 + 1) \cdot 2 + (3 \cdot 3 \cdot 2 + 1) \cdot 2 + \ (2 \cdot 6 \cdot 6 + 1) \cdot 10 = 824$
Learning rate: 0.01
batch size: 32
Number of epochs: 5



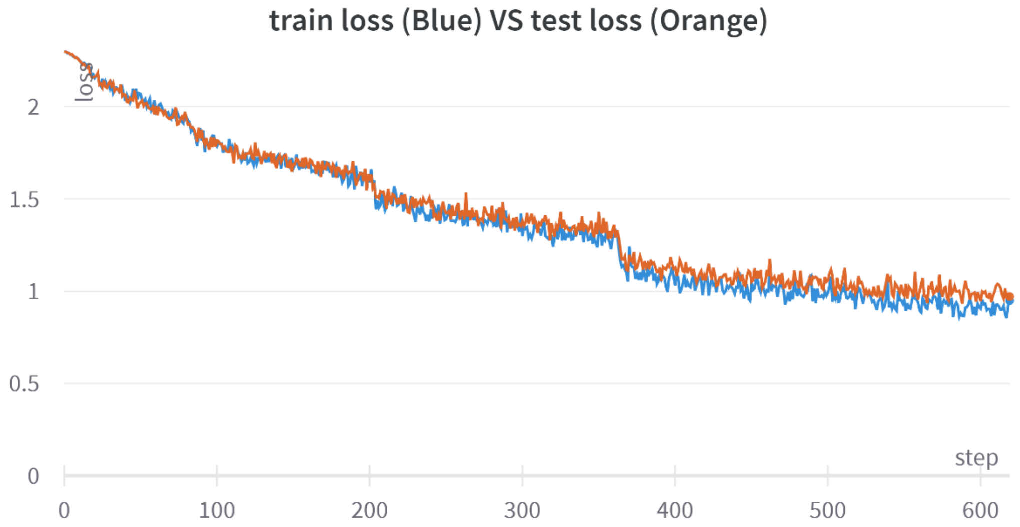
Comment: In order to achieve underfitted model I did:
- Decreased the number of learnable parameters to create more simple hypnosis class.
- Decreased the number of epochs, such that the model didn't go through enough steps to reach to local minimum in SGD process.
- Decreased the learning rate, so the model can miss a local minimum.

Best model (I found):

network structure: (bais included)

- Convolution layer: 3 input channels, 18 output channels, 5x5 kernel size
- RELU
- Max pooling: 2x2 Window size
- Convolution layer: 18 input channels, 36 output channels, 5x5 kernel size
- RELU
- Max pooling: 2x2 Window size
- Fully connected layer: 900 input features, 10 output features

Total learnable parameters:
$(5 \cdot 5 \cdot 3 + 1) \cdot 18 + (5 \cdot 5 \cdot 18 + 1) \cdot 36 + (36 \cdot 5 \cdot 5 + 1) \cdot 10 = 26614$
Learning rate: 0.0005
batch size: 16
Number of epochs: 10

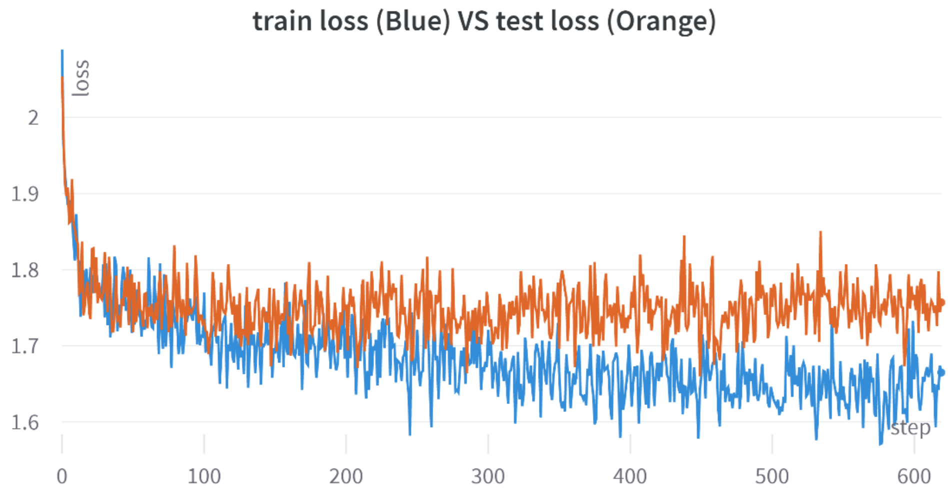


Comment: In order to adjust the model to achieve better results I did:
- Used 10 epochs as a trade off regards to the underfitted and overfitted models.
- Used not too many and not too less parameters, regard to the underfitted and overfitted models.
- Decrease learning rate to not miss a local minimum during SGD process.

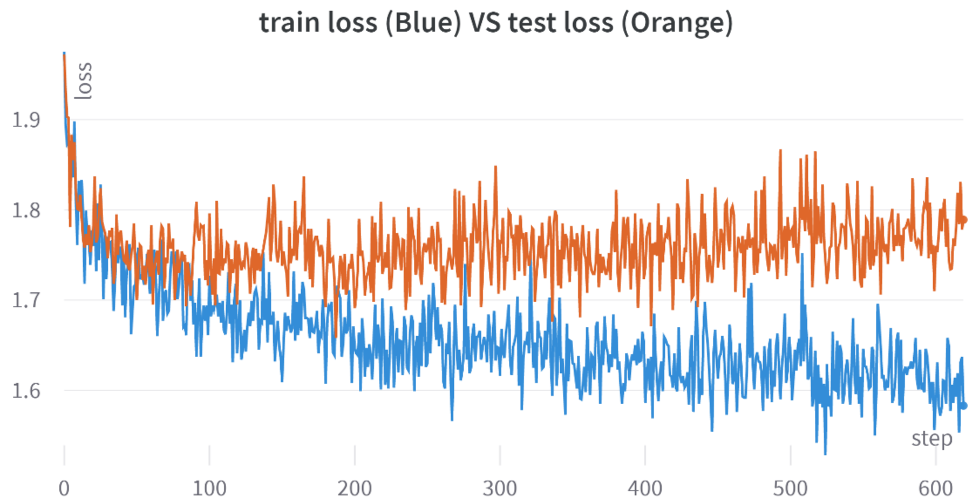
Q2:

The network's non-linear components are RELU and max pool operators.
Removing them will result in the following network structure:

- Convolution layer: 3 input channels, 18 output channels, 5x5 kernel size
- Convolution layer: 18 input channels, 36 output channels, 5x5 kernel size
- Fully connected layer: 20736 input features, 10 output features



Increasing the network's size didn't not changed much:



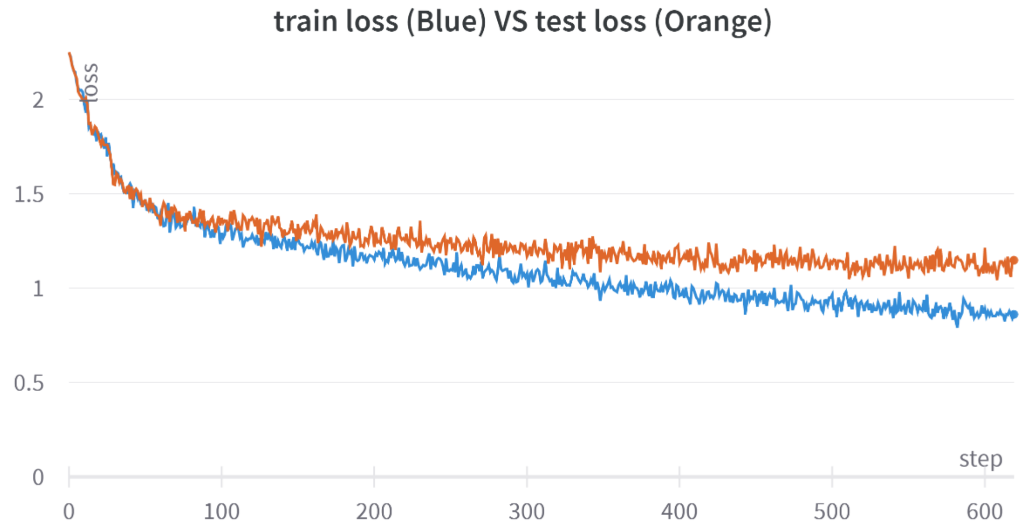
<u>The reason that non-linear components are important:</u>

Supposing that we have a neural network with T linear layers.
Since each layer L_t is linear, we can express its operation as a matrix multiplication W_t
Thus, the whole network can be represented by a **single** linear layer, which represented by the matrix:
$W_T \cdot W_{T-1} \cdot \cdots \cdot W_1$.
For that reason, adding only linear layers will not increase the approximation power of a linear neural network at all.
This is the reason we get underfitting model with poor results.

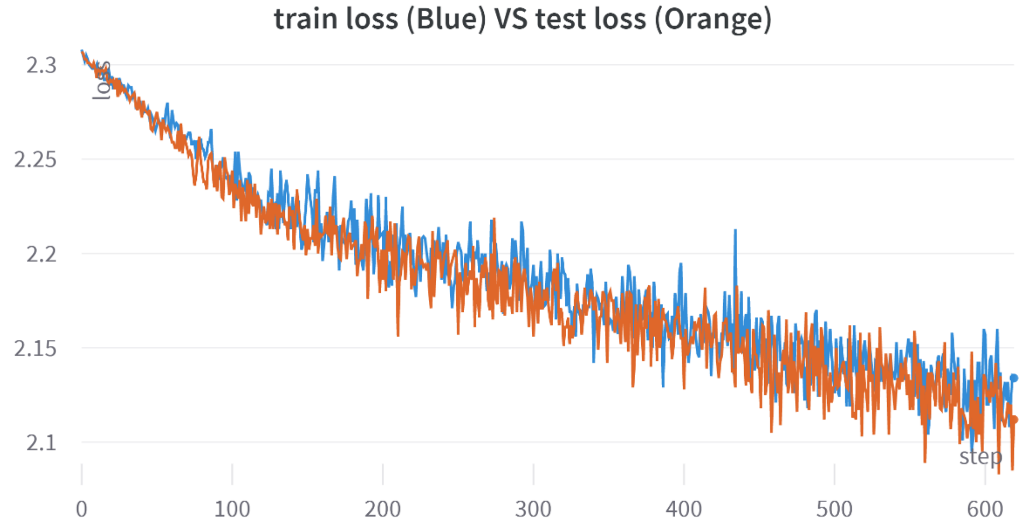
Q3:

With the current network architecture (from Q1) this are the plots I got for each option:

Option1: A FC layer on the activations of the first conv layer



Option2: A “global average pooling” operator that compute the average of the activations in each channel and then apply FC layer.



We can see that we get better results on the first option, so we choose **option number 1.**

This makes sense, because when we use the global average pooling, we basically only consider only the average value on each channel to make a further decision about the image subject.
But since many images with subjects of different classes can share similar average values on each channel, we can expect to many misclassifications.

However, after I adapted the network architecture with option1, the new network didn’t perform better compare to the original network from Q1.

This is makes sense, because in the new network, the receptive field of each neuron in the output layer is **smaller** compare to the original network.
Thus, the new module will be difficult in recognize basic shapes that spread in a large area of the image, which can give a useful information regards to the image's subject classification.

חלק ג' מעורב - קרן מזור ת.ז. 203765698.

1) צ"ל: מתקיים כי $(Lx_{i+k})_j = (Lx_i)_{j+k}$ אינווריאנטי להזזה.

עבור אופרטור L המוגדר וקטור סיגנל $x \in \mathbb{R}^n$ עם

$k \in \mathbb{N}$, ומתאים לקונבולוציה.

הוכחה: יהי $w \in \mathbb{R}^m$ וקטור המשקלים שאנו מפעילים איתו קונבולוציה.

מתקיים כי עבור שכבת הפלט Lx_{i+k} כי

$$(Lx_{i+k})_j = \sum_{a=1}^{m} (x_{i+k})_{j-a \pmod n} \cdot (w)_a \underset{(*)}{=} \sum_{a=1}^{m} (x_i)_{j+k-a \pmod n} \cdot (w)_a = (**)$$

(*) מכיון ש x_{i+k} זה וקטור הזזה של x_i ב k אינדקסים נקבל כי לכל $1 \le j \le n$ $(x_{i+k})_j = (x_i)_{j+k \pmod n}$

לכן

$$(Lx_{i+k})_j = (**) \underset{\text{מהגדרת הקונבולוציה}}{=} (Lx_i)_{j+k \pmod n}$$

לכן נקבל כי שכבת קונבולוציה L אינווריאנטי להזזה. מ.ש.ל.

2) עבור שכבה $FC: \mathbb{R}^n \to \mathbb{R}^m$ ווקטור קלט V_{in}

האם חשוב הסדר של הקואורדינטות של V_{in}?

<u>תשובה:</u> הסדר <u>לא</u> משנה.

<u>הסבר:</u> ניתן לייצג את הפעולה הלינארית בשכבה FC

כמטריצה $W \in \mathbb{R}^{m \times n}$

נניח כי יש לנו רשת נוירונים עם רשת FC ומשקולות

מטריצה $(W_1 \dots W_n) = W \in \mathbb{R}^{m \times n}$

כעת תהי פרמוטציה σ על הקבוצה $[n] = \{1, \dots, n\}$

ונפעיל את הרשת כך שהשכבה FC תקבל

כקלט את $\sigma(V_{in})$. כלומר לכל $1 \le j \le n$: $(\sigma(V_{in}))_j = (V_{in})_{\sigma^{-1}(j)}$.

אז מכיוון שהשכבה FC <u>מחברת</u> כל נוירון לכל נוירון בווקטור הקלט (מחובר לכל הנוירונים בווקטור הפלט),

<u>מתקיים</u> כי אם נאמן את הרשת המעודכנת ונלמד עבור שכבת ה-FC מטריצה חדשה $(W'_1, \dots, W'_n) = W' \in \mathbb{R}^{m \times n}$

אז לכל קואורדינטה $1 \le j \le n$:

$$W'_j = W_{\sigma^{-1}(j)}.$$

לכן לכל ווקטור $V_{in} \in \mathbb{R}^n$ מתקיים לכל $1 \le j \le n$:

$$(W' \cdot \sigma(V_{in}))_j = \langle - \underset{\substack{\uparrow \\ \text{ווקטור שורה } j \text{ ב-} W'}}{W'_j} - , \sigma(V_{in}) \rangle = \sum_{k=1}^{n} (W')_{jk} \cdot (\sigma(V_{in}))_k =$$

$$= \sum_{k=1}^{n} W_{j,\sigma^{-1}(k)} \cdot (V_{in})_{\sigma^{-1}(k)} \underset{\substack{\uparrow \\ \sigma \text{ זו פרמוטציה}}}{=} \sum_{k=1}^{n} W_{jk} \cdot (V_{in})_k =$$

$$= \langle - \overset{\substack{\text{ווקטור שורה } j \text{ ב-} W \\ \wedge}}{W_j} - , V_{in} \rangle = (W \cdot V_{in})_j$$

<u>מסקנה:</u> הפלט משכבת ה-FC בשני המקרים הוא <u>זהה</u>. לכן נובע שאין חשיבות לסדר.

3) נראה אף האופרטורים הבאים הם LTI:

א) <u>ReLU:</u>

כאשר עבור וקטור $V\in\mathbb{R}^n$, לכל $1\le j\le n$ נגדיר

$$\left(\mathrm{ReLU}(V)\right)_j = \max\{0, V_j\}$$

מתקיים כי ReLU אינה פונקציה <u>לינארית</u>

$$\mathrm{ReLU}\begin{pmatrix}2\\2\end{pmatrix}+\mathrm{ReLU}\begin{pmatrix}-1\\-1\end{pmatrix}=\begin{pmatrix}2\\2\end{pmatrix}\neq\begin{pmatrix}1\\1\end{pmatrix}=\mathrm{ReLU}\left(\begin{pmatrix}2\\2\end{pmatrix}+\begin{pmatrix}-1\\-1\end{pmatrix}\right)$$

לכן בפרט G אינו LTI (לינאריות זה תנאי הכרחי)

ב) <u>Strided pooling Layer:</u>

אם עבור קלט מטריצה M בגודל $n\times n$,

$$L(M) = M_{1,n} \text{ (top right)}$$

אם נראה כי L לא LTI, לכן נציג עבור

$$M_1=\begin{pmatrix}3&3&3&3\\0&0&0&0\\1&1&1&1\\2&2&2&2\end{pmatrix}\quad M=\begin{pmatrix}0&0&0&0\\1&1&1&1\\2&2&2&2\\3&3&3&3\end{pmatrix}$$

$$L(M_1)=\begin{pmatrix}3&3\\1&1\end{pmatrix}\quad L(M)=\begin{pmatrix}0&0\\2&2\end{pmatrix}$$

אך $L(M_1)$ לא מזוזה בהזזה 1 (מעלה) של $L(M)$.

ד) addition of a bias:

אם עבור וקטור $V \in \mathbb{R}^n$, ווקטור $0 \neq b \in \mathbb{R}^n$

נגדיר את הטרנספורמציה הנ"ל ב $F(V) = V + b$

ברור כי F אינה לינארית כי $F(0) = b \neq 0$

ולכן בפרט F אינה LTI. (לינאריות זה תנאי הכרחי)

ה) הכפלה עם Fully connected matrix:

אם עבור וקטור $V \in \mathbb{R}^n$ נגדיר את הטרנספורמציה

הנ"ל ב $F(V) = M \cdot V$ עבור $M \in \mathbb{R}^{m \times n}$.

יהי V_k הוקטור הנזז ב-k של V. אז

$$(MV_k)_j = \sum_{t=1}^{n} M_{jt} \cdot (V_k)_t = \sum_{t=1}^{n} M_{jt} \cdot V_{t-k \pmod n}$$

$$\underset{\text{לא בהכרח}}{\neq} \sum_{t=1}^{n} M_{j+k \pmod n, t} \cdot V_t = (MV)_{j+k \pmod n}$$

למשל נקח: $M = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$, $V = \begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}$,

אז $M \cdot V = \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix}$

אבל אם, עבור $k=1$ $V_k = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$!

$M \cdot V_k = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}$

והרי $(MV_k)_2 \neq (MV)_1$

לכן אינה לא בהכ' LTI.