**סיכום מצגות אחרונות בקורס רשתות נוירונים**

**שבוע 13:**

**מצגת Autoencoder**

An **autoencoder** is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning). The encoding is validated and refined by attempting to regenerate the input from the encoding. The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data (“noise”).

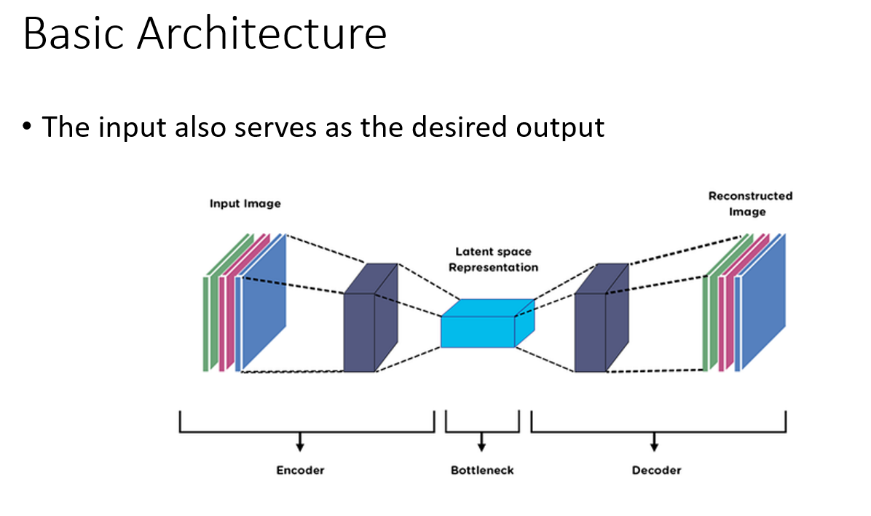
כלומר autoencoder היא רשת נוירונים שמטרתה ללמוד יצוג מקודד של מידע על ידי למידה של מידע מקודד(דחוס) וניסיון לפרוס אותו למקור.

Graphical user interface, text, application

Description automatically generated

בקורס ראינו 4 שימושים עיקריים:

1. Codec-דחיסת מידע ופריסתו
2. הפחת מימדים
3. זיהוי חריגות
4. שימוש ברשתות VAE

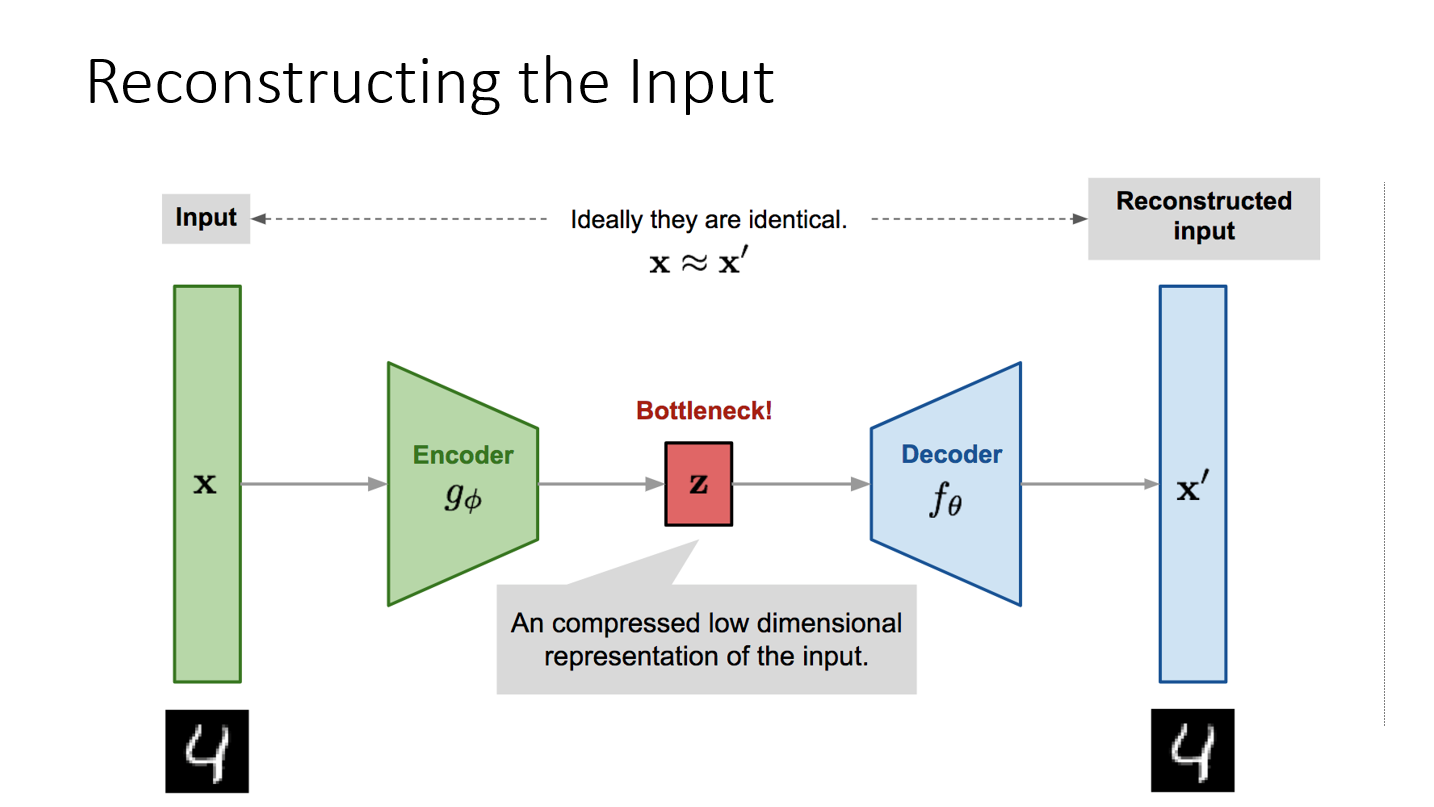




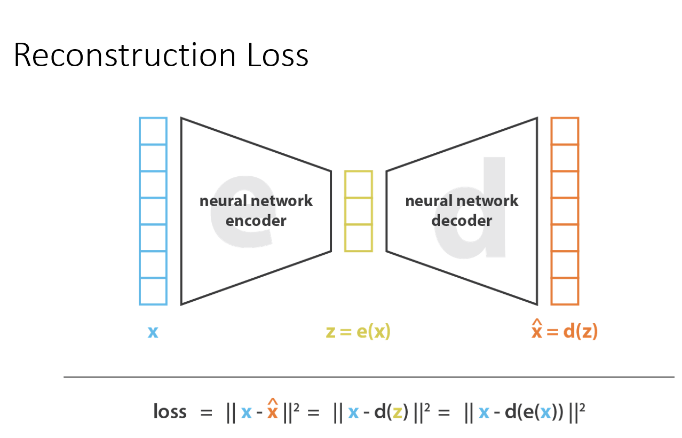
**דוגמא לרשת מסוג זה**

בדוגמא ניתן לראות כי הרשת נחלקת ל3(מקודד,Bottleneck ומפענח)

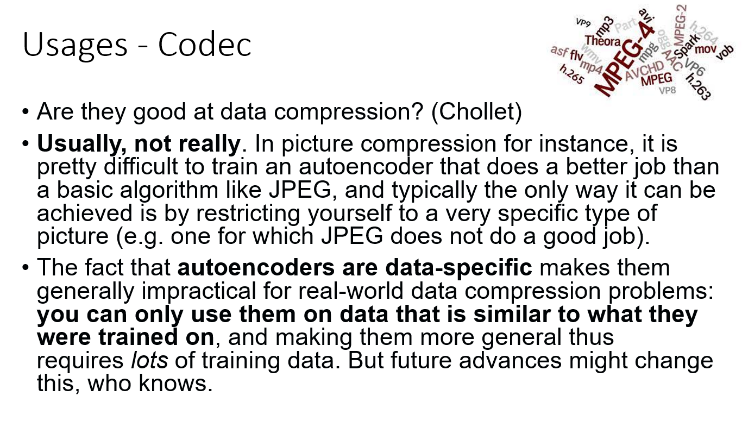
**שחזור הקלט:**

****

כלומר נקבל תמונה נדחוס אותה ואז ננסה לשחזר אותה מן הערכים שיצאו לאחר הדחיסה.

**פונקציית ההפסד של הרשת תהיה Reconstruction Loss כלומר כמות המידע שאיבדנו במהלך הדחיסה והפענוח או כך שx זה הקלט המקורי וd(e(x)) זה הפלט שלנו(מה שקיבלנו לאחר הפענוח של המידע הדחוס)**

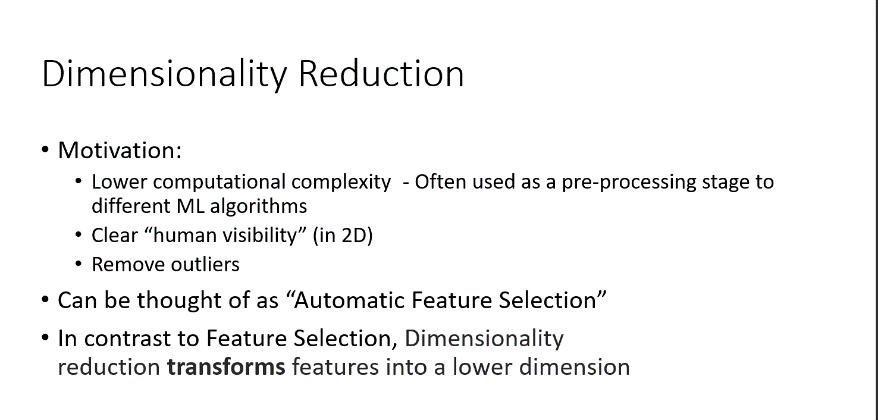
**1.CODEC(Coder/Encoder)**



השימוש הראשון עליו דיברנו הוא CODEC כלומר לדחיסת מידע בכדי להקטין את נפחו ולעזור להעביר את הנתונים בצורה קלה יותר לדוגמא כמו JEPG שהוא אלגוריתם לדחיסה ופריסה של תמונות ובכך מאפשר לנו להוריד קבצי תמונה ששוקלים מעט מאוד. לרוב מדובר כרגע בתחום שאינו ברמה מפותחת כל כך כי יש לנו אלגוריתמים קיימים ברוב התחומים שעושים עבודה טובה ועל מנת להתחרות בהם בעידן הנוכחי נצטרך להגביל את עצמנו מאוד לדברים ספציפיים כלומר אם הרשת ראתה בעיקר תמונות של חתולים היא תעשה עבודה טובה בדחיסת תמונות של חתולים אבל לא תעשה עבודה בדחיסת תמונות חדשות שהיא לא מכירה.

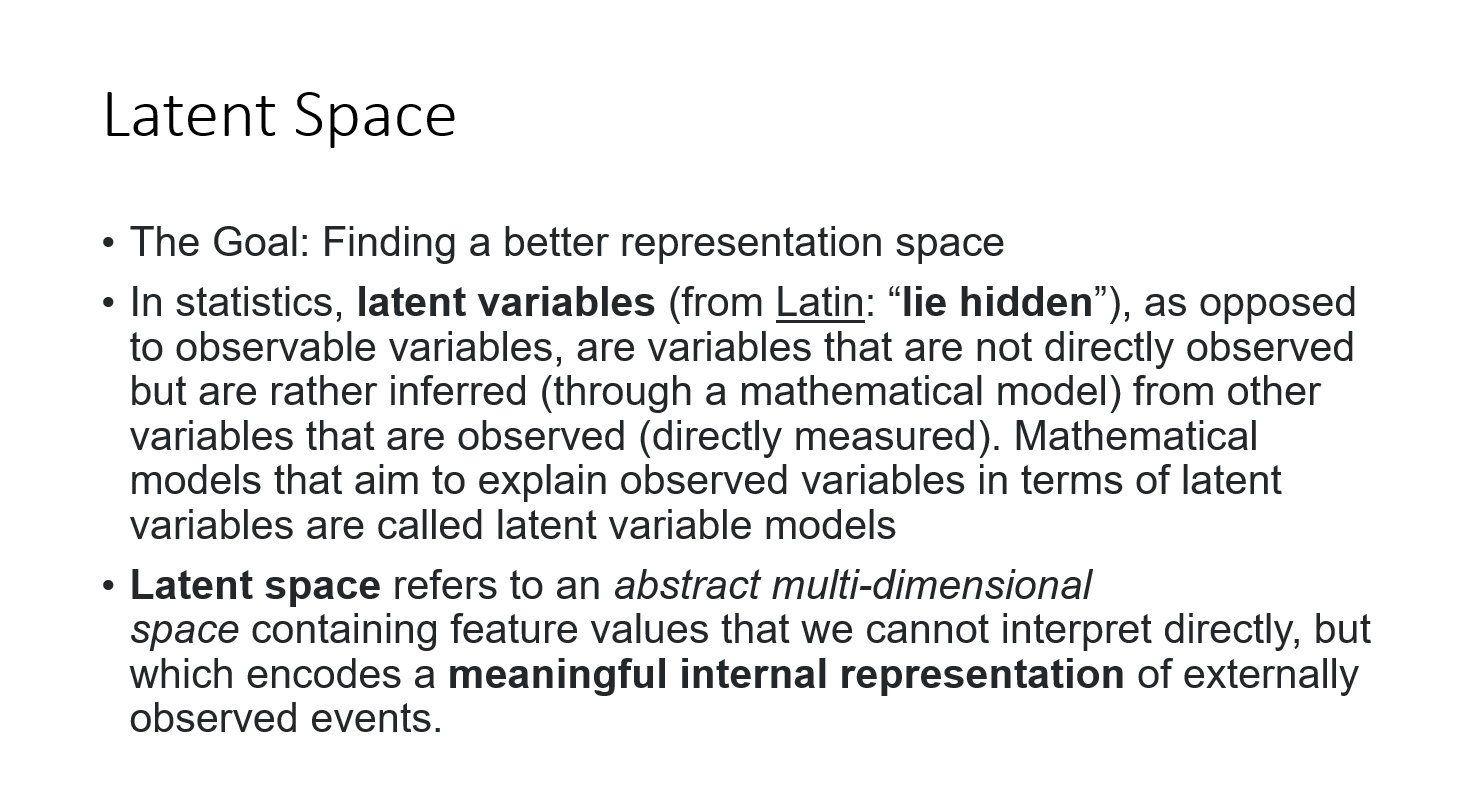
בגלל התכונה הזו שautoencoder הם data-specific כלומר לומדים על דאטה ספציפי הם לא מעשיים לשימוש כCODEC היות והפיכתם לכללים דורשת המון דאטה ולא רק דאטה ספציפי אולי בעתיד המצב ישתפר.

**הפחתת מימדים**



הפחתת מספר המימדים שמייצגים את הבעיה, עוזר בהפחתת המעמסה החישובית, יצוג המידע בצורה שברורה לעין האנושית(לא יכולים לראות יותר מ3 מימדים) ויכול לעזור בהסרת ערכי קיצון.

בניגוד לתהליך הFeature Selection שאנו מכירים כאן אנחנו לא מחסירים פיצ'רים אלא מפחיתים את המשמעות של פיצ'רים מסוימים בכך שאנחנו משלבים אותם עם פיצ'רים אחרים ליצירת מימד משותף.

**Latent Space**

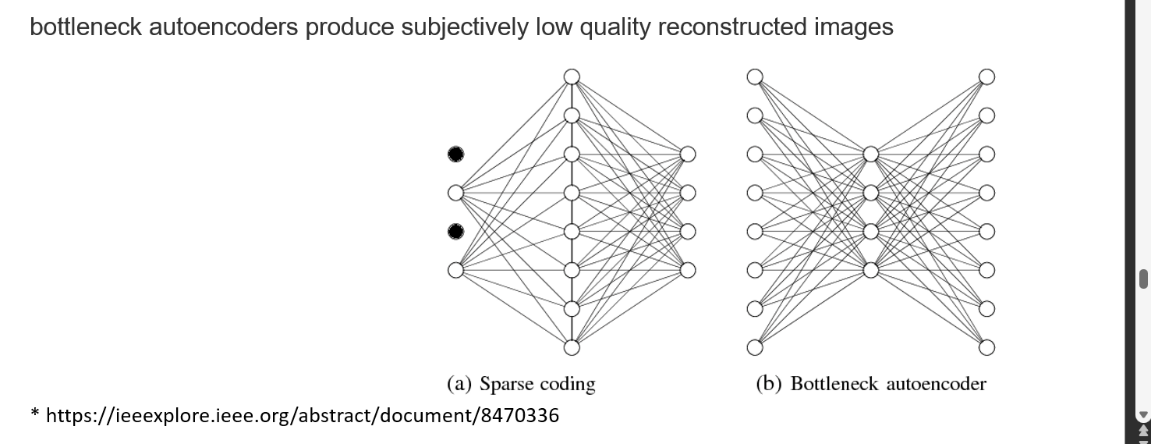
משתנה נסתר – משתנה שבניגוד למשתנים הנצפים הוא לא בהכרח נתון לנו אלא ניתן לנו במרומז באמצעות מודל מתמטי כלשהו על ידי שימוש במשתנים הנתונים.

המרחב הנסתר הוא מרחב אבסטרקטי רב מימדי המכיל פיצ'רים אותם אנחנו לא יכולים לפרש ישירות אבל הם מייצגים ידע ומידע בעל משמעות חשובה מאוד לאירועים המדוברים.

לדוגמא ניתן לחשוב על DNA כמרחב הנסתר של אורגניזמים חיים

או לדוגמא קרני שמש בתהליך הפוטוסינתזה.

**יצירת הLatent space**

יש לנו 2 שיטות ליצירת המרחב הנסתר(Bottleneck וSparse)

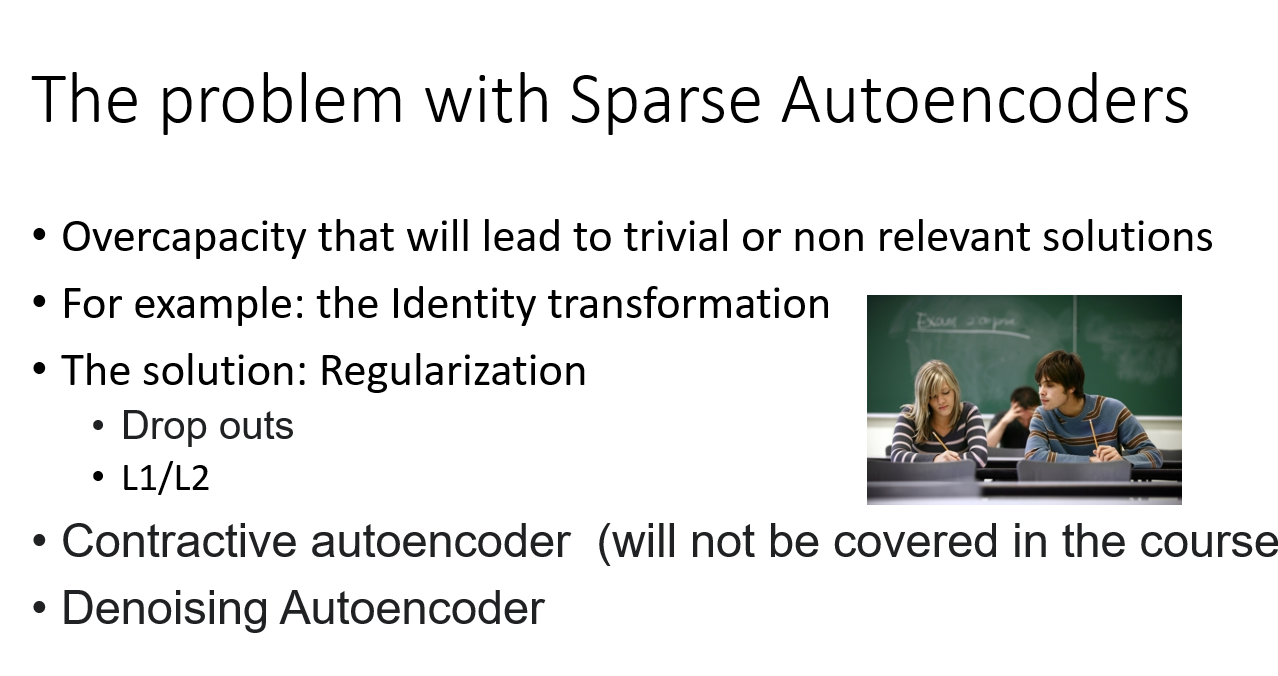
A Sparse Autoencoder is designed to learn a compact representation (or encoding) of the input data that captures the most important features while reducing the dimensionality of the data. The model is trained to produce sparse encodings, meaning that the majority of the activations in the encoding are zero. This helps the model to learn more robust and interpretable features, as it is forced to focus on only a few important activations at each layer.

On the other hand, a Bottleneck Autoencoder is designed to learn a compact representation of the input data that passes through a bottleneck layer, meaning a layer with a much lower number of neurons than the input and output layers. The purpose of the bottleneck layer is to enforce a strong constraint on the capacity of the model, forcing it to learn a compact representation of the input data that is easy to decode. This results in a compact and dense encoding, rather than a sparse encoding as in the case of Sparse Autoencoders.

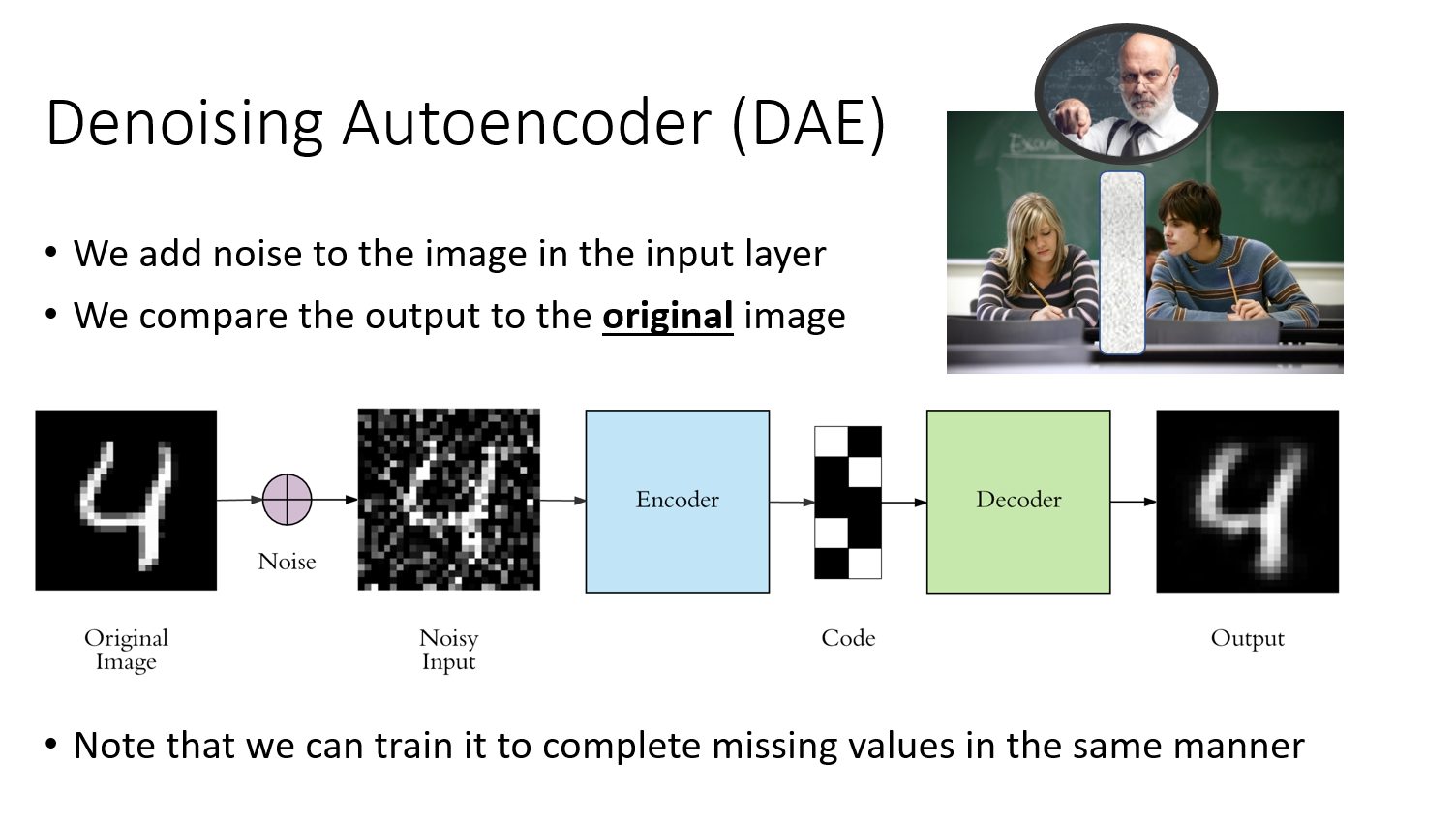
In conclusion, the main difference between Sparse Autoencoders and Bottleneck Autoencoders lies in the type of constraint imposed on the capacity of the model. Sparse Autoencoders use sparsity constraints to force the model to learn more interpretable features, while Bottleneck Autoencoders use capacity constraints to enforce a compact representation of the input data.

כלומר בארכיטקטורה שהיא Sparse אנחנו מנסים לתת לרשת ללמוד את הפיצ'רים החשובים ביותר, אנחנו משתמשים בפונקציות אקטיבציה שרבות מהן מתאפסות ולכן המודל יישאר בסופו של דבר עם הפיצ'רים החשובים.

בארכיטקטורת Bottleneck אנחנו יוצרים צוואר בקבוק כלומר דוחסים את המידע לכמות נוירונים קטנה יותר מהכמות המקורית ובכך מכריחים את הרשת לבחור את הפיצ'רים החשובים ביותר כדי שהיא תוכל לפרוס את המידע לאחר מכן.



**Denoising Autoencoder (DAE)**



כלומר נבצע את מה שביצענו קודם רק שהפעם נוסיף לתמונה רעש לפני שנדחס אותה וננסה לפרוס אותה ולהגיע למקור שקיבלנו ובכך הרשת שלנו תלמד להתמודד עם רעש, להכליל טוב יותר וגם ליצור תמונות בצורה טובה יותר.

**Text Generation**

Graphical user interface, text

Description automatically generated

נשתמש ברשתות מסוג RNN או Transformers על מנת ליצור טקסט, נאמן רשת לחזות את המילה/תו הבא במשפט.

מודל שפה- רשת המסוגלת למדל את ההסתברות של המילה/תא הבא בהינתן המילים/תווים קודמים.

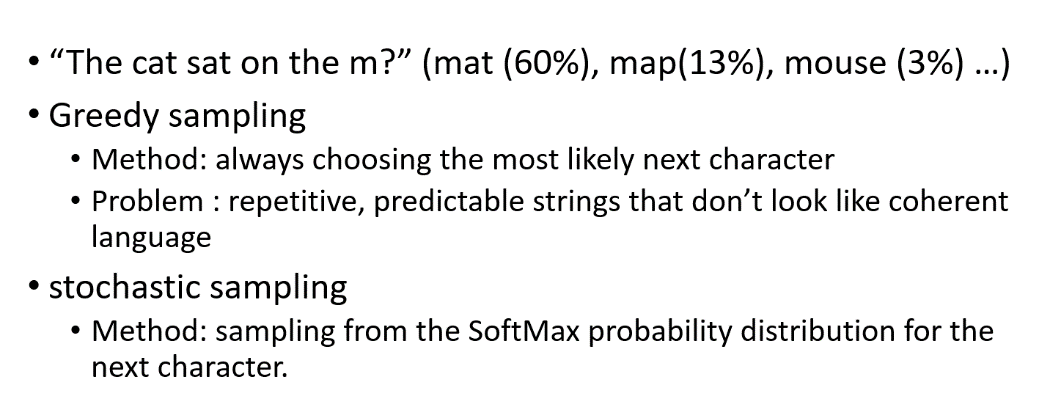
מודל כזה מסוגל להבין את המרחב הנסתר של השפה(המבנה הסטטיסטי שלה)

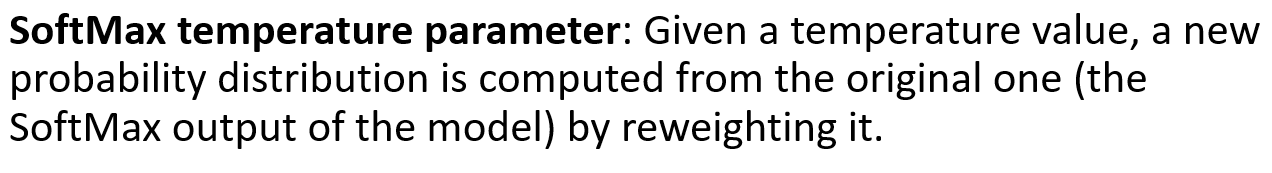
Diagram

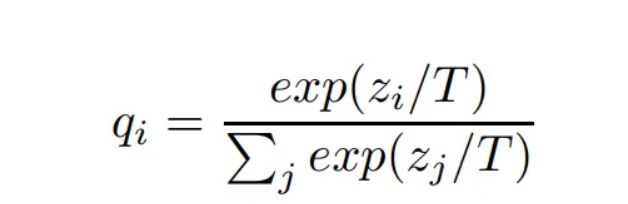
Description automatically generated

ברגע שיש לנו מודל מסוג זה אנחנו יכולים להשתמש בו על מנת לדגום ממנו.

נכניס לו מחרוזת ראשונית ונבקש ממנו ליצור את החלקים הבאים במשפט, נוסיף את המידע שחזינו לקלט ונחזור על התהליך כמה שבא לנו.

**שיטות דגימה**





כלומר ככל שהטמפרטורה גדלה ככה ההתפלגות תהיי שיווניות יותר ונקבל יותר גיוון אבל גם נקבל יותר ג'יבריש ומשפטים לא הגיוניים, יש להתאים את הטמפרטורה בהתאם למטרה שלנו.

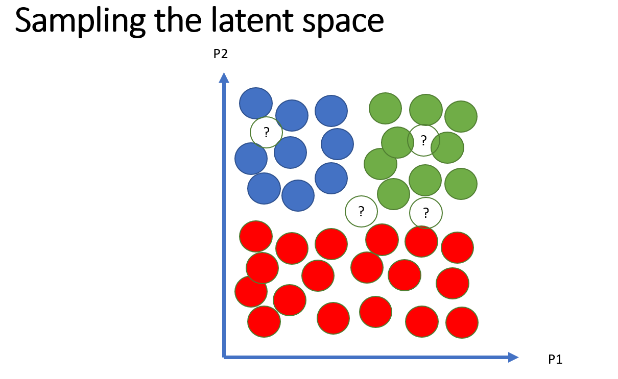
**Generative Adversarial Networks (GAN)**

Diagram

Description automatically generatedרשת נוירונים המורכבת מ2 רשתות שונות(זייפן ושוטר) מטרת הזייפן היא ליצור תמונות שהשוטר יחשוב שהן אמיתיות ומטרת השוטר היא לתפוס את הזייפן ולהגיד שהתמונה מזוייפת.

נפסיק לאמן את הרשת כאשר פונקציית ההפסד שלנו היא 0.5 כלומר הוא לא יכול לדעת האם מדובר בזיוף או אמת ומנחש באופן רנדומלי.

נשים לב כי מדובר במשחק סכום אפס בעצם בין הזייפן לשוטר וכל אחד רוצה להצליח ולכן פונקציית ההפסד שלהם משותפת, הזייפן רוצה לצמצמם אותה והשוטר רוצה למקסם אותה כי השפעת השוטר היא בלוג שבו הוא רוצה להיות כמה שיותר חיובי(כדי לנחש נכון) והזייפן רוצה להיות כמה שיותר קרוב ל0(כך שהשוטר לא ניחש נכון).

**Variational Autoencoders (VAE)**

Chart, bubble chart

Description automatically generated

כאן אנחנו רואים דוגמה למרחב נסתר כאשר אחד הוא מסודר יחסית ורציף והשני מבולגן ובעייתי.

Chart, diagram

Description automatically generated

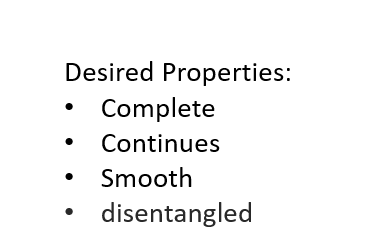
כלומר אנחנו רואים כי שימוש בAutoencoders רגילים יכול להיות בעייתי וליצור לנו מרחב נסתר לא בהכרח טוב.

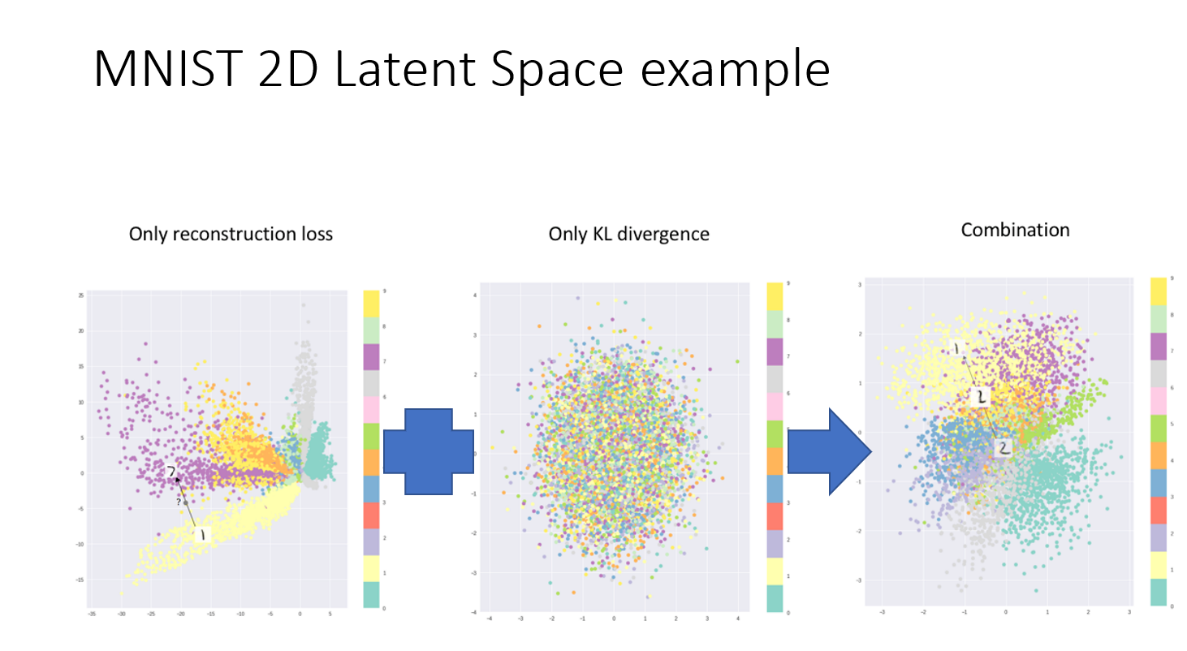
Autoencoders try to learn a compact representation of the input data, but there is no guarantee that this representation will be meaningful or that it will capture the underlying structure of the data. This can lead to suboptimal or even random representations.

VAEs, on the other hand, impose a probabilistic structure on the latent space, typically by assuming that the latent variables are sampled from a simple prior distribution, such as a normal distribution. This allows VAEs to learn a more meaningful and structured latent space that can be used for a variety of tasks, such as generative modeling, data visualization, and representation learning.

**the autoencoder is solely trained to encode and decode with as few loss as possible, no matter how the latent space is organized**

כלומר בעזרת Autoencoder אנו מצפים ללמוד יצוג קומפקטי של המידע שלנו אבל אין לנו הכרח לדעת כי היצוג שלמדנו הוא בעל משמעות או שהוא אכן תופס את המהות הנסתרת של המידע ולכן זה יכול להוביל ליצוג לא טוב או אפילו רנדומלי.

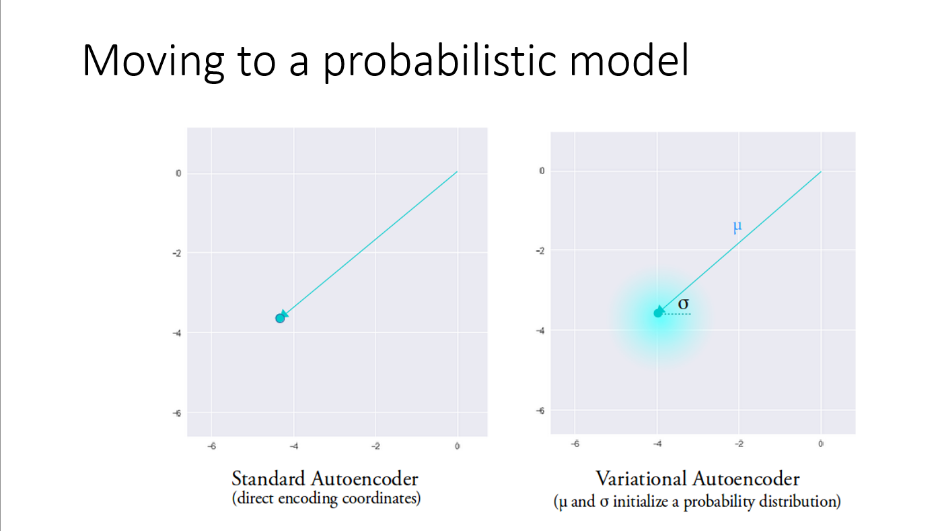
בVAE אנחנו משתמשים במודל הסתברותי לבנות את המרחב הנסתר ובכך אנחנו יכולים ליצור מרחב נסתר טוב יותר.



לדוגמא המרחב הנסתר של MNIST 2D מורכב מהReconstruction loss שזה ההפרש בין התמונה המקורית לתמונה ששוחזרה

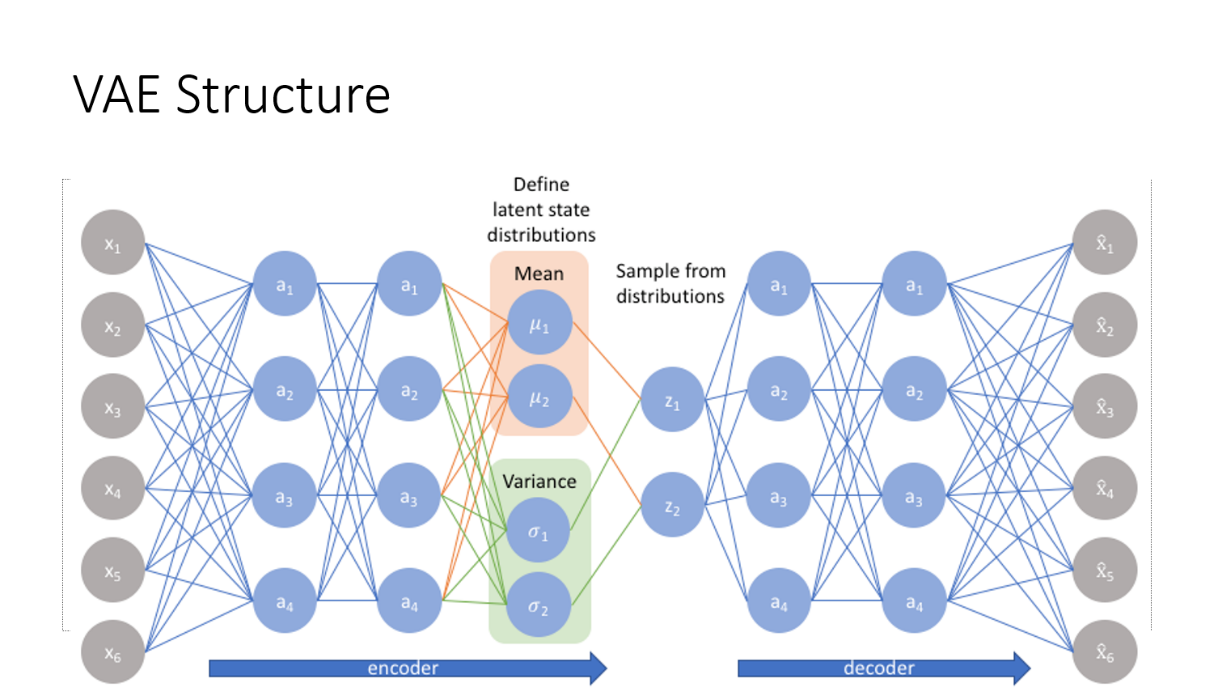
KL-Divergence ההפרש בין 2 התפלגויות, במקרה שלנו ההתפלגות של המרחב הנסתר שלנו וההתפלגות הפריורית(ההתפלגות שלדעתנו מתארת את המידע שלנו בצורה הטובה ביותר)

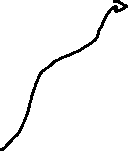
**How do we regularize the latent space?**



בניגוד לautoencoder רגיל בו אנו יוצרים וקטור אחיד לכל encoder בVAE אנחנו יוצרים התפלגות כלומר אנחנו יוצרים מרחב נסתר שבו לכל מימד יש התפלגות נורמלית משלו ויש לנו 2 וקטורים שאחד מהם מכילים לכל מימד במרחב את הממוצע שלו והשני מכיל לכל מימד במרחב את סטיית התקן שלו.

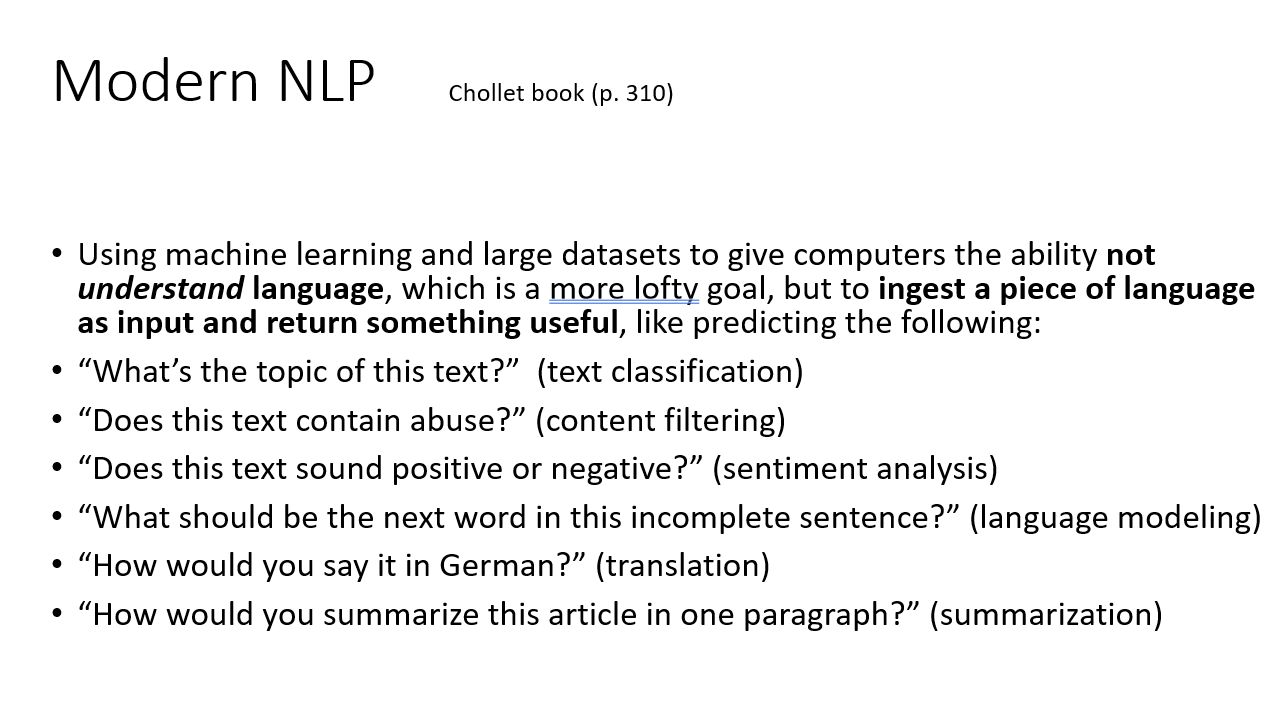
במידה ונרצה ליצור דוגמא חדשה נוכל פשוט לדגום מהמרחב הנסתר את כל הממוצעים בתוספת רעש מסוים ובכך נקבל דוגמא חדשה או במידה ונרצה לשנות מידע קיים נקודד אותו ואז נוסיף לו מידע מדגימת המרחב הנסתר.

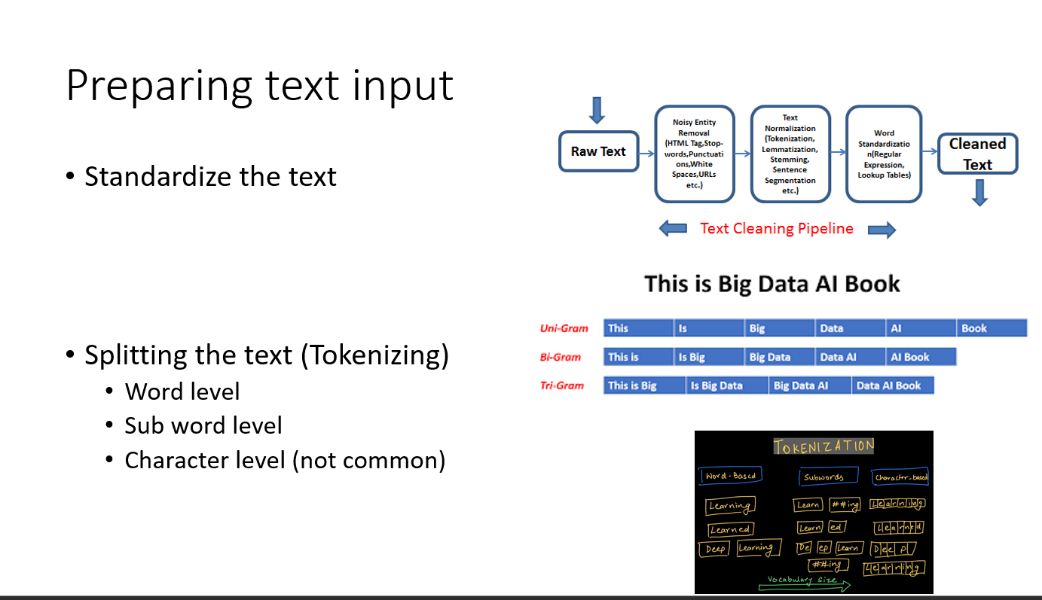


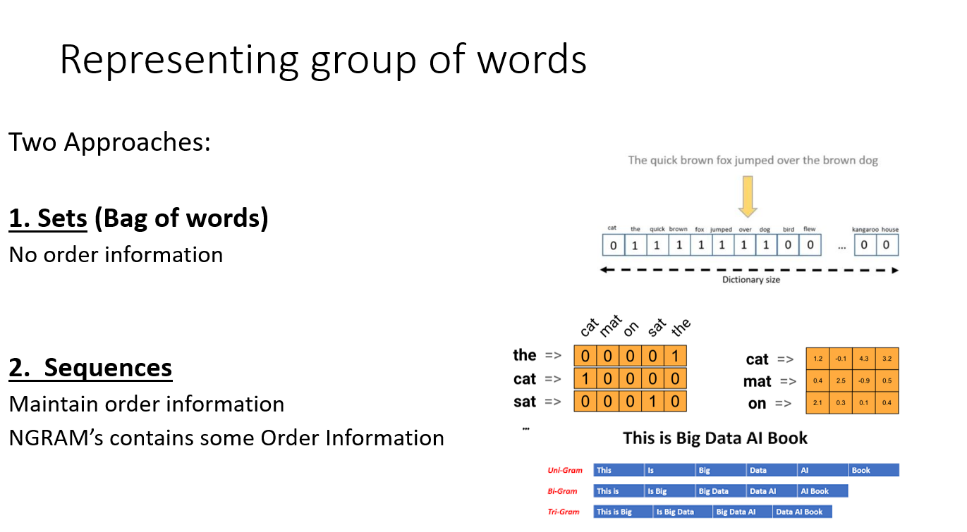


**פונקציית הפסד**

**Transformers**

רשת נוירונים מיוחדת, משנת 2018 הרשת הנפוצה ביותר בתחום הNLP

**שלב 1 הכנסת הטקסט**

נפרק את המשפט לפי הרמה שמתאימה לנו(מילים בודדות, מספר מילים, חלק ממילים וכו'...)

**Attention**

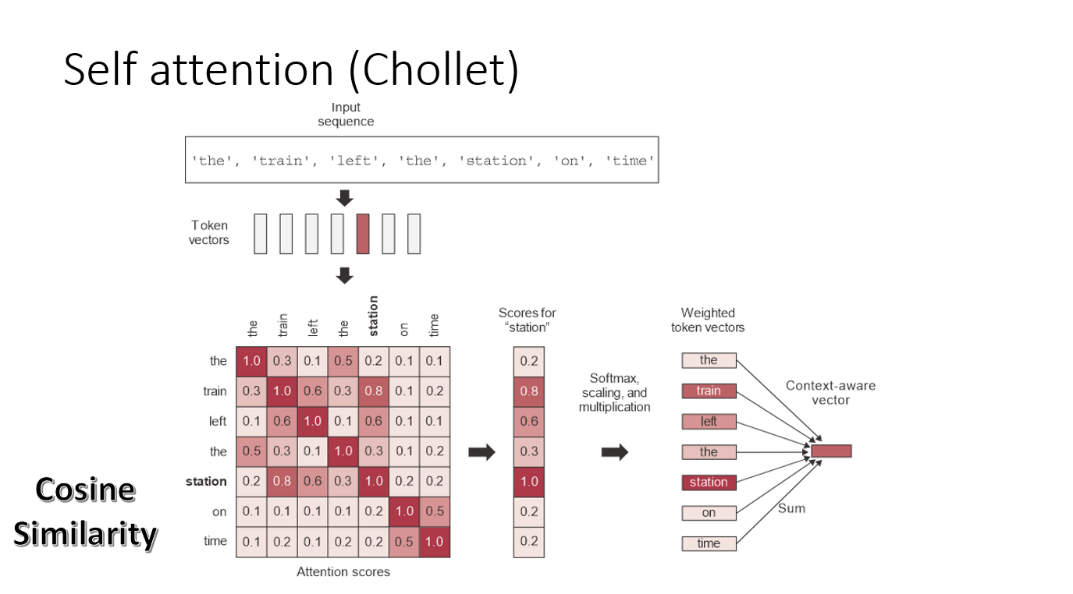
מנגנון המאפשר לנו להבין הקשר בצורה טובה יותר, בדומה לחשיבה אנושית נשים דגש על החלקים שאנו חושבים שהם חשובים להבנת המשפט וכך יהיה לנו קל יותר לדעת מה קורה.

לדוגמא I love playing soccer, but it's too hot outside במשפט שלנו ניתן להגיד כי המילים love וhot

הן המילים הכי חשובות ולכן יקבלו את ציון הattention הגבוה ביותר והן יהיו המפתח להסקת מסקנות והוצאת מידע חשוב על המשפט.

Chart, histogram

Description automatically generated

כלומר נוסיף למשקל את המידע ההקשרי.

Diagram, schematic

Description automatically generated

ברשתות מסוג Transformer אנחנו משתמשים במה שנקרא Attention head והוא מורכב מ3 חלקים

K,Q,V

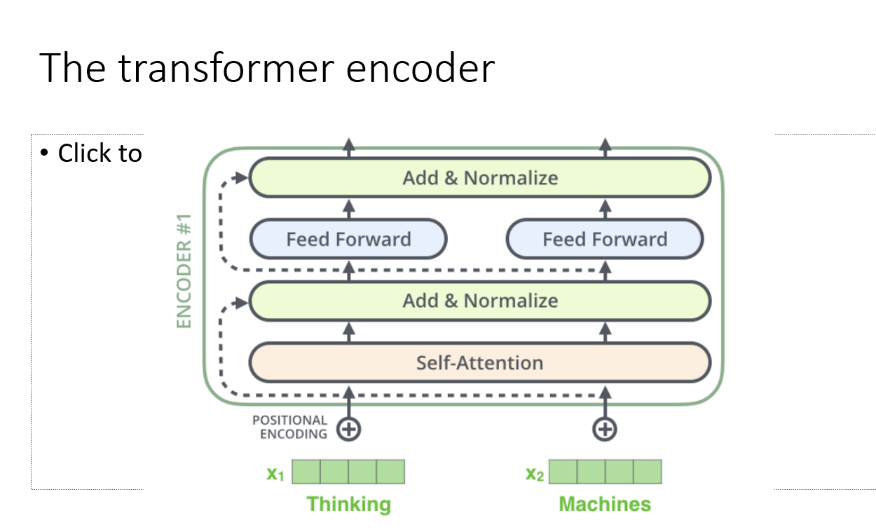
אנחנו לוקחים את המטריצה שלנו ומכניסים אותה לV,Q,K לאחר מכן מעבירים כל אחד מהם בשכבה לינארית עם משקולות שונים.

מבצעים הכפלה בין הQ והK לאחר השינוי, מנרמלים את התוצאה(חלוקה בשורש הגודל של הוקטור) מעבירים אותה בSoftmax כדי לקבל ציונים בין 0 ל1 וכך נוצר לנו Attention filter שאותו נכפול במטריצה שיצאה לנו בValue וזה סופו של הראש הספציפי אבל משתמשים ביותר מראש אחד ולכן בשלב הסופי נחבר את המטריצות שיצאו מכל הראשים ולאחר מכן נעביר אותם בשכבה לינארית נוספת כדי לצמצם את גודל הוקטור הסופי לגודל הוקטור המקורי שקיבלנו כדי שנוכל לעבוד אתו וזה יהיה וקטור הAttention שלנו.

**Positional encoding**

Graphical user interface, website

Description automatically generatedדבר נוסף שיש לקחת בחשבון הוא המיקום של המילה במשפט שכן הוא משפיע על משמעותה ובניגוד לRNN שמקבל את המידע בצורה סדרתית בTransformer כל המידע מגיע במכה אחת אז כיצד נדע מה היה המיקום של כל מילה? לכל וקטור של מילה נוסיף גם וקטור מיקום



כלומר אנחנו מוסיפים את הוקטור שיצרנו בMulti Head לקלט שקיבלנו ומבצעים נרמול עליו ולהמשיך להשתמש בו,נשים לב שזה כמו Resnet עם חיבורים שקופצים בין שכבות.

אנחנו עורמים Encoder כך שכל Encoder יכול ללמוד צורה שונה של Attention בהתאם למשקולות בMulti head שלו וכך יכול לשפר את החיזוי

