

Gen AI Architectures



תוכן עניינים

6.....	AutoEncoders
6.....	שימושים
7.....	תיאור הארכיטקטורה
7.....	פורמלית
9.....	דוגמה לשימוש – Denoising AutoEncoder
10.....	דוגמה לארQUITקטורה
11.....	Probabilistic AutoEncoders
12.....	Probabilistic AutoEncoders Based Normal Distribution Example
12.....	המודוביציה לשימוש ב敦oders
13.....	Variational AutoEncoder – VAE
13.....	הסבר הVAE בעJECTIVE
14.....	פירוט על המשמעות של קירוב התפלגיות
15.....	המשמעות של Maximum Likelihood Estimation
16.....	Maximization of Maximum Log Likelihood in VAE
17.....	פירוט על KL Divergence
18.....	VAE Likelihood
19.....	בעיות בחישוב ההסתברות בהגדרת OBJECTIVE
20.....	בעית חישוב הגראדיינט ו- Reparameterization Trick
21.....	Generative Adversarial Network – GAN
22.....	אופן פעולה הGenerator
23.....	על אופן פעולה GAN לאורך האיטרציות Overview
24.....	Discriminator Loss Function and Objective
25.....	משמעות גרפית של ה-Discriminator Objective
25.....	Generator Loss Function and Objective
26.....	GAN Objective
27.....	Optimal Discriminator
31.....	Optimal Generator
34.....	JSD (Jensen Shannon Divergence)
35.....	Transformers
36.....	ארQUITקטורת ה-Transformer
37.....	Transformer Overview
38.....	Model's Input

38Input Embedding
39Positional Encoding
40Encoder
41Self-Attention
42Query
43Key
44הגדרת ה-Attention
45Value
47Multi Head Attention
48Add & Norm
49Feed Forward Neural Network (FFNN)
50Decoder
51 אלגוריתם הרצה נומרית עבור <i>Inference</i>
51Inputs
51Inputs Embedding
51Positional Encoding
53Encoder 1
53Attention Head 1
56Add & Norm 1
57Feed Forward Neural Network 1
59Encoder 2
60Decoder 1
60Masked Multi Head Attention 1
63Add & Norm 1 - Decoder
63Multi Head Attention 1
66Add & Norm 2 - Decoder
67Feed Forward Neural Network 1 - Decoder
68Decoder 2
69Linear Layer
69SoftMax
70 ארQUITטורות מבוססות טרנספורמר
70 BERT
71Pre-Trained Methods
71Masked Language Model (MLM)
72Next Sentence Prediction (NSP)
73Fine-Tune Methods
73Text Classification
73Question Answering
74 T5 – Transfer Text To Text Transformer
75 TFT (Temporal Fusion Transformer)
76 ערק הפרדיקציה ב-TFT

77	דוגמא נומרית ל-TFT
78	Model Architecture
79	GRN (Gated Residual Network)
80	דוגמת הרצה נומרית ל-GRN
82	GLU (Gated Linear Unit)
83	VSN (Variable Selection Network)
84	דוגמת הרצה נומרית ל-VSN
85	SCE (Static Covariate Encoders)
86	LSTM Encoder/Decoder
88	Temporal Fusion Decoder
88	Static Enrichment
89	Temporal Self Attention
90	Position Wise Feed Forward
91	– Quantile and Quantile Loss – הסבר כללי
92	TFT – Quantile and Quantile Loss
93	Vision Transformer – ViT
94	ViT – Architecture Overview
95	Images To Patches
95	Patches to Features
96	Linear Projection
97	Conv2D – Alternative of Linear Projection
98	Positional Encoding
99	Transformer Encoder
99	Prediction Head
100	SimCLR
101	– פונקציית ה-loss ב-SimCLR – Contrastive Loss
102	הקדמה ל-ConVIRT - CLIP
103	– פונקציית ה-loss ב-ConVIRT – Contrastive Loss
104	CLIP
105	שלב האימון – CLIP
106	שלב ה-Inference – CLIP – Inference
107	U-Net
108	מאפייני הארכיטקטורה
109	Difussion Model

110	סקירה מודלים גנרטיביים מובילים עד כה....
111	מאפייני ארכיטקטורת ה-Diffusion
112	תיאור מתמטי של ארכיטקטורת ה-Diffusion
112	נווטציות
113	הרעשה
115	Denoising
116	Variational Lower Bound – Using KL Divergence
124	תיאור האלגוריתם
124	שלב האימון
125	שלב ה-Sampling
126	Stable Diffusion
127	תיאור הארכיטקטורה
128	Encoder
128	Decoder
128	Encoder – Decoder Objective
129	תהליך ה-Diffusion (הרעשה)
130	Denoising
130	Cross Attention
131	U-Net
132	Stable Diffusion Objective
132	Autoencoder Objective
134	LDM Loss

AutoEncoders

[Video Link](#)

בעידן המידע בו אנו חיים, כמות הנתונים שנוצרת מדי יום מציבה אתגרים והזדמנויות חדשים לפני מדעני הנתונים והמחקר. במרקם של אתגרים אלו עומד הצורך למצוא דרכים יעילות לדוחס, לנתח ולפרש את הנתונים האדירים הללו. בהקשר זה, אחת הטכנולוגיות המרכזיות המשמשות היום היא ארכיטקטורת AutoEncoders, שפותחה בשנות ה-80 ומאפשרת דחיסת נתונים באופן אוטומטי ויעיל. היא מתמקדת בלמידה רפרזנטטיבית מצומצמת ועמוקות של הנתונים, באמצעות מודל שמחולק עצמו לשני חלקים: Encoder, האחראי להפחתת ממד הנתונים, ו-Decoder, שתפקידו לשחזר את הנתונים מהרפרזנטציה המצומצמת.

חשוב לציין כי ב-AutoEncoders, הקלט הוא גם הפלט וה-Objective שיפורט בהמשך חל בין הקלט לבין עצמו. למעשה במהלך תהליכי הלמידה מצמצם כל הניתן את הקלט למרחב לטני כלשהו ועם זאת לשמר את התכליות המהוויות בדאטא כך שבבסיסו של דבר הרפרזנטציה הנלמדת מהוות תמצאות ויזוק של הדאטא ולאחר מכן משמשת לטובות הרכבתו מחדש.

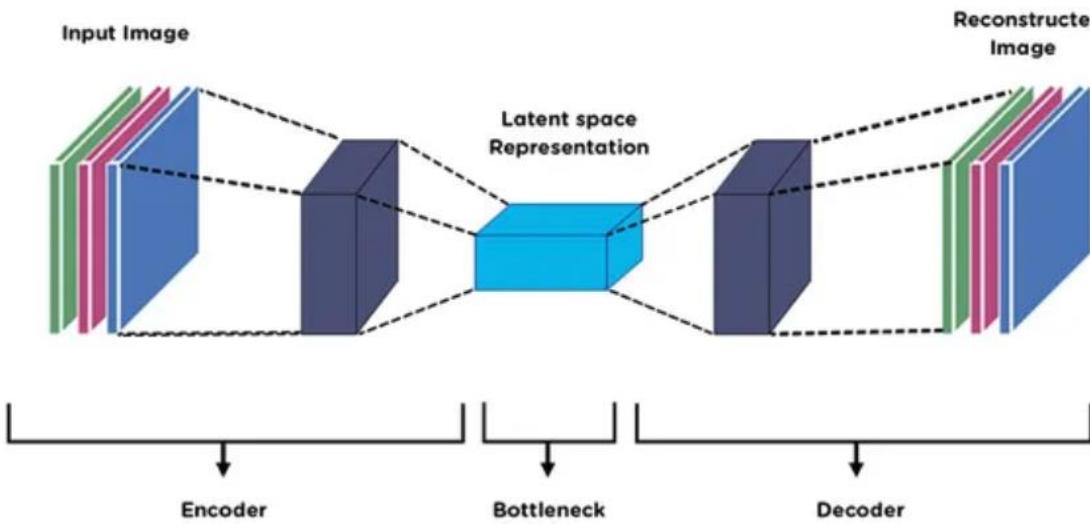
שימושים

1. **ארQUITקטורת AutoEncoders מתמקדת במספר שימושים עיקריים המבוססים על הייצוג הלטנטי: דחיסה (Compression)** – מאפשר דחיסת נתונים על ידי הפקחת ממד הקלט, דרך הלמידה של רפרזנטציות נמוכות יותר מבלי ל앗ז מידע מהותי. זהו שימוש חשוב במיוחד כאשר רוצים לצמצם את נפח הנתונים לפני עיבודם או שימושם.
2. **ניקוי רעש (Denoising)**: דרך הטמעת שכבה רעשה במהלך האימון, AutoEncoders יכולים ללמוד לשחזר את הקלט המקורי מתוך קלט רועש, ובכך להפחית או להסיר את הרעש מהנתונים עבור דאטא חדש במהלך תהליכי inference.
3. **ייצוג דליל (Sparse Representation)**: מאפשרות נוספת היא למידת ייצוג דליל שבו רוב הערכים הם אפסים.
4. **יצירת נתונים (Data generation)**: בהינתן ה-AutoEncoder כבר מאומן נכון, על ידי ידיעת התפלגות הווקטור בממד הלטנטי ניתן לבצע הרכבה (reconstruction) על מנת לקבל דאטא מג'ונרט.

תיאור הארכיטקטורה

התמונה מציגה את ארכיטקטורת ה-AutoEncoder, המורכבת משלושה חלקים עיקריים:

1. **Encoder**: החלק הראשון של הרשת, שבו התמונה המקורית מוצנת. ה-Encoder מתמקד בכך שהוא לומד את הנתונים הממדים הגבויים וודוחם אותו לתוך מרחב ממדים נמוך יותר, הנקרא מרחב הלטנטי. ה-Encoder מכיל סט משקولات לממדות.
2. **Bottleneck**: זה החלק המרכזי של ה-AutoEncoder, כאן למעשה מוצג הדטה בצורתו הדחוסה, המידע מיוצג על ידי וקטור קטן יותר של תכונות, המכונה גם "היצוג במרחב הלטנטי".
3. **Decoder**: בחלק הזה של הרשת, המידע שנדחס על ידי ה-Encoder מזין CUT-L Decoder, שמטרתו לשחזר את התמונה המקורי מהייצוג הדחוס שלה. התמונה שモפקת כאן מכונה "התמונה המשוחזרת". ה-Decoder מכיל סט משקولات לממדות.



פורמלית

מבחן פורמלית, המשווה המוצגת מבטא את ה-Objective שבה אנו שואפים למצער במהלך האימון של מודל AutoEncoder. המטריה היא למצוא את הערכים האופטימליים של המשקولات w , שבhem ההבדל בין נתוני הקלט x_n לבין תמונה הקלט המשוחזרת \tilde{x}_n , המיוצרת על ידי המודל, יהיה הקטן ביותר. הפונקציה מחושבת בסכום של חצי הנורמה האוקלידית בריבוע, עבור כל נקודות נתונות בסט האימון. המשקولات w_f מייצגות את ה-Encoder, והמשקولات w_g מייצגות את ה-Decoder.

$$\min_w \frac{1}{2} \sum_n \|w_g w_f x_n - \tilde{x}_n\|_2^2$$

חשיבות לציון כי ה-Objective המוצג כאן למעשה מייצג linear AutoEncoder מהסיבה שהפעולות המוצגות כאן הן פעולות של כפל מטריצוני (טרנספורמציה לינארית) ולכן ה-Objective המשתמש ב-Objective זה הוא לינארי.

עם זאת, בהינתן פונקציות לא לינאריות המוגדרות f, g , עברו ה-Encoder ו-Decoder בהתאם או להילופין אסופה של פונקציות לא לינאריות המצוירות בין קומפוננטות Encoder וDecoder ונראה כי נקבל Non-linear AutoEncoder בו הייצוג הלטנטי שהתקבל מסוגל להכיל תלויות שאין לינאריות וכן כמו כן אופן ההרכבה בהתאם יכול להיות לא לינארי.

מבחן פורמלית ניתן לתאר את ה-Non-linear AutoEncoder הבא:

$$\min_w \frac{1}{2} \sum_n \|g(f(x_n; w_f); w_g) - \tilde{x}_n\|_2^2$$

רפרנציה דليلה:

כאמור, תיארנו את אחד השימושים של AutoEncoder כזאת המאפשר ללמידה רפרנציה דليلה (Sparse Representation) באופן בו הוקטור המצרי במרחב הלטנטי מכיל ברובו אפסים. לצורך ביצוע פעולה זו על פניו היינו רוצים להשתמש בפונקציית Objective ה-*h*:

$$\min_w \frac{1}{2} \sum_n \|g(f(x_n; w_f); w_g) - \tilde{x}_n\|_2^2 + c \operatorname{nnz}(f(x_n; w_f))$$

זה מציין *non-zero entries* כלומר היינו רוצים פונקציית Objective שמטרתה למנם ככל הניתן את מספר הסקלרים בוקטור הלטנטי שאינם אפס. מהסיבה שזו דרישת בעייתי לאופטימיזציה נוכל לבצע רדוקציה לדרישה ולתאר אותה באופן הבא:

$$\min_w \frac{1}{2} \sum_n \|g(f(x_n; w_f); w_g) - \tilde{x}_n\|_2^2 + c \|f(x_n; w_f)\|_1$$

ambil' להיכנס להוכחה שמתארת את השקילות, אנחנו יכולים לראות כי רגולרייזציה מסוג L1 שකולה לאופטימיזציה שרצינו לבצע על ידי מינימום ה-*h* אך ורק Objective זה עמיד ליצור ווקטור ליטני דليل. כמו כן אנחנו יכולים לראות כי C הוא היפרפרמטר שניון לקביעה במהלך תהליכי האימון כיאה objectives מסווג ה-*h*.

דוגמא לשימוש – Denoising AutoEncoder

כאמור, כפי שציינו מקודם אחד השימושים של AutoEncoder הוא לטובת ניוקי רעש. האופן בו אנו יכולים לרטום את ה-*AutoEncoder* לניקוי רעשים הוא באופן הבא:

שלב 1 – הכנת דאטא מולולר

בהינתן דאטא כלשהו, נוסיף לו רעש בתפלגות גausיינית (σ, μ) N . לרוב כאשר אנו מושפעים רעש בתפלגות גausיינית, ערך ממוצע התפלגות הוא 0 וערך סטטיסטיקת התקן משתנה לפי הצורך. על ידי הוספת רעש בצורה זו נקבל כי רוב הפיצרים בדאטא יקבלו ערכי שונים קטנים מהסיבה שהם ממוקמים במרכז התפלגות. עם זאת יהיו כאלה שיקבלו ערכים גדולים יותר תחת ההנחה שהערכים שיתווסףו יהיו ממוקמים בשולי התפלגות. לאחר שהוספנו לדאטא רעש גausייני קיבלנו דאטא מורעש שנוצר כתוצאה מהדאטא המקורי.

שלב 2 – שלב האימון

בשלב זה אנחנו מאמנים את ה-*AutoEncoder* באופן בו הקלט f -*to*-*Decoder* הוא הדאטא המורעש. בתהילך האימון אנחנו מעבירים את הדאטא המורעש לאורך *Encoder* וה-*Decoder* כך שלבסוף אנו מודדים את *Loss* בין הדאטא המקורי לפני הרעשה לבין הדאטא המנוקה לכואה. אנחנו ממשיכים בפועל באופן איטרטיבי עד להתקנות.

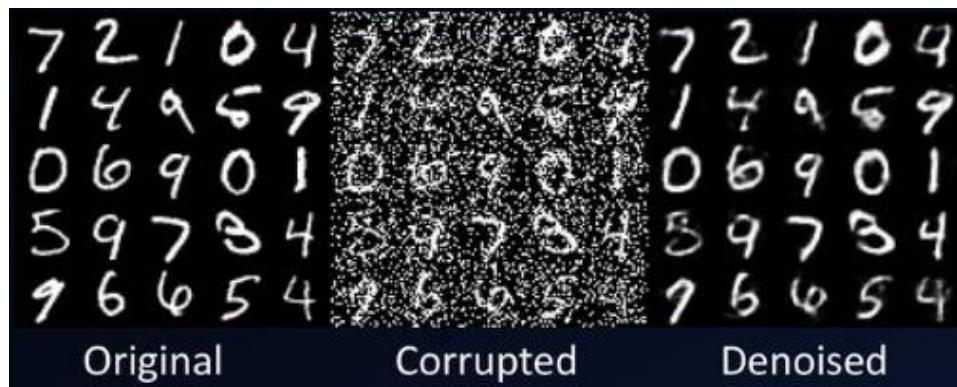
ניתן לראות את ה-*Objective* המתואר פורמלית במשווהה הבאה:

$$\min_w \frac{1}{2} \sum_n \|g(f(x_{noisy}; w_f); w_g) - \tilde{x}_{original}\|_2^2 + c \|f(x_n; w_f)\|_1^1$$

שלב 3 – שלב ה-*Inference*

ב-*Inference* קיבלנו על פניו מודל מאומן ולכן כל שעליינו לעשות הוא להכניס דאטא מולולר והרשת עתידה לספק לנו את הדאטא מנוקה מהסיבה שהיא למדה רפרזנטציות של תחילן הניקוי.

$$g(f(x_{noisy}; w_f); w_g) = x_{clean}$$



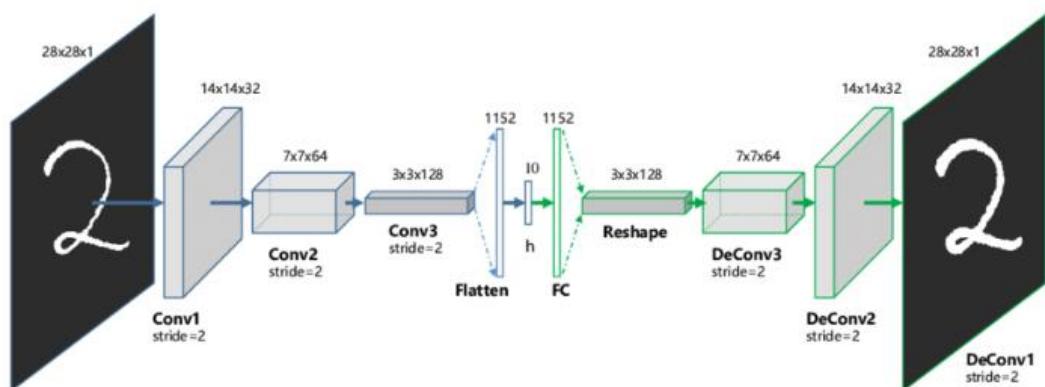
דוגמא לארכיטקטורה

התמונה מתארת את ארכיטקטורת AutoEncoder עם מרכיבי בניין שמרחיבים את התהילה הפשט של Encoder וה-Decoder.

בחלק השמאלי של האיר, אנו רואים את Encoder, שמתחל בקלט של תמונה בגודל $1x28x28$. התמונה עוברת דרך מספר שכבות קונволוציה (Conv1 ו-Conv2), כאשר כל שכבה מבצעת סינון של הנתונים ומפחיתה את הממדים דרך פעולה stride. לאחר שכבות הקונволוציה, הנתונים מועברים לשכבת Flatten לקלט וקטור ישר אחד, ואז מועברים לשכבת חיבור מלאה (FC) לקבלת הייצוג הוקטורוני במרחב הלטנטי (Latent space).

בחלק הימני של האירור, אנו רואים את Decoder שמנסה לשחזר את התמונה המקורית מהייצוג הלטנטי. הוא מתחילה משכבה חיבור מלאה (FC) ולאחר מכן משתמש בפעולות Reshape כדי להחזיר את הנתונים למבנה מרוחבי שמתאים לפעולות קובולוציוניות. שתי שכבות קובולוציוניות הפוקוט (DeConv1 ו-DeConv2), שכל אחת מהן מבצעת *Upsampling*, נמצאות בפעולה כדי להרחב את הממדים ולהגיע לממד התמונה המקורי של 28×28 .

לסייע לנו לומר כי אמן AutoEncoder מוגדר בצורה כללית על ידי Encoder, Decoder וDecoder.



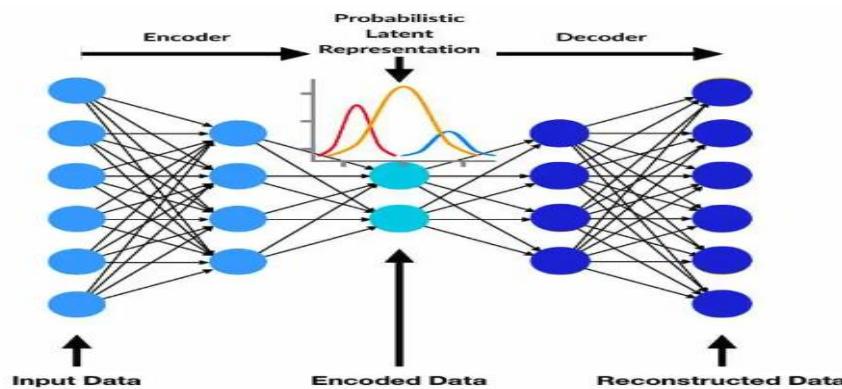
Probabilistic AutoEncoders

העיקרון של AutoEncoders הסתברותיים מאוד דומה לעיקרון של AutoEncoders עליהם דיברנו עד כה. עם זאת, השוני כאן הוא שה-s-AutoEncoders הללו מכילים פונקציות הסתברותיות המוגדרות על סמך התפלגות מסוימת. הכוונה היא שהווקטור במרחב הלטנטי כמו כן הפלט מוגדרים על פי הקלט שנכנס אל הפונקציה, מועבר דרך estimator המוגדר על פי התפלגות, ורק לאחר מכן מתקבל ערך המשנה במרחב הלטנטי וערך המשנה בפלט.

$$f: \Pr(h|x; w_f)$$

$$g: \Pr(\tilde{x}|h; w_g)$$

בתמונה ניתן לראות כי המבנה הארכיטקטורי של-h-AutoEncoder ההסתברותי זהה לחלוין לבנייה של-h-AutoEncoder אך עם זאת ערכי הווקטור במרחב הלטנטי מוכתבים על ידי ההתפלגות עברו כל אחד מהאלמנטים בווקטור הלטנטי. הפלט ההסתברותי שמוכתב על ידי ההתפלגות הוא גם אלמנט הניתן למידה במהלך תהליכי האימון. בהתאם לסוג ההתפלגות, הערכים המכתיים את ההתפלגות (למשל ס, גaussian עבור ההתפלגות נורמלית), עתידיים להשנות במהלך תהליכי האימון כדי שלאחר ההרכבה ב-Decoder של מהווקטור הלטנטי נקבל פלט שדומה ככל הניתן לקלט אותו התחלונו.

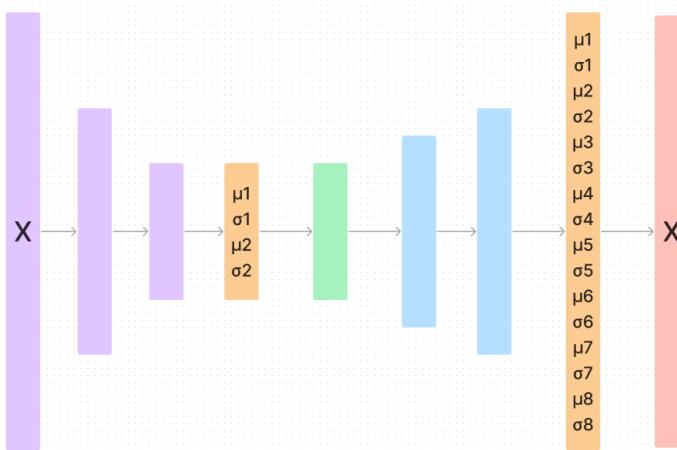


מכיוון שהערכים בווקטור הלטנטי מוכתבים על ידי ההתפלגות, הרשת מסוגלת לייצר פלטים מורכבים ולא דטרמיניסטיים, דבר שמאפשר גיון רחב יותר בנתונים שהרשות מסוגלת להתמודד איתם ולשחזר. היכולת לקבוע את ההתפלגות של כל אלמנט בווקטור הלטנטי מאפשרת גישה יותר גמישה ומותאמת אישית לטיב הנתונים, ובכך משפרת את איכות הלמידה והשחזור של הרשת.

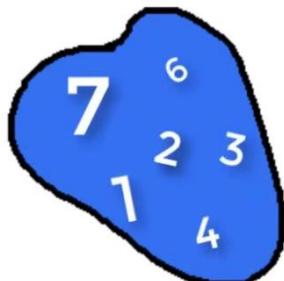
Probabilistic AutoEncoders Based Normal Distribution Example

לצורך ההמחשה נניח והקлат שלנו הוא בגודל 8 ומוגדר $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ ומווגדר $X = h(h_1, h_2)$. כמו כן נניח והווקטור במרחב הלטנטי הוא מגודל 2 ומוגדר (μ_1, μ_2) .

אנחנו מתחילה עם וקטור בגודל 8 ולאחר העברתו לשכבות ה-Encoder אנחנו מקבלים כי הפלט של השכבה האחורונה ב-Encoder (הכטומה) מכיל את ערכי התפלגות ומיצרים את הווקטור הלטנטי (השכבה הירוקה) תהליך האימון. לאחר מכן אנו אנו דוגמים מן התפלגות ומיצרים את הווקטור הלטנטי (השכבה הירוקה) בגודל 2. הווקטור זה בתורו עבר דרך Decoder עד שהוא מגיע לשכבה האחורונה של הDecoder. שמכילה 8 זוגות ערכים סעודיים הנקראים עבור כל אחד מהאלמנטים בווקטור אנו מקבלים את הפלט בגודל 8.

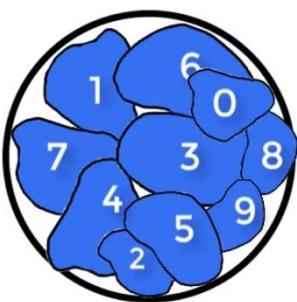


המודוליבציה לשימוש ב-AutoEncoders



על פניו, היינו יכולים להמשיך ולהשתמש ב-AutoEncoders, לקבל ייצוגים לטנטיטים ולהרכיב אותם לכדי ערך הפלט כמו למשל שהדגמנו בשימוש ב-AutoEncoder Denoising.

עם זאת, אם אנחנו מסתכלים על הייצוג של הווקטורים במרחב הלטנטי נדמה כי הם לא מספרים לנו סיפור מעניין במיוחד. אוסף של מספרים שהתקבלו מהתהילך של עדכון משקלות חלק מחולק Backpropagation. אם מסתכלים למשל על מספרים המוצגים אל אותם וקטוריים לטנטיטים נראה כי הם למעשה אוסף של וקטורים שנלמדו על סמך סט האימון ואין להם כל קשר ממשום התפלגות.



לכן, באמצעות שימוש ב-AutoEncoders הסתברותיים אנו יכולים לאפיין את הווקטורים הלטנטיטים כלוקחים מtower הסתברות רציפה כלשהי. כך בהינתן וקטור לטנטיטי אותו נרצה ליצר תחת הרגשה שנרצה לקבל פלט הגיוני, נוכל לדגסמן התפלגות באזוריים המיוחסים לואותו פלט ובכך לקבל תוצר הקרוב למזה שציפינו. למעשה, באמצעות שיטה זו אנחנו יכולים לייצר דאטא מלאכותי באמצעות ידיעת התלות בין הפלט, הווקטור הלטנטי וההתפלגות. זו למעשה הקדמה ל-VAE וליכולת היגנרטיבית שהאכטיקטורה מאפשרת לנו עליה גורחיב כעת.

Variational AutoEncoder – VAE

[Video Link](#)

Variational Autoencoders, או בקיצור VAE, מהווים חידוש בתחום של למידת מכונה בשנים האחרונות. מדובר בסוג של רשתות ניורוניים שמאפשרות גישה חדשה לצירוף פריזנטציה מצומצמת של נתונים. עיקר ההבדל בין VAE ל-Autoencoders הקלאסי הוא בהנחה הבסיסית שעל פיה המרכיב הלטנטי אינו מרחיב דטרמיניסטי אלא מרחיב הסתברותי, מה שמאפשר ל-VAE ליצור דגימות חדשות ומגוונות באמצעות חישוב של התפלגיות.

ב-VAE, ה-Encoder מוגדר כפונקציה סטטיסטית המשקפת את הפרמטרים של התפלגות המרכיב הלטנטי, וה-Decoder לatkח דגימות מהרכיב הלטנטי ומשחרר את הנתונים. המטריה בلمידה היא למקסם את פונקציה ה-Loss של VAE. פונקציה זו נועדה להתמודד עם שני גורמים: האחד הוא השגת דחיסה גבוהה של הנתונים ומיקסום ה-likelihood, והשני הוא קידוד הנתונים לכדי התפלגות נורמלית שדומה לתפלגות המקורית – למעשה רצון לקירוב התפלגיות.

$$\max_w \sum_n \log \Pr(\tilde{x}|x, w_f, w_g) - c \text{KL} (\Pr(\mathbf{h}|x_n; w_f) || N(\mathbf{h}; \mathbf{0}, I))$$

הסבר ה-Objective ב-VAE

את המשוואה שמייצגת את ה-Objective של VAE ניתן לחלק ל-2 חלקים:

חלק 1 – Maximum log likelihood

אנו משתמשים ב-log likelihood loss כפונקציית loss בחלק הראשון של ה-objective. הסיבה לכך היא שפונקציית log likelihood מודדת את הסיכוי שהנתונים המשוחזרים \tilde{x} יהיו דומים לנ נתונים המקוריים x . מיקסום ה-log likelihood מתרחש כאשר הסיכוי לשחזר מדויק של התמונה המקורית הוא הגבוה ביותר. במקרה אחר, ככל שההבדין בין \tilde{x} לא- x יהיה גבוה יותר, כך ערך ה-log likelihood $\log \Pr(\tilde{x}|x, w_f)$ יהיה יותר��, והמודל יתקרב יותר למטרה שלו.

חלק 2 – קירוב התפלגיות – (KL Divergence)

בחלק השני של ה-objective, אנו משתמשים ב-Kullback-Leibler divergence כדי לקרב את התפלגות של הוקטור הלטנטי \mathbf{h} לתפלגות נורמלית רב-ממדית ($I(\mathbf{h}; \mathbf{0}, I)$). הסיבה לכך היא שדגימה מההתפלגות נורמלית פשוטה יחסית, ומאפשרת יצירת תמנונות חדשות בקלות. כמו כן, חשוב לציין שאנו לא יכולים לדגם ישירות מההתפלגות של \mathbf{h} כתלות ב- w_f, x , והסיבה לכך היא שתפלגות זו עשויה להיות מורכבת וקשה לדגימה. לכן, אנו משתמשים ב- KL Divergence כדי למצוא התפלגות קרובה (התפלגות נורמלית) שקל יותר לדגם ממנה.

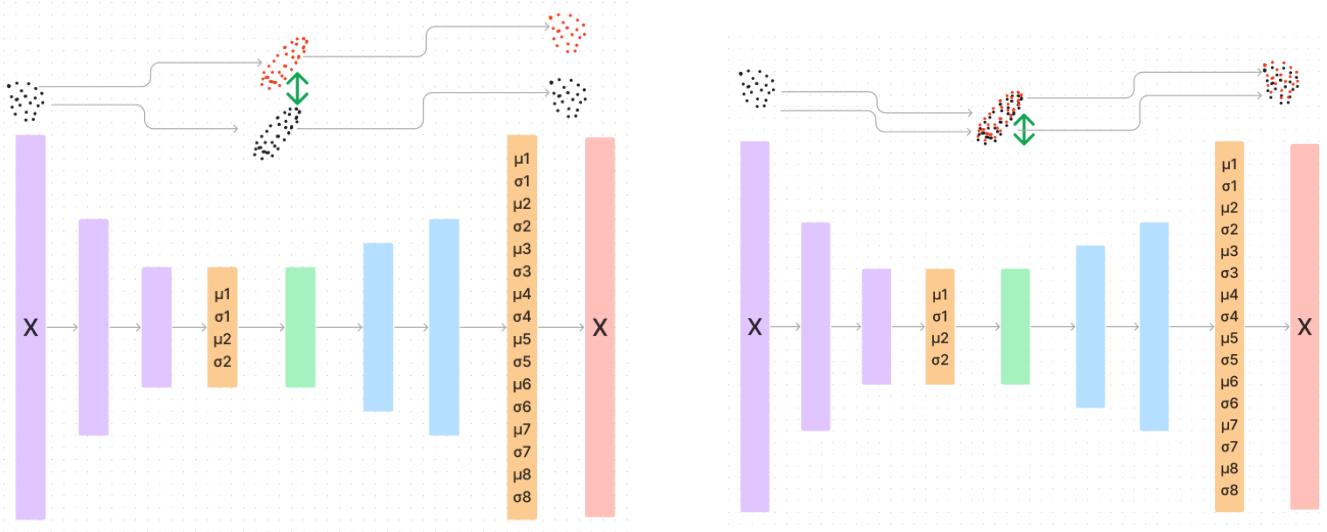
פירוט על המשמעות של קירוב התפלגיות

קירוב התפלגיות במרחב הלטנטי הוא פעולה מרכזית בפעולות של Variational Autoencoders (VAEs). המנגנון זהה מאפשר למודלים להתמודד עם אחד האתגרים הבסיסיים בלמידת מכונה: ייצור והבנה של התפלגיות מורכבות של נתונים. ב-VAEs, נעשה שימוש בתפלגות נורמלית רב-ממדית ($I(\mathbf{z}; \mathbf{\mu}, \mathbf{\sigma}^2)$) כדי להתמודד עם המרחב הלטנטי של הנתונים. תהליך זה מאפשר לנו לאשר בין הפערים הנובעים מהמרחב המקורי של הנתונים לבין המרחב שבו יכולים לנו לנהל ולשלוט בו מבחינה חישובית.

הकושי לדגום יישרות מתפלגות המורכבת של \mathbf{z} , \mathbf{x} נובע מהאופי הלא דטרמיניסטי והלא ליניארי של רוב הנתונים בעולם האמיתי. נתונים אלו יכולים להיות בעלי תכונות של רעש, פיזור או אינטראקציות דינמיות שאין מתווארות היטב על ידי התפלגות פשוטות. דוגמה מתפלגות כזו יכולה להיות מורכבת מאוד ולדרושים כמות גדולה של חישובים. ב网讯וד לכך, התפלגות נורמלית רב-ממדית היא יחסית פשוטה להבנה ולדגם, מכיוון שככל אלמנטים בתפלגות זו נתונים לתיאור סטטיסטי בסיסי.

בשימוש ב- KL Divergence כחלק מה-objective ב-VAEs, אנו מצאים למצוא התפלגות נורמלית שהיא הקרובה ביותר להתפלגות המקורי המורכבת. זהו למעשה קירוב שמאפשר לנו ליצור פתרון חישובי עיל ומוקדק. כשהאנו מבצעים דוגמה מתפלגות הנורמלית המקורבת, אנו מוצאים פתרון שהוא אمنם לא מושלם, אבל הוא מפנה לנו הבנה ובקירה טובה יותר על התהליך הכלול.

המשמעות העומקה של תהליך זה היא שאנו יכולים ליצור מודלים שמנצחים את המידע הלטנטי העמוק של הנתונים, ולא רק את המאפיינים הפנימיים שלהם. זה מאפשר ל-VAE להיות לא רק כל' לדחיסת נתונים, אלא גם כל' לייצרת נתונים חדשים באופן שמחקה את הממציאות בצורה טבעית ו邏輯ית.



בשתי התמונות אנו יכולים לראות במרחב הלטנטי 2 התפלגיות. התפלגות השחורה היא זו המתארת את התפלגות הוקטור הלטנטי \mathbf{h} כתלות ב- $\mathbf{z}_f; \mathbf{x}_n$ $\Pr(\mathbf{h}|\mathbf{x}_n; \mathbf{z}_f)$. כמו כן התפלגות האדומה מייצגת את התפלגות הנורמלית הרבה ממדית המוגדרת ($I(\mathbf{z}; \mathbf{\mu}, \mathbf{\sigma}^2)$).

בתמונה השמאלית ניתן לראות כי התפלגיות במרחב הלטנטי מרווחות זו מזו באופן ייחודי דבר הגורם לקבלת פרדייקציות רחוקות זו מזו. עם זאת, כאשר אנו מקרבים את התפלגיות כפי שניתן לראות בתמונה הימנית באמצעות כלילת רכיבת הנורמליזציה הזה בפונקציית הסיגנום ניתן לראות כי גם התפלגיות הפרדייקציות מתקשרות זו לזו לתיאור טוב יותר של הממציאות ביחס לקלט \mathbf{x} .

(בעמודים 10,11,12 יש הסברים נלוויים להבנה טוביה יותר של הנושא – להמשר רציף אפשר לעבור לעמוד 13)

maximum Likelihood Estimation

[Video Link](#)

ננצל את הבמה על מנת לדון בחישובות של MLE כפונקציית loss עבור מודלי למידת מכונה בכלל ועבור VAEs ומדידת ה-Reconstruction Loss בין \hat{x} לא- x .

כאשר אנו מדברים על likelihood, הנראות, אנו למעשה מדברים על הנראות של הפרדייקציות אל מול הליבלים האמתיים. הנראות יכולה להשתקף במספר אופנים אך המשותף לכל האופנים הללו הוא שהם כולם מצינים מידת התאמה בין פרדייקציות המודל לנוטונים. ככל שהפרדייקציות שהמודל מייצר מתאימות יותר לתוצאות או לנוטונים האמתיים שאנו נתקלים בהם, כך הנראות תהיה גבוהה יותר, וכן המודל נחשב למדויק יותר.

פונקציית loss log (נקראת גם NLL) מודדת את הפער בין התפלגיות ההסתברויות לבין התיאוג האמתי בבעיות סיווג. ניתן לחשב אותה באמצעות הנוסחה הבאה:

$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^n [y_i \cdot \log(p_i) + (1 - y_i) \cdot \log(1 - p_i)]$$

נשים לב כי המשוואה מכילה שני חלקים:

$y_i \cdot \log(p_i)$ – מבטא את הענישה על הערך החזוי כאשר התווית האמיתית היא 1

$(1 - y_i) \cdot \log(1 - p_i)$ – מבטא את הענישה על הערך החזוי כאשר התווית האמיתית היא 0

שני החלקים הללו נדרשים כדי להבטיח שה-logLoss יתואר באופן מדויק לשתי האפשרויות הבינאריות של התיאוג, ורק החלק הרלוונטי יתווסף לחישוב הփסד עבור כל תצפית. תחת ההנחה שאחד משני חלקים המשווה יהיה שווה ל-0 וכן תהיה התעלמות מהחלק שאינו רלוונטי לתיאוג הנתון.

ניתן להרחיב את התיאור באותו אופן למספר מחלקות ולא רק לקלסיפיקציה ביןארית:

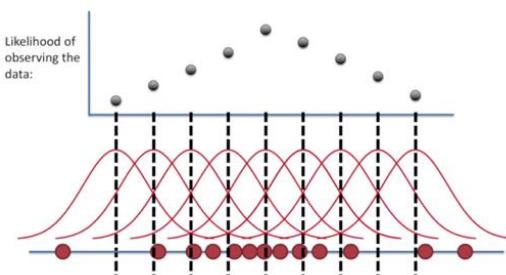
$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^n \sum_{c=1}^n y_{ic} \cdot \log(p_{ic})$$

דוגמא:

נניח ויש לנו 3 מחלקות [C,B,A] ונניח שעבור איטרציה מסוימת התיאוג הוא A כלומר הווקטור הוא [1,0,0].

כמו כן נניח והתפלגות ההסתברויות היא [0.7,0.2,0.1] אז נקבל כי:

$$\text{Log Loss} = -(1 \cdot \log(0.7) + 0 \cdot \log(0.2) + 0 \cdot \log(0.1))$$



Maximization of Maximum Log Likelihood in VAE

עד כה תיארנו את ה-Likelihood, הנראות, עבור בעיות קלסיפיקציה (בינארית וmulti-class) אך כעת נרצה לתאר את מושג הנראות שישמש אותנו ב-VAEs.

במודלים כמו VAE, המטרה היא מיקסום ה-log likelihood של הפלט המשוחזר בהינתן המודל בהשוואה לקלט. לכן ניתן לתאר את ה-objective באופן הבא:

$$NLL = -\log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp \left(-\frac{(\tilde{x} - x)^2}{2\sigma^2} \right) \right)$$

ניתן לפשט את הביטוי באופן בו אנחנו יכולים לחלק ממנו חלקים קבועים שאינם משתנים וכן נקבל:

$$\text{Reconstruction Loss} = \frac{(\tilde{x} - x)^2}{2\sigma^2} + \text{Constant Value}$$

דוגמה:

ניתן כתת דוגמא לחישוב של Reconstruction Loss וכך נסביר את החשיבות של מיקסום מיציה של ערך זה ואיך הוא בא לידי ביטוי מבחינה מתמטית.
נניח ויש לנו אופטימיזציה עבור פיקסל יחיד המכיל ערך ייחיד וכך נניח ואנו מספקים 2 פרדייקציות. בסכム כתת את הנתונים:

- 0.5 – Original Pixel Value
- 0.6 – Reconstruction Pixel Value Case 1
- 0.52 – Reconstruction Pixel Value Case 2

نبצע כתת חישוב עבור הערך המוגדר כמינוס ערכי ה- σ (0.1) Reconstruction Loss מהמקרים:

$$\text{Negative Reconstruction Loss}_{\text{Case 1}} = -\frac{(0.5 - 0.6)^2}{2 \cdot 0.1^2} = \frac{(0.1)^2}{2 \cdot 0.01} = \frac{0.01}{0.02} = -0.5$$

$$\text{Negative Reconstruction Loss}_{\text{Case 2}} = -\frac{(0.5 - 0.52)^2}{2 \cdot 0.1^2} = \frac{(0.02)^2}{2 \cdot 0.01} = \frac{0.0004}{0.02} = -0.02$$

כך, על ידי מיקסום ה-log likelihood של Reconstruction Loss, אנו למעשה מבטיחים שהתמונה המשוחזרת מתקרובות ככל האפשר לתמונה המקורית, דבר המאפשר שיפור ניכר בדיקוק השחזור ובאיכות המודל הכללי.

- נקודה חשובה היא שהטרמינולוגיה (יש לומר המבלבלת) היא שאנו רוצים לדוחף למינום ערך הנראות (Maximize Likelihood) – הגדרה שקופה היא שאנו רוצים לדוחף למינום ערך הנטאות. למעשה יש שקלות בין מיקסום הנראות לבין מינום ערכי ה-NLL.

פירוט על KL Divergence

[Video Link](#)

KL Divergence הוא ממד מתמטי המודד את המרחק בין שתי התפלגיות הסטברותיות. בהינתן שתי התפלגיות P , Q ועבור משתנה מסוים $x=X$ נראה כי המשוואה מוגדרת באופן הבא:

$$D_{KL} = (P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right) = p_1 \log \frac{p_1}{q_1} + \dots + p_n \log \frac{p_n}{q_n}$$

באופן עקורי Q , P יכולות להיות 2 התפלגיות כלשהן אך לטובות התקשרות לעולם המושגים שלנו ולטובות ההסביר נוכל לדמות את P להתפלגות האמיתית (Oracle) ו- Q להתפלגות ההסתברויות כפלט של מודל כלשהו.

- נכל לבדוק ב-2 נקודות מעניינות:
 - 1. $0 = q_i \rightarrow \log \frac{p_i}{q_i} = 0$ – עבור כל ערכי הדגימות בהינתן וערך הסתברות של P הפתלוגות הפרדי-קייזות שווה לערך התפלגות האמיתית נקבל כי $0 = \log$ וכך ממד KL Divergence יהיה 0 ככלור ההתפלגיות זהות.
 - 2. כאשר ערך i בהתפלגות האמיתית הוא 0, תרומתו לממד KL Divergence היא אפס, מכיוון שהמונה בחישוב $\log \left(\frac{P(x)}{Q(x)} \right)$ הופך לאפס. זה מגדיש שرك הנקודות שבין התפלגות האמיתית מציגה ערכים חיוביים תורמים לחישוב KL Divergence, ומאפשרת התמקדות בא-התאמות המשמעותיות בין התפלגיות.

שקליות KL Divergence and Log Loss

השקליות בין פונקציית log loss לממד KL Divergence נובעת מהמבנה המתמטי הדומה של שניהם. שתי הפונקציות מודdot את ההבדל בין התפלגיות הסטברותיות, אך מתוור זוויות שונות. פונקציית log loss מחשבת את log loss-information כאשר מודל מיצג נכון או לא נכון נתונים קלים, בעוד ש-*KL Divergence* מודדת את "מරחק" הכללי בין התפלגות המודלית לבין התפלגות האמיתית. השקליות נוכנוה מכיוון שבשתי המקדים, אנו מוחפשים למצער את הטיעויות בחיזוי של המודל ביחס למציאות הקיימת. כאשר ההפרש הלוגריטמי ממוצע, התפלגות שהמודל מיצע קרוביה יותר להתפלגות האמיתית, מה שmobiel למינימום של KL Divergence ולכן לשקליות בין שתי המדדים. הפיתוח המתמטי מצוין בתמונה הבאה:

$$\begin{aligned}
 D_{KL}(P^*||P) &= \sum_y P^*(y|x_i) \log \frac{P^*(y|x_i)}{P(y|x_i; \theta)} \\
 &= \sum_y P^*(y|x_i) [\log P^*(y|x_i) - \log P(y|x_i; \theta)] \\
 &\quad \text{DOESN'T DEPEND ON } \theta \\
 &= \sum_y \boxed{P^*(y|x_i) \log P^*(y|x_i)} - \sum_y P^*(y|x_i) \log P(y|x_i; \theta)
 \end{aligned}$$

(P^* – פונקציה יתירה y היא התפלגות האמיתית (Oracle) (או צוינו בתור P)
 ותפלגיות ההסתברות היא P (או צוינו Q))

$$\operatorname{argmin}_{\theta} D_{KL}(P^*||P) \equiv \operatorname{argmin}_{\theta} - \sum_y P^*(y|x_i) \log P(y|x_i; \theta)$$

VAE Likelihood

از תיארנו את ה-VAEObjective ב-VAE ככח התלוי בשני חלקים. על החלק השני המודד את המרחק בין ההתפליגיות הרוחבנו בפירוט בעמודים הקודמים.

עת נרצה להתמקד בחלק הראשון של objective המוגדר $\Pr(\tilde{x}|w_f, w_g)$.
נשאלת למשה השאלה איך ניתן לחשב אותו. על מנת לענות על השאלה זו נרצה לפשט את הבעיה ולומר כי נניח ולוקטור הlatentי h הינו שני ייצוגים וקטוריים בלבד: Vec A, Vec B.
אז הינו יכולים לומר כי התיאור הבא מתאר את ההסתברות:

$$\Pr(\tilde{x}|w_f, w_g) = \Pr(\tilde{x}|\mathbf{h} = A, W_g) \cdot \Pr(\mathbf{h} = A|x_n; W_f) + \Pr(\tilde{x}|\mathbf{h} = B, W_g) \cdot \Pr(\mathbf{h} = B|x_n; W_f)$$

כמו כן בהינתן ולוקטור הlatentי h הינו ח' ייצוגים וקטוריים הינו יכולים להכליל ולומר כי החישוב מתואר:

$$\Pr(\tilde{x}|w_f, w_g) = \sum_{i=1}^n \Pr(\tilde{x}|\mathbf{h} = i; W_g) \cdot \Pr(\mathbf{h} = i|x_n; W_f)$$

עם זאת מאחר כי הוא יציג latent מתוך מרחב כלשהו ממימד גבוה אזי מתקיים כי הוא יכול להכיל אינסוף ערכים \mathbb{R}^d ולכן יש צורך לחשב את ההסתברות באופן תחת ההנחה כי היא רציפה:

$$\Pr(\tilde{x}|w_f, w_g) = \int_h \Pr(\tilde{x}|\mathbf{h}; W_g) \cdot \Pr(\mathbf{h}|x_n; W_f) d\mathbf{h}$$

הסבר פשוט לאופן חישוב ההסתברות בצורה רציפה:

כשכתבתי את הסיכום, מצאתי את אופן חישוב ההתפלגות למבלבל מעט ולכן החלטתי לכלול גם דוגמא פשוטה יותר שתמחיש את העקרון המנחה באופן חישוב זה.

נניח וננו רוצים לצלם תמונה של גינה המוגדרת x התלויה בזמן כלשהו $time$. אילו בזמן הינו שני ערכים בלבד הינו יכולים לתאר את ההסתברות בטור:

$$\Pr(x_n) = \Pr(x_n|morning) \Pr(morning) + \Pr(x_n|evening) \Pr(evening)$$

עם זאת, מאחר זמן הוא משתנה מקרי רציף אזי נדרש לחשב על חישוב ההסתברות בצורה רציפה:

$$\Pr(x_n) = \int_{time} \Pr(x_n|time) \Pr(time) d(time)$$

על פניו הינו רוצים לתחום את החישוב ב $(time_{low}, time_{high})$ $\Pr(x_n) = \int_{time=low}^{time=high} \Pr(x_n|time) \Pr(time) d(time)$
כלומר תחינה וחישוב הסתברות על סמך מסגרת זמן כלשה. אך למעשה באופן בו אנו מחשבים את ההסתברות בצורה על כל המרחב אנו למשה יוצרים חישוב רציף עבור כל רגע בזמן.

בעיות בחישוב ההסתברות בהגדרת ה-Objective

כאמור, הגדרנו באופן הבא את ההסתברות: $\Pr(\tilde{x}|w_f, w_g) = \int_h \Pr(\tilde{x}|h; W_g \cdot \Pr(x_n|W_f \cdot dh)$.
 נשים לב שבאופן חישוב זה אנחנו מגדירים את ההסתברות להיות תליה באופן חישוב רציף המוגדר על כל המרחב ועל כל הערכים שהוא קטרו h יכול לקבל, על זה מתבצעת האינטגרציה. מבחינה פרקטית וחישובית דבר זה אינו ישים מהסיבה שהמרחב הוא מרחב רציף המוגדר \mathbb{R}^d .

לפיך נרצה להגדיר את ההסתברות $\Pr(x_n|h; W_f)$ להיות כך שאינה מחושבת על פני כל המרחב אלא למשה נדגם מתוך התפלגות נורמלית. ערכיו התפלגות הנורמלית מוגדרים להיות ככלי המוחשבים על סמך רשף נוירונים שהמטרה שלהם זה למצאו את ערכי התפלגות המספקים רפרזנטציות התואמות לתוצאות. נשים לב כי ערכי ההסתברות של ה-Encoder מוגדרים באופן הבא:

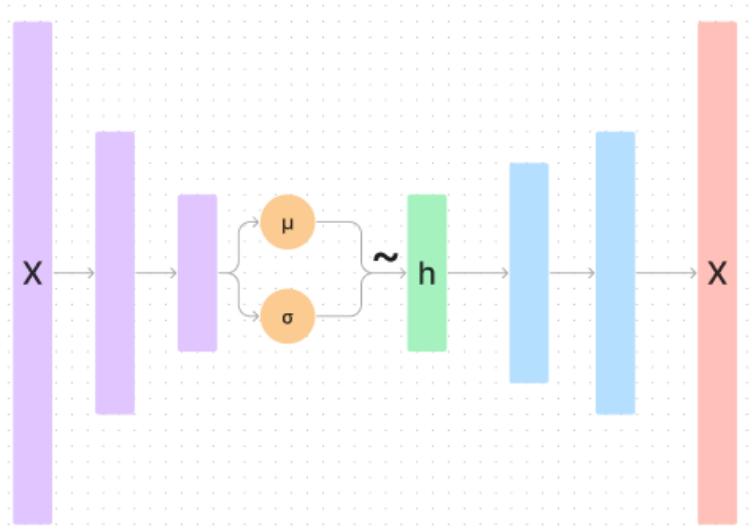
$$\Pr(h|x_n; W_f) = N(h; \mu_n(x_n; W_f), \sigma_n(x_n; W_f)I)$$

באופן זה ניתן לשים לב כי אנחנו יכולים להגדיר את ה- e-VAE עבור Objective VAE כולו באופן מעט שונה התלוי ב h_n כruk h_n כשלעצמם נדגם מהתפלגות נורמלית וכן אין צורך בחישוב האינטגרציה:

$$\Pr(\tilde{x}_n; W_f, W_g) \approx \Pr(\tilde{x}_n|h_n; W_g)$$

$$h_n \sim N(h; \mu_n(x_n; W_f), \sigma_n(x_n; W_f)I)$$

נסים לב כי אנו למשה משורכים את האינטגרל באופן בו אנו דוגמים בכל פעם את h מתוך התפלגות אוטונומית המורכבת מאוסף של μ_n, σ_n הנלמדים באמצעות רשף נוירוניים. ההנחה היא שאומנם אנו דוגמים בכל פעם ייחודה מתוך התפלגות הנלמדת אך אחר ותהליך מתבצע פעמים רבות, בתוחלת אנחנו נקבל קירוב לערך האינטגרל.

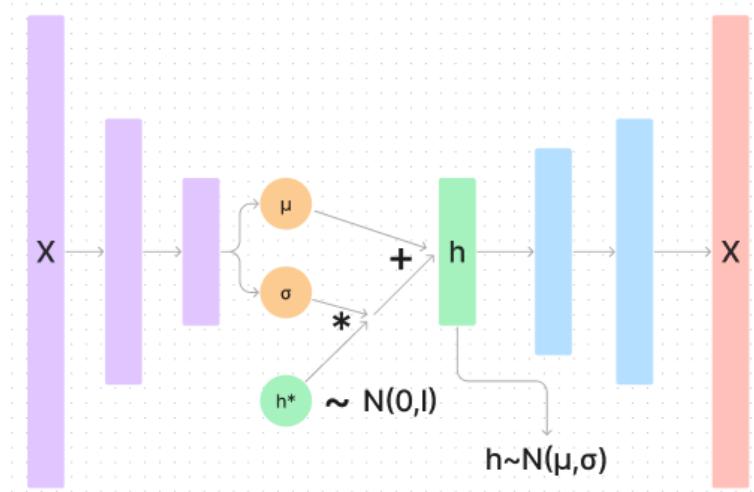


בעיית חישוב הגראדיינט ו- Reparameterization Trick

از אמם פתרנו את הבעיה של חישוב באינטגרל באמצעות דגימה מהתפלגות גausianית שבתווחת מתכנסת לתיאור התפלגות האמיתית אך עצה אנחנו נתקלים בעיה נוספת.

בעיה נוספת היא שאנו דוגמים מהתפלגות נלמדת כלשהו במהלך הפרופוגציה לאורך הרשת – דבר המוביל לעוביה בעית חישוב הגראדיינט לטובע עדכון המשקولات. בפועל, אחר ופועלת הדגימה היא פעולה רנדומית היא לא נחשבת לפועלה חלקה מתמטית ולפיכך הנגזרת אינה מוגדרת.

הפתרון לעוביה הוא שימוש Reparameterization Trick באופן בו אנו נדגום תחיליה מתוך גausian "טיפש" שלא מכיל שום ערכים נלמדים ואל ערך הדגימה נוסיף בצורה אריתמטית את ערכי σ_n, μ_n



ניתן לראות כי מתבצעת דגימה $(I|0)N \sim h$ ולאחר מכן יש הכפלת ב σ והוספת μ לקבלת ערך דגימה המשקף דגימה מתוך $(\sigma_n, \mu_n)N \sim h$ כפי שהיינו רוצים. כמו כן, לאחר ופועלות הכפל ופעולות החיבור הין פעולות אריתמטיות אזי גזירות ולפיכך ניתן לחשב עבורן גראדיינט ולפיעוף את ערכיו לאורך הרשת בתהיל' ה-Backpropagation

אופן ביצוע ה Inference

כאמור, חלק מה**objective** כולל קירוב התפלגיות בין $(I|0)N(h; \mathbf{w}_f)$ ו- $\Pr(h|x_n; \theta)$ באמצעות KL Divergence. המשמעות של הדבר היא שבעת דגימה בשלב ה **Inference** ניתן לדגם וקטור h מתוך גausian "טיפש" תחת ההנחה כי הרפרזנטציות כבר נלמדו במהלך תהליכי האימון, ולכן המודל מסוגל להפיק פלטיהם ממשמעותיים גם מנקודות שבhn לא נראה דוגמאות במפורש במהלך האימון. זהו מרכיב מרכזי ביכולת הganerativiy של המודל, שמאפשר לו ליצור על פניו דוגמאות חדשות שלא נראה מעולם מתוך התפלגות שקרובה להתפלגות המקורית.

Generative Adversarial Network – GAN

[Video Link](#)

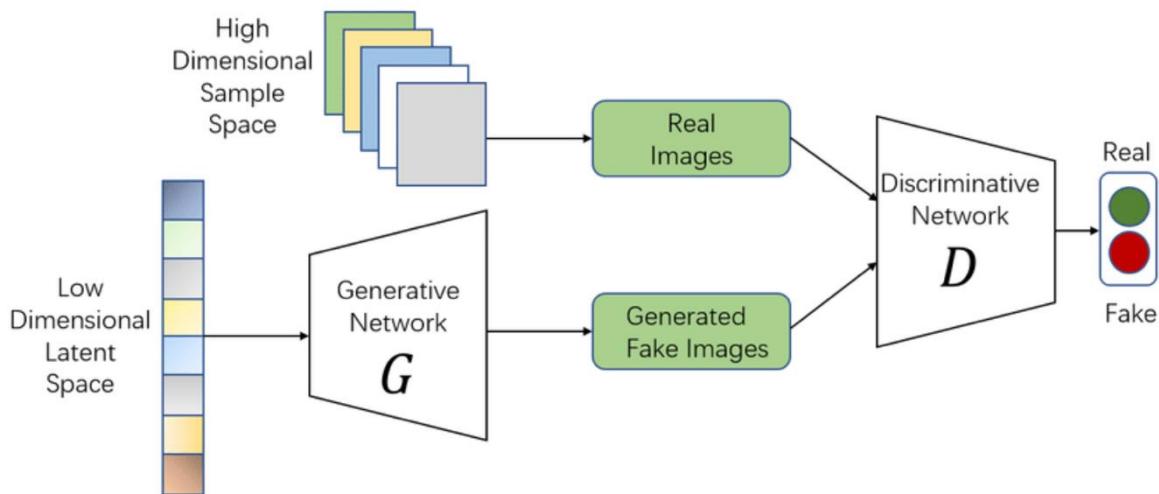
עד כה דיברנו על VAE כמודל גנרטיבי המסוגל לדגם וקטור לטנטי מתוך התפלגות, להעביר את הוקטור דרך Decoder המכיל שכבות נלמדות וליצור מהן דגימות חדשות מההתפלגות.

כשאנו מדברים על GAN, או Generative Adversarial Network, אנו מתיחסים למודל גנרטיבי שפותח בשנת 2014. GAN מורכב משני רכיבים עיקריים (Generator, Discriminator) (Generator, Discriminator) שמתחרים זה בזה במהלך האימון. השניים יוצרים דינמיקה שבה כל אחד מהם משפר את ביצועיו בתגובה לשינויים של השני. זהה טכניקה עילה מאד לייצור דגימות חדשות שנראות כאלו הן לקוחות מההתפלגות הנתונים המקוריים.

1. Generator – הGENERATOR הוא רכיב האחראי לייצור הדגימות החדשנות. המטריה של הGENERATOR היא ללמידה כיצד לייצר דגימות שנראות כמו שיותר אמינות ואמיות, כלומר דוגמאות לדגימות מההתפלגות הנתונים האמיתית שמננה הוא מנסה ללמידה. GENERATOR מקבל קלט וקטור לטנטי של רעש אקראי (לרוב מtower התפלגות נורמלית סטנדרטית) ומשתמש בראשת נוירונים כדי להמיר את הרעש הזה לדגימה שمدמה דגימה אמיתית.

2. Discriminator – הDISCRIMINATOR הוא הרכיב השני ב-GAN, ותפקידו לבדוק בין דגימות אמיתיות לבין דגימות שהופקו על ידי GENERATOR. זהו למעשה Classifier שמנסה לקבוע אם דגימה נתונה היא "אמיתית" (לקוקה מההתפלגות הנתונים המקוריים) או "מזויפת" (ונזרה על ידי GENERATOR). DISCRIMINATOR גם כן מבוסס על ראש נוירונים ומשתף במהלך האימון ככל שהוא נחשף יותר ויוטר לדגימות מנוי הסוגים.

במהלך האימון, GENERATOR וDISCRIMINATOR מתחרים זה בזה באופן בו GENERATOR מנסה להונאות את DISCRIMINATOR על ידי יצירת דגימות הדומות כמה שיותר לא-אמיות, והDISCRIMINATOR משתפר ביכולתו להבחין בין אמיתי למזויף. המטריה הכללית היא להגיע לשינוי משקל, שבו GENERATOR יוצר דגימות שהDISCRIMINATOR לא יוכל להבחין בהן מדגימות אמיתיות, מה שמעיד על כך שהGENERATOR למד היטב את התפלגות הנתונים.



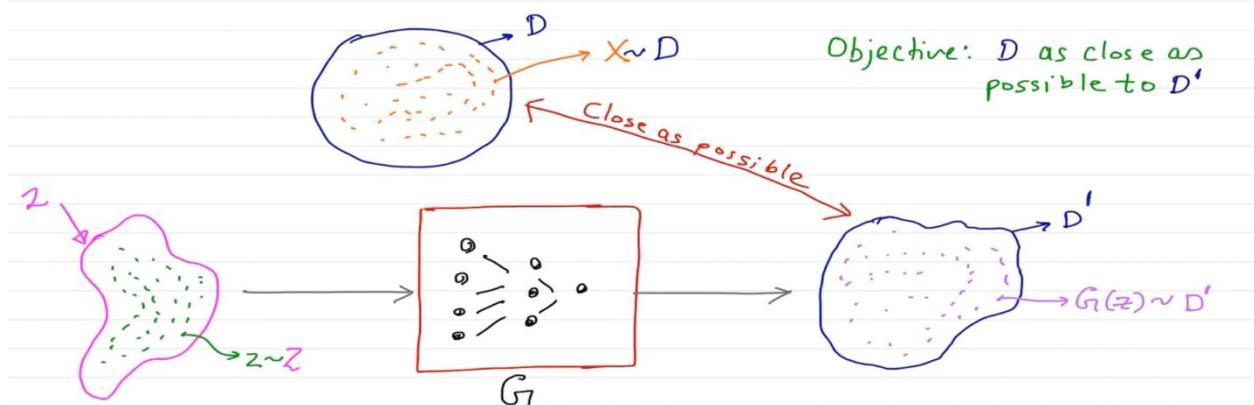
אופן פעולה ה Generator

- על מנת להבין את אופן פעולה הגנרטור, נרצה לתאר את התפלגיות המעורבות בפעולת הגנרטור:
1. Z – ניתן להסתכל על Z כעל מרחב לטני המתפרק למ��טפלוגות רנדומית (למשל גאוסיאן)
 2. D – ההתפלגות המקורית שמתארת את התצפיות העולם האמיתי
 3. D' – ההתפלגות המתבקשת כתוצאה מההעברה סך כל הווקטורים הלטנטיים בגנרטור לאחר תהליכי האימון

כדי להמיחס את ההתפלגיות ניתן דוגמאות ממשיות לכל אחת מההתפלגיות על בסיס MNIST:

1. Z – ההתפלגות מתארת וקטורים מימייד כלשהו הקטן מימייד התמונהות ב-MNIST Dataset
2. D – התמונהות ב-MNIST, כל תמונה $D \sim x$ לקויה מתוך ההתפלגות המתוארת כ- D' כלו
3. D' – התמונהות שהתקבלו כתוצאה מההעברה סך כל הווקטורים הלטנטיים בגנרטור. למעשה מתקבלות תמונהות שדומות לתמונהות הלקוחות מההתפלגות המקורי אך לא זהות אליהן לחלוין

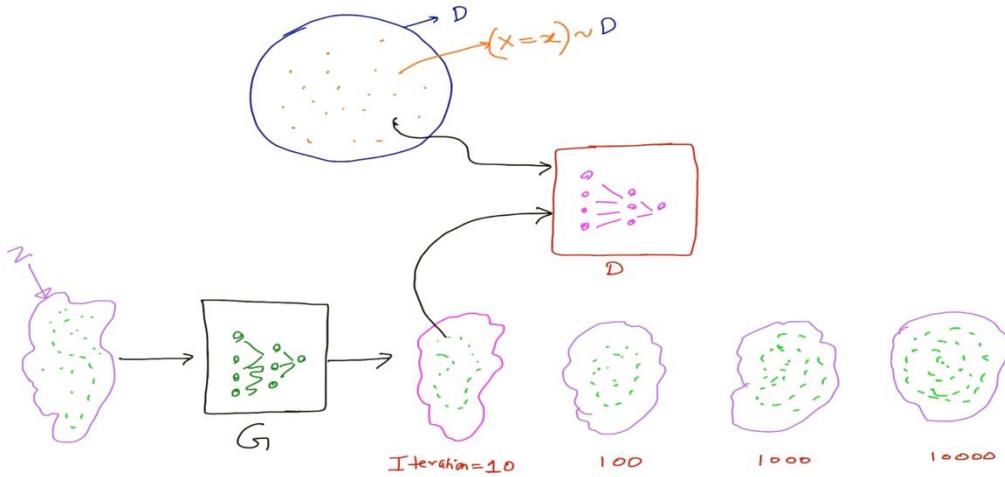
בתרשים הבא ניתן לראות את אופן פעולה רכיב הגנרטור ב-GAN:



תחילה ניתן לראות כי אנו דוגמים $Z \sim z$ מתוך ההתפלגות הרנדומית. לאחר מכן z עובר לאורק כל שכבות הגנרטור כך שמתקבל $D' \sim G(z) = x$ המהווה דוגמה מתוך ההתפלגות D' שהגנרטור מייצר באופן פעולתו. המטרה היא להביא לקירוב בין ההתפלגיות D' ל D דבר שייעיד על כך שהגנרטור מצילח לייצר דוגמאות אשר נראות כלקוות מתוך ההתפלגות האמיתיות ובכך יצליח להרים על הדיסקרימינטור כפי שיפורט בהרחבת המשך.

Overview על אופן פעולה GAN לאורך האיטרציות

לאחר שתיארנו את אופן פעולה הגנרטור והאופן בו הוא מנסה ליצור התפלגות שדומה ככל הנימין לההתפלגות המקורי, נרצה להציג כי ההליך הזה מתבצע בצורה איטרטיבית לאורך תהליך האימון. כמפורט בתמונה, תחילת התפלגות שמתהווה על ידי הגנרטור היא התפלגות שרחוקה מההתפלגות המקורי. לאורך תהליך הלמידה עם התקדמות באיטרציות, הגנרטור למד רפרזנטציות מדוייקות יותר של ההתפלגות המקורי בהתאם להחלטות הדיסקרימינטור.



- נרצה לפרט מספר נקודות שיבילו להעמקת ההבנה על האופן בו הארכיטקטורה פועלת:
1. מהסיבה שבעת תחילת האימון קיימים בפנינו דוגמאות מההתפלגות המקוריים וכמו כן, קיימת גם דוגמאות שאיגונרטו על ידי הגנרטור, אנו יכולים לティיג את הדוגמאות מההתפלגות המקוריות כאמיתיות (Real) ואת הדוגמאות שייצרו על ידי הגנרטור כمزוייפות (Fake).
 2. בשלב הראשון אנו רוצים לאמן את הדיסקרימינטור לסוג את הדוגמאות לאמיתיות ומזויפות. למען האמת זה משמשת קלסיפיקציה קלה מהסיבה שהגנרטור מפיק דוגמאות שאינן קרובות כלל להתפלגות המקוריים מהסיבה שהוא עדין לא עבר תהליכי אימון. לאחר סיום שלב זה יש לנו דיסקרימינטור שיזען לסוג את הדואטא בצורה ד' π משכנעת.
 3. בשלב הבא אנחנו מתחילהם באיטרציות האימון עבור הגנרטור באופן בו הדוגמאות שמיוצרות על ידי נראות יותר ויתר ככלו הלקוחות מתוך ההתפלגות המקוריים מכל שמתקדמים באיטרציות. האופן בו הגנרטור למד הוא על סמך ה loss שמחושב לאחר קלסיפיקציית הדיסקרימינטור, ככל שהדיסקרימינטור יהיה מסוגל להבחין בклות רבה יותר בדוגמאות המזויפות שנוצרו על ידי הגנרטור, כך צעד העדכון ב **Backpropagation** יהיה חזק יותר.
 4. ממשיך את תהליך האימון לאורך האיטרציות ובכך נקרב את התפלגות הגנרטור 'D' להתפלגות המקורית D. (כפי שראים בתמונה היא הולכת ומתעלגת)

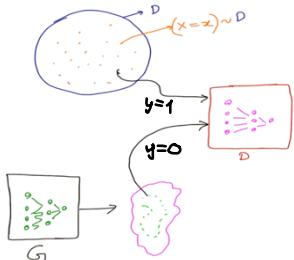
נקודות חשובות לשים אליהן לב:

1. האימון של הדיסקרימינטור ושל הגנרטור יבוצע בנפרד באופן בו הדיסקרימינטור יאמן בנפרד בעוד הגנרטור מקובע, והגנרטור יאמן בנפרד בעוד הדיסקרימינטור מקובע.
2. המטריה לאורך תהליכי האימון הוא שהדיסקרימינטור יספק ערך פרדי-קייזה זהים עבור שתי המחלקות – האמיתיות והמזויפות. ערכי הפרדי-קייזה יעמדו על 0.5 בקירוב ויעידו על כך שהגנרטור מצילח ליצור דוגמאות מתוך ההתפלגות שיצור שדומה מאוד להתפלגות המקורי.

Discriminator Loss Function and Objective

ה Loss Function של הדיסקרימינטור הוא Binary Cross Entropy לרוב כ-KLao. בעיות קלסיפיקציה ביןארית. Binary Cross Entropy נראה כי י מייצג את התיאוג האמתי ו י מייצג את ערך הפרדיקציה שבמקרה שלנו מקורה בפרדיקציית הדיסקרימינטור.

$$L(\hat{y}, y) = [y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$



- במקרה שלנו הדיסקרימינטור מספק פרדיקציות עבור אחד משני מצבים:
1. הדיסקרימינטור מספק פרדיקציות עבור דגימות הלקוחות מההתפלגות האמיתית – במקרה זה ערך y הוא 1
 2. הדיסקרימינטור מספק פרדיקציות עבור דגימות הלקוחות מההתפלגות הгенרטור – במקרה זה ערך y הוא 0

- חשב לנו כי כעת שנרשום ס הכוונה היא לדיסקרימינטור

לפיכך, ניתן לתאר את פונקציית loss במקרה זה ככזו התלויה בשני מקרים אל:

מקרה 1 – הפרדיקציות לקוחות מההתפלגות האמיתית – $L(D(x), 1)$
עבור מקרה זה נקבל כי למעשה פונקציית loss מוגדר כ $L(D(x), 1)$ באופן בו ערך הפרדיקציה הוא ערך ההחזרה של הדיסקרימינטור על x הלקוח מתוך ההתפלגות המקורית. כשנطען במשווה נקבל:

$$L(D(x), 1) = y \log \hat{y} + (1 - y) \log (1 - \hat{y}) = 1 \log D(x) + 0 \log(1 - D(x)) = \log D(x)$$

מקרה 2 – הפרדיקציות לקוחות מההתפלגות הгенרטור – $L(D(G(z)), 0)$
עבור מקרה זה נקבל כי למעשה פונקציית loss מוגדר כ $L(D(G(z)), 0)$ באופן בו ערך הפרדיקציה הוא ערך ההחזרה של הדיסקרימינטור על z הלקוח מתוך ההתפלגות הгенרטור. כשנטען במשווה נקבל:

$$L(D(G(z)), 0) = y \log \hat{y} + (1 - y) \log (1 - \hat{y}) = 0 \log D(G(z)) + (1 - 0) \log(1 - D(G(z))) = \log(1 - D(G(z)))$$

נראה כי קיבלנו ביטוי ל Objective של הדיסקרימינטור המורכב מ-2 ביטויים. נראה כי המטרה של הדיסקרימינטור הוא למקסם את ערך שני הביטויים על מנת להיות אופטימלי.
הסיבה למקסימיזציה נועוצה בכךון למינם את loss-Loss באמצעות מקסימיזציה של the-objective,
למעשה הדיסקרימינטור צריך לספק ערכי 1 עבור כל הערכים הלקוחים מההתפלגות האמיתית וערך 0 עבור כל הערכים הלקוחים מההתפלגות הгенרטור דבר היוביל לתוצאות הבא:

$$1. 0 = \log(1 - D(x)) \rightarrow \max_D \log D(x)$$

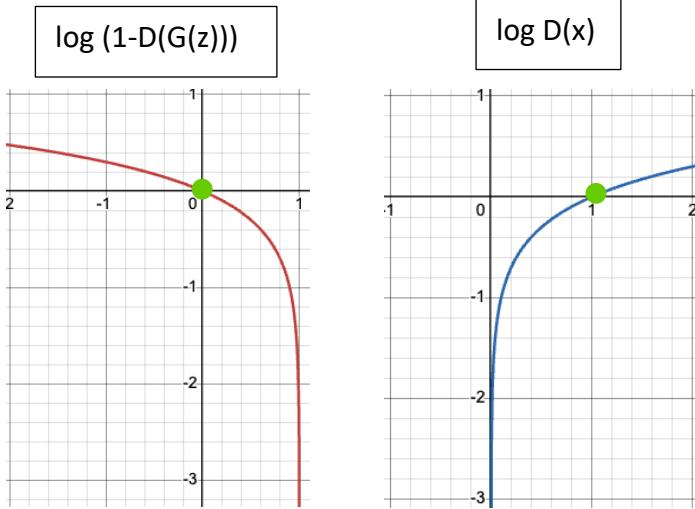
$$2. 0 = \log(1 - D(G(z))) \rightarrow \max_D \log(1 - D(G(z)))$$

נשים לב כי כל ערך אחר שאינו 1 עבור פרדיקציית הדיסקרימינטור עבור התפלגות המקורית ו-0 עבור פרדיקציית הדיסקרימינטור עבור התפלגות הgentrator הינה מובילה לכך ערך loss גדול מ-0.
סכום הכל כי the-objective של הדיסקרימינטור מוגדר להיות:

$$\max_D \log(D(x)) + \log(1 - D(G(z)))$$

משמעות גרפית של ה-objective של Discriminator

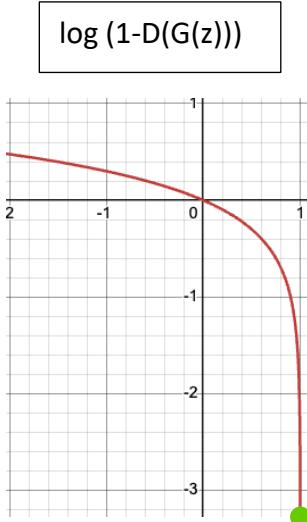
לאחר שתיארנו בצורה מילולית וסבירנו מוטיבציה למיקסום ה-objective הדרוש למינימיזציה ערך ה-loss והתכונות הדיסקרימינטור נרצה כעת לספק תיאור גרפי.



הגרפים המוצעים לשמאלי מתארים את הפונקציות המתמטיות הקשורות בערך objective. ניתן לראות כי הגרף הימני מתאר את הביטוי התלוי בערכי פרדיקטיביות הדיסקרימינטור עבור התפלגות המקורית. מהסיבה שהפונקציה מונוטונית עולה וחסומה בין 0 ל-1 נראה כי היא מקבל ערך מקסימלי כאשר ערך הפרדיקטיבית הוא 1. כמו כן הגרף השמאלי מתאר את הביטוי התלוי בערכי הפרדיקטיבית עבור התפלגות הגנרטור. מהסיבה שהפונקציה מונוטונית יורדת חסומה בין 0 ל-1 אז היא מקבלת ערך מקסימלי כאשר הפרדיקטיבית 0.

לפיכך, על סמך התצוגה הגרפית ועל סמך התיאור המילולי ניתן לומר כי סיפוק פרדיקטיביות של 1 עבור התפלגות המקורית ו-0 להתרגלות הגנרטור תוביל להשאפת ערכי הפרדיקטיבית אל עבר הנקודות הירוקות ובכך תוביל למיקסום objective הדיסקרימינטור.

Generator Loss Function and Objective



לאחר שתיארנו את ה-objective עבור הדיסקרימינטור נרצה לתאר את ה-objective עבור הגנרטור. על סמך אותו הගיון שהנחה אותנו למיקסם את ה-objective של הדיסקרימינטור, נראה כי הרצון כעת הוא למנמם את ה-objective של הגנרטור. הנחת היסוד היא שהמטרה של הגנרטור היא להערים על הדיסקרימינטור באופן בו הדיסקרימינטור יתבלבל ויחשוב שהדגימות שמייצרת על ידי הגנרטור הן אמיתיות ולכן ישן להן ערך פרדיקטיבית גבוהה וושאפים ל-1. מטעינה 1 בערך המשווהה נקבל את הביטוי הבא: $\log(0) = \log(1 - 1) = \log(1 - D(G(z)))$

נשים לב כי במקרה בו הדיסקרימינטור טועה בprdikativa שלו על דגימה מתווך בתוך ה-loss וכן ערך הלוג ישאף לא- ∞ . במקרים אחרים, ככל שהגנרטור יצליח להערים יותר על הדיסקרימינטור כך הוא יוביל לערכי פרדיקטיביות שמתאימים להתרגלות המקורית וכן מטרת הגנרטור היא למנמם את ערכי ה-objective. סיכום, ניתן לומר כי ה-objective של הגנרטור מוגדר להיות:

$$\min_G \log(D(x)) + \log(1 - D(G(z)))$$

GAN Objective

לאחר שפירטנו על ה-objective של הדיסקרימינטור והגנרטור בנפרד נרצה לתאר את ה-objective של GAN אשר מוגדר להיות מבחינה לא פורמלית:

$$\min_G \max_D \log(D(x)) + \log(1 - D(G(z)))$$

מבחינה מילולית כפי שתואר, מטרת הדיסקרימינטור היא מיקסום objective ומטרת הגנרטור היא מיקסום objective וכאן ה-objective הכללי של GAN מתייחס ל-objective של שני הרכיבים.

במאמר המקורי עם זאת ה-objective תואר באופן הבא:

$$\min_G \max_D V(D, G) = \min_G \max_D \left\{ \mathbb{E}_{x \sim p_{data}(x)} \log D(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z))) \right\}$$

ניתן לראות כי הכתיבה מעט שונה אבל המשמעות זהה ודומה למשמעות objective הכללי שתואר. כמו כן ניתן לב לשקלות $V(D, G) = \mathbb{E}_{x \sim p_{data}(x)} \log D(x) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z)))$ כלומר הפונקציה V מתארת את ביטוי התוצאות המופיע בצד ימין.

עת נסתכל על שני הביטויים הנוספים:

1. $\mathbb{E}_{x \sim p_{data}(x)} \log D(x)$ – הביטוי מייצג את התוחלת של הלוגריתם של הסיכוי שהדיסקרימינטור יחשב שדגימה x שנלקחה מההתפלגות של הנתונים האמיתיים p_{data} לכך אמיתית.

2. $\mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z)))$ – הביטוי מייצג את התוחלת של הלוגריתם של ההסתברות שהדיסקרימינטור לא יצליח לזהות את שדגימה שהגנרטור יוצר מהווקטור z הנלקח מההתפלגות הרנדומית p_z היא מצויה.

- הנקודת החשובה לקחת היא שכל תכליתו של objective היא משחק $\min \max$ בין הדיסקרימינטור לגנרטור באופן בו מטרת הגנרטור היא להערים על הדיסקרימינטור באופן בו הוא יתקשה לזהות את הדגימות שיצר מההתפלגות הרנדומית ויחשוב שאלה נוצרו מההתפלגות המקורית. מצד שני, מטרת הדיסקרימינטור היא לאגלות מה הן הדגימות המזויות שיוצרים על ידי הגנרטור.

- בתהליך איטרטיבי של שיפור הדיסקרימינטור והגנרטור באופן נפרד, השיפור הנפרד של כל אחד מהם מוביל לשיפור גם ביכולותיו של الآخر כך שהמטרה הסופית היא שיפור ממשמעותי ביכולותיו של הגנרטור באופן בו הוא בסופו של דבר יצליח ליצור דגימות הדומות ככל הנימין להתפלגות המקורית.

Optimal Discriminator

از כאמור, כפי שהזכירנו מספר פעמים ישנו קרב מתמיד בין הדיסקרימינטור לגנרטור. נרצה להגדיר מושג חדש: דיסקרימינטור אופטימלי. הדיסקרימינטור אופטימלי הוא הדיסקרימינטור אליו מגיעים בהינתן גנרטור מסוובע. יש לציין כי השם כאן מעט מבלב אלל האופטימלית של הדיסקרימינטור היא ביחס ל-GAN כמודל אמיתי ולא ביחס לדיסקרימינטור כלעצמו כאופטימלי. לעומת זאת, אין הדבר שהדיסקרימינטור אופטימלי מהסיבה שהסיווג שלו מושלם.

ניגש להגדירה המתמטית, בהינתן גנרטור מסוובע, דיסקרימינטור מסוובע יקיים:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

לפנינו שנצלו להוכחה המתמטית נרצה דוקא לספק מוטיבציה שתסביר את התלות של דיסקרימינטור אופטימלי בקיום התלות.

לטובות ישור הכו נתאר את שני הביטויים הבאים:

1. $p_{data}(x)$ – צפיפות ההסתברות של הדיסקרימינטור עבור הדגימות האמיתיות
2. $p_g(x)$ – צפיפות ההסתברות של הדיסקרימינטור עבור דגימות שיוצרו על ידי הגנרטור

עבור מצב בו $0.2 = p_g(x) = p_{data}(x)$ נקבל כי $D_G^*(x) = 0.8$. כלומר עבור הדגימה הספציפית הדיסקרימינטור האופטימלי משער כי בהסתברות של 80% הדגימה שייכת להתפלגות האמיתית. ככל שהערך קרוב יותר ל-1 כהה הדיסקרימינטור מצביע על כך שהדगימה שייכת להתפלגות האמיתית וככל שהערך קרוב יותר ל-0 כך הדיסקרימינטור מצביע על כך שהדגימה מזויפה.

באופן כללי, נרצה לקבל $\frac{1}{2} \approx D_G^*(x)$ עבור רוב הדגימות, דבר שצביע על כך שהדיסקרימינטור "מבלבל" והוא נוטה לשערق את הדגימות האמיתיות והמזויפות באותו אופן. דבר שירמז על כך שהגנרטור הצליח להעתים על הדיסקרימינטור ומכאן שהגנרטור הצלח ללמידה ופרצנטציות נכונות של התפלגות המקורית.

הוכחה מתמטית

השאלה הנשאלת היא בהינתן גנרטור מקבוע, עבור אילו ערכים (x) D_G^* מקבל ערך מקסימלי. ופורמלית:

$$D_G^*(x) = \arg \max_D V(D, G) = \arg \max_D \mathbb{E}_{x \sim p_{data}(x)} \log D(x) + \mathbb{E}_{z \sim p_z(z)} \log (1 - D(G(z)))$$

עד כה אין חידוש, למשווה תיארנו את המשווה המוכרת והשינוי היחידי שביצענו הוא לקיחת ערכי argmax מהסיבה שהיא שמעניין אותנו זה אילו ערכים מקסימים את הביטוי.

מהגדרת התוחלת נקבל: $\mathbb{E}_{p(x)}[x] = \int_x x p_x(x) dx$ וכן לאחר שנעשה פלאג לביטוי במשווה נקבל:

$$V(D, G) = \int_x p_{data}(x) \log D(x) dx + \int_z p_z(z) \log (1 - D(G(z))) dz$$

גם כאן, אין חידוש ושום דבר מורכב, בסך הכל עשינו פלאג להגדרת התוחלת לשווהות המתארת את הפונקציה V .

лемה 1

על מנת להמשיך ולהתקדם בהוכחה נרצה להוכיח את השקילות הבאה על מנת לקבל ביטוי שתליי ב- x בלבד ואיןו תלוי ב- z .

$$\int_z p_z(z) \log (1 - D(G(z))) dz = \int_x p_g(x) \log (1 - D(x)) dx$$

השקלות זו מתקיימת הודות לקונספט הנקרא "החלפת משתנה" והוא מוגדר באופן הבא:
בhinint מ�טנה מקרי X המוגדר (x) מ נתן לחשב את פונקציית צפיפות ההסתברות של משתנה
כלשהו אחר Z התלוי ב- X ומוגדר להיות $(X) = f = Z$ באופן הבא: $p_x(f^{-1}(y)) = p_z(y)$.
הסיבה לכך היא מאחר ש- f היא פונקציה הפיכה אז הפעלה על איברים מהטוויה טוביל לקבלת איברים מהתחום. במקרה שלנו, $(y)^{-1} f$ למעשה נותן ערך שקול לא כלשהו מהתחום.

$$\begin{aligned} z &\xrightarrow{\text{פונקציית}} G(z) \\ &= x \end{aligned}$$

$$\begin{aligned} p_z(z) &= p_x(G^{-1}(x)) \frac{dx}{dz} \end{aligned}$$

במקרה שלנו, Z מייצג את הוקטור שנלקח מההטפלות הרנדומית בעלת פונקציית צפיפות ההסתברות (z) ומועבר אל הגנרטור לצורך קבלת דוגימה מתוך הטפלות הגנרטור. מהסיבה שהגנרטור הוא בסופו של דבר רשות ניירונים ולפיכך על פניו ניתן לפישוט כפונקציה הפיכה (מבחינה תיאורית ניתן להרצה מהסוף להתחילה) אז ניתן להחיל את הקונספט של "החלפת משתנה" ובכך לומר כי $(z) = p_z\left(\frac{dG^{-1}(x)}{dx}\right) = p_g(x)$.

- דבר שחשוב לציין הוא ש $G(z) = x$ הוא לא x הלווה מטור ההתפלגות המקורית אלא דוגימה שיוצרה על ידי הגנרטור. אנו מסוגלים לעשות את הקירוב הזה תחת ההנחה שבסוף תהליך האימון התפלגות הגנרטור וההתפלגות המקורית קרובות באופן יחס.

כעת מה שנרצה לעשות, באמצעות הבדיקה שביצענו נרצה לצאת מ $\int_z p_z(z) \log(1 - D(G(z))) dz$
ולגיע לביטוי השקל המוגדר כ $\int_x p_g(x) \log(1 - D(x)) dx$

$$\begin{aligned} \int_z p_z(z) \log(1 - D(G(z))) dz &= \int_x p_z(G^{-1}(x)) \log(1 - D(x)) dG^{-1}(x) \\ &= \int_x p_z(G^{-1}(x)) \log(1 - D(x)) \frac{dG^{-1}(x)}{dx} dx \end{aligned}$$

כמה דברים חשובים לשים אליהם לב:

1. המעבר בין $p_z(z) G^{-1}(x)$ מאפשר בעקבות ההגדרה ש $G(z) = x$ ומכאן $z = G^{-1}(x)$
2. המעבר בין $1 - D(x)$ ל $1 - D(G(z))$ מאפשר בעקבות ההגדרה ש $G(z) = x$
3. המעבר בין dz ל dx מגיע בעקבות כלל שרשרת. שהרי בהכללה $dz = \frac{dg}{dx} dx$

הדבר הבא שנרצה לעשות הוא לחת את שני הביטויים ולהגיע באמצעותם אל השקילות הסופית:

$$\int_x p_z(G^{-1}(x)) \log(1 - D(x)) \frac{dG^{-1}(x)}{dx} dx .1$$

$$.p_g(x) = p_z(G^{-1}(x)) \frac{dG^{-1}(x)}{dx} .2$$

מאחר החלק ימני של הביטוי אינו תלוי ב $(x)^{-1} G$ וכן כן לאחר ופעולת הכפל הינה פועלה קומוטטיבית אז ניתן לכתוב את הביטוי בצורה שונה באופן הבא:

$$\int_x p_z(G^{-1}(x)) \log(1 - D(x)) \frac{dG^{-1}(x)}{dx} dx = \int_x p_z(G^{-1}(x)) \frac{dG^{-1}(x)}{dx} \log(1 - D(x)) dx$$

ונכל לשים לב כי החלק האדום שקול ל $p_g(x)$ ולכן סך הכל נקבל:

$$\begin{aligned} \int_x p_z(G^{-1}(x)) \frac{dG^{-1}(x)}{dx} \log(1 - D(x)) dx \\ = \int_x p_g(x) \log(1 - D(x)) dx = \int_z p_z(z) \log(1 - D(G(z))) dz \end{aligned}$$

סוף למה 1

לאחר שהוכחנו את הלמה נרצה להמשיך ולהתקדם בהוכחת הדיסקרימינטור האופטימלי.

הביטוי הבא הוא ביטוי שאנו כבר מכירים ומתאר את החישוב של התוחלות:

$$V(D, G) = \int_x p_{data}(x) \log D(x) dx + \int_z p_z(z) \log (1 - D(G(z))) dz$$

נחליף את dz $\int_x p_g(x) \log (1 - D(x)) dx = \int_z p_z(z) \log (1 - D(G(z))) dz$ (זו השקלות שהוכחנו בلمה 1) ונקבל:

$$V(D, G) = \int_x p_{data}(x) \log D(x) dx + \int_x p_g(x) \log (1 - D(x)) dx$$

마וחר ומדובר באינטגרל מעל x ניתן לכנס את האינטגרלים ולקבל:

$$V(D, G) = \int_x p_{data}(x) \log D(x) + p_g(x) \log (1 - D(x)) dx$$

על מנת לקבל ערך עבור הדיסקרימינטור האופטימלי נרצה לבדוק מה הערך שמקסם את הערך שמצוין בתוך האינטגרל: $p_{data}(x) \log D(x) + p_g(x) \log (1 - D(x)) dx$. לשם כך נרצה לגזר את הביטוי זהה ונקבל:

$$\frac{p_{data}(x)}{D(x)} - \frac{p_g(x)}{1 - D(x)}$$

- חשוב לציין כי יש לבצע בדיקה באמצעות נגזרת כפולה כדי לוודא שאכן מדובר בנקודת מקסימום ולא בנקודת מינימום.

לאחר שנכנסו איברים נקבל את הביטוי הבא שכבר מוכר לנו:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

סך הכל קיבלנו כי מיקסום ה-objective מוביל לכך דיסקרימינטור אופטימלי המקיים את הביטוי.

Optimal Generator

از לאחר שדיברנו על הדיסקרימינטור האופטימלי נרצה לדבר קצת על הגנרטור האופטימלי. יש לציין שאופטימליות הגנרטור היא אף יותר חשובה בסופה של דבר מהסיבה שבפועל inference time הוא המודל שייצר את הדגימות.

כפי שקיבענו את הגנרטור בתהילך ההסקה לפני הדיסקרימינטור האופטימלי, כאן נקבע את הדיסקרימינטור באופן בו הדיסקרימינטור שנקבע הוא הדיסקרימינטור האופטימלי.

כאמור, בגנרטור יש רצון למנם את objective ולכן המטרה היא למצוא גנרטור G^* המקיים:

$$G^* = \arg \min_G V(D_g^*, G)$$

מהסתכלות על ביטוי התוצאות (בשילוב הלמה) נקבל:

$$G^* = \arg \min_G \int_x p_{data}(x) \log D_G^*(x) + p_g(x) \log (1 - D_G^*(x)) dx$$

כעת נעשה פלאג ל D_G^* אל תוך המשווה ונקבל:

$$G^* = \arg \min_G \int_x p_{data}(x) \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) + p_g(x) \log \left(1 - \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \right) dx$$

על מנת להמשיך בפתרונות ולהגיע להתכנסויות בהמשך נרצה לבצע טרייק מתמטי והוא הוספה והחסרה של $p_g(\log 2)$ ($\log 2$) p_{data} , ($\log 2$) p_g :

$$\begin{aligned} G^* = \arg \min_G \int_x & (\log 2 - \log 2) p_{data} + p_{data}(x) \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \\ & + (\log 2 - \log 2) p_g + p_g(x) \log \left(1 - \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \right) dx \end{aligned}$$

- מה שמופיע בירוק הוא הוספה והחסרה של הביטוי תוך כינוס איברים

ובוצע עוד כינוסים מתמטיים ונקבל:

$$\begin{aligned} G^* = \arg \min_G \int_x & -\log 2 \left(p_{data}(x) + p_g(x) \right) + p_{data}(x) \left[\log 2 + \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \right] \\ & + p_g(x) \left[\log 2 + \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \right] dx \end{aligned}$$

נרצה לפרק את הביטוי ל 3 אינטגרלים נפרדים לטובת המשך העבודה:

$$\begin{aligned} G^* = \arg \min_G -\log 2 & \int_x \left(p_g(x) + p_{data}(x) \right) dx \\ & + \int_x p_{data}(x) \left[\log 2 + \log \left(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right) \right] dx \\ & + \int_x p_g(x) \left[\log 2 + \log \left(\frac{p_g(x)}{p_g(x) + p_{data}(x)} \right) \right] dx \end{aligned}$$

- פירוק האינטגרל לחלקים מתרכחת בעקבות תכונת לינאריות האינטגרל: $p(x)dx + q(x)dx = \log 2 + \log(p(x)/q(x))dx$
- כמו כן באינטגרל הראשון ישנה הוצאה של $\log 2$ החוצה מהביטוי שהוא אין תלוי ב- x

נמשיך ונבצע פישוט, נשים לב כי עבור 3 הרכיבים, רכיב 1 הוא ביטוי קבוע סקלארי. רכיבים 2,3 למשהו
הם ביטוי של KL Divergence שהם מהצורה $\int p(x) \log \left(\frac{p(x)}{q(x)} \right)$
המשמעות היותר عمוקה של

$$\begin{aligned} G^* = \arg \min_G -\log 2(1+1) & + \int_x p_{data}(x) \log \left(\frac{\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}}{\frac{2}{2}} \right) dx \\ & + \int_x p_g(x) \log \left(\frac{\frac{p_g(x)}{p_g(x) + p_{data}(x)}}{\frac{2}{2}} \right) dx \end{aligned}$$

- המעבר בין $\int_x (p_g(x) + p_{data}(x)) dx = -\log 2(1+1) - \log 2 \int_x (p_g(x) + p_{data}(x)) dx$ מתקיים כתוצאה מהגדרת פונקציית צפיפות ההסתברות שהרי $\int_x p(x)dx = 1$
- המעבר עבור רכיבים 2,3 מתקיים באמצעות השקילות $\log(AB) = \log(A) + \log(B)$

לאחר שacen זיהינו כי אכן מדובר ב-**KL Divergence** נוכל לכתוב את הביטוי באופן הבא:

$$G^* = \arg \min_G -\log 4 + KL \left[p_{data}(x) \parallel \frac{p_{data} + p_g}{2} \right] + KL \left[p_g(x) \parallel \frac{p_g + p_{data}}{2} \right]$$

כמו כן הביטוי $KL \left[p_{data}(x) \parallel \frac{p_{data} + p_g}{2} \right] + KL \left[p_g(x) \parallel \frac{p_g + p_{data}}{2} \right]$ הוא לא אחר מאשר JSD ולכן נקבל את הביטוי:

$$G^* = \arg \min_G \left\{ -\log 4 + 2 JSD \left(p_{data}(x) \parallel p_g(x) \right) \right\}$$

סימטרי עליי יפורט מעט יותר בהרחבה בהמשך. מה שחשוב לנו לדעת לרגע זה הוא sh-JSD אכן מוגדר להיות $JSD = \text{mean} \left(D_{KL} \left(P \parallel \frac{P+Q}{2} \right) + D_{KL} \left(Q \parallel \frac{P+Q}{2} \right) \right)$.

במקרה שלנו, מהסיבה ואנו מנסים להגעה לביטוי מינימלי עבור הגרנרטור $\min_G \arg \sum p_g \log p_g$ וכך נקבע:

$$G^* = \arg \min_G \{-\log 4 + 0\} = -\log 4$$

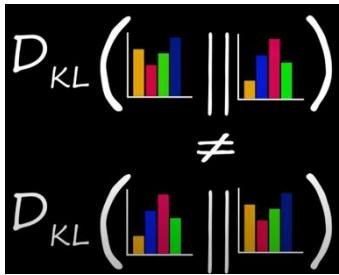
סך הכל נקבע כי עבור מצב בו $JSD = 0$ הערך התיאורי המינימלי שאנו מקבלים עבור הגרנרטור המינימלי הוא $-\log 4$. עם זאת, השורה התחתונה היא לא דוקא הערך הזה אלא העובדה שהוא מתקיים בעקבות העובדה שהובילנו למצב בו $p_g = p_{data}$ כלומר הגרנרטור האופטימלי שאף לקירוב ההסתברויות. מהסיבה sh-JSD יכול לקבל רק ערכים חיוביים, כל מצב אחר בו $p_g \neq p_{data}$ יוביל ל- $-\log 4 >$

כמו כן אנחנו יכולים לבצע הסתכלות מפרוספקטיבית שונה על הערך שקיבלנו – דוקא מזוויות הדיסקרימינטור. אם מסתכלים על הערך של (D_G^*, G) ניתן לראות כי עבור מצב בו $D_G^* = \frac{1}{2}$ אזי מתקובל הביטוי הבא:

$$\begin{aligned} V(D_G^*, G) &= \int_x p_{data}(x) \log \frac{1}{2} + p_g(x) \log \frac{1}{2} dx = \\ &= -\log 2 \int_x p_{data}(x) + p_g(x) dx = -\log 4 \end{aligned}$$

כזכור ההסתכלות היא שcolaה משני הכוונים, הגרנרטור שואף למצב בו $p_g = p_{data}$ כלומר ההסתגלויות שוות עבור הדאטא האמתי והדאטא שנוצר כתוצאה מהסתגלות הגרנרטור. במידה והוא יכול להגיעה למצב זה, הוא יכול להרים על הדיסקרימינטור ויגרום לו לפלוט ערכי פרדיקציה השווים ל- $\frac{1}{2}$ עבור כל דוגמה ומכך נקבל את המסקנה שהגרנרטור מייצר דוגמאות טובות שנראות כאילו לקוחות מהסתגלות האמיתית.

JSD (Jensen Shannon Divergence)



עד כה כאשר דיברנו על ממד למדידת מרחק בין שתי התפלגיות דיברנו על KL Divergence. עם זאת, נקודה חשובה לשחוב לשים אליה לב היא העובדה ש-KL Divergence הוא ממד שאינו סימטרי. ככלומר בהינתן שתי התפלגיות מסוימות P, Q , אין מתכאים עבורן כי: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$ ($JSD(P||Q) \leq D_{KL}(P||Q)$).

ממד JSD לוקח את העובדה זו בחשבון והוא מציג ערך אחד למרחק בין שתי התפלגיות באופן בו ישנה קומוטטיביות במיקומי התפלגיות בנוסחת הממד. האופן בו JSD עושה זאת הוא באמצעות קיצות ממוצע ערכי ה-KL Divergence עבור כל אחד מההתפלגיות. מתמטית:

$$JSD = \text{mean} \left(D_{KL} \left(P \parallel \frac{P+Q}{2} \right) + D_{KL} \left(Q \parallel \frac{P+Q}{2} \right) \right)$$

דוגמה:

$$\text{לטובת הדוגמא נגדיר } JSD(P||Q) = \frac{D_{KL}(P+M)+D_{KL}(Q+M)}{2} = M \text{ ומכאן: } \frac{P+Q}{2}$$

נסתכל על ה-*the-h*-chiontion Probability Distribution של 2 התפלגיות P, Q :

$$P = \{0.2, 0.3, 0.4\}, Q = \{0.3, 0.4, 0.3\}$$

מכאן:

$$M = \frac{P+Q}{2} = \frac{\{(0.2+0.3), (0.3+0.4) + (0.5+0.3)\}}{2} = \{0.25, 0.35, 0.4\}$$

cutet נסתכל על $JSD(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$ – KL Divergence ונחשב:

$$D_{KL}(P||M) = \left(0.2 \log \left(\frac{0.2}{0.25} \right) \right) + \left(0.3 \log \left(\frac{0.3}{0.35} \right) \right) + \left(0.5 \log \left(\frac{0.5}{0.24} \right) \right) = 0.1193$$

$$D_{KL}(Q||M) = \left(0.3 \log \left(\frac{0.3}{0.25} \right) \right) + \left(0.4 \log \left(\frac{0.4}{0.35} \right) \right) + \left(0.3 \log \left(\frac{0.3}{0.24} \right) \right) = 0.1109$$

לס"ום, נבצע מיצוע של שני העריכים ונקבל את הערך של JSD:

$$JSD(P||Q) = \frac{D_{KL}(P+M) + D_{KL}(Q+M)}{2} = \frac{0.1193 + 0.1109}{2} = 0.1151$$

גם כאן מבחינה מתמטית, נשים לב כי עבור מצב בו $P=Q$ ככלומר התפלגיות זהות, אז נקבל מצב בו ערכיו ה-KL Divergence יהוו 0 ומכאן נקבל כי JSD יקבל ערך של 0.

Transformers

[Video Link 1](#)

[Video Link 2](#)

הטרנספורמר היא אול' אחת הארכיטקטורות המשמעותיות ביותר ביזור בכמה השנים האחרונות. במאמר בשם *Attention Is All You Need* תיארו בשנת 2017 חוקרים מוגול את הארכיטקטורה שבאמצעות מנגנון-h-Attention שבה מסוגלת להتمודד עם Sequential Data ללא תופעת לוזאי וביעות שרואה אותה תקופה לגבי מודלים כמו M, LSTM ורומים.

מאז, הטרנספורמר שולב במספר ארכיטקטורות שמהוות כיום SOTA בתחום השפה (NLP), תמונה ואף ידאו מה שהופך אותו לבסיס למודלים מתקדמים בתחוםים אלו אשר הולכים ומתפתחים בקצב מהיר כמוision GPT, CLIP, Stable Diffusion ועוד הרבה נוספים. בין היתר, הארכיטקטורה מאפשרת ייצרת Autoregressive Model לטובות יצירת תוכן גנרטיבי של טקסט, ייצרת קישור בין תמונות וtekst או כל שני סוגים נתונים והבאתו לכדי ייצוג וקטורי אחד ווד.

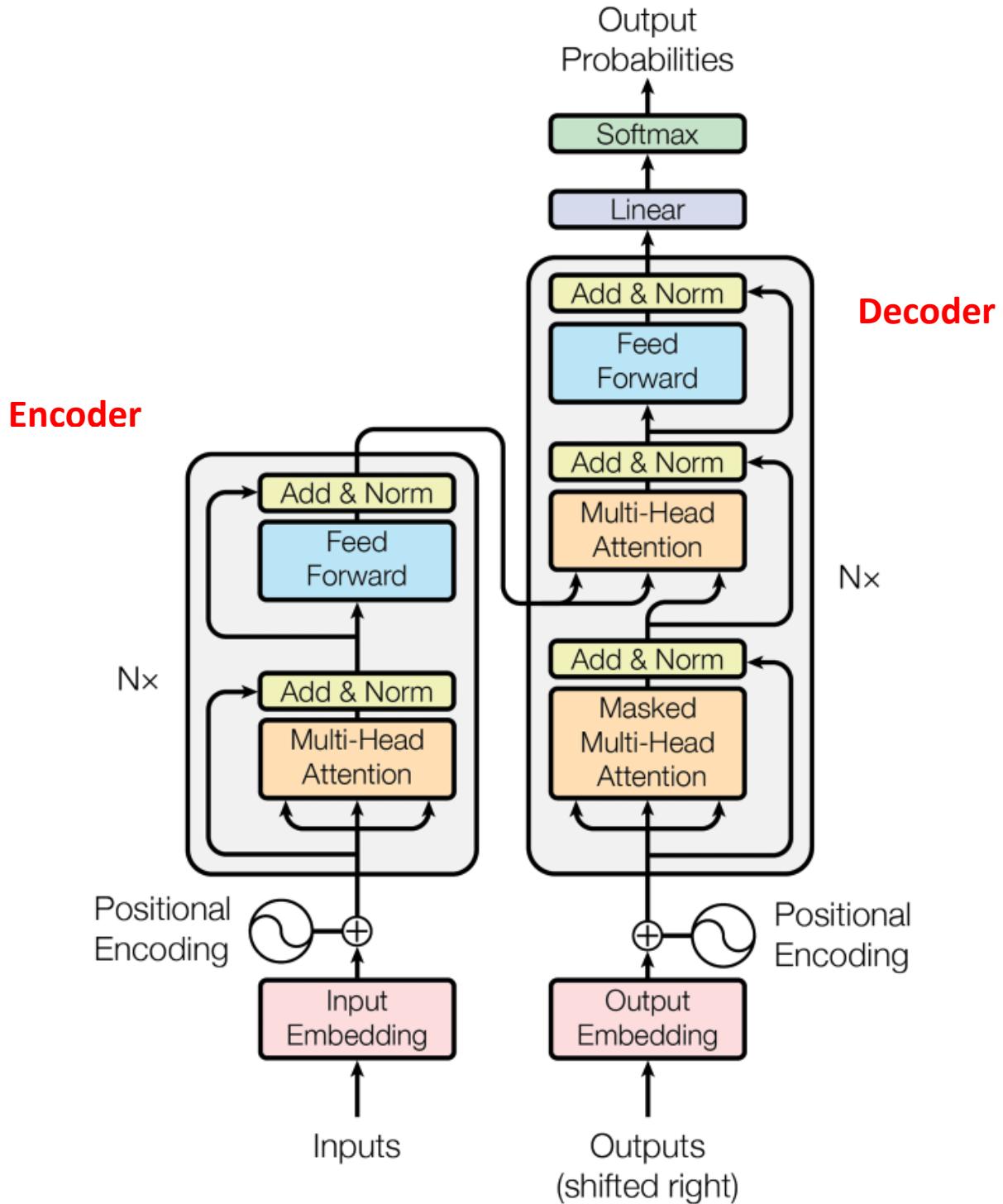
כפי שצווין, הקפיצה המחשבתית שהביא הטרנספורמר מבוססת בעיקר על יכולת להתמודד עם תלויות ארוכות טווח בין אלמנטים בדאטא, בזכות מנגנון-h-חסון המאפשר למודל להתמקד באזוריים המגליים תלויות רלוונטיות בדאטא שמודלים אחרים התקשו לזהויים.

חשוב לציין כי הארכיטקטורה מורכבת יחסית ובונה מספר לא קטן של קומפוננטות וכן תחיליה יתוארו חלק מהקומפוננטות ב-High Level לטובות הבנת החישבות הכלליות שלן ולאט לאור הסיכון אניATAR At Depth In יותר את הקומפוננטות.

חשוב לציין כי מאחר והאררכיטקטורה יחסית מורכבת לא אוכל לפרט לפרט פרטם קונספטים יותר יסודיים למען שמירה על הרצף אך עם זאתאגע בנושאים המצויים בארכיטקטורה וחשובים לטובות הבנתה.

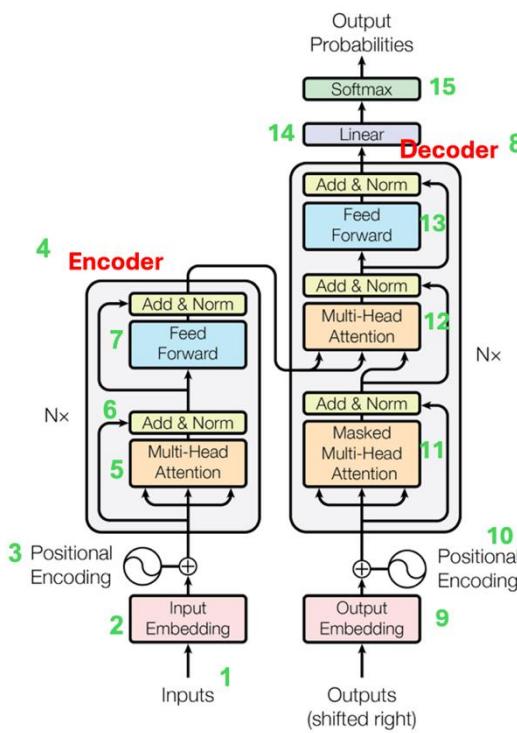
הדוגמה בה השתמש היא דוגמא לטרנספורמר אוטו-רגרטיבי כך שבהינתן פרומפט יוצר את התשובה על סמך פרדיקציה על המילה הבאה ברכף עד לכדי השלמת התשובה המלאה.

ארכיטקטורת ה-Transformer



Transformer Overview

נרצה כעת לתת תיאור High Level לקומפוננטות:



1. Inputs – הפלט המקורי של המודל המורכב מטוקנים שמקורם במילים.
2. Input Embedding – המפה של הטוקנים לייצוגים וקטוריים מספריים.
3. Positional Encoding – הוספה וקטוריית המיצגת את מיקום הטוקן בפרומפט.
4. Encoder – הרכיב בארכיטקטורה שמטרתו לקובד את המידע בפרומפט לכדי ייצוג מספרי מזוקק ומורכב מילויים. Encoder-Decoder. בפועל אין Encoder יחיד אלא ח Callable, כל Encoder הנסמן באוטו ז מעביר את הפלט שלו ל-Decoder-הו. לבסוף ה-Decoder יעביר את הפלט הסופי אל כל אחד מה-Decoders-ים.
5. Multi-Head Attention – הרכיב החשוב בארכיטקטורה. מורכב ממספר תתי יחידות הנקראות Self Attention. מטרתה של כל אחד מהתאי היחידות הללו זה למצוא טוקנים בעלי משמעות סמנטית והשפעה זה על זה.
6. Add & Norm – בשכבה זו מוסיפים את הערך של ה-Residual Connection לערך הפלט משכבות Multi Head Attention (Add). כמו כן מבצעים נורמליזציה על הערך הסופי באמצעות (Norm).
7. Feed Forward Neural Network – רשת נירונים לנמדת המקלט כקלט את פלט שכבת ה-Attention. מכילה שכבות לא לינאריות שמשמשות בלימידת הרפרזנטציות עבור המודל.
8. Decoder – הרכיב האוטו-רגרטיבי בארכיטקטורה שמטרתו ליצר בצורה איטרטיבית ורציפה Probability Distribution over the corpus לטובות בחירת המילה הבאה המתאימה ביותר. כמו ב-Encoder גם כאן יש ח שכבות של Decoder באוף בו פלט ה-Decoder-הו נכנס אל כל אחד מה-Decoders-ים ופלט כל ה-Decoder-הו עבר אלDecoder-הו+1. כך שלבסוף מקבלים ה-Decoder-הו שיציר את מה שהפוך להיות הפרדיקציה הסופית.
9. Output Embedding – ייצוג וקטורי מספרי של כל המילים שייצרו עד כה על ידי ה-Decoder.
10. Positional Encoding – הוספה וקטוריית המיצגת את מיקום הטוקן בפרומפט.
11. Masked Multi-Head Attention – שכבת Masked Multi-Head Attention. השוני שלה משכבות ה-Attention היא שהיא מודדת שפדי-קיציות עבור טוקן במיקום מסוים והוא קאוזליות ועל סמך טוקני עבר בלבד ולא על סמך טוקני עתידי שיכולים להטוט את תוצאות הבדיקה-קייזות.
12. Multi-Head Attention – שכבת Multi-Head Attention נוספת הפעם ב-Decoder שמקבלת כקלט את הפלט של ה-Encoder-הו וכמו כן את הפלט של Masked Multi-Head Attention לאחר נורמליזציה.
13. Feed Forward Neural Network – רשת נירונים לנמדת המקלט כקלט את פלט שכבת ה-Attention. מכילה שכבות לא לינאריות שמשמשות בלימידת הרפרזנטציות עבור המודל.
14. Linear – שכבה לינארית לטובות מיפוי פלט ה-Decoder לווקטור בגודל ה-Corpus.
15. SoftMax – לאחר שקיבלונו וקטורי בגודל ה-Corpus שמכיל בתוכו את המידע לגבי הפרדיקציה למילה הבאה, נבצע SoftMax לטובות הפיקת Chosptail Probability Distribution על ה-Corpus. על פניו המילה עם ההסתברות הגבוהה ביותר להיות המילה הבאה בקלט (עם זאת יש לוגיות).

Model's Input

הקלט של הטרנספורם יכול להיות מגוון אך לטובת המוחשת הארכיטקטורה אטמוך בקלט טקסטואלי. אך למעשה מחר והקלט הוא טקסטואלי, קרי פרומפט, אזי מורכב ממילים. המילים הללו בתהיליך טוקניזציה הופכים להיות טוקנים. טוקנים הם למעשה פיסת טקסט (לעתים גם מילה) שמייצגת יחידה תפקודית לטובת המודול. למשל את המילה working אפשר לפצל ל-2 טוקנים: work + ing.

```
In the vast realm of Natural Language Processing (NLP), text processing holds a fundamental position. It acts as the initial step in transforming raw textual data into a more digestible format for subsequent algorithms and models. Among the core techniques in text processing, **Tokenization** and handling of **Stopwords** are foundational. Let's delve into each.
```

דוגמאות ל- Tokenizations :
 Unbelievable: "un", "believ", "able"
 Transformation: "trans", "form", "ation"
 Intercontinental: "inter", "continent", "al"
 Misunderstanding: "mis", "under", "stand", "ing"

באופן כללי נוכל לתאר את התהיליך באופן הבא:

$$\forall word \in Prompts \rightarrow Tokens = tokenization(word)$$

כלומר הטוקנים הנשלחים אל המודול הן כל המילים המופיעות בפרומפט.

Input Embedding

לאחר שייצרנו את הטוקנים, אנו רוצים לקבל ייצוגים וקטוריים מסווגים. דבר זה נעשה באופן בו מתרחש מיפוי בין כל טוקן לבין idique בעקבות look up table ואחר מכן שליפה של הייצוג הווקטורי במקומות index-ים מתוך Embedding Matrix ייצוג הווקטורי המתאר את הטוקן.

$$W_E = \begin{bmatrix} \text{look} & \text{door} & \text{is} & \text{the} & \text{end} & \text{of} & \text{a} & \text{sentence} \\ -4.0 & +1.0 & +0.5 & -0.4 & -1.6 & +7.5 & 2.5 & -9.9 & -5.0 & -3.6 & \dots & +7.1 & -0.8 & -1.1 & -3.2 & +7.5 & +8.5 & +9.7 & -2.4 & +0.2 & +6.8 \\ +3.5 & -0.1 & -5.6 & -6.8 & +0.4 & -4.1 & -0.9 & -0.1 & +0.5 & +0.8 & \dots & -7.1 & -1.4 & -6.8 & +6.3 & -7.9 & -6.5 & -3.9 & -8.4 & +1.5 & -7.8 \\ -1.4 & -0.6 & +1.9 & -7.8 & +0.4 & +0.6 & -2.1 & -5.1 & 4.9 & 0.3 & \dots & -9.1 & +2.8 & -1.8 & -2.4 & +3.1 & +4.1 & -9.0 & 2.9 & -7.9 & -4.3 \\ -2.8 & -2.4 & -4.2 & -7.4 & -7.7 & -0.7 & -6.3 & -1.9 & -4.9 & -0.5 & \dots & -0.2 & -9.9 & -1.5 & -8.6 & -5.8 & +8.6 & -5.6 & +7.1 & +0.0 & -0.7 \\ -2.1 & -7.6 & 14.5 & 12.7 & 16.2 & -0.4 & 8.2 & 8.9 & -4.1 & 14.3 & \dots & -1.6 & -6.5 & 7.8 & 16.3 & 0.5 & 7.6 & 4.6 & 1.8 & 2.5 & 0.5 \\ -7.7 & +4.7 & -0.8 & +3.8 & +1.3 & -4.2 & -6.4 & -0.3 & -7.1 & -2.8 & \dots & +8.7 & +0.4 & -4.3 & -3.2 & +2.0 & +0.2 & -7.0 & -4.8 & -7.4 & -0.2 \\ 17.9 & -6.2 & 10.6 & -3.6 & -3.6 & -1.1 & -1.3 & -2.8 & +5.2 & -4.6 & \dots & +4.5 & -4.2 & -1.5 & 3.5 & -3.9 & -3.1 & 1.4 & -4.7 & 7.1 & 1.2 \\ -1.2 & -0.3 & -1.0 & +1.3 & +2.4 & +0.0 & +7.3 & +2.5 & -2.0 & -1.6 & \dots & +6.2 & -3.0 & -5.7 & -8.7 & +7.4 & +8.3 & -7.6 & -3.3 & -6.4 & -7.6 \\ \vdots & \dots & \vdots \\ +7.9 & -8.8 & +0.5 & -8.0 & +7.2 & +1.3 & -2.6 & -3.1 & +5.1 & -3.7 & \dots & +3.1 & -0.3 & -0.5 & -7.9 & +1.1 & +6.5 & +4.5 & -9.1 & +5.4 & -5.6 \end{bmatrix}$$

Embedding matrix

נניח וה- token הוא door. ולפי ה- table נשים מיפוי בין door לאינדקס Embedding. 54323. כתת נمر אל ה- Embedding Matrix ונשלוף את הווקטור במיקום .54323

חשוב לציין כי בעבר שיטות Embedding היו למשל Word2Vec, Glove, אך כיום כאשר אנו משתמשים ב-GPT, Embedding Matrix כלולה בתוך המודול ולמעשה הייצוגים הווקטוריים נלמדים במהלך תהליך האימון של המודול. על קצה המציג, למשה מתחילהים בייצוגים וקטוריים רנדומליים או מבוססים על ייצוגים וקטוריים קודמים ובמהלך תהליכי האימון נדחפים גראדיינטיים כחלק מה- Backpropagation ומודכנים את הייצוגים הווקטוריים של הטוקנים.

בפועל, למשה Embedding Matrix היא למעשה סט משקولات נלמד שמתעדכן ומשתנה לאורך תהליכי האימון ומטרתו למדוד רפרנציאציות לגבי הtekst שבסופו של דבר מגולמות בייצוגים וקטוריים עבור הטוקנים.

מייד וקטור ה- Embedding שתואר במאמר הוא 512 אך עם זאת במודלי שפה מתקדמים יותר כמו GPT3, המיד גדול בהרבה ועומד על 12288.

Positional Encoding

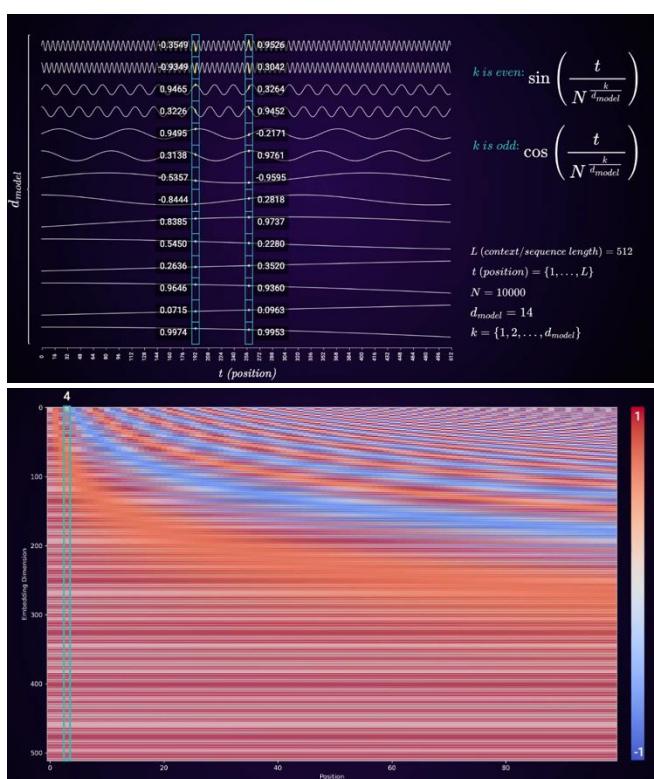
[Video Link](#)

ל-Positional Encoding יש משמעותות גדולות בתרנספורמרים והסיבה לכך נועזה באופי המודול ובעיקר בשכבות ה-Attention. בעוד שלמודלים שמותאים לעבוד בצורה ישירה עם Sequential Data יש יכולת עצמאית לייחס משמעותות למיקומי היקלט לטרנספורמר המשמש ב-h-Attention אין את היכולת הזאת. כפי שנראה בהמשך לשכבת ה-Attention אין יכולת להבדיל לגבי המיקומים של הtokנים ברצף כך למשל, אלא Positional Encoding המשפטים "I walk to the park" ו- "The park walk to the park" יהיי זהים לוחלוטין עביניו הטרנספורמר על אף לנו כבני אדם קל להבחין כי הם שונים בתכלית. לכן, יש צורך בקידוד המידע הזה בצורה יעודית, כאן Positional Encoding נכנס לתמונה.

- ה-Positional Encoding הוא ערך ווקטור מממד Embedding שמתווסף Element-Wise אל הווקטור המספרי המייצג את הTokן. ה-Positional Encoding צריך לקיים כמה תכונות עיקריות בניהו:
1. פונקציה חד-חד ערכית ועל – הפונקציה המשמשת ליצור הווקטור שיתווסף ל-Embedding צריכה להיות צזו היוצרת ווקטורים ייחודיים ולכן בהינתן position position של מילה, יוצר ווקטור ייחיד עבור position זה.
 2. דטרמיניסטי – עבור ערך position יוצר בכל פעם אותו ערך ווקטורי ייחיד (לא יכול להיות מצב שעבור position מסוים יוצר פעמי אחד ווקטור a ופעם אחר ווקטור b)
 3. מדיד אמפירית – עבור כל שני ווקטורים הלוקחים מההתפלגות, ניתן לתאר את המרחק ביניהם באמצעות כל מסויים כלשהו

שיטת המקובלת לבצע זה באמצעות Cosine and Sine Positional Encoding

$$k \text{ is even} = \sin\left(\frac{t}{N^{\frac{k}{d_{model}}}}\right), \quad k \text{ is odd} = \cos\left(\frac{t}{N^{\frac{k}{d_{model}}}}\right)$$



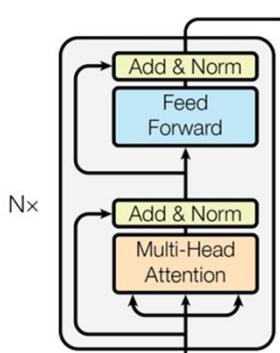
בראה כי t מייצג את position של הTokן הפרומפט, k מייצג את position של הסקלר בווקטור ה-Embedding, N מייצג מספר גדול אותו אנחנו מעלים בחזקה, d_{model} מיצג את ממד ה-Embedding אליו אנחנו עובדים. סך הכל עבור כל סקלר בווקטור ה-Embedding יש פורמוללה ייחודית ליצירתו. לאחר שיצרונו את ווקטור ה-Positional Encoding אנו מוציאים אותו Element-Wise לכלי אחד מהאלמנטים בווקטור ה-Embedding כך שcutut יש קידוד של האינפומציה המייצגת את מיקומם ה Tokנים לאורור המשפט באופן בו שכבת ה-Attention והטרנספורמר באופן כללי יכול ללמוד את הרפרנציות הסדרתיות.

שיטת נוספת ל-Positional Encoding היא Color Map באופן בו ציר ה- x -אxis מייצג את המיקום של הTokן במשפט וציר ה- y -axis מייצג את האלמנט ה- i -בוווקטור. אוננו נסיף ליצוג ה-Embedding של הTokן. צבע ה-cell מתאר את הערך שיתווסף עבור אותו אלמנט כאשר צבע אדום מייצג ערך גבוה וצבע כחול מייצג ערך נמוך.

Encoder

רכיב Encoder במודל הטרנספורמר הוא אחד משני חלקים הבסיסיים של הארכיטקטורה, השני הוא Decoder. כל חלק במודל כולל מספר שכבות של Encoders או Decoders, שכל אחת מהן תורמת לעיבוד הנתונים בשלבים רבים ומורכבים.

בפועל, Encoder מתחילה את פעולתו על ידי קבלת סדרת קלט, שבדרך כלל מורכבת מזוקטוריהם של טוקנים שהומרו מtekst ליצוג וקטורי מסווני. מטרתו העיקרית של Encoder היא להפוך את הסדרה זו ליצוגים עשירים ומורכבים יותר, שיכולים להכיל מידע רב על ההקשרים שבין האלמנטים השונים בטקסט. זה מאפשר Decoder לפענוח את המידע ביעילות רבה יותר בהמשך.



הטכניקה המרכזית שבה משתמש Encoder היא מנגןון ה-*Chosenn*, שמאפשר לו להתקדם בחלקים הרלוונטיים ביותר של הטקסט בזמן עיבודו. כל שכבה encoder כוללת שני רכיבים עיקריים:

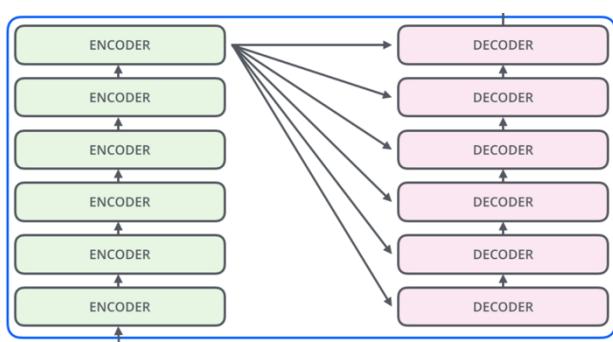
1. Multi-Head Attention
2. Feed Forward Neural Network

ה-*Multi-Head Attention* מסייע בהבנת הקשרים המרחביים בין המילים בטקסט, כאשר מספר ראשים של *Attention* עובדים במקביל לחפש אינטראקציות שונות. *Multi-Head Attention* משמש להפעלת תהליכי נוספים של למידה אחרי כל שכבה *Attention*, מושך ל-*Feed Forward Neural Network*, שמאפשר למדידת שגיאות טקסטואליות על המידע שנאוסף, ולהגברת ההבנה של ההקשרים הטקסטואליים.

חשוב לציין של אף שבתמונה נראה כי *Feed Forward Neural Network* מופיע ככללותו רק לאחר *Self Attention* בפועל הוא מופיע אחרי כל *Self Attention* כך שבහינת Encoder עם k שכבות של *Self Attention*, אז יהיו בו k שכבות של *Feed Forward Neural Network*.

כמו כן לאחר כל *Multi-Head Attention* ולאחר כל *Feed Forward Neural Network* ישנה שכבה של *Residual Connection* וbijzou נורמליזציה של ערך הקומפוננטה.

במהלך העיבוד, כל שכבה Encoder מוסיפה ידע ומורכבות לייצוג הטקסט, מה שמאפשר בסופו של תהליך לייצר וקטורים עשירים שמשמעותם העמוקה והקשרים המורכבים של הטקסט המקורי.

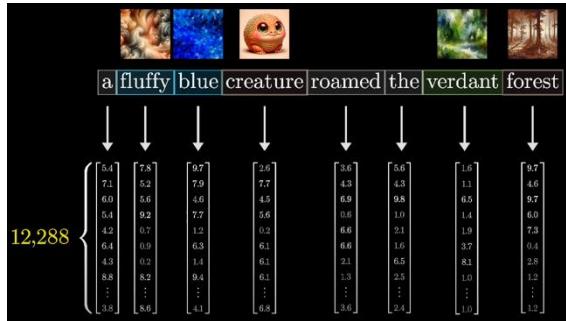


כמו כן, פרט חשוב לציין הוא שבפועל אין Encoder ייחיד אלא חסיאלי, כל Encoder הנזכר באוטר או מעביר את הפלט שלו ל-*Decoder* ה-1+i. לבסוף ה-*Decoder* ה-i מעביר את הפלט הסופי אל כל אחד מה-*Decoders* כקלט. בארכיטקטורה המקורית תוארה ארכיטקטורת הטרנספורמר ככך המכילה 6 שכבות *Encoders* ו-6 *Decoders*. אך עם זאת במודלים גדולים יותר למשל כמו GPT3, המספר זהה גדול משמעותית ועומד על 96-24.

Self-Attention

לפנינו שנדבר על Multi-Head Attention חשוב קודם לכך לדבר על Self-Attention הרכיב שהוא למעשה גולת הכותרת של הטרנספורמר ופתח את הדלת לכל החידושים האחרונים ב-GenAI. עד אז, מודלים ל-Sequential Data נתקלו בעויהות רבתות כמו Exploding/Vanishing Gradients וכמו כן התקשו למצוא להתמודד עם תלויות מורכבות ברצפים ארוכים.

במנגנון-Attention למשה תיאורטי אין הגבלה לגבי אורך הקונטקסט ובאופן עקרוני אורך הרץ יכול להיות אינסופי, הגבלה בפועל היא משיוקלי זמן ריצה ומשאבים.



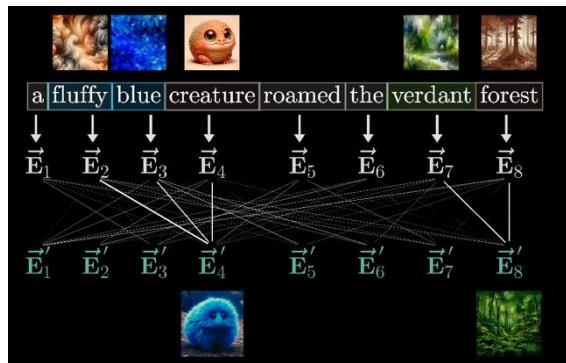
על מנת להסביר את מנגנון ה-Attention נפתח דוחא בדוגמא שמציגה את העיקרון. נניח ואנו ניצבים אל מול המשפט הבא:
a fluffy blue creature roamed the verdant forest

- בתרור בני אדם, ישר קופצות לנו לעין התלויות הבאות:

 - creature Fluffy, blue 1.
 - forest Verdant 2.
 - creature Roamed 3.
 - creature roamed Forest 4.

כלומר, בתור בני אדם שיעודים לפרש שפה, מיד אנחנו מבקינים בתלויות שגורמות לכך שאחנו מוצאים היגיון במשפט ויעודים לספוג את האינפורמציה שבו. יכולת זו היא יכולת שאחנו רוצים גם שהמודל יפתח, שידע למצוא תלויות ברצף ולתרגם אותן לכדי רפרזנטציות שהוא למד על הקונטקט בפרט ועל האינפורמציה המובאת לידי ביטוי בשפה אנושית בכלל.

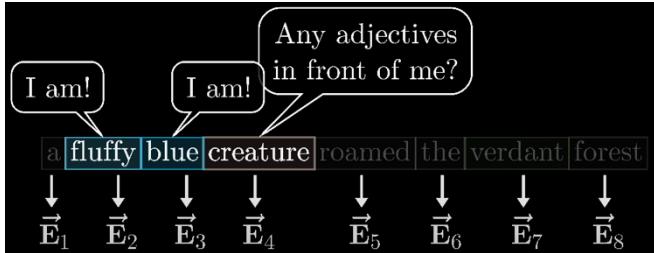
מבט נוסף בתמונה חשוב לי לציין גם מה אנחנו רואים, למשה המתואר הוא אוסף של ווקטורים המיצגים את הトーונים (שבדוגמא זו נתיחס אליהם כמלילים). כפי שתואר, אנחנו כבר בתוך ה-Encoder ולכן הוקטוריהם הללו כבר עברו PE (Positional Encoding) ועתידייםicut להיכנס אל עבר הkomponeneta של ה-Self Attention.



באופן כללי ניתן לומר כי המודל מתחילה מה"יצוגים הוקטוריים הלבנים שלמעה מייצגים את הערך שנשלה מtower-h-x Embedding Matrix בתוספת PE והמטרה של במהלך תהליכי Attention הוא למצוא את ה"יצוגים הוקטוריים הכהולים שמצקקים את המשמעות הסמנטית

כך למשל ניתן לראות למשל כי E4 הוא ייצוג ווקטור של המילה creature שספוגה לתוכה את המשמעות הסמנטית של 2 המילים הקודומות לה, *blue* and *fluffy*.

בדוגמא זו הטעמךנו בתלות בין המילים *blue*, *fluffy*, *creature*, *crush* והו שם עצם - *fluffy*, *blue* הן מיליות תואר וכך גם עברו *verdant*, *forest*. עם זאת חשוב לציין שגם דוגמא בלבד והוא ניתן רק כי לנו בთור בני אדם קל ל תפופו אותה. ניתן לחשב על כל שכבת *Attention* שכך התופסת תלויות שאלו בטקסט זהה גם מה שנעשה לשם הפשטות, אך בפועל נתפסות תלויות ורפרזנטציות על ידי המודל שבפועל לנו בני אדם אין יכולת להבחן בהן. עם זאת, לטובת המשך ההסבירים נמשיך לדבוק בדוגמה זו מהסבירה שהיא מתיחסת בצורה טובה עם היגיון האנושי.



פה בדיק נכנס לתמונה מנגנון-h-questions, זה בדיק אופן הפעולה שלו ובאופן עבודתו הוא מمدל בדיק את ההתנהגות זו אותה אנו רצים להשיג. כתע נסקור את האופן שבו הוא עשו זאת. מבחן פשטיינט, אנחנו רצים רכיב במנגנון שיעוד לשאול את השאלה (**Query**), רכיב במנגנון שיעוד לספק תשובה לשאלה (**Key**) ורכיב במנגנון שיעוד לבצע את ההוספה לווקטור במצב בו התשובה לשאלה חיובית (**Value**).

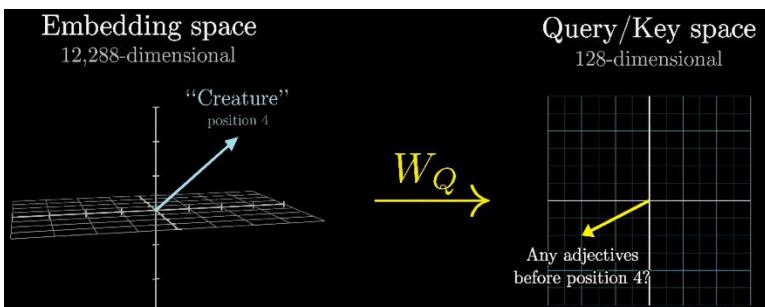
Query

בහינתן רצף טקסטואלי באורך n המורכב מ- $E_1 \dots E_n$ טוקנים (밀ים במרקחה שלנו), אנחנו רצים שיהיה לנו אלמנט שאנו יכולים להפעיל על כל ווקטור $E_i \in E_1 \dots E_n$ הקשורו לשאלה הרצiosa עבור הטוקן.

$$\underbrace{W_Q}_{\text{Matrix}} \begin{bmatrix} \vec{E}_i \\ \vec{E}_1 \\ \vec{E}_2 \\ \vec{E}_3 \\ \vdots \\ \vec{E}_n \end{bmatrix} = \vec{Q}_i$$

האלמנט במרקחה זהה הוא W_Q . האלמנט זהה הוא מטריצה בעלת משקלות נלמדות. אני אוהב להסתכל עליו כמו קאוצ'ירית, היא כביכול מנהה את הווקטור E_i לשאול את השאלה Q_i על ידי כך שמתבצעת מכפלה מטריציונית המתוארת $W_Q \otimes E_i = Q_i$.

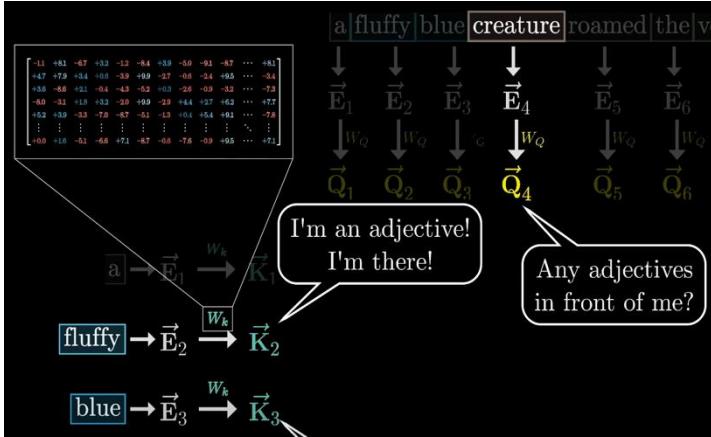
חשוב לציין כי W_Q הולכת ומשתרעת לאורכו תהליך האימון אך כתם אחר וצינו שאנו מתארים את המנגנון-**Inference** אנו נתיחס אליה ככל קאוצ'ירית נוספת שיעודעת להוביל לשאלות הנכונות לכל ווקטור.



בדוגמא שלנו ווקטור-h-embedding הוא בגודל של 12288×1 . עם זאת המטריצה W_Q היא בגודל של 128×12288 וכך מהמכפלה המטריציונית מתקבל ווקטור ממימד 128×1 . ניתן לחשב על כך שבעצם המטריצה הנלמדת W_Q ממנה את הווקטור ממימד הגובה לווקטור אחר ממימד נmor בהרבה. שמצויה את השאלה הנשאלת עבור הווקטור.

למעשה אנו מקבלים ממימד נmor יותר שהוא המימד אליו מופיעות השאלות וכפי שנראה נכון שם הן גם מבלות מענה על ידי ווקטור מאותו מימד. לפ"ז המענה לשאלות ממימד זה המודל יידע האם לבצע עדכן חזק לווקטור הטוקן או עדכן חלש.

Key

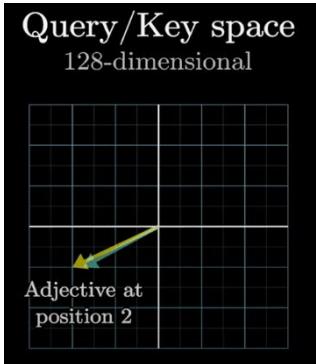


אנו יכולים להסתכל על ה-Key כעל וקטור מענה עבור ה-Query. לכל TOKEN $E_1 \dots E_n$ נבצע מכפלה באלמנט נלמד נוספת בשם W_K . זו למעשה מטריצה נלמדת שמקילה משקלות אותן אנו כופלים בכל וקטור E_i לקבלת תשובה K_i המוגדרת באופן:

$$W_K \otimes E_i = K_i$$

כזכור לכל וקטור מתבצעת מכפלה מטריצית במתリיצת המשקלות הנלמדת לקבלת תשובה K_i לשאלה Q_i .

גם כאן חשוב לציין כי במהלך תהליכי האימון המשקלות של המטריצה משתנות בהתאם לדאטא אך גם כאן מאחר ואנו מדברים על Inference נניח וערך המטריצה כבר מכויים באופן מספק.



סך הכל, לכל וקטור Embedding אנו מחשבים את המכפלה המטריצונית שלו עם שתי מטריצות משקלות W_Q , W_K לקבלת שני וקטורים מממד נמוך יותר K_i, Q_i .
מהchine גראפית ניתן לחשב על התלות בין K_i ל- Q_i כעל ניסיון להגעה לחיפוי מושלם בין הווקטורים במאיה ויש התאמה טוביה בין ה-Key ל-Key Query-Query וווקטורים. הסיבה לכך היא שכי שתכף נראה אנו מחשבים dot-product בין שני הווקטורים באופן הבא:

$$\forall i \rightarrow Q_i \cdot K_i = \text{Key Value Matrix}$$

כזכור לכל זוג ווקטורים i , אנחנו מחשבים מכפלה dot-product המסתכמה לסקלר. בהנחה והסקלה גדול אידי קיים דמיון רב יותר בין הווקטורים, בהנחה והסקלה קטן יותר קיים דמיון פחות יותר בין הווקטורים.

הסיבה לכך נעוצה באופן בו מכפלת dot-product מחושבת. בהינתן שני וקטורים a, b , נוכל להגיד את המכפלת dot-dot-product באופן הבא: $\sum_{i=1}^n a_i b_i = a \cdot b$. בהינתן הגדרה זו נשים לב כי הביטוי מקבל ערך מקסימלי כאשר $b = a$ מהסיבה ש $\sum_{i=1}^n a_i^2 = a \cdot a$ תמיד מקבל ערך חיובי.

	\downarrow E_1	\downarrow E_2	\downarrow E_3	\downarrow E_4	\downarrow E_5	\downarrow E_6	\downarrow E_7	\downarrow E_8
$[a] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{fluffy}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{blue}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{creature}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{roamed}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{the}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{verdant}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$
$[\text{forest}] \rightarrow$	$\vec{E}_1 \xrightarrow{w_k} \vec{K}_1$	$\vec{E}_2 \xrightarrow{w_k} \vec{K}_2$	$\vec{E}_3 \xrightarrow{w_k} \vec{K}_3$	$\vec{E}_4 \xrightarrow{w_k} \vec{K}_4$	$\vec{E}_5 \xrightarrow{w_k} \vec{K}_5$	$\vec{E}_6 \xrightarrow{w_k} \vec{K}_6$	$\vec{E}_7 \xrightarrow{w_k} \vec{K}_7$	$\vec{E}_8 \xrightarrow{w_k} \vec{K}_8$

בתרמונה ניתן לראות את Key Value Matrix באופן בו לכל תא במטריצה יש עיגול שמייצג את ערך מכפלת dot-dot-product באופן בו עיגולים גדולים מייצגים ערך גבוה כלומר הלימה בין Key-Query-Query. קטענים מייצגים חוסר התאמה בין Key-Query-Query לבין מצב בו יש התאמה קטנה בין הזוג הסדור של הטוקנים. התאמה גדולה כאמור מייצגת מצב בו יש Attention ותלות בין שני הטוקנים וההתאמה קטנה מייצגת מצב בו אין Attention ותלות בין הטוקנים. כפי שנראה נכון, ערך ה-dot-product עברו כל זוג סדור של טוקנים יישמש עבורנו לטובת עדכון הייצוגים הווקטוריים של כל אחד מהטוקנים ב-inference-while לחיזוי עבור המילה הבאה שהכי מתאימה להשלמת הרץ.

הגדרת ה-Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

	Q_1	Q_2	Q_3	Q_4	Q_5	\dots	Q_n
K_1	$\frac{Q_1 \cdot K_1}{\sqrt{d_k}}$	$\frac{Q_2 \cdot K_1}{\sqrt{d_k}}$	$\frac{Q_3 \cdot K_1}{\sqrt{d_k}}$	$\frac{Q_4 \cdot K_1}{\sqrt{d_k}}$	$\frac{Q_5 \cdot K_1}{\sqrt{d_k}}$	\dots	$\frac{Q_n \cdot K_1}{\sqrt{d_k}}$
K_2	$\frac{Q_1 \cdot K_2}{\sqrt{d_k}}$	$\frac{Q_2 \cdot K_2}{\sqrt{d_k}}$	$\frac{Q_3 \cdot K_2}{\sqrt{d_k}}$	$\frac{Q_4 \cdot K_2}{\sqrt{d_k}}$	$\frac{Q_5 \cdot K_2}{\sqrt{d_k}}$	\dots	$\frac{Q_n \cdot K_2}{\sqrt{d_k}}$
K_3	$\frac{Q_1 \cdot K_3}{\sqrt{d_k}}$	$\frac{Q_2 \cdot K_3}{\sqrt{d_k}}$	$\frac{Q_3 \cdot K_3}{\sqrt{d_k}}$	$\frac{Q_4 \cdot K_3}{\sqrt{d_k}}$	$\frac{Q_5 \cdot K_3}{\sqrt{d_k}}$	\dots	$\frac{Q_n \cdot K_3}{\sqrt{d_k}}$
K_4	$\frac{Q_1 \cdot K_4}{\sqrt{d_k}}$	$\frac{Q_2 \cdot K_4}{\sqrt{d_k}}$	$\frac{Q_3 \cdot K_4}{\sqrt{d_k}}$	$\frac{Q_4 \cdot K_4}{\sqrt{d_k}}$	$\frac{Q_5 \cdot K_4}{\sqrt{d_k}}$	\dots	$\frac{Q_n \cdot K_4}{\sqrt{d_k}}$
K_5	$\frac{Q_1 \cdot K_5}{\sqrt{d_k}}$	$\frac{Q_2 \cdot K_5}{\sqrt{d_k}}$	$\frac{Q_3 \cdot K_5}{\sqrt{d_k}}$	$\frac{Q_4 \cdot K_5}{\sqrt{d_k}}$	$\frac{Q_5 \cdot K_5}{\sqrt{d_k}}$	\dots	$\frac{Q_n \cdot K_5}{\sqrt{d_k}}$

כפי שהוזג במאמר המקורי, הגדרת ה-Attention מוגדרת באופן הבא:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

הסימון היחיד המוכר לנו הוא QK^T המיצג כפי שדובר את ה-Attention Matrix. Key Value Matrix עם זאת, עד כה דיברנו על שני רכיבים מתוך הגדרת ה-Attention שהם Query, Key ו-Value. לפני שנותמקד ברכיב השלישי שהוא Value המוצג באות V, נרצה לתאר 3 גורמים נוספים:

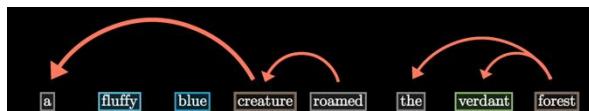
1. חלוקה ב- $\sqrt{d_k}$ -Masking
2. SoftMax
- 3.

חלוקת ב- $\sqrt{d_k}$

חלוקת ב- $\sqrt{d_k}$ היא straight forward חלוקה של כל תא במטריצה ב- d_k קלומר בשורש המימיד הווקטורי. הסיבה לכך היא רצון לשמור על יציבות נומרית ולהימנע מלעבוד עם מספרים גדולים שיכולים לגרום לקשיי בערך המשקولات ולמידת הרפרזנטציות של המודל.

Masking

שימוש ב-Masking מונע מהמודול ללמידה רפרזנטציות עבור טוקנים באמצעות טוקני עתיד במהלך היליך האימון או במהלך היליך Inference ואנחנו משתמשים ב-Cross Attention כשהאנו כשאנו באים לפטור בעיית תרגום למשל.



אם נסתכל לדוגמה על התא המסומן באדום בתמונה העילונה, נראה כי זו למעשה המילה a שמספריקה את Key-Query המילה creature שהיא מילת עתיד.

למעשה, המילה a לא השפיעה בכלל למילה creature ולכן נרצה למסך את כל האלכסון התיכון שמייצג מילות עבר שמקבלות מענה על ידי מילות עתיד.

+3.53	+0.80	+1.96	+4.48	+3.74	-1.95	
$-\infty$	-0.30	-0.21	+0.82	+0.29	+2.91	
$-\infty$	$-\infty$	+0.89	+0.67	+2.99	-0.41	
$-\infty$	$-\infty$	$-\infty$	+1.31	+1.73	-1.48	
$-\infty$	$-\infty$	$-\infty$	$-\infty$	+3.07	+2.94	
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	+0.31

האפקן שבו נבצע את ה-Masking הוא על ידי השמת ערך של מינוס אינסוף בכל התאים שמצויים במשולש התיכון מתחת לאלכסון הראשי. של המטריצה.

באופן זה הערכים שנוטרו מייצגים אך ורק מילות נכון-Shunting. שמתבססות אך ורק על מילות עבר לצורך חישוב ה-Attention. חשוב לציין כי Masking אינם הכרחי תמיד ויש ארכיטקטורות שימושות את המנגנון הזה.

SoftMax

לאחר החישוב של QK^T , חלוקה ב- $\sqrt{d_k}$ וביצוע Masking שכאמור אופציוני בלבד,Cut יש חישוב של SoftMax על כל עמודות המטריצה בנפרד.

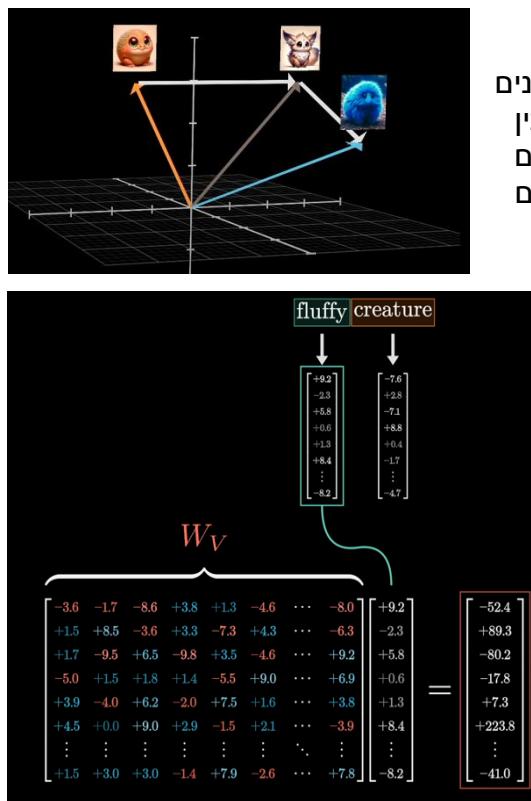
+3.53	+0.80	+1.96	+4.48	+3.74	-1.95	
$-\infty$	-0.30	-0.21	+0.82	+0.29	+2.91	
$-\infty$	$-\infty$	+0.89	+0.67	+2.99	-0.41	
$-\infty$	$-\infty$	$-\infty$	+1.31	+1.73	-1.48	
$-\infty$	$-\infty$	$-\infty$	$-\infty$	+3.07	+2.94	
$-\infty$	$-\infty$	$-\infty$	$-\infty$	$-\infty$	+0.31	

1.00	0.75	0.69	0.92	0.46	0.00	
0.00	0.25	0.08	0.02	0.01	0.46	
0.00	0.00	0.24	0.02	0.22	0.02	
0.00	0.00	0.00	0.04	0.06	0.01	
0.00	0.00	0.00	0.00	0.24	0.48	
0.00	0.00	0.00	0.00	0.00	0.03	

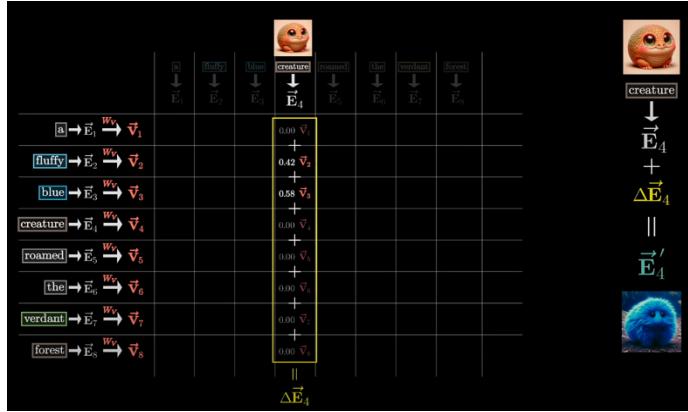
באופן זה אנו מקבלים כי עבר כל עמודה, המיצגת טוקן, אנו מקבלים ערכי הסתברות בין 0 ל-1 המסתכניםים ל-1. ככלمر אנו מקבלים ערך Probability Distribution עבור ה-h-hochetion לכל טוקן ביחס לטוקני העבר שלו. התא שמכיל את הערך הגבוה ביותר הוא התא שמייצג את הטוקן בעל ההשפעה הגדולה ביותר עבור הטוקן הנוכחי.



از אמם דיברנו על ה-Query וה-Key Attention שבמציאות נמדדת התלות בין הטוקנים לאור הרצף. עם זאת, לאחר שהמנגן מכתת את התלות בין המילים ברצף יש לבצע שניי ביצוגים הווקטוריים של הטוקנים על מנת שייגלמו ייצוגים חדשים התלויים בתלותו של הטוקנים שקדמו להם. כאן נכנס לתמונה וקטור ה-Value.



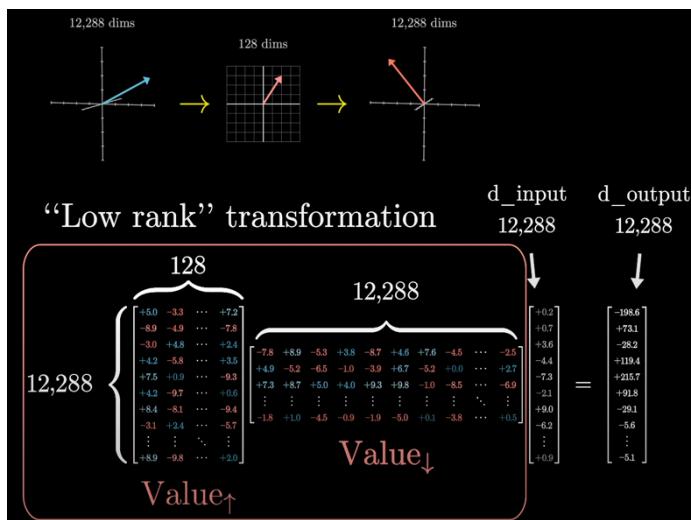
לכל טוקן $E_1 \dots E_n$ נבצע מכפלה במטריצה נלמדת נוספת בשם W_V . זו למעשה מטריצה נלמדת שמקולות את אונן כופלים בכל וקטור E_i לקבלת תשובה V_i המוגדרת באופן: $V_i = W_V \otimes E_i$. אני אהוב להסתכל על המטריצה הנלמדת W_V כמו תהליך של חיליתת תה באופן בו המטריצה הנלמדת יודעת למצאות מהווקטורי E_i את מהות השינויים שיזובילו לעדכון האפקטיבי ביותר של הייצוג הווקטורי של הטוקן הבא. למעשה ניתן לחושב על E_i כמו תמצית הפרחים, על W_V כל המים הזורמים על תמצית הפרחים (המים יודעים איך לחלוות ולמצאות) ונשפיקים אל הכוון בו התחה שלעצמם הוא הווקטור V שמייצג את המיצוי של הרפרחנטציות הרלוונטיות שיזובילו לכך שהטוקן הבא ישפע מהטוקן שקדם לו באופן נכון ואפקטיבי.



מתמטית את הפעולה שאחננו עושים כ- $V \cdot \frac{QK^T}{\sqrt{d_k}}$ softmax שזו בדיקת הגדרת ה-Attention.

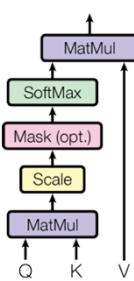
לכן, עבור כל וקטור E_i באמצעות פעולה Attention אנו למשה יוצרים וקטור שקול בגודלו שסימונו הוא ΔE_i שמייצג את השינוי שצריך לבצע בייצוג הווקטורי של הטוקן על מנת לתכל את הטוקנים שקדמו לו והשפעתם על הייצוג הווקטורי שלו וככל זאת בהתאם לערכי ה-Query, Key, Value.

בדוגמא שלנו ניתן לראות כי הטוקן הנוכחי הוא המציג את המילה creature, ומכפלת dot-product של blue, fluffy, גתנה ערכים של 0.42, 0.58, 0.42 בהתאמה. לפיכך, הם משפיעים על המילה creature ולכן צריך לשנות את הייצוג הווקטורי של היחסים של תוצאות SoftMax שmobilia לכך יצירת וקטור שמכפל בו וקטור V ליצירת ΔE_i שמתווסף ΔE_i אל E_i .



המטריצה הנלמדת W_V מוגדרת כ- $W_V = W_{Value↑} + W_{Value↓}$. כלומר, מטריצת W_V היא בפועל מכפלת מטריצות קטנות יותר: $W_{Value↑} \cdot W_{Value↓}$.

ניתן לשים לב כי תחילת מתבצעת מכפלה של $W_{Value↑}$ עם הייצוג הווקטורי של הטוקן V , ולאחר מכן מתקבל מילוי למינד נמור ולאחר מכן מתבצעת מכפלה של $W_{Value↓}$ עם המילוי $W_{Value↑} \cdot V$.



לסיום, ניתן לומר כי התמונה מצד שמאל ממחישה את השלבים שתוארו לגבי הגדרת ה-Attention(Q, K, V) = $softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V$.

כמו כן כדי שתואר נשים לב כי בשלב ה-Masking הוא אופציונלי ולא הכרחי למשתמש. כמו בסוגי Attention אחרים כמו Cross Attention בו אנו עוסקים בסוגי DATA מתוחומים שונים או למשל בראפי טקסט משפות שונות.

מתיאור התהילה נשים לב כי כל אחד מהיצוגים הוקטוריים $E_1 \dots E_n$ מכפל ב- W_V לקבלת $V_1 \dots V_n$. לאחר מכן, מתבצעת מכפלה של V_i ב- $\frac{Q_i K_i^T}{\sqrt{d_k}}$ לקבלת הערך הממושך של השינוי שכל טוקן קודם מוסיף לטוקן הנוכחי.

חשוב לציין כי עבור V_i, Q_i, K_i , ניתן להתייחס אליהם בתור ווקטורים המכפלים dot-product אר-אר ניטן להתייחס אליהם גם כאוסף וקטורים, קרי – מטריצה ולכן ניתן ליצג

מתמטית את הפעולה שאחננו עושים כ- $V \cdot \frac{QK^T}{\sqrt{d_k}}$

נקודה חשובה שחייב לציין היא גודל המטריצה הנלמדת W_V . בעוד שאמורנו שגם W_Q וגם W_K הן בגודל של 128x128, גודל W_V הוא בגודל של 12288x12288.

דבר ששוביל לਮרכיבות מהסיבה שזה הרבה פרמטרים נלמדים שיכולים לעקב את התהילה האימון את התהילה ה-*Inference*. הפתרון לכך הוא לפרק את המטריצה W_V לשתי מטריצות קטנות יותר:

1. $W_{Value↑}$ בגודל 12288x128
2. $W_{Value↓}$ בגודל 128x12288

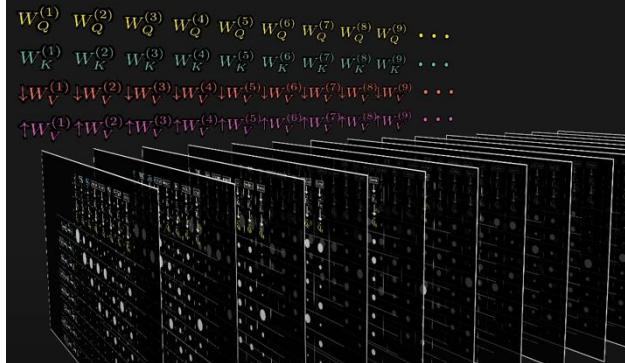
ניתן לשים לב כי תחילת מתבצעת מכפלה של $W_{Value↑}$ עם הייצוג הווקטורי של הטוקן V , ולאחר מכן מתקבל מילוי למינד נמור ולאחר מכן מתבצעת מכפלה של $W_{Value↓}$ עם המילוי $W_{Value↑} \cdot V$.

הגודל של W_V שווה לפעמיים הגדל של W_Q / W_K .

Multi Head Attention

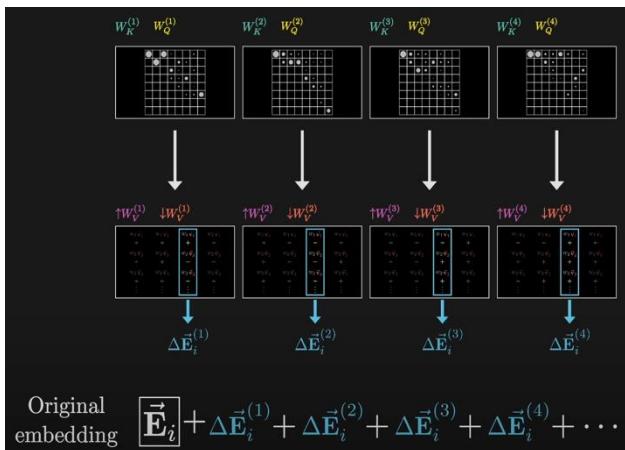
עד כה דיברנו על Self-Attention ועל הפקיד שלו במציאת תלויות לאורך הרצף. בדוגמה שהבנו, ראיינו כי שכבת Self-Attention מטרתה למצוא תארים לפני שמות עצם, אך כאמור זו הייתה רק דוגמא והتلויות שנמצאות בפועל יכולות להיות תלויות שאנו חנו כבני אדם בכלל לא יכולים להבחין בהן. עם זאת, אנו תיארנו תלות אחת בלבד אך יכולות להיות רבות ומגוונות ולכן לא נרצה להסתפק בשכבת Attention ייחידה אלא נרצה שיהיה לנו מספר ראשי Heads Attention שהמטרה שלהם למצוא תלויות שונות ומגוונות בDATA ובדוק לשם כך נכנס לתמונה מנגנון Multi Head Attention שמורכב ממשפר ראשיהם.

Multi-headed attention



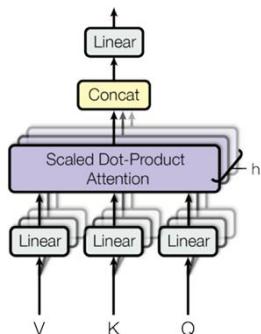
לכל אחד מהראשים יש את כל אחת ממטריצות המשקولات W_Q , W_K , $W_{Value\uparrow}$, $W_{Value\downarrow}$ וכל אחד מהראשים מפיק בסופו של דבר את שלושת הוקטורים מהיממד הנמור: V , K , Q .

למעשה מה שאנו מקבלים זה שכל ראש Attention למעשה מציע את השינוי שלו לייצוג Embedding של וקטור הtokenizer בהתאם לסת הערכים הנלמדים שמצויים בו. כל ראש למעשה מסיים עם הוצאות לשינוי שנסמן ב- ΔE_i כפלט של שכבת ה-Attention הספציפית.



האוף בו מטבחע העדכון של הייצוג ה-**Embedding** בסופו של דבר עבר ערך הווקטור של הטוקן הוא על ידי סכימה של כל השינויים שהוצעו בכל אחד מראשי ה-Attention אשר יתווסף אל הערך הווקטורי המקורי של הטוקן.

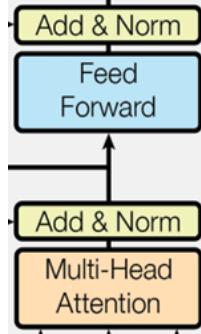
בתרמונה ניתן לראות כי עבור ייצוג ווקטור E_i ישנה הוספה של כל $\Delta E_i^{(n)}$... $\Delta E_i^{(1)}$ באופן בו המספר הקטן העליון הוא אינדיקס של ראש ה-Attention באופן בו הוא מציין את השינוי שהראש רצה לבצע בוקטור הטוקן כתלות בדפוס אותו הוא אומן למצוא.



היתרון המשמעותי ב-**Multi Head Attention** היא יכולת שלו לróż בזיכרון GPU דבר שմבינה הנדסית יכול להוביל לזמן ריצה מהירים ו-**Inference** ומהירות מהירה. בעודו שלמודל יש מספר רב של פרמטרים, הריצה שלו במקביל מוביילו לכך זמן הריצה יכול להיות קצר אף עד 3 סדרי גודל מהטיבה שלא קיימת תלות מושרשת בין הפלט של ראש **Attention** כלשהו לבין ראש **Attention** שבא אחריו וכן יכולים לזרז במקביל, לסכם את תוצאות $E_i \Delta$ עבור כל ראש אל וקטור הטוקן ולהמשיך אל עבר החישוב הבא.

Add & Norm

קומפוננטת-hAdd Norm מופיעה מספר פעמים בארכיטקטורה. כמו כן, היא מופיעה בין היתר לאחר שכבות<->Feed Forward Neural Network<->Multi Head Attention<->



Add
בשכבה<->Add Norm מتبצע הוספה ערך<->Residual Connection<->מהשכבה<->הקודמת<->(Add). אופן<->פעולות<->Add<->עובדת<->באופן<->ה הבא<->נסמן<->ב-f<->את<->הטרנספורמציה<->שمتבצעת<->על<->הקלט<->המקור<->למשל<->Feed<->Forward<->Output<->f(x) = f(x) + x<->כלומר<->או<->Multi<->Head<->Attention<->azi<->הפלט<->יה<->שווה<->-x<->his<->output<->his<->ישירה<->של<->ערך<->הקלט<->אל<->ערך<->הפלט<->לאחר<->הטרנספורמציה<->הוספה<->ישירה<->בארכיטקטורת<->הטרנספורמר<->הערך<->מתווסף<->בצורה<->ישירה<->ובמלואו<->אך<->ישנו<->ארכיטקטורות<->בהן<->חלק<->קטן<->יתר<->מתווסף<->למשל<->אם<->כופלים<->את<->x<->במספר<->כלשהו<->בין<->0<->-1.

Norm

למעשה מדובר כאן<->Batch Normalization<->באוון<->הנורמליזציה<->התרחשת<->בנפרד<->עבור<->כל<->עמודה<->הנוסחה<->המתמטית<->لتיאור<->Batch Normalization<->היא:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \quad , \quad \sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$$

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$x_{i_{final}} = \gamma \hat{x}_i + \beta = BN_{\gamma, \beta}(x_i)$$

למעשה עבור<->כל<->עמודה<->מוחשב<->ערך<->הממוצע<->והשונות<-> σ^2 ,
 μ <->ולאחר<->מכן<->עבור<->כל<->עמודה<->יש<->חסורה<->של<->ערך<->הממוצע<->וחולקה<->בערך<->השונות<->ועוד<->אפס<->ילון<->כדי<->למנוע<->חולקה<->-0. לבסוף,<->יש<->הכנסה<->של<->הערך<->אל<->עבור<->הערך<->הסوفي<->של<->הסקלר<->בעמודה,<->לאחר<->מיניפולציה<->באמצעות<->שני<->פרמטרים<->נלמדים<-> β , γ .

עצם<->העובדת<-> γ , β , γ <->נלמדים<->מאפשרת<->למודל<->לשנות<->על<->גמישות<->הנורמליזציה<->ולבצע<->אותה<->באופן<->חלקי<->בטווח<->הרציף<->עבור<->הערכים<->אותם<->יכולים<->הפרמטרים<->הלו<->לקבל. כמו<->כן,<->אחד<->הטיסיות<->שיש<->בפרמטרים<->הלו<->שימוש<->באופן<->נלמד<->כך<->לא<->אפשר<->מצב<->בו<->המודל<->למעשה<->כל<->לא<->ישתמש<->בפרמטרים<->ובכך<->נפחית<->את<->bias<->שעלול<->להיווצר<->ולגרום<->overfitting. נראה<->כי<->עבור<->מצב<->בו<->0<->=<->1<->=<->\beta<->=<->\gamma<->מקבלים<->למעשה<->זהות<->ל<->\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}

סך הכל,<->נראה<->כי<->שכבות<->Add Norm<->מופיעה<->פעמים<->רבות<->בארכיטקטורה<->ומהווים<->תפקיד<->חשוב<->באופן<->ל מידת<->המודל<->ולמידת<->הרפרזנטציות. היא<->دواגת<->למניעת<->Exploding<->/Vanishing<->Gradients<->ומובילת<->להתכנסות<->בצורה<->יעילה<->יתר.

Feed Forward Neural Network (FFNN)

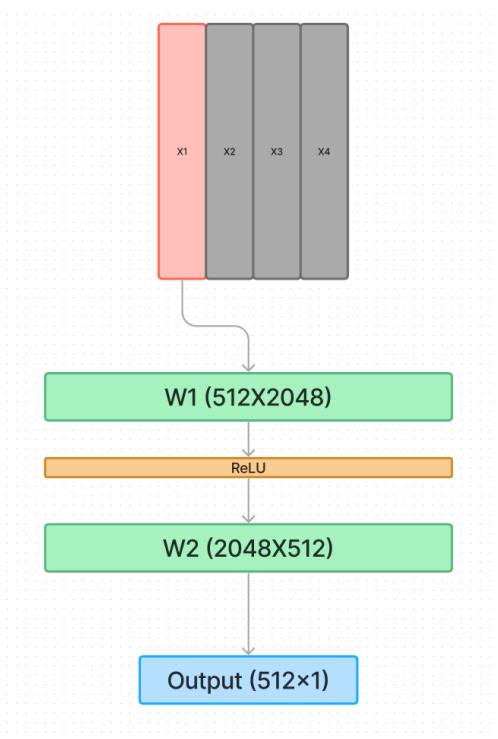
בארQUITטורת ה-Transformer גם ב-Decoder יש רכיב של FFNN שמטרתו ללמידה רפרזנטציות נוספות. דבר חשוב שיש לציין לפני שנותחיל ונתאר את האופן בו ה-FFNN מבצע הוא שעד כה כאשר דיברנו על Attention ראיינו איך ורק פועלות לינאריות כמו מכפלות מטריצות וחיבורים שלහן. עם זאת, ב-FFNN נראה תקופה כי ישנה שימוש בפונקציית אקטיבציה ReLU שמאפשרת למודל ללמוד גם רפרזנטציות לא לינאריות על הדטה.

במאמר המקורי FFNN מתוארת באופן הבא:

$$FFNN(x) = \max(0, xW_1 + b_1) W_2 + b_2$$

נשים לב כי x הוא ווקטור יחיד מתוך הפלט של שכבת-h Norm & Add הקודמת אשר מוכפל בסט משקלות ולאחר מכן מ被执行 ReLU($0, res$) max כפונקציית אקטיבציה לשכבה ולאחר מכן יש מכפלה בסט משקלות נוספת לטובת החזרה למידה המקורי.

הסיבה שאנו לוקחים ווקטור יחיד היא מאחר ואננו אנו יודעים את מידת Embedding מראש אנחנו לא יודעים מה הוא מספר הטוקנים בפרומפט. מאחר ו-FFNN היא רשף קבועה בה יש גודל קבוע של סט משקלות קבוע מראש הפעלת עבור כל אלמנט בנפרד אנחנו לא יכולים להפעיל אותה ישירות על כל הפלט של h Norm & Add מכיוון צרכיים להפעיל אותה על כל ווקטור בנפרד שהוא כאמור בעל גודל קבוע העומד על 1xEmbedding_Size.

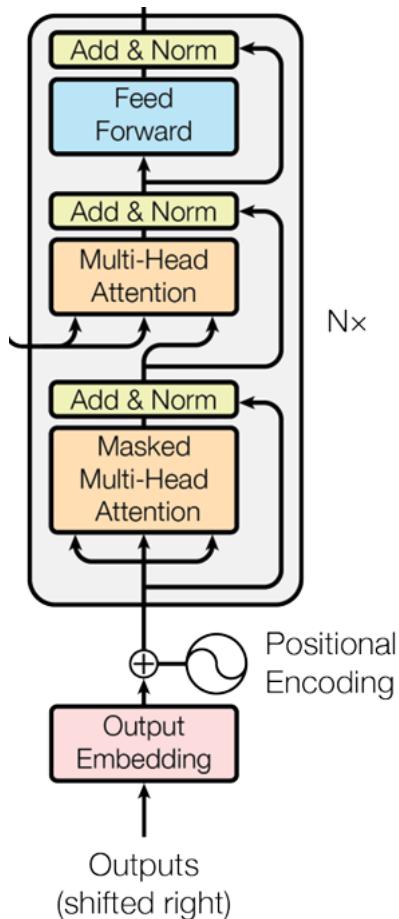


בתרשים ניתן לראות את המתואר בכתב, $X_1 \dots X_4$ מייצגים את רפרזנטציות הטוקנים שהתקבלו לאחר ה-Add & Norm. ניתן לראות כי כל שהגיעה לאחר ה-Attention Is All You Need. מושם מוכפל בנפרד ב-2 סטי משקלות עם שכבת ReLU אחת מהם מושם מושם מוכפל בנפרד המ מיידם כפי בינהן. כמו כן המימדים שרואים בתמונה הם המימדים כפי שהוצעו במאמר המקורי. בדוגמה שלנו המימד הווקטור 512 שקול ל-12288 וכאן ניתן להעריך גם שיטך והמיופי ב-FFNN (恬才) למימד גבוה יותר מ-2048 למשל כמו $49152 = 12288 \times 4$ (אבל זו רק השערה).

כמו כן מאחר ומדבר במטריצות גדולות ומאחר ומתבצע עיבוד של כל ווקטור בנפרד דבר שעלול להיות יקר בזמן ריצה יתכן וישנם פתרונות הנדסיים שמטרתם להקל על התהילה למשל כמו עבודה ב-batches, Low Ranking וכו'.

מה שכן, ניתן לשים לב כי הפלט הסופי של ה-FFNN זהה בגודלו לקלט הנקנס אל ה-FFNN עבור כל ווקטור (512×1) והשוני הוא בערכים שהשתנו בעקבות רפרזנטציה שהמודל למד והובילה לשינויים.

Decoder

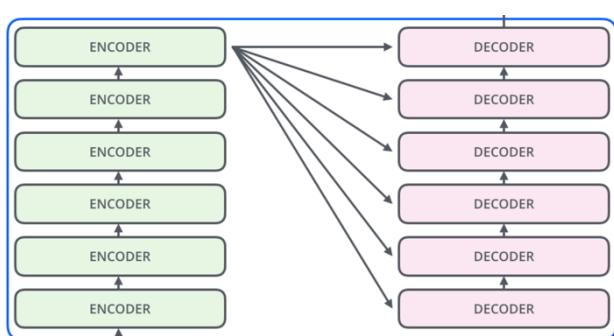


כפי שכבר דובר, Decoder הוא החלק בארכיטקטורה שמטרתו ללמידה ופריזנטציה עבור חיזוי המילה הבאה ברצף. לאחר מכן Encoder לומד ופריזנטציות עבור קידוד הפורומפט והפקה של המהות שלנו באמצעות למידת התלות בין הטוקנים השונים, ניתן להסתכל על Decoder ככזה שלוקח את המידע המקיים, יודע לעכל אותו ולפיכך להפיק תובנות משמעותיות שיובילו לייצור Probability Distribution מעל ה-s-*Corpus* לטובות ניבוי המילה הבאה ברצף.

מהסתכלות על התמונה ועל ארכיטקטורת ה-Decoder ניתן לראות כי היא מורכבת מעט יותר מארכיטקטורת ה-Encoder אך עם זאת, מכילה כבר קומפוננטות שכבר ראיינו ו她们 לידי ביטוי בהרכבת שונות.

כמו ה-Encoder גם Decoder יש N העתקים שנקראים בוצרה איטרטיבית אחד לאחרו. בדומה ל-Encoder, גם כאן הפלט של ה-Decoder במקומות 0-1 מועבר ל-Decoder ה- $i+1$. יוצא מכך ה-Decoder הוא ה-Encoder במקומות 0-1 שמקבל את הקלט של מה-Encoder בלבד.

Decoder ניתן לראות שיש שני Multi Head Attention, הראשון מקבל את הקלט מה-Encoder הקודם והשני מקבל את הקלט מה-Encoder ה- i -י ובנוסף גם מה-Encoder שמצוי לפני אותו ה-Decoder.



כמו כן ניתן להבחין באյור כי יש שני חיצים שנכונים אל עבר Encoder Multi Head Attention מה-Encoder וחז' יחיד שנcono מה-Decoder. Masked Multi Head Attention ב-Decoder.

החיצים שנכנים מה-Encoder יהוו את הקלט שיוכפל בסוט המשקولات W_V , W_K ויבלו לקבלת V , K . והקלט שיגיע מה-Decoder יוכפל ב- W_Q ויבל לתקבלת Q .

אילוסטרציה הרצה נומריית עבור Inference

cut נרצה לבצע אילוסטרציה מספורית עבור Inference בארכיטקטורה. האילוסטרציה תתמקד בפרומפט המכיל 3 מילים שיהו גם כטוקנים ולהם נרצה לחזות את המילה הרביעית. הארכיטקטורה בה נבחר היא צזו בה יציג ה-embedding הוא מימד 3, ישנים 2 ראשים ב-head Multi head Attention וכן כמו כן יש 2 Encoders ו-2 Decoders.

Inputs

ה-input הוא הפרומפט "love eating love I" המורכב מהמלים: "love", "eating", "love", "I". לאחר ואנו מבצעים inference אזי כבר קיימת Embedding Matrix שהתקבלה לאחר תהליכי האימון ולכן נניח כי האינדקס במטריצה המתאים לכל מילה הוא: "I" = 1, "love" = 2, "eating" = 3.

Inputs Embedding

על סמך האינדקסים שהבאנו מה-table look up עבור כל מילה ניגש CUT אל ה-x-axis ונשלוף את הייצוגים הווקטוריים המתאימים לכל אחת מהמלים. נניח והערכיהם הם:

$$\begin{aligned} I &= [0.1, 0.2, 0.3] \\ \text{love} &= [0.2, 0.1, 0.5] \\ \text{eating} &= [0.4, 0.4, 0.3] \end{aligned}$$

Positional Encoding

נזכיר בקורס ה-PE ונזכיר עבור כל אחד מהייצוגים הווקטוריים את ערכיהם במשווהות הבאות:

$$k \text{ is even} = \sin\left(\frac{t}{N^{\frac{k}{d_{model}}}}\right), \quad k \text{ is odd} = \cos\left(\frac{t}{N^{\frac{k}{d_{model}}}}\right)$$

תזכורת:

t – מייצג את ה-position של הトーון (במקרה שלנו מילה) בפרומפט

k – מייצג את ה-position של הסקלר בווקטור ה-embedding

N – מייצג מספר גדול אותו אנו מעלים בחזקה (10000)

d – מייצג את מידת ה-embedding אליו אנו עובדים (במקרה שלנו 3)

כמו כן, חשוב לציין כי הקולט לפונקציות Sin, Cos הוא ברדייאנים ולא במעלות

Position 0: "I"

$$PE(0,0) = \sin\left(\frac{0}{10000^{\frac{0}{3}}}\right) = \sin(0) = 0$$

$$PE(0,1) = \cos\left(\frac{0}{10000^{\frac{1}{3}}}\right) = \cos(0) = 1$$

$$PE(0,2) = \sin\left(\frac{0}{10000^{\frac{2}{3}}}\right) = \sin(0) = 0$$

Overall Positional Vector: [0, 1, 0.002]

Position 1: "love"

$$PE(1,0) = \sin\left(\frac{1}{10000^{\frac{0}{3}}}\right) = \sin(0) = 0$$

$$PE(1,1) = \cos\left(\frac{1}{10000^{\frac{1}{3}}}\right) \approx \cos(0) = 0.99$$

$$PE(1,2) = \sin\left(\frac{1}{10000^{\frac{2}{3}}}\right) \approx \sin(0.002) = 0.002$$

Overall Positional Vector: [0, 0.99, 0.002]

Position 2: "eating"

$$PE(2,0) = \sin\left(\frac{2}{10000^{\frac{0}{3}}}\right) \approx \sin(0) = 0.9$$

$$PE(2,1) = \cos\left(\frac{2}{10000^{\frac{1}{3}}}\right) \approx \cos(0) = 0.99$$

$$PE(2,2) = \sin\left(\frac{2}{10000^{\frac{2}{3}}}\right) \approx \sin(0) = 0.004$$

Overall Positional Vector: [0.9, 0.99, 0.004]

סך הכל נראה כי לאחר חיבור הווקטורים שמייצגים את ה-PE בצורה element-wise אל הווקטורים המקוריים שהגיעו מה-Embedding Matrix נקבל:

$$I = [0.1, 0.2, 0.3] + [0, 1, 0.002] = [0.1, 1.2, 0.302]$$

$$love = [0.2, 0.1, 0.5] + [0, 0.99, 0.002] = [0.2, 1.09, 0.502]$$

$$eating = [0.4, 0.4, 0.3] + [0.9, 0.99, 0.004] = [1.3, 1.39, 0.304]$$

Encoder 1

כאמור, בדוגמה זו יש לנו שני Encoders. כל אחד מה-Encoders מכילים קומפוננטת Multi Head Attention וככל אחד מה-Multi Head Attentions מכיל קומפוננטת Attention Head :Attention Head 1

Attention Head 1

כאמור לאחר שלב ה-PE אנחנו נכנסים ל-Encoder עם הייצוגים הוקטוריים הבאים:

$$\begin{aligned} I &= [0.1, 1.2, 0.302] \\ \text{love} &= [0.2, 1.09, 0.502] \\ \text{eating} &= [1.3, 1.39, 0.304] \end{aligned}$$

היצוגים הוקטוריים הללו יכולים גם להיות מוצגים במטריצה מגודל 3×3 הבא:

0.1	0.2	1.3
1.2	1.09	1.39
0.302	0.502	0.304

כמו כן, נגידר 4 מטריצות באופן שרירותי שייצגו את W_Q , W_K , W_V , $W_{Value \uparrow}$, $W_{Value \downarrow}$ (חשיבות לציין כי ההגדירה היא שרירותית ולכן לא נתיחס לאפייניות של המילים האחת לשניה אלא לעקרון הכללי). כמו כן, נניח כי המימד אליו מתמפים ה-Q,K,V הוא 2 (בהתאם לתיאורתי המיפוי היה מממד ווקטור של למימד נמוך יותר 128 – כאן המיפוי הוא מממד ווקטורי 3 למימד נמוך יותר 2) 12288

W_Q (2x3)	0.1 0.4	0.2 0.5	0.3 0.6
----------------	------------	------------	------------

W_K (2x3)	0.15 0.45	0.25 0.55	0.35 0.65
----------------	--------------	--------------	--------------

W_V (2x3)	0.2 0.5	0.3 0.6	0.4 0.7
----------------	------------	------------	------------

$W_{Value \uparrow}$ (3x2)	0.25 0.35 0.45	0.55 0.65 0.75
-------------------------------	----------------------	----------------------

nbצע את החישובים הבאים:

$$W_Q \otimes E = Q$$

W_Q (2x3)	0.1	0.2	0.3
	0.4	0.5	0.6

0.1	0.2	1.3
1.2	1.09	1.39
0.302	0.502	0.304

Q (2x3)	0.341	0.389	0.499
	0.821	0.926	1.397

$$W_K \otimes E = K$$

W_K (2x3)	0.15	0.25	0.35
	0.45	0.55	0.65

0.1	0.2	1.3
1.2	1.09	1.39
0.302	0.502	0.304

K (2x3)	0.421	0.478	0.649
	0.901	1.016	1.547

$$W_{Value\downarrow} \otimes E = Value \downarrow$$

$W_{Value\downarrow}$ (2x3)	0.2	0.3	0.4
	0.5	0.6	0.7

0.1	0.2	1.3
1.2	1.09	1.39
0.302	0.502	0.304

$Value \downarrow$ (2x3)	0.501	0.568	0.799
	0.981	1.105	1.697

$$W_{Value\uparrow} \otimes W_{Value\downarrow} = Value$$

$W_{Value\uparrow}$ (3x2)	0.25	0.55
	0.35	0.65
	0.45	0.75

$Value \downarrow$ (2x3)	0.501	0.568	0.799
	0.981	1.105	1.697

$Value$ (3x3)	0.665	0.750	1.133
	0.813	0.917	1.383
	0.961	1.084	1.632

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

cut נמייר וنبצע חישוב עבור:

Q (2x3)	Q1=I	Q2=love	Q3=eating
	0.341	0.389	0.499
	0.821	0.926	1.397

K (2x3)	K1=I	K2=love	K3=eating
	0.421	0.478	0.649
	0.901	1.016	1.547

$$Q \otimes K^T =$$

	Q1	Q2	Q3
K1	$K_1 \cdot Q_1$	$K_1 \cdot Q_2$	$K_1 \cdot Q_3$
K2	$K_2 \cdot Q_1$	$K_2 \cdot Q_2$	$K_2 \cdot Q_3$
K3	$K_3 \cdot Q_1$	$K_3 \cdot Q_2$	$K_3 \cdot Q_3$

$$Q \otimes K^T =$$

	Q1	Q2	Q3
K1	0.883	0.998	1.469
K2	0.997	1.049	1.588
K3	1.491	1.685	2.485

$$\frac{Q \otimes K^T}{\sqrt{d}} =$$

	Q1	Q2	Q3
K1	$\frac{0.883}{\sqrt{3}}$	$\frac{0.998}{\sqrt{3}}$	$\frac{1.469}{\sqrt{3}}$
K2	$\frac{0.997}{\sqrt{3}}$	$\frac{1.049}{\sqrt{3}}$	$\frac{1.588}{\sqrt{3}}$
K3	$\frac{1.491}{\sqrt{3}}$	$\frac{1.685}{\sqrt{3}}$	$\frac{2.485}{\sqrt{3}}$

$$\text{softmax} \left(\frac{Q \otimes K^T}{\sqrt{d}} \right) =$$

	Q1	Q2	Q3
K1	0.258	0.284	0.286
K2	0.276	0.292	0.306
K3	0.464	0.422	0.407

$$\text{softmax} \left(\frac{Q \otimes K^T}{\sqrt{d}} \right) V =$$

	Q1	Q2	Q3
K1	1.138	0.755	0.669
K2	1.201	0.796	0.706
K3	1.802	1.195	1.060

חשוב לבדוק ולצין כי החישוב שעשינו הוא עבור 1 Attention Head וכפי שציינו בהתחלה, בדוגמה זו השתמש ב-2 ראשי Attention. על מנת לא להעמעס על הסיכום ניצור שרירותית מטריצה המייצגת את תוצאות החישוב של 2 Attention Head וונכתב את 2 המטריצות זו לצד זו כעת:

Attention Head 1				Attention Head 2			
	Q1	Q2	Q3		Q1	Q2	Q3
K1	1.138	0.755	0.669	K1	0.347	0.553	0.556
K2	1.201	0.796	0.706	K2	-0.230	0.483	0.809
K3	1.802	1.195	1.060	K3	1.160	1.540	1.243

מחיבור element-wise של 2 המטריצות קיבל $\Delta E = \Delta E_1 + \Delta E_2$ השווה ל- ΔE כולם לערך הכלל שמופק עבור Multi Head Attention.

	Q1	Q2	Q3
K1	1.485	1.308	1.225
K2	0.971	1.279	1.515
K3	2.962	2.735	2.303

Add & Norm 1

בשלב ה-Add נבצע הוספה ישירה של המטריצה שהתקבלה מה-Multi Head Attention למטריצה המקורי:

$$\begin{array}{|c|c|c|} \hline 0.1 & 0.2 & 1.3 \\ \hline 1.2 & 1.09 & 1.39 \\ \hline 0.302 & 0.502 & 0.304 \\ \hline \end{array} + \begin{array}{|c|c|c|} \hline 1.485 & 1.308 & 1.225 \\ \hline 0.971 & 1.279 & 1.515 \\ \hline 2.962 & 2.735 & 2.303 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 1.585 & 1.508 & 2.525 \\ \hline 2.171 & 2.369 & 2.905 \\ \hline 3.264 & 3.237 & 2.607 \\ \hline \end{array}$$

בשלב הנורמליזציה נבצע נורמליזציה לכל עמודה בנפרד באופן בו הפרמטרים הנלמדים β, γ יקבעו באופן שרירותי $\beta = 0.6, \gamma = 0.5$. בנוסף נגדיר $\epsilon = 10^{-8}$ ונקבל:

$$\begin{array}{|c|c|c|} \hline 1.585 & 1.508 & 2.525 \\ \hline 2.171 & 2.369 & 2.905 \\ \hline 3.264 & 3.237 & 2.607 \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|} \hline 0.057 & -0.011 & 0.128 \\ \hline 0.478 & 0.598 & 1.292 \\ \hline 1.263 & 1.213 & 0.379 \\ \hline \end{array}$$

Feed Forward Neural Network 1

בשלב ה-FFNN למשה אנחנו עושים מיפוי של כל ווקטור מהמטריצה שהתקבלה בשכבה הקודמת של ה-Norm & Add. למינד גובה יותר על ידי הכפלה ב-Layer הראשון ב-FFNN, לאחר מכן מבצעים ReLU ולבסוף מחזירים את הווקטור למינד המקורי. כפי שצווין כאן פועלים על כל ווקטור בצורה ייחידית אך עם זאת לבטח כי מנות אופטימיזציות הנדרסיות שמאפשרות עבודה על מטריצה כמקשה אחת.

לצורך הדוגמא, נניח ה-Layer הראשון ממפה את הווקטור למינד 4 ולאחר מכן מחזיר אותו בחזרה למינד 3 באמצעות ה-Layer השני לאחר שביצעReLU. נגידר את השכבות באופן שרירותי:

Layer 1			Layer 2			
0.033	0.313	-0.012	0.542	-0.313	-0.989	-0.222
-0.128	0.552	1.312	-0.982	1.121	-0.876	1.122
1.872	-0.876	1.231	1.523	0.665	0.872	1.878
1.277	-0.438	-0.279				

чисוב עבור ווקטור 1:

$$Layer1 \otimes Vec1 = \begin{array}{|c|c|c|} \hline 0.033 & 0.313 & -0.012 \\ \hline -0.128 & 0.552 & 1.312 \\ \hline 1.872 & -0.876 & 1.231 \\ \hline 1.277 & -0.438 & -0.279 \\ \hline \end{array} = \begin{array}{|c|} \hline 0.057 \\ \hline 0.478 \\ \hline 1.263 \\ \hline \end{array} \begin{array}{|c|} \hline 0.136 \\ \hline 1.914 \\ \hline 1.243 \\ \hline -0.489 \\ \hline \end{array}$$

$$ReLU(Layer1 \otimes Vec1) = \begin{array}{|c|} \hline 0.136 \\ \hline 1.914 \\ \hline 1.243 \\ \hline 0 \\ \hline \end{array}$$

$$Layer 2 (ReLU(Layer1 \otimes Vec1)) = \begin{array}{|c|c|c|c|} \hline 0.542 & -0.313 & -0.989 & -0.222 \\ \hline -0.982 & 1.121 & -0.876 & 1.122 \\ \hline 1.523 & 0.665 & 0.872 & 1.878 \\ \hline \end{array} = \begin{array}{|c|} \hline 0.136 \\ \hline 1.914 \\ \hline 1.243 \\ \hline 0 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline -1.755 \\ \hline 0.923 \\ \hline 2.564 \\ \hline \end{array}$$

חישוב עברור וקטור 2:

$$Layer1 \otimes Vec2 = \begin{array}{|c|c|c|} \hline 0.033 & 0.313 & -0.012 \\ \hline -0.128 & 0.552 & 1.312 \\ \hline 1.872 & -0.876 & 1.231 \\ \hline 1.277 & -0.438 & -0.279 \\ \hline \end{array} = \begin{array}{|c|} \hline -0.011 \\ \hline 0.598 \\ \hline 1.213 \\ \hline \end{array} \begin{array}{|c|} \hline 0.172 \\ \hline 1.923 \\ \hline 0.949 \\ \hline -0.614 \\ \hline \end{array}$$

$$ReLU(Layer1 \otimes Vec2) = \begin{array}{|c|} \hline 0.172 \\ \hline 1.923 \\ \hline 0.949 \\ \hline 0 \\ \hline \end{array}$$

$$Layer 2 (ReLU(Layer1 \otimes Vec2)) = \begin{array}{|c|c|c|c|} \hline 0.542 & -0.313 & -0.989 & -0.222 \\ \hline -0.982 & 1.121 & -0.876 & 1.122 \\ \hline 1.523 & 0.665 & 0.872 & 1.878 \\ \hline \end{array} \begin{array}{|c|} \hline 0.172 \\ \hline 1.923 \\ \hline 0.949 \\ \hline 0 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline -1.447 \\ \hline 1.155 \\ \hline 2.368 \\ \hline \end{array}$$

חישוב עברור וקטור 3:

$$Layer1 \otimes Vec3 = \begin{array}{|c|c|c|} \hline 0.033 & 0.313 & -0.012 \\ \hline -0.128 & 0.552 & 1.312 \\ \hline 1.872 & -0.876 & 1.231 \\ \hline 1.277 & -0.438 & -0.279 \\ \hline \end{array} = \begin{array}{|c|} \hline 0.128 \\ \hline 1.292 \\ \hline 0.379 \\ \hline \end{array} \begin{array}{|c|} \hline 0.404 \\ \hline 1.194 \\ \hline -0.426 \\ \hline -0.508 \\ \hline \end{array}$$

$$ReLU(Layer1 \otimes Vec3) = \begin{array}{|c|} \hline 0.404 \\ \hline 1.194 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}$$

$$Layer 2 (ReLU(Layer1 \otimes Vec3)) = \begin{array}{|c|c|c|c|} \hline 0.542 & -0.313 & -0.989 & -0.222 \\ \hline -0.982 & 1.121 & -0.876 & 1.122 \\ \hline 1.523 & 0.665 & 0.872 & 1.878 \\ \hline \end{array} \begin{array}{|c|} \hline 0.404 \\ \hline 1.194 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}$$

$$\begin{array}{|c|} \hline -0.155 \\ \hline 0.942 \\ \hline 1.409 \\ \hline \end{array}$$

סך הכל נראה שמה-FFNN יצאונו עם שלושת הוקטורים הבאים:

-1.755	-1.447	-0.155
0.923	1.155	0.942
2.564	2.368	1.409

אשר נוכל לאחדם לכדי מטריצה:

-1.755	-1.447	-0.155
0.923	1.155	0.942
2.564	2.368	1.409

cutת נרצה להפעיל Add & Norm גם על הפלט של ה-FFNN ונקבל:

המטריצה מה-MHA	+	המטריצה שהתקבלת מה-FFNN	=	מטריצה לאחר Add
1.485 1.308 1.225 0.971 1.279 1.515 2.962 2.735 2.303		-1.755 -1.477 -0.155 0.923 1.155 0.942 2.564 2.368 1.409		-0.27 -0.169 1.1 1.894 2.434 2.457 5.526 5.103 3.712

בשלב הנורמליזציה נבצע נורמליזציה לכל עמודה בנפרד באופן בו הפרמטרים הנלמדים β, γ יקבעו באופן שירוטי $\beta = 0.7, \gamma = 0.3$. בנוסף נגדיר $\epsilon = 10^{-8}$ ונקבל:

-0.27	-0.169	1.1
1.894	2.434	2.457
5.526	5.103	3.712

→

0.367	0.334	0.327
0.638	0.696	0.709
1.094	1.068	1.062

המטריצה הכתומה היא המטריצה של Encoder 1 לאחר כל ביצוע FFNN, Multi Head Attention. ישים לב כי הפלט של Encoder 1 זהה בגודלו לקלט של Encoder 1 ושתי שכבות של Add & Norm. השכבות שיופיעו ב-Encoder 2 זהים לחילוטין לשכבות שביצענו והוא מהווה את הקלט ל-Encoder 2. השכבות שיופיעו ב-Encoder 1 הן מטריצה שירוטית שתהוויה לנו את הפלט של Encoder 2 איתה נמשיך ונתקדם לעבר Decoder 2.

Encoder 2

0.262	0.424	0.764
0.538	1.696	1.199
0.174	0.028	1.222

Decoder 1

לאחר שתיארנו את החלק של ה-Encoder, נרצה כעת לעבור לחלק של ה-Decoder שמקבל כקלט את החלק של ה-Encoder ב-Multi Head Attention אשר תחילתה נשים לב כי יש קומפוננטה התחלתיות נוספת של Masked Multi Head Attention שמקבלת את הפלט של ה-Decoder הקודם. מאחר ואנחנו בשכבה הראשונה של ה-Decoder (מתוך 2) וכן כן אנחנו מיצרים את המילה הראשונה אזי אין Decoder קודם ואין אף מלל קודם ולכן הטוקן שייכנס הוא טוקן של **<SOS>**, שהמשמעות שלו היא **.Start Of Sentence**.

0.2
1.76
0.702

Masked Multi Head Attention 1

נייצג את הקלט של **<SOS>** באופן הבא (אופן שרירותי):

יש לציין כי הערך זהה שנבחר שרירותית הוא לאחר Positional Encoding ומוקן ומצוון כבר להיכנו אל ה-Decoder

כמו כן עבור קומפוננטת ה-Masked Multi Head Attention נגדיר את המשקלות הנלמדות הבאות:

W_Q	0.6	0.5	0.4
(2x3)	0.3	0.2	0.1

W_K	0.65	0.55	0.45
(2x3)	0.35	0.25	0.15

$W_{Value\downarrow}$	0.7	0.6	0.5
(2x3)	0.4	0.3	0.2

$W_{Value\uparrow}$	0.75	0.45
(3x2)	0.65	0.35
	0.55	0.25

נבצע את החישובים הבאים:

$$W_Q \otimes E = Q$$

W_Q (2x3)	0.6	0.5	0.4	0.2
	0.3	0.2	0.1	1.76

0.2
1.76
0.702

Q (2x1)	1.281
	0.482

$$W_K \otimes E = K$$

W_K (2x3)	0.65	0.55	0.45	0.2
	0.35	0.25	0.15	1.76

0.2
1.76
0.702

K (2x1)	1.414
	0.615

$$W_{Value\downarrow} \otimes E = Value \downarrow$$

$W_{Value\downarrow}$ (2x3)	0.7	0.6	0.5	0.2
	0.4	0.3	0.2	1.76

0.2
1.76
0.702

$Value \downarrow$ (2x1)	1.547
	0.748

$$W_{Value\uparrow} \otimes W_{Value\downarrow} = Value$$

$W_{Value\uparrow}$ (3x2)	0.75	0.45
	0.65	0.35
	0.55	0.25

$Value \downarrow$ (2x1)	1.547
	0.748

$Value$ (3x1)	1.497
	1.267
	1.038

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

cut נמשיך ונבצע חישוב עבור:

Q (2x1)	Q1=<SOS>
	1.281
	0.482

K (2x1)	K1=<SOS>
	1.414
	0.615

$$Q \otimes K^T =$$

	Q1
K1	$K_1 \cdot Q_1$

$$Q \otimes K^T =$$

	Q1
K1	2.108

$$\frac{Q \otimes K^T}{\sqrt{d}} =$$

	Q1
K1	$\frac{2.108}{\sqrt{3}}$

$$\frac{Q \otimes K^T}{\sqrt{d}} =$$

	Q1
K1	1.217

$$\text{softmax} \left(\frac{Q \otimes K^T}{\sqrt{d}} \right) V =$$

0.429
0.324
0.245

חשוב לבדוק ולציין כי החישוב שעשינו הוא עבור 1 Attention Head וכפי שציינו בהתחלה, בדוגמה זו השתמש ב-2 ראשי Attention. על מנת לא להעמעס על הסיכום ניצור שרירותית מטריצה המייצגת את תוצאת החישוב של 2 Attention Head וונכתב את 2 המטריצות זו לצד זו如下:

Attention Head 1

0.429
0.324
0.245

Attention Head 2

1.023
0.003
0.120

מחיבור element-wise של 2 המטריצות נקבל $\Delta E_1 + \Delta E_2 = \Delta E$ השווה לערך הכלל שמופיע עבור ה-Multi Head Attention.

1.452
0.327
0.365

Add & Norm 1 - Decoder

בשלב הנורמליזציה נבצע נורמליזציה לכל عمودה בנפרד באופן בו הפרמטרים הנלמדים β, γ יקבעו באופן שרירותי $\beta = 0.3, \gamma = 0.3$. בנוסף נגדיר $\epsilon = 10^{-8}$ ונקבל:

$$\begin{array}{c} 1.452 \\ 0.327 \\ 0.365 \end{array} + \begin{array}{c} 0.2 \\ 1.76 \\ 0.702 \end{array} \rightarrow \begin{array}{c} 0.735 \\ 1.048 \\ 0.315 \end{array}$$

Multi Head Attention 1

לאחר שסיימנו עם הקומפוננטה של ה-Masked Multi Head Attention ב-Decoder הראשון כמו כן ביצענו Add & Norm, אנחנו ממשיכים עם הפלט אל עבר קומפוננטת ה-Multi Head Attention& Decoder וגם מה-Encoder (הקלט אליו אנו ממשיכים הלאה) שכאמרם מקבלת את הקולט שלה גם מה-Decoder וגם מה-Encoder.

חשוב לציין כי בקומפוננטת ה-Multi Head Attention שיכפלו ב- W_K , $W_{Value\downarrow}$, $W_{Value\uparrow}$, המטריצות שיוכפלו ב- W_Q יגעו מה-Encoder ה-n-i (האחרון) ואילו המטריצה שתוכפל ב- W תגיע מה-Decoder (מה שחשבנו כתעט).

עבור ה- Decoder Multi Head Attention- Value הנקודות הנלמדות הבאות:

$W_{Value\downarrow}$	0.4	0.3	0.2
(2x3)	0.7	0.6	0.5

$W_{Value\uparrow}$	0.45	0.75
(3x2)	0.35	0.65
	0.25	0.55

W_Q	0.3	0.2	0.1
(2x3)	0.6	0.5	0.4

W_K	0.35	0.25	0.15
(2x3)	0.65	0.55	0.45

נבצע את החישובים הבאים:

$$W_Q \otimes E = Q$$

W_Q	0.3	0.2	0.1
(2x3)	0.6	0.5	0.4

0.735
1.048
0.315

Q	0.462
(2x1)	1.091

$$W_K \otimes E = K$$

W_K	0.35	0.25	0.15
(2x3)	0.65	0.55	0.45

0.262	0.424	0.764
0.538	1.696	1.199
0.174	0.028	1.222

K	0.252	0.577	0.750
(2x3)	0.545	1.221	1.706

$$W_{Value\downarrow} \otimes E = Value \downarrow$$

$W_{Value\downarrow}$	0.4	0.3	0.2
(2x3)	0.7	0.6	0.5

0.262	0.424	0.764
0.538	1.696	1.199
0.174	0.028	1.222

$Value \downarrow$	0.301	0.684	0.910
(2x3)	0.541	1.320	1.499

$$W_{Value\uparrow} \otimes W_{Value\downarrow} = Value$$

$W_{Value\uparrow}$	0.45	0.75
(3x2)	0.35	0.65
	0.25	0.55

$Value \downarrow$	0.301	0.684	0.910
(2x3)	0.541	1.320	1.499

$Value$	0.541	1.298	1.534
(3x3)	0.457	1.097	1.293
	0.373	0.897	1.052

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V$$

כעת נמשיך ונבצע חישוב עבור:

Q (2x1)	0.462
	1.091

K (2x3)	0.252	0.577	0.750
	0.545	1.221	1.706

$$Q \otimes K^T =$$

	Q1
K1	$K_1 \cdot Q_1$
K2	$K_2 \cdot Q_1$
K3	$K_3 \cdot Q_1$

$$Q \otimes K^T =$$

	Q1
K1	0.711
K2	1.588
K3	1.491

$$\frac{Q \otimes K^T}{\sqrt{d}} =$$

	Q1
K1	$\frac{0.711}{\sqrt{3}}$
K2	$\frac{1.588}{\sqrt{3}}$
K3	$\frac{2.208}{\sqrt{3}}$

$$\text{softmax} \left(\frac{Q \otimes K^T}{\sqrt{d}} \right) =$$

	Q1
K1	0.410
K2	0.916
K3	1.274

$$\text{softmax} \left(\frac{Q \otimes K^T}{\sqrt{d}} \right) V =$$

	Q1
K1	3.365
K2	2.840
K3	2.315

גם כאן, חשוב להבחין ולציין כי החישוב שעשינו הוא עבור 1 Attention Head וככפי שציינו בהתחלה, בדוגמה זו נשתמש ב-2 ראשי Attention. על מנת לא להעמעס על הסיכום ניצר שרירותית מטריצה המציגת את תוצאות החישוב של 2 Attention Head זו לצד זו כעת:

Attention Head 1

3.365
2.840
2.315

Attention Head 2

0.763
1.002
0.008

חיבור של 2 המטריצות נקבע $\Delta E = \Delta E_1 + \Delta E_2$ השווה ל- ΔE כלומר לערך הכלל שמופיע עבור ה-Multi Head Attention.

4.128
3.842
2.323

Add & Norm 2 - Decoder

בשלב הנורמליזציה נבצע נורמליזציה לכל עמודה בנפרד באופן בו הפרמטרים הנלמדים β, γ יקבעו באופן שרירותי $\beta = 0.7, \gamma = 0.3$. בנוסף נגדיר $\epsilon = 10^{-8}$ ונקבל:

$$\begin{array}{c} \begin{matrix} 4.128 \\ 3.842 \\ 2.323 \end{matrix} + \begin{matrix} 0.2 \\ 1.76 \\ 0.702 \end{matrix} \rightarrow \begin{matrix} 0.702 \\ 1.066 \\ 0.331 \end{matrix} \end{array}$$

Feed Forward Neural Network 1 - Decoder

בשלב ה-FFNN למשה אנחנו עושים מיפוי של כל וקטור מהמטריצה שהתקבל בשכבה הקודמת של ה-Norm & Add & Layer <SOS>. FFNN הראשון ב-Layer, מקבל מכאן מבצעים ReLU ולבסוף מחזירים את הווקטור למילד המקורי. נשים לב כי במקרה זה עברו Decoder ה-MLP מעתה ותחלנו עם וקטור שמייצג רק את <SOS> אזי אנחנו עובדים עם וקטור יחיד לאחר הפלט של שכבה ה-Multi Head Attention.

לצורך הדוגמא, נניח ה-Layer הראשון ממפה את הווקטור למילד 4 ולאחר מכן מחזיר אותו בחזרה למילד 3 באמצעות ה-Layer השני לאחר שביצע ReLU. נגיד את השכבות באופן שritable:

Layer 1			Layer 2			
0.023	0.313	-0.412	0.122	-0.313	-0.489	0.255
-0.228	0.322	0.415	1.982	-1.221	-0.842	1.123
0.432	-0.876	1.121	1.523	0.635	0.322	-1.178
1.877	-1.438	-0.229				

чисוב עבור וקטור 1:

$$Layer1 \otimes Vec1 = \begin{array}{|c|c|c|} \hline 0.023 & 0.313 & -0.412 \\ \hline -0.228 & 0.322 & 0.415 \\ \hline 0.432 & -0.876 & 1.121 \\ \hline 1.877 & -1.438 & -0.229 \\ \hline \end{array} = \begin{array}{|c|} \hline 0.213 \\ \hline 0.321 \\ \hline -0.260 \\ \hline -0.291 \\ \hline \end{array}$$

$$ReLU(Layer1 \otimes Vec1) = \begin{array}{|c|} \hline 0.213 \\ \hline 0.321 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}$$

$$Layer 2 (ReLU(Layer1 \otimes Vec1)) = \begin{array}{|c|c|c|c|} \hline 0.122 & -0.313 & -0.489 & 0.255 \\ \hline 1.982 & -1.221 & -0.842 & 1.123 \\ \hline 1.523 & 0.635 & 0.322 & -1.178 \\ \hline \end{array} = \begin{array}{|c|} \hline -0.074 \\ \hline 0.030 \\ \hline 0.528 \\ \hline \end{array}$$

cut נרצה להפעיל Add & Norm FFNN ונקבל:

המטריצה מה-MHA	+	המטריצה שהתקבלת מה-FFNN	=	המטריצה לאחר Add
0.702 1.066 0.331		-0.074 0.030 0.528		0.628 1.096 0.859

בשלב הנורמליזציה נבצע נורמליזציה לכל עמודה בנפרד באופן בו הפרמטרים הנלמדים β, γ יקבעו באופן שירוטי $\beta = 0.3, \gamma = 0.7$. בנוסף נגדיר $\epsilon = 10^{-8}$ ונקבל:

0.628 1.096 0.859	→	0.334 1.068 0.696
-------------------------	---	-------------------------

המטריצה הכתומה היא המטריצה של **Decoder 1** לאחר ביצוע כל השלבים: Add & Norm, Multi Head Attention, Masked Multi Head Attention ו-3 שכבות של FFNN. נשים לב כי הפלט של Decoder 1 זהה בגודלו לקלט של Decoder 2 והוא מהווים את הקלט לשכבת Decoder 2. השלבים שהיו ב-2 זהים לחלוון לשכבות שביצעו כעט ב-1-Decoder 2 מנת לא להעמיס על הסיכום, נגדיר מטריצה שירוטית שתהווה לנו את הפלט של Decoder 2 אותה המשיך ונתקדם.

Decoder 2

0.444 0.212 1.312

הפלט זהה הוא הפלט של Decoder השני והאחרון. כעט הפלט זהה יכנס אל השכבה הלינארית. פרט חשוב לי לציין גם כאן הוא שאנו מיצרים כאן את המילה הראשונה ולכן נראה כי המטריצה היא מוגדל 3×3 המייצגת מילוי Embedding בגודל 1 ועל כך שמדובר במילה 1. עם זאת, חשוב לציין כי אם הינו מיצרים את המילה השנייה אזי המטריצה הסופית יהיה בגודל 2×3 וב הכללה בכלל להציג שבחינותן מילוי כלשהו d ובහינתן מילה שאינדקסה הוא ch , אזי גודל המטריצה הסופית יהיה בגודל $d \times d$. אני מצין זאת כי אז נשאלת השאלה, איך מטריצות בגודל שונה יכולות להיכנס לשכבה לינארית בגודל אחד?

התשובה לכך היא שלא כל המטריצה נכנסת אלא רק המילה האחרונות שתמיד תהיה בגודל של $1 \times d$ נכנסת אל השכבה הלינארית והסיבה לכך היא שהיא מגלה בתוך הייצוג הווקטורי שלה את כל החישובים שבוצעו בשכבות-h-Attention ו-FFNN כך שscr הכל תמיד הקלט אל השכבה הלינארית הוא בגודל אחד.

Linear Layer

השכבה הילינארית מקבלת קלט בגודל של $1 \times d$ והוא מכילה וקטור בגודל של $d \times c$ באופן בו ס מייצג את גודל ה-*Corpus*. סך הכל נקלט וקטור שנסמן V_{Cor} בגודל של $1 \times c$.

השכבה הילינארית	פלט ה-Decoder	ווקטור בגודל הקורפוס V_{Cor}
0.232	...	0.456
.		.
.		.
.		.
0.003	...	1.312
		0.012

SoftMax

הדבר האחרון שנותר לעשות זה להפעיל SoftMax על הווקטור V_{Cor} על מנת לקבל בסופה של דבר Probability Distribution העובר כל המילים בקורסוס נבחר את המילה בעלת ההסתברות הגבוהה ביותר.

Moon	0.002
Child	0.003
Bee	0.001
Apples	0.456
	.
	.
	.
Card	0.003
Sun	0.001
Day	0.020
Nice	0.001

$$= \text{SoftMax}(V_{cor})$$

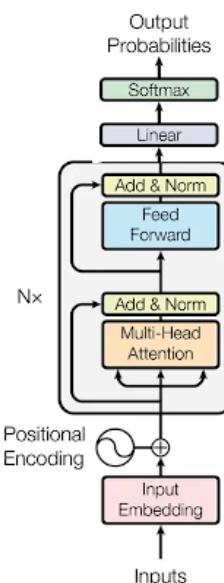
סך הכל נראה כי המילה שקיבלה את ההסתברות הגבוהה ביותר היא המילה Apples ולכן המילה הבאה שתציגו לנו היא המילה Applesvr כך שנקבל כי המשפט שיתקבל יהיה "I love eating apples" ו- "I". באיטרציה הבאה אל ה-Decoder "יכנס הרץ" <SOS>, Apples, Apples" והגינרט ימשך עד שיתקבל הטוקן <EOS> (End Of Sentence) שייעיד על סיום יצירת הטקסט.

ארQUITקטורות מבוססות טרנספורמר

BERT

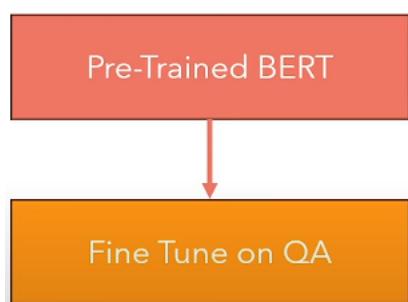
[Video Link](#)

ארQUITקטורה מבוססת טרנספורמר שפותחה גם היא על ידי חוקר גול בשנת 2018. BERT Bidirectional Encoder Representations from Transformers או בקיצור BERT



כשמה כן היא, BERT היא ארQUITקטורה דו כיוונית. בעוד במודל הטרנספורמר הונילה השתמשנו ב-Masking על מנת להבטיח כי המודל לא יוכל ללמידה רפרזנטציות בהתבסס על טוקני עתיק, ב-BERT אנחנו דואקים מאפשרים למודל לעשות שימוש בטוקני עתיק על מנת ללמידה רפרזנטציה עשרה יותר על השפה. זה מעניין כי זה מתאר יכולת שהיא במידת מה אנושית, בהינתן ואני רוצה להביע איך שהשי נקודה מסוימת במהלך המשפט אני אברור את המילים ההתחלתיות שלី בתחילת המשפט כדי שיבלו אותו לראיון המסייע שאותו ארצה לדבר בהמשך המשפט.

- כמו כן, מהתמונה ניתן לראות כי האՐQUITקטורה פשוטית יותר ונראית כי מזכירה יותר את רכיב ה-Encoder, הגדייר 2 סוג א\RQUITקטורות עבור BERT: עבור המודל המקורי, הגדייר 2 שכבות FFNN Layer 3072, Encoders .1. גרסה בסיסית – 12 שכבות .2. ראשיהם של .768 Attention, מרחב וקטורי בגודל 4096, Encoders .2. גרסה מוחבה – 24 שכבות .16 ראשיהם של .1024 Attention, מרחב Embedding וקטורי בגודל 768.



בניגוד לטרנספורמר שתהליכי האימון קורא במקביל ל-Encoder ול-Decoder על אותה סוג מטלה באופן בו יש פעוף גראדינטיים בתהליכי ה-Backpropagation לאורך כל הקומפוננטות, ב-BERT יש תהליכי מעת שונה והוא מורכב מ-2 חלקים: 1. Pre-Trained – אימון ראשון של המודל בהתאם לוגיקה כללית שמטרתה ללמד את המודל לתפוס רפרזנטציות כליליות על השפה 2. Fine Tune – מיקוד המודל במשימה הייעודית אותה הוא יctrיך לבצע ולרכוש בה מיומנות.

אפשר לחשב על 2 החלקים ועל תהליכי לימודים של רופא. תחיליה לרופא יש מספר שנים בהן הוא רוכש ידע בתחום כללים כמו ביולוגיה, כימיה, היסטולוגיה אנטומיה וכו'. ולאחר שלמת התשנים הראשונות בהן למד הרופא רפרזנטציות כליליות מעולם הרופאה הואicut יכול לעבור התמחות בתחום הספציפי אליו הוא מיועד. בעמודים הבאים נסקור 2 שיטות עבור כל חלק ונראה כיצד הן משויות לאופן הפעולה של BERT.

Pre-Trained Methods

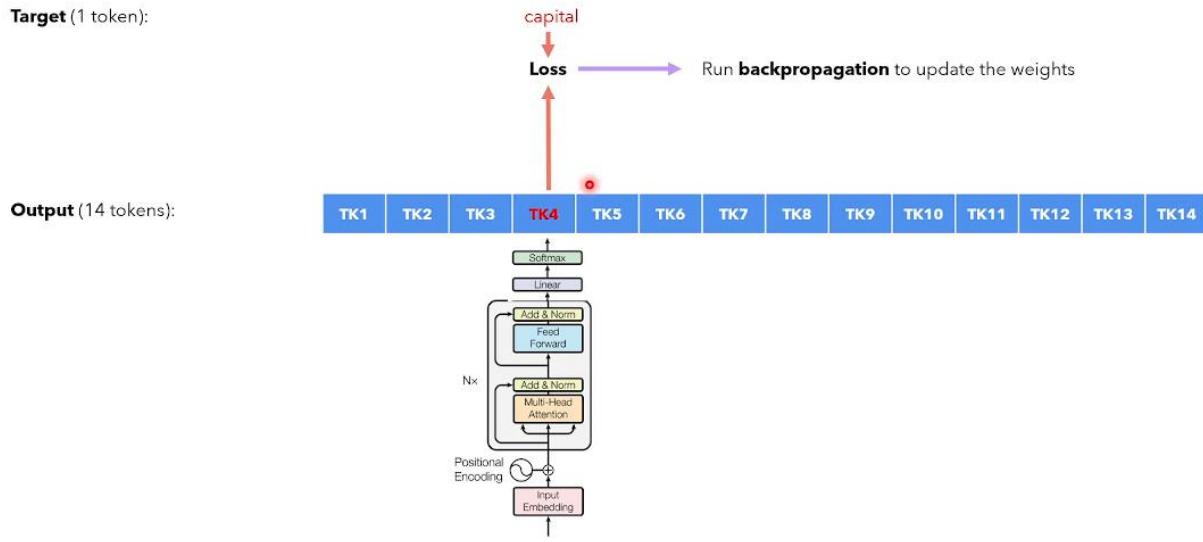
נרצה כעט לסקור 2 שיטות כלליות עבורין יכול המודל ללמידה רפרזנטטיבית כללית לגבי השפה ולהבין אותה בצורה עשירה.

Masked Language Model (MLM)



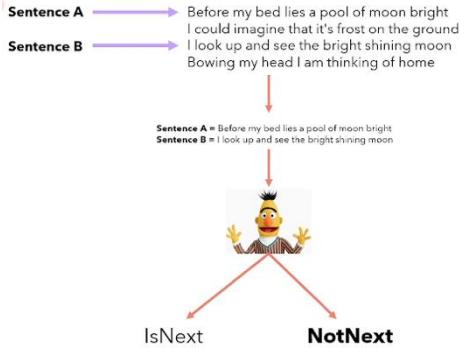
בשיטת זו אנו ממסכים חלק מהמילים במשפט ומעודדים את המודל לחזות את המילים הממוסכotas. לרוב המיסוך יבוצע על 15% מהמילים במשפט ובמהלך תהליכי איטרטיבי המודול ילמד להשלים את המילים באופן בו הן יובילו לקוherenceיות במשפט.

בתמונה ניתן לראות כי המילה שנבחרה להיות ממוסכת היא המילה **capital**. המילה הוחלפה בטוקן מיוחד **[mask]** בעל ייצוג וקטורי ייחודי כך שבמהלך חישוב ה-Attention על סמך המילים הסמוכות ל-[mask] ישתנה הייצוג הווקטורי ויבוא לכך שהייצוג החדש שיתקבל יוביל בסופו של דבר לערכים וקטוריים שיביאו ל-Probability Distribution על ה-Corpus שיבילו לבחירת מילה מתאימה למילה שהייתה **masked**.



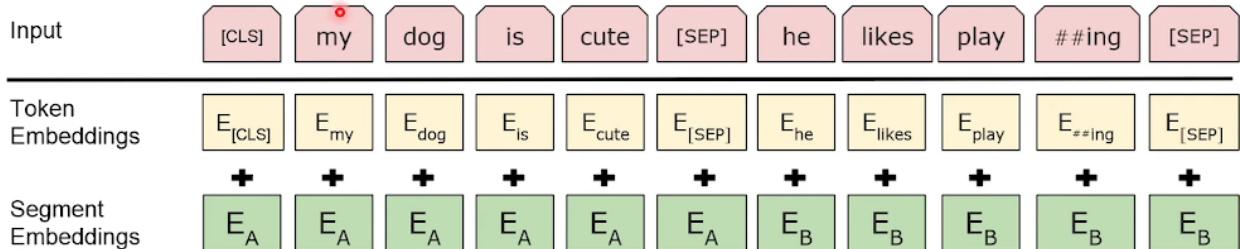
במהלך תהליכי האימון ולאחר מכן ה-Backpropagation יוחשב loss בין המילה שנבחרה לבין המילה שהיא אמורה להיות מלכתחילה באופן בו הגרידינט שנוצר יפעע לאורך כל שכבות הרשת ויישנה את המשקولات בהתאם.

Next Sentence Prediction (NSP)

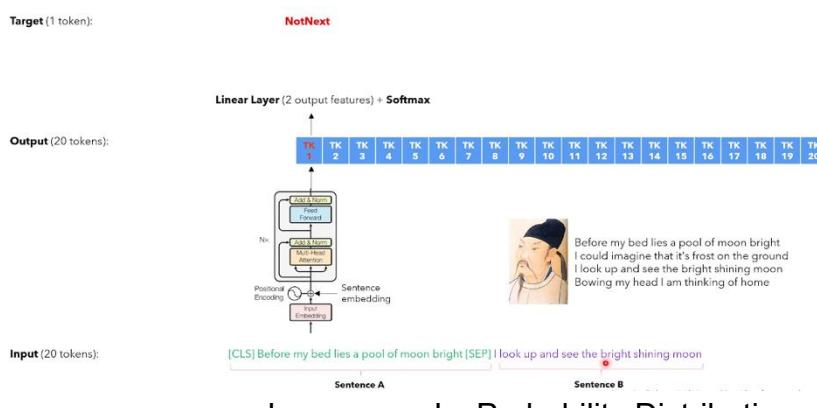


שיטה נוספת לביצוע Pre-Trained היא NSP. בשיטה זו למשה בוחרים בצורה אקראית משפט A. עבור המשפט זהה בהסתברות של 50% בוחרים את המשפט העוקב שלו. ובהתברות של 50% בוחרים משפט שאין קשור אליו כלל. המשפט הנבחר הוא משפט B. לאחר מכן מאמנים את המודל להיות מודל קלטי-פיזי ביןarity שתפקידו להכריע האם בהינתן משפט A, משפט B הוא המשפט העוקב שלו או לא. אמנם המודל מייצר ערך פרדיקציה ביןאי 0/1 אך עצם מדידת ה-loss ועדכו המשקولات בתהליך ה-Backpropagation עוזרת למודל למדוד רפרזנטציות לגבי השפה.

נשאלת השאלה, בהינתן מלא{n} הנכeno{l}-BERT, איך ניתן להפרידו בצורה סמנטית כך שהמודול יידע שמדובר בשני משפטיים שונים. אם היינו מכנים את שני המשפטים ללא כל שינוי בהזה אחר זה, המודול היה חושב שמדובר במשפט אחד ולא היינו יכולים לבצע את המטלה.



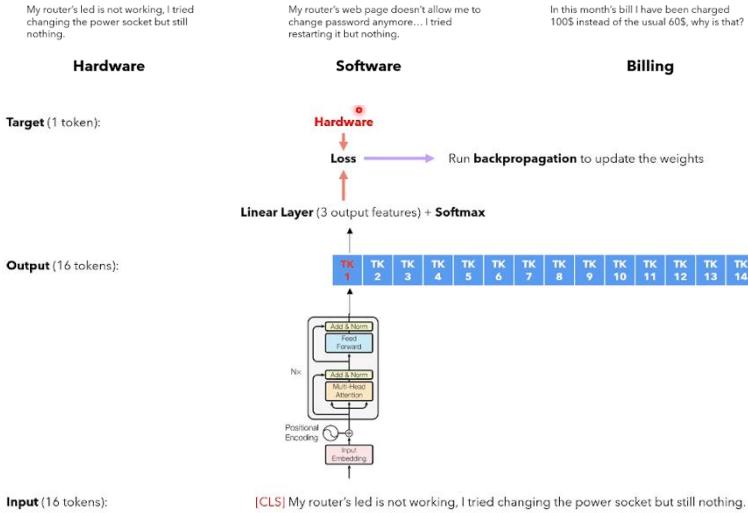
האופן בו אנו עושים זאת היא באמצעות שימוש בטוקנים מיוחדים CLS, SEP וכמו כן ב- Segment Embedding שבודוק כמו element-wise Positional Encoding ערכאים וקטוריים לכל יצוג Embedding Makro עבור כל טוקן classification token CLS הוא שמאכיל יצוג וקטורי כלשהו. Target (1 token): NotNext



כלאנו שעליינו נפעיל SoftMax גם כאן ונחשב loss בין ערך הפרדיקציה לבין התיאוג האמתי וונפער את הגרADING לאורך הרשת לטובות עדכון סטי המשקولات.

Fine-Tune Methods

לאחר שתיארנו את אופן ביצוע Pre Train-hochsitz, נרצה לתאר את אופן ביצוע-hochsitz. כזכור הינה ספציפית יותר למודל. כאשר אנו ניגשים ל-Fine Tune, אנו ניגשים עם מודל שהוא כבר Pre Train ומכיר רפרנציאציות לגבי השפה וכל שנותר הוא להכשיר אותו לשימוש החדש. גם כאן נרצה לתאר 2 סוגים:



Text Classification

בשימוש זו נרצה שבהינתן טקסט כלשהו, המודל יצליח להכריע לאיזה מחלוקת לסוגו את הטקסט.

גם כאן נעשה שימוש בטוקן CLS. לאחר העברת הטקסט במודול, עבריר את הייצוג הווקטורי של CLS דרך שכבה לינארית ועל הפלט נבצע SoftMax לקבלת הסתברות לכל אחת מהמחלקות באופן בו המחלוקת בעלת הערך הגדול ביותר תהיה הפרדיקציה של המודל. לעדכן המשקלות ונחשב loss ונפעפם לאורח שכבות המודול.

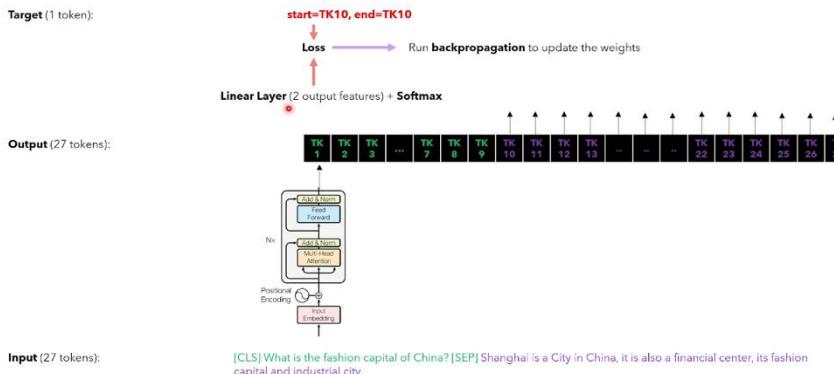
Question Answering

Context: "Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city."

Question: "What is the fashion capital of China?"

Answer: "Shanghai is a City in China, it is also a financial center, its fashion capital and industrial city."

ב-Question Answering אנו רוצים שבהינתן שאלה וקונטקסט המודל יידע לסמן את המיקום בקונטקסט בו מצוי התשובה לשאלת (במקרה שלנו שנגחאי).



האופן בו נבצע את-hochsitz יכול שאבור כל טוקן נשלח את הייצוג הווקטורי שלו לשכבה לינארית עליה נפעיל SoftMax לקבלת 2 ערכים: start, end. ערך start קובע האם הטוקן הוא הטוקן ההתחלתי של התשובה או לחילופין הטוקן הסופי של התשובה. עבור שני הטוקנים שעבור אחד מהם ערך start הוא הגבוה ועבור השני שעבורו ערך end הוא הגבוה, ניקח את כל הטוקנים שבניהם (כולל) זהה יהיה ערך ההצהרה. גם כאן, לאחר ערך-end הוא הגבוה, ניקח את כל הטוקנים שבניהם והטוקנים הסופיים עברו כל משפט, ונחשב באימון loss בין הערכים הללו.

לערך הפרדיקציה ונעדכן את המשקלות במהלך-hochsitz Backpropagation.

T5 – Transfer Text To Text Transformer

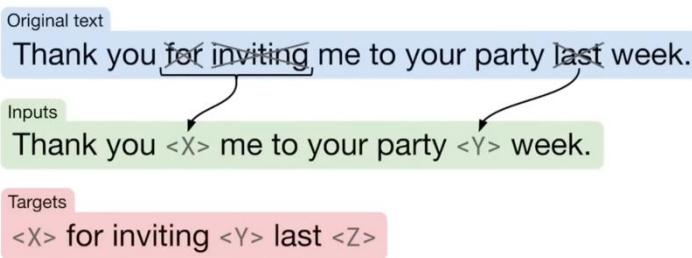
[Video Link](#)

ארQUITקטורה נוספת מושתת על טרנספורמר מבית Google היא T5. למעשה לא מדובר ממש בארכיטקטורה אלא יותר במודול שאומן על ידי חוקרים מ-Google על סמך מתודולוגיות מסוימות וכעת הוא Open Source וניתן לשימוש בו שימוש לטובת ארכיטקטורות נוספות.

היחידיות הרבה של המודל באהה לידי ביטוי בדאטא הייחודי עליו הוא אומן:

Corpus Colossal Clean Crawled Corpus או בקיצור C4. הדטאזה הוא למעשה scraping של חלקים נרחבים מהאינטרנט ומביא לכך קבלת 20TB מידע חדש. ל-C4 מבוצע תהליך ניקוי של ה-HTML כך שנשארים עם טקסט בלבד. כמו כן, חוקרי גוגול ביצעו ניקוי נוסף לטובת הcontentו למודל:

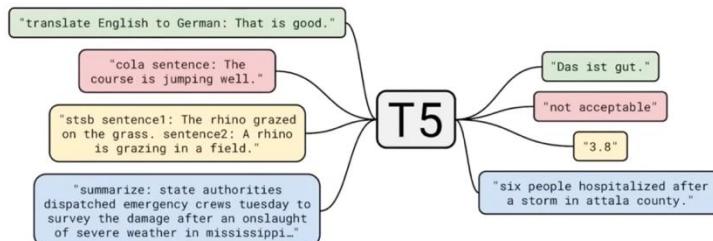
1. הסירו חלקיק טקסט שלא נגמר בסימן פיסוק כמו נקודה, סימן שאלה סימן קריאה וכו'.
2. הסירו חלקיק טקסט שמקורם בעמודים המכילים פחות מ-5 משפטים
3. הסירו תכנים המכילים מילים אסורות מיותרות שהגדירו מראש
4. הסירו קטעי קוד ב-Java Script או בשפות קוד אחרות
5. הסירו קטעי מלשניהם ipsum lorem骚扰 שallow נועד לטובת בדיקות עיצוב הטקסט
6. התמקדו במלל באנגלית ולכך מילים שבסבירות של פחות מ-0.99 זוהו כמילים באנגלית הוסרו



בדומה למה שכבר רأינו, גם כאן השתמשו בשיטת Masking בה הסירו בזורה מכונת חלקים גדולים מהtekסט וגרמו למודל להשלים את החלקים הללו בעצמו תוך כדי שהוא לומד רפרזנטציות על השפה. זו כאמור שיטה שהיא סוג של Self-Supervised עברור רצפי טקסט ארכוכים.

עם זאת, על אף שהפעילו את שיטת-h-Masking לטובת מידת רפרזנטציות על ידי המודל, הם גם ביצעו סוג נוסף של אימון שהוביל את המודל לכך למידת רפרזנטציות נוספת. למעשה הם נתנו למודל מספר סוגים מטלות שנთארן כעת:

1. תרגום – תרגום משפה כלשהי לשפה אחרת באופן בו התיאוג הוא התרגום עצמו
2. תקיןות שפה – בדיקה האם קלט טקסט כלשהו נכון מבחינת תחבירית והגינותית באופן בו התיאוג הוא GLUE בינארי – ה-Benchmark ששומש כאן הוא
3. דמיון בין משפטים – מתן שני קלטים טקסטואליים ומודל צריך לברור ולהחליט מה מידת ההתאמאה בין שני המודלים – גם כאן ה-Benchmark ששומש כאן הוא GLUE
4. סיכום – סיכום טקסט באופן בו התיאוג הוא הטקסט המסכם



סך הכל קיבל כי כפי שציין, T5 אינה ארכיטקטורה העומדת בפני עצמה כסוג חדש של ארכיטקטורה אלא מודל טרנספורמר שאומן בזורה ייחודית ולמד רפרזנטציות טובות של טקסט בו ניתן לבצע שימוש במודלים וארכיטקטורות שונות לטובת מיטוב התוצאות

TFT (Temporal Fusion Transformer)

TFT היא ארכיטקטורה מבוססת טרנספורמר שפותחה על ידי Bryan Lim ומספר חוקרים מוגול בשנת 2020. עד כה כאשר תיארנו את ארכיטקטורת הטרנספורמר, דיברנו עליה בערך בהקשר של טקסט באופן בו המידע הטקסטואלי חסר אלמנט של זמן. TFT היא ארכיטקטורה המתאימה לעולם בעיות שלאו דזוקא קשור לעולמות שפה אלא דזוקא לעולמות בהם הדadata הוא Time Series ואנו רוצים לספק פרדיקציות על הדadata בהתאם למספר Horizons קלומר מספר טווחי זמן (למשל מחיר מניה בשבועות הקרוב, בחודש הקרוב ובשנה הקרובה).

טיור הבעיה

על מנת להבין טוב יותר את הבעיה איתה אנו מתמודדים נרצה לנתח בצורה פורמלית ולתת דוגמא. יש לציין כי כאן הדadata מורכב יותר והוא מורכב ממספר אלמנטים שאוותם נרצה לפרש כתעת.

נניח ואני מנהלים רשות של חניות עליים בשם "אופנת דייקסטרה", הרשות מצלילה באופן ייחסי ולה מסpter סכינים בכל רחבי הארץ. בתור מנהלי הרשות אנו רוצים לחזות מה תהיה התחזית המכירות ומאחר אנו מאמנים בעולמות הטכנולוגיה שכרכנו צוות מפתחים שיפתחו לנו אלגוריתם תחזית שכזה המשמש ב-TFT.

להלן הפרמטרים של הרשות שנרצה לתת לצוות המפתחים:

I - קבוצה של כל-h-entities (ישויות/דגימות) בדאטאסט שלנו. כל ישות יכולה להיות היא בעלת timestep ייחודי שנסמן $t \in [0, T_i]$. סך הכל נוכל להכליל ולומר כי ההגדירה של I היא:

$$\{ (i_1, t_1), \dots, (i_n, t_n) \} = I$$

$$t_i \in [0, T_i]$$

עתנו נותר לדבר על מה זה i ? כלומר מה ההגדירה לשוט. ישוט במקרה שלנו היא פריט מידע שי יכול להיות אחד מ-2 קטגוריות:
1. $s_i \in \mathbb{R}^{m_s}$ – זה תכמה המתארת את הדadata שלא משתנה לאורך הזמן. למשל מיקום החנות, גודל חנות וכו'. למעשה אנו מעבירים כפרמטר List של כל המשתנים הסטטיים $I \subseteq S \in \mathbb{R}^{m_s}$.

2. $\chi_{i,t} \in \mathbb{R}^{m_s}$ – כל הנתונים הדינמיים המשתנים לאורך הזמן.
אם $\chi_{i,t}$ מתחולק ל-2:
1. $x_{i,t}$ – Known Inputs – כל הנתונים ההיסטוריים עברו אותה נקודת זמן למשל מספר האנשים שעברו ברחוב, האם היה יום חמ, האם יש פסטיבל בעיר וכו'.
2. $z_{i,t}$ – Unknown Input – כל הנתונים העתידיים עברו אותה נקודת זמן למשל משביר אספקה חמורה באותו יום, מלחמה שפרצה באופן פתאומי וכו'.
למעשה הנתונים הללו הם נתונים שלא היו וודאים לגבי העתיד באותה נקודת זמן אך עם זאת יכולים להועיל לנו במתן הפרדיקציה מהסיבה שהם משפיעים על אופן קבלת החלטות.

$y_{i,t}$ – ה-target אותו אנו מנסים לחזות. הוא יכול להיות רציף או לחילופין, מחלוקת או כל סוג כלשהו של לייבל אותו אנו רוצים לחזות. במקרה שלנו ב-"אופנת דייקסטרה" אנו רוצים לחזות לייבל רציף המתאר את תחזית המכירות עברו כמה סוג Horizons.

ערך הפרדיקציה ב-TFT

עד כה תיארנו את הפרמטרים השונים לשוגרים שנקננים אל מודל ה-TFT וכעת נרצה להגדיר באופן פשוטי את האופן בו פרדיקציה מסופקת במהלך ה-Inference.

חשוב לציין, כי המימוש הרווח ב-TFT הוא באופן בו הפרדיקציה איננו מספר יחיד אלא על סמך חלוקה למספר ערכי פרדיקציה לפי quantiles באופן בו כל אחד מהם מייצג רמת בטחון של המודל לגבי ערך הפרדיקציה.

נרצה לתאר את ערך הפרדיקציה באופן הבא:

$$\hat{y}_i(q, t, \tau) = f_q(\tau, y_{i,t-k:t}, z_{i,t-k:t}, x_{i,t-k:(t+\tau)}, s_i)$$

נרצה לפרט קצת את המשתנים שבמשוואה:

(τ, q, t, \hat{y}_i) – ערך הפרדיקציה שתלויה במספר משתנים. q מייצג את ה-quantiles הרלוונטי עבורו מסופקת הפרדיקציה, t מייצג את הזמן העכשווי עבורו רוצים לספק את הפרדיקציה, τ מייצג Horizon עבורו אנו רוצים לספק פרדיקציה.

f_q – מתאר אה-TFT כקופסה שחורה אליה אנו מעבירים פרמטרים והוא מבצעת את החישוב ומחזירה את ערך הפרדיקציה.

τ – כפי שצווין, τ הוא ערך horizon עבורו מספקים פרדיקציה למשל עבור הדआה של היום ספק פרדיקציות 7 ימים אל העתיד.

$y_{i,t-k:t}$ – סט התוצאות שיש ברשותנו מזמן של $k-t$ עד זמן t (למשל עבור $k=14$ ימים אחרוניים), במרקחה של "אופנת דיקסטרה" הנתון מתייחס למשל להכנסות היום עבור כל יום ב-14 הימים האחרונים.

$z_{i,t-k:t}$ – נתונים שלא היו זמינים לגבי העתיד באותה נקודת זמן אך התחווו עבור אותו רגע בזמן. הם יכולים להוביל לנו במתן הפרדיקציה מהסיבה שהם משפיעים על אופן קבלת החלטות.

$x_{i,t-k:(t+\tau)}$ – כל הנתונים ההיסטוריים עבור אותה נקודת זמן למשל מספר האנשים שעברו בדרך, האם היה יום חמ, האם יש פסטיבל בעיר וכו'.

s_i – זה תכונה המתארת את הדआה שלא משתנה לאורך הזמן למשל מקום החנות, גודל חנות, סוג חנות וכו'.

דוגמא נומרית ל-TFT

נרצה לتاאר דוגמא נומרית עברו מצב בו אנו רוצים לחזות עבור תאריך של היום (05/06/2023) ולספק פרדייקציות עם Horizon של 3 ימים (08/06/2023) ובהסתמכוות על>Data של השבוע האחרון (29/05/2023).

נתונים

תאריך של היום (t) – 05/06/2023

חלון זמן היסטורי ($t - k$) – 29/05/2023 – 05/06/2023 – ($t - k:t$)

תאריך של היום ($t + \tau$) – 08/06/2023 – ($t + \tau:t+k$)

לייבל היסטורי ($y_{i,t-k:t}$) –

סך סכום המכירות שהיא בכל יום – [220, 210, 250, 260, 220, 230, 210]

– ($z_{i,t-k:t}$) – **Unknown Output at this moment**

בעיה באספקה באותו היום – [0, 1, 0, 0, 0, 1, 0]

סבירות לחסימה משטרתית בהמשך הרחוב – [0.2, 0.3, 0.8, 0.4, 0.3, 0.9, 0.1]

– ($x_{i,t-k:t+\tau}$) – **Known Historical Inputs**

מספר האנשים באותו יום ברחוב – [400, 410, 430, 390, 380, 400, 410, 420, 420, 370]

טמפרטורה ממוצעת באותו יום – [27, 26, 27, 26, 27, 29, 34, 35, 32, 28]

– (s_i) – **Static Covariates**

מאפיינים יבשים –

גודל חנות – 340 מ"ר

אזור חנות – עירוני

מספר קומות – 3

פרדייקציות

– **פרדייקציות לפי quantiles**

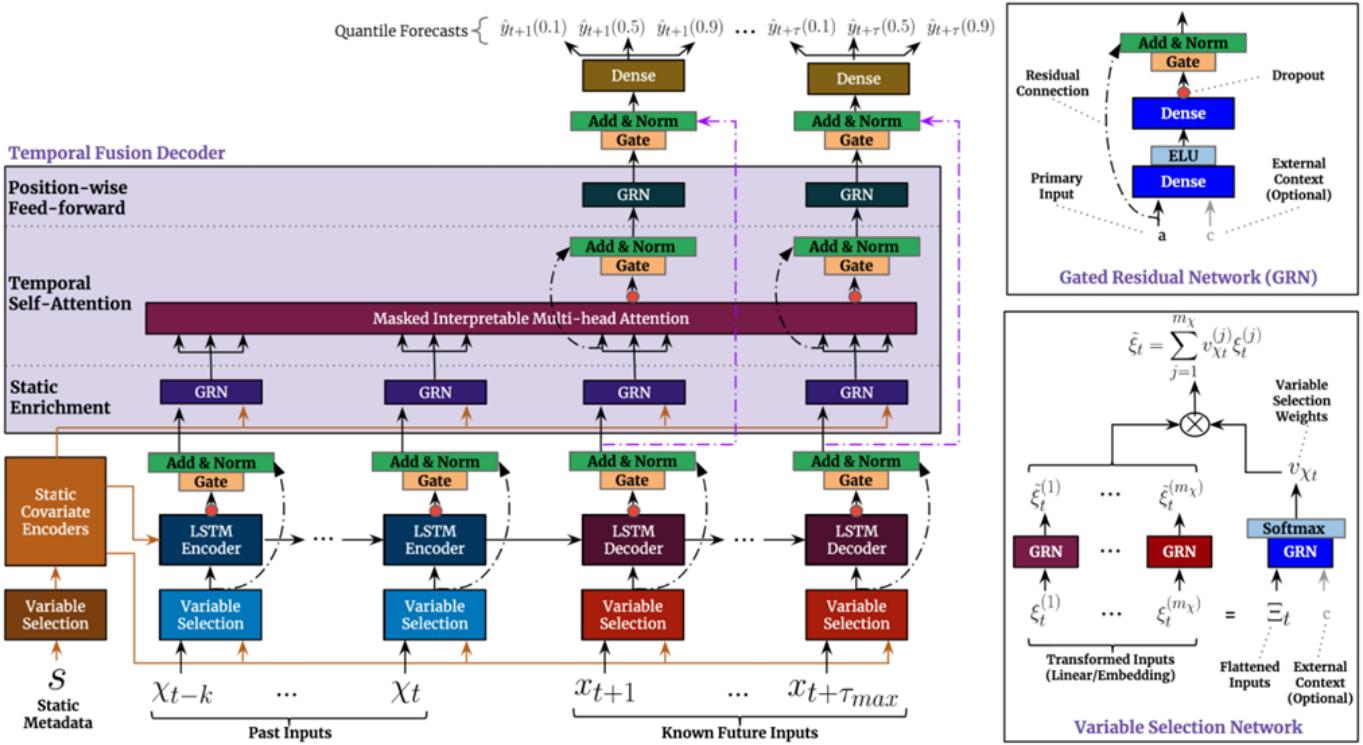
220 – 10th quantiles

240 – 50th quantiles

250 – 90th quantiles

Model Architecture

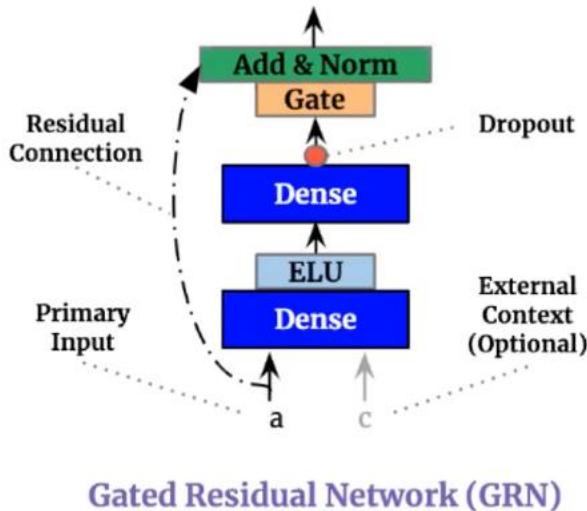
בתמונה מתוארת ארכיטקטורת המודל ונitinן לראות כי היא מורכבת מלה מושפעות.



בعمודים הבאים נרצה לסקור ולתאר את הקומפוננטות השונות של מודל-h-TFT, לכל קומפוננטה יש את התפקיד החשוב והיחודי שלו ואת התפקידו שלה לכולות המודל. מכילול כל הקומפוננטות מובילות לכך למידת רפרנציאיות אינדיבידואלית והן יתאזרו בצורה מעמיקה יותר בעמודים הבאים באופן בו כל קומפוננטה תפורט בנפרד ולבסוף תתבצע הרכבה של כל הקומפוננטות כדי השלמת התמונה הכללית ותיאור הרפרנציאיות הנלמדות על ידי המודל.

GRN (Gated Residual Network)

GRN הוא רכיב ב-TFT שמשמש כמנגנון טרנספורמציה המשמש ב-some Gate mechanism כדי ללמידה ולשלוט בזרימת המידע, מה שמאפשר למודל לנחל ביעילות **Residuals** ובכך ללמידה תלות מורכבת בצורה אדפטיבית. GRN מסייעים בשימירה של מידע שימושי באמצעות **Residuals** והתאמת תרומות הקלט לפולט הסופי.



מבט ראשון בקומפוננטה ניתן לראות כי היא מקבלת קלט a ו- c הינה. a הוא הקלט עצמו למשל בהינתן ואנו מדברים על פרדיקציה עבור מג אויר יכול להיות קריואט טמפרטורה, כמוות לחות ומהירות רוח. ו- c הוא מידע נוסף שמתווסף על הקלט כמו מידע גיאוגרפיה, גובה תחנת מג האויר, מידע עונתי. כמו כן ניתן לראות בתמונה כי הקלט והkontext עוברים סט של טרנספורמציות ומוכפלים בסדרה של משקלות נלמדות, פונקציות אקטיבציה ושכבות נורמליזציה לקבלת הפולט של ה-GRN שימושי אל עבר השכבה הבאה.

מבחינה מתמטית ה-GRN מתואר באופן הבא:

$$GRN_{\omega}(a, c) = LayerNorm(a + GLU_{\omega}(\eta_1))$$

$$\eta_1 = W_{1,\omega}\eta_2 + b_{1,\omega}$$

$$\eta_2 = ELU(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega})$$

$$GLU_{\omega}(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega}) \odot (W_{5,\omega}\gamma + b_{5,\omega})$$

מבחינה מילולית למעשה סט המשוואות מתאר את האירור באופן בו η_2 הוא הפולט של ה-Dense Layer הראשון למשה הקלטים a, c מוכפלים בסט משקלות + biases ולאחר מכן מעבר דרך פונקציית אקטיבציה מסווג ELU לקבל η_2 .

לאחר מכן הפולט של ה-Dense Layer הראשון η_2 עובר מכפלה מטריצונית עם סט משקלות נוספים η_1 . η_1 בטורו נשלח ל- GLU שmagdar להיות מכפלה של η_1 עם שתי סטי משקלות שונים, מכפלה(element-wise) בין שני הפלטים של כל אחד מסטי המשקלות ולאחר מכן הפעלה של פונקציית אקטיבציה מסווג סיגמאoid על הפולט. הדרך האחרוןינו הפעלה של שכבת Add & Norm על הפולט של GLU והוספה של הקלט המקורי כחלק מה-Residual Connection.

דוגמה הרצה נומרית ל-GRN

תחילה נרצה לתאר את הפרמטרים הבאים עבור דוגמת הרצה:

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad \text{ווקטור הקלט } a -$$

$$\begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \quad \text{ווקטור הקונטקסט } c -$$

$$\begin{bmatrix} 0.6 & 0.4 \\ -0.3 & 0.8 \end{bmatrix} \quad \text{מטריצת משקלות } W_1 -$$

$$\begin{bmatrix} 0.5 & -0.5 \\ 0.5 & 0.5 \end{bmatrix} \quad \text{מטריצת משקלות } W_2 -$$

$$\begin{bmatrix} 0.2 & 0.1 \\ -0.1 & 0.2 \end{bmatrix} \quad \text{מטריצת משקלות } W_3 -$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{ביאו } b_1 -$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{ביאו } b_2 -$$

$$\begin{bmatrix} 1.0 & -0.5 \\ 0.7 & 0.4 \end{bmatrix} \quad \text{מטריצת משקלות } W_4 -$$

$$\begin{bmatrix} 0.5 & 0.2 \\ 0.1 & 0.6 \end{bmatrix} \quad \text{מטריצת משקלות } W_5 -$$

$$\begin{bmatrix} 0.1 \\ -0.1 \end{bmatrix} \quad \text{ביאו } b_3 -$$

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \text{ביאו } b_4 -$$

לאחר שהגדכנו את כל הפרמטרים הדרושים לטובת הריצה נרצה כעת לחשב את השלב הראשון:

$$\eta_2 = \text{ELU}(W_{2,\omega}a + W_{3,\omega}c + b_{2,\omega})$$

0.5	-0.5
0.5	0.5

+

1.15
0.05

+

0.2	0.1
-0.1	0.2

+

1.15
0.05

+

0
0

=

1.15
0.05

כעת נרצה להפעיל ELU על הווקטור שהתקבל:

ELU

1.15
0.05

סך הכל מנוסיבנה שני/almost האלמנטים בווקטור הם חיוביים והוא-ELU לינארי עבור חיוביים, נקבל:

1.15
0.05

כעת נרצה לחשב את השלב הבא:

$$\eta_1 = W_{1,\omega}\eta_2 + b_{1,\omega}$$

0.6	0.4
-0.3	0.8

+

1.15
0.05

+

0
0

=

0.71
-0.305

לאחר מכן נרצה לחשב:

$$GLU_\omega(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega}) \odot (W_{5,\omega}\gamma + b_{5,\omega})$$

σ

1.0	-0.5
0.7	0.4

+

0.71
-0.305

+

0.1
-0.1

•

0.5	0.2
0.1	0.6

+

0.71
-0.305

+

0
0

=

σ

0.9625
0.275

•

0.294
-0.112

=

0.2126
-0.0636

לאחר שהשכנו את החישובים המטריצוניים,icut נרצה לחשב Residual Connection, Dropout, Add & Norm.

נתחיל בביטוי $\text{Dropout} = 0.1$ והגדנו Dropout=0.1 ואך אחד מהאלמנטים לא נבחר בהסתברות זו. וلن נותרו עם אותו וקטור שקיבלנו בשלב הקודם.

icut נרצה לחשב Residual Connection באופן הבא:

$$\text{Residual Output} = \alpha + \gamma$$

1
-1

+

0.2126
-0.0636

=

1.2126
-1.0636

לבסוף, בשלב הנורמליזציה נבצע נורמליזציה באופן בו הפרמטרים הנלמדים β, γ יקבעו באופן שרירותי. בנוסף נגדיר $\epsilon = 10^{-8}$, $\beta = 0.7$, $\gamma = 0.3$ ונקבל:

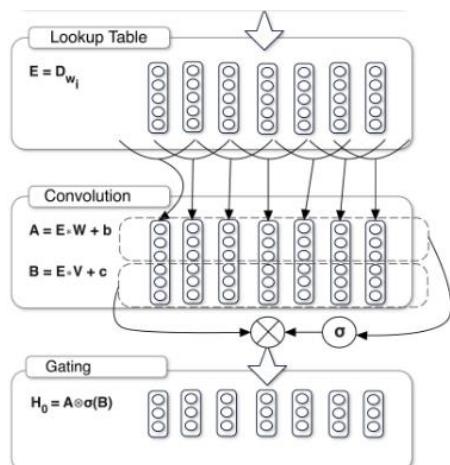
1.2126
-1.0636

→

1
0.4

סך הכל נראה כי וקטור קלט a ובנוסף וקטור קונטקסט c , קיבלנו וקטור זהה למינם ומיצג את התוצאה לאחר ההכפלה בסטי המשקولات והפעלת המニアולציות המתמטיות ב-GRN.

GLU (Gated Linear Unit)



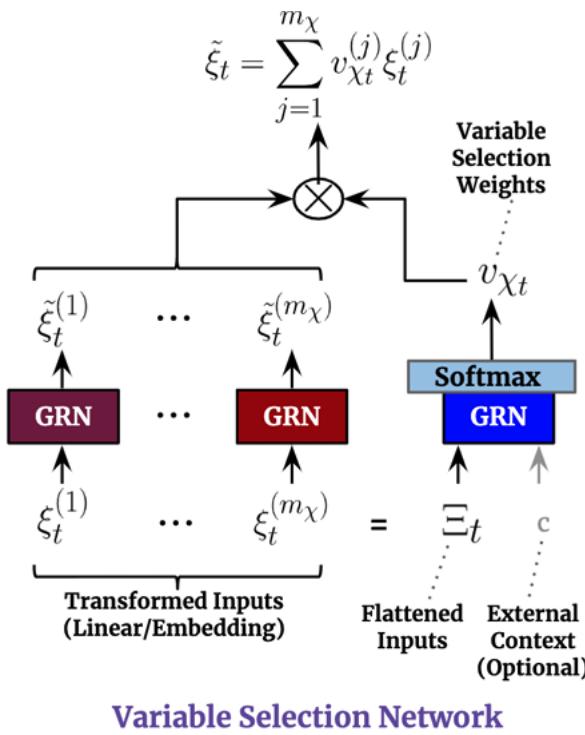
מבחן גראפית נוכל לפחות את ה-GLU לקלט כלשהו E המוכפל בשתי סטי משקولات שונים V, W באופן בו הפלט $w * E$ עובר אקטיבציה באמצעות פונקציית אקטיבציה סיגמאoid. לאחר מכן יש מכפלה מטריצונית בין שני הפלטים לקבלת מטריצה אחת H_0 .

סך הכל נוכל באמת להגדיר את ה-Gating במקורה שלנו:

$$GLU_\omega(\gamma) = \sigma(W_{4,\omega}\gamma + b_{4,\omega})\Theta(W_{5,\omega}\gamma + b_{5,\omega})$$

VSN (Variable Selection Network)

המטרה של הקומפוננטה זו כשמה כן היא הוא ביצוע Feature Selection על סדר המידע שהמודל למד במהלך תהליכי האימון. חשוב לציין שהאופן בו ה-Feature Selection נעשה הוא באופן Instance-Wise – כלומר בדעתו יבצע Feature Selection בצורה פרטונית ומותאם אישית אליה על סמך הרפרנציות שנלמדו לגבי הפיצרים במהלך תהליכי האימון. הרפרנציות הללו במהלך הבדיקה על ידי סט המשקولات בקומפוננטה.



מבוט בשיר ניתן לראות כי קומפוננטת ה-VSN מרכיבת מספר תת-יחידות GRN באופן בו כל יחידת GRN ייעודית עבור פיצר כלשהו ובנוסף יחידת GRN נוספת העוסקת על כל הפיצרים ביחד לטובת ביצוע ה-Feature Selection.

מבחינת סימונים אנו יכולים לראות כי הסימון $\xi_t^{(j)}$ מייצג את הפיצר ה- j -בזמן t . לפיקר ניתן להבחן כי עבור זמן כלשהו t , כל פיצר $\{1 \dots m_\chi\}$ באותה נקודת זמן נכנס ל-GRN (האדומים) נפרד וועבר עיבוד נפרד. ניתן לחשב על התהליך זה בו כל פיצר בנפרד עבר digestion GRN-ב-GRN להפקת רפרנציות המתיחסות עם תפיסת העולם של המודל. כמו כן ב-GRN הכהול ניתן לראות כי הקלט שלו הוא Ξ_t וקונטיקט נוסף c . Ξ_t מייצג עבורנו concatenation של כל הפיצרים בדעתו לכדי GRN-הכהול הוא לייצר וקטור Feature Selection המיציר מכון להכפיל זאת באמצעות Softmax ולאחר מכן ברפרנציות הפיצרים שייצרו על ידי ה-GRNs-האדומים המוצגות כ- $\{\tilde{\xi}_t^{(1)} \dots \tilde{\xi}_t^{(m_\chi)}\}$.

1. בפועל, יש מכפלה element-wise בין הרפרנציות של הפיצרים בעני המודל לבין וקטור בחירת הפיצרים לקבלת הווקטור הסופי של הפיצרים שיועברו לשכבות הבאים בארכיטקטורה.
2. נרצה למנות כאן מספר תכונות של ה-Instance-Wise Selection Layer – כפי שתואר, בחירות הפיצרים היא עברו כל דגימה כשלעצמה. לכן נוכל לקבל למשל דגימה שערci פיצר כלשהו שלא יבחרו ולעומת זאת עברו דגימה אחרת ערך אותו הפיצר יוביל לכך שהוא יתאפשר ולא יוכל לשכבות הבאים.
3. Enhanced Input Representation – עצם העבודה שיש מכפלה element-wise בין פלט ה-GRN הכהול ל-GRNs האדומים מובילה למצב בו ה-GRNs האדומים למעשה ממצים את המהות של הפיצרים והמכפלה בפלט ה-GRN הכהול למעשה מושקלת את ה-Essences. הללו בהתאם לחסיבות הפיצרים עבור המופיע הספציפי.
4. Efficiency and Noise Reduction – עצם העבודה שאנו מסנים חלק נכבד מהפיצרים אנו מעשה מתעלמים חלק נכבד בקהל שיכל לטעטו במודול ולגרום לו ללמידה רפרנציות שלאו דווקא בהכרח משקפות את המציגות בשל ריבוי פיצרים שלאו דווקא בהכרח מספרים את סיפור ההתפלגות.

דוגמת הרצה נומרית ל-VSN

לטובת דוגמת הרצה נניה ויש לנו 2 פיצרים: טמפרטורה (רציף), סוג יומ (קטגוריאלי). לצורך פשוטות אין וקטור קונטקטו C, אך עקרונית הוא גם יכול להיות מובא ולהיכנס לכל VSN.

0.5
-0.5
1
0

טמפרטורה (recurrent) – $\xi_t^{(1)}$ (embedded-encoded)

סוג יומ (one hot vector encoding) – $\xi_t^{(2)}$

תחילה נרצה לחשב את ערכי **GRNs האדומים** (אלו שיודעים לזקק את מהות הפיצרים). מאחר וכבר ביצענו דוגמת הרצה ל-GRNs, לא נרשום את כל התהילר אלא רק את פלטי הקומפוננטה:

$$\begin{aligned}\tilde{\xi}_t^{(1)} &= GRN_1(\xi_t^{(1)}) = GRN_1([0.5, -0.5]) = [0.142, 0.023] \\ \tilde{\xi}_t^{(2)} &= GRN_2(\xi_t^{(2)}) = GRN_2([1, 0]) = [0.339, 1.023]\end{aligned}$$

סך הכל נראה כי יצאנו עם 2 וקטורים שמייצגים את מהות הפיצרים:

0.142
0.023
0.339
1.023

cutת נרצה לחשב את ה-GRN הכחול. נשים לב כי יש תהילר מקדים של חישוב Ξ המוגדר באופן הבא:

0.5
-0.5
1
0

$$\Xi_t = [\xi_t^{(1)} \ \dots \ \xi_t^{(m_\chi)}]$$

כלומר בתור flatten vector שמכיל בתוכו את כל ערכי הפיצרים בצורה משורשת ובמקרה שלנו:

מבצע חישוב ונקבל:

$$v_{\chi_t} = \text{Softmax}\left(GRN_{v_\chi}(\Xi_t, c)\right) = \text{Softmax}\left(GRN_{v_\chi}([0.5, -0.5, 1, 0])\right) = [0.425, 0.575]$$

נראה כי הפלט שקיבלנו עבור χ_t מייצג את המשקל על הפיצרים לפי חישובותם. כל שנותרCut הוא

להכפיל את $\tilde{\xi}_t^{(1)}$, $\tilde{\xi}_t^{(2)}$ – ב- v_{χ_t} בצורה element-wise לקבלת היצוגים הממשוקלים:

$$\tilde{\xi}_t = \sum_{j=1}^{m_\chi} v_{\chi_t}^{(j)} \tilde{\xi}_t^{(j)} = (0.425 * [0.142, 0.023]) + (0.575 * [0.339, 1.023]) =$$

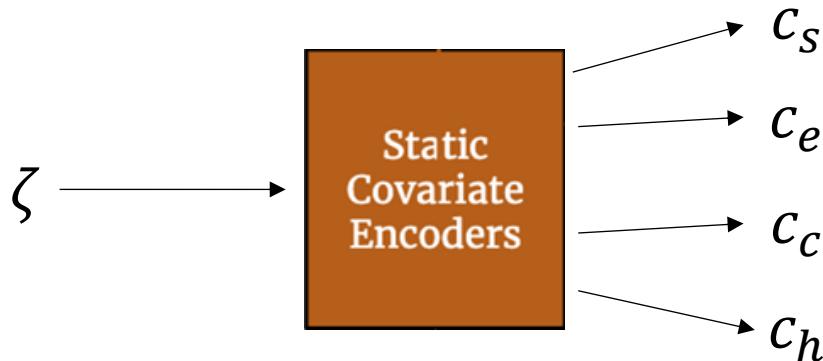
$$(0.425 * 0.142 + 0.575 * 0.339), ((0.425 * 0.023) + (0.575 * 1.023)) = [0.255, 0.005]$$

סך הכל נראה כי הפלט הסופי של ה-VSN המוגדר להיות $\tilde{\xi}_t$ (כל הפיצרים עבור רגע כלשהו בזמן) הוא:

0.225
0.005

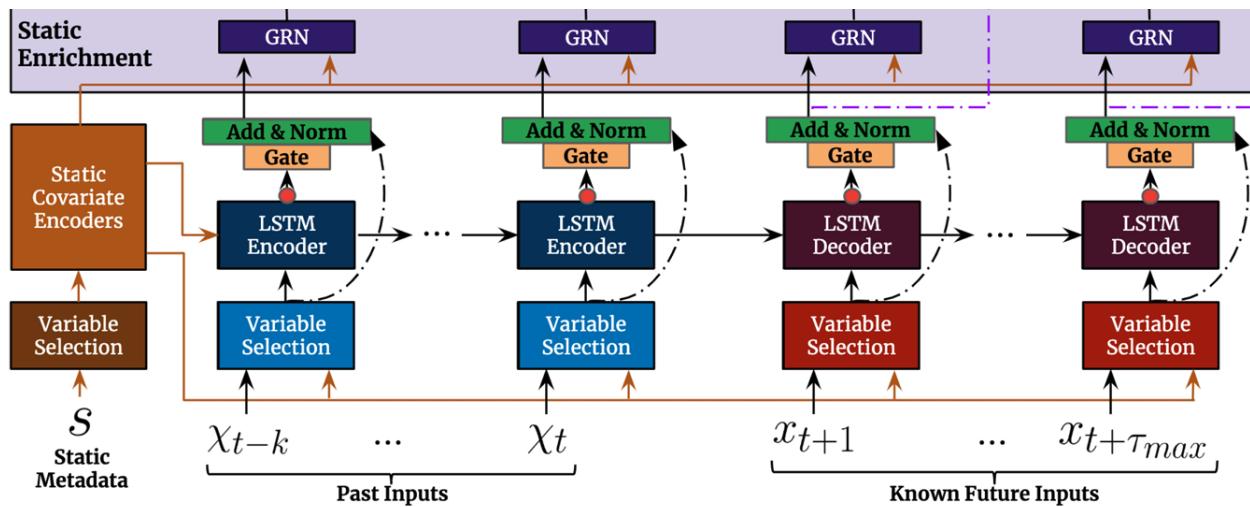
SCE (Static Covariate Encoders)

בניגוד למודל סדרות זמן אחרים, TFT עוצב על מנת להפיק מידע בצורה איקוית מנתונים סטטיים. הוא עושה זאת באמצעות כלילה של שני רכיבים עיקריים. על הראשון כבר דיברנו והוא למעשה VSNet העובר המשתנים הסטטיים. לאחר מכן הפלט של VSNet שנסמן ζ מועבר אל עבר השכבה הבאה שהיא השכבה מכילה מספר GRNs שוכלים מקבלים כקלט את ζ , כל GRN מפיק פלט שונה אשר נשלח למקום אחר בארכיטקטורה.



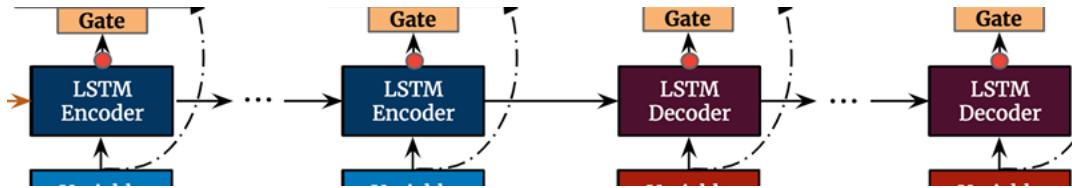
סך הכל נראה כי אנו מקבלים 4 פלטים:

- c_s – מתקבל כתוצאה מחישוב של $GRN_{c_s}(\zeta)$ – מועבר אל כל אחד מה-GRNs האדומים והכחולים
- c_e – מתקבל כתוצאה מחישוב של $GRN_{c_e}(\zeta)$ – מועבר אל כל אחד מה-GRNs ב-Static Enrichment
- c_c – מתקבל כתוצאה מחישוב של $GRN_{c_c}(\zeta)$ – מועבר כקלט ל-LSTM Encoder הראשון
- c_h – מתקבל כתוצאה מחישוב של $GRN_{c_h}(\zeta)$ – מועבר כקלט ל-LSTM Decoder הראשון

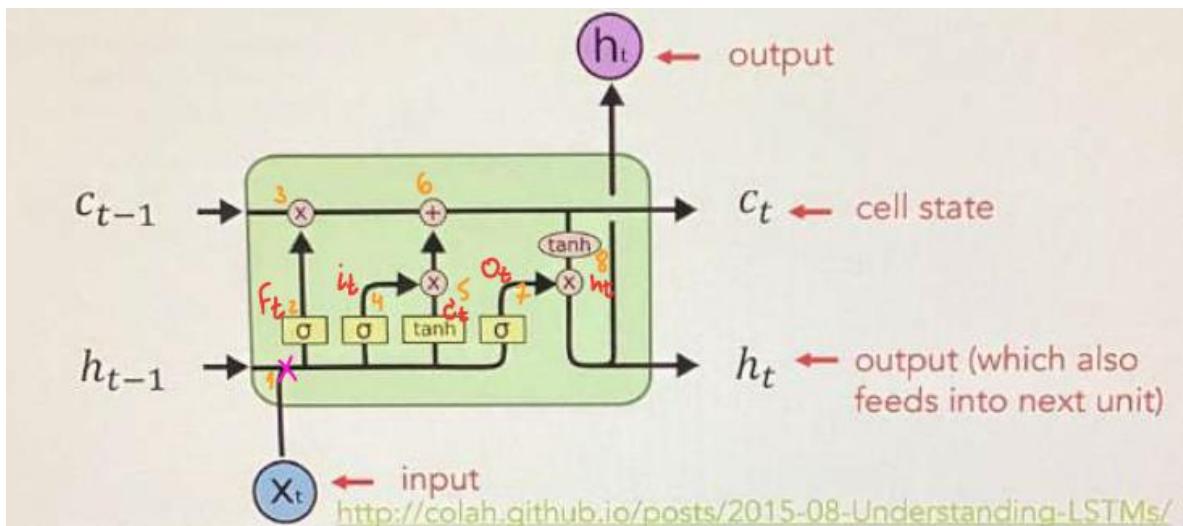


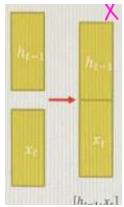
LSTM Encoder/Decoder

ניתן לראות שלאחר שלב ה-VSN יש שכבה של תמי יחידות LSTM המחברות האחת לשניה. עבור כל תט יחידה ניתן לראות כי היא מקבלת את הקלט שלה מה-VSN במקום t_i ומעבירה את הפלט שלה אל עבר תט היחידה במקום t_{i+1} ובנוסף אל עבר שכבת Gating הממוקמת במקום t_i שבתורה מעבירה את הפלט אל עבר המשך הארכיטקטורה.



נרצה כעת לתאר את הפעולות המתבצעות בתוך תט היחידה. תט היחידה עצמה נרצה לתאר בפרט היא את היחידה הראשונה של ה-LSTM – ממנה נוכל להשליך על תטי היחידות האחרות כתלות בפלטי תטי היחידות הקודמות להן בצורה אינדוקטיבית. כמו כן ניתן לראות קלט 3 קלטיים: c_{t-1} משול ל- c_t , h_{t-1} משול ל- c_t המתקבלים כפלט ב-SCE. כמו כן ניתן לראות קלט נוסף המוגדר כ- x_t המיצג את פלט ה-VSN במקום t . כמו כן ניתן לראות 3 פלטיים: h_t שמחווה את הקלט לשכבות ה-Gating, h_t , c_t שמחווים את הקלט לתט היחידה הבאה של ה-LSTM.





הסבר מילולי על המתרחש בתת יחידת ה-LSTM:

1. ניתן לראות כי **X** מייצג את נקודת כניסה הקלטים x_t, c_h אל ה-LSTM. ברגע שמתבצעת הכניסה, מבוצע concat ביןיהם לקבלת ווקטור גדול יותר שמכיל את ערכיו שניהם.
2. לאחר ביצוע הconcat וקבלת הווקטור $[x_t, c_h]$ יש זרימה של המידע לסת המשקولات הראשון בתנת היחידה שנקרא **f_t** המוגדר להיות $f_t = \sigma(w_f[x_t, c_h] + b_f)$, מכפלת dot-product בין ווקטור ה concat לבין סט המשקولات והבאים ולאחר מכן אקטיבציה sigmoid כך שנקבל כפלט f_t הוא ווקטור ערכים בין 0 ל-1 (עד כדי כפל בקבוע בהתאם להגדלת הסיגמאoid).
3. בשלב הבא ניתן לראות כי ווקטור הפלט f_t מוכפל ב-state cell של תת היחידה הקודמת הוא במקורה שלנו מאחר ואנו מדברים על ה-Encoder הראשון אז c_s המגיע מה-SCE. המכפלה בין פלט f_t לבין c_s מתבצעת element-wise באופן בו כל איבר מוכפל במשנהו. רכיב זה ב-LSTM מטרתו לקבוע את הזיכרון לטוויה קצר/ארוך של המודל באופן בו ערכי c_{t-1} קטנים יובילו להקטנת הערך ולזיכרון חלש וערכי c_{t-1} גדולים יובילו להגדלת הערך ולזיכרון חזק.
4. ניתן לראות כי ישנה זרימה של $[x_t, c_h]$ גם לכיוון נוסף אל עבר שתי סטי משקولات נלדים המוגדרים i_t, c'_t . ב- i_t יש מכפלה המוגדרת $(i_t * w_i[x_t, c_h] + b_i) \sigma = i_t$ כך שגם כאן יש פלט המכיל ערכים סופיים בין 0 ל-1 (עד כדי כפל בקבוע בהתאם להגדלת הסיגמאoid).
5. בשלב זה מתבצע החישוב $c'_t = \tanh(w_c[x_t, c_h] + b_c) * c_t$ כך שגם יש פלט המכיל ערכים סופיים בין 0 ל-1 (עד כדי כפל בקבוע בהתאם להגדלת הסיגמאoid).
6. בשלב זה מקבל את c_t באופן להסתכול על $c'_t = c_t * i_t + c_s * f_t$ ב- c_t מיצגת מכפלת element-wise. סך הכל מקבל את c_t באופן בו ניתן להסתכול על $c_t = f_t * c_s + i_t * c'_t$ בעוד מה רוצים לזכור מהמצב הקודם ועל הרכיב של c'_t מציין מה רוצים לזכור מהמצב החדש. ב- c_t אין משקלות ולכן לא רכיב נלמד.
7. ב- o_t יש מכפלה המוגדרת $(o_t * w_o[x_t, c_h] + b_o) \sigma = o_t = \tanh(c_t * w_o[x_t, c_h] + b_o)$ כך שגם יש פלט המכיל ערכים סופיים בין 0 ל-1 (עד כדי כפל בקבוע בהתאם להגדלת הסיגמאoid). למעשה תפקיד הרכיב הזה הוא לקבוע איזה סוג של מידע נרצה להוציא החוצה כפלט תת היחידה ובאיזה אופן.
8. ב- h_t יש מכפלה המוגדרת $(c'_t * \tanh(o_t)) = h_t$ כלומר מכפלת element-wise בין o_t לבין c'_t . הארכים כאן הם בין -1 ל-1. אקטיבציה נוספת של c'_t .

סך הכל מקבל כי כפוי שתואר כל קלט של תת יחידת LSTM מתקבל בין היתר על ידי תת היחידה הקודמת לה וכמו כן כל פלט של תת יחידת LSTM מועבר אל עבר שכבת Gating נוספת המכונה Add & Norm. ובצורה פורמלית:

$$\tilde{\phi}(t, n) = \text{LayerNorm}(\tilde{\xi}_{t+n} + \text{GLU}_{\tilde{\phi}}(\phi(t, n))) \quad , \quad n \in [-k, \tau_{max}]$$

ונכל לראות כי $(n, t) \tilde{\phi}$ מתאר את הפלט הסופי של כל תת יחידת. כמו כן $\text{GLU}_{\tilde{\phi}}(\phi(t, n))$, מתאר את פלט שכבת-h-Gating שקיבלה כקלט את פלט LSTM- $\tilde{\xi}_{t+n}$ כמו כן $\text{LayerNorm}(\tilde{\xi}_{t+n})$ מתאר את פלט ה-VSN-Residual. כל הביטוי בתווך עבר Norm & Add לקבלת $(n, t) \tilde{\phi}$ עבר כל תת יחידת.

Temporal Fusion Decoder

Temporal Fusion Decoder הוא אבן הבניין בארכיטקטורה המוציה בבלוק הסגול. למשה לאחר שביצענו digestion למידע ברכיבים הקודמים הוא עוברCutת ל-*Temporal Fusion Decoder* – קומפוננטה בעלת 3 רכיבים שמטרתה למצוא תלויות בדאטא ולהפוך תבניות שיובילו בעtid לייצור פרדייקציות ותחזיות.

3 רכיבים שמרכיבים את ה-*Temporal Fusion Decoder* ועליהם נרצה לפרט בהרחבה הם:

- 1. Static Enrichment
- 2. Temporal Self Attention
- 3. Position Wise Feed Forward

Static Enrichment

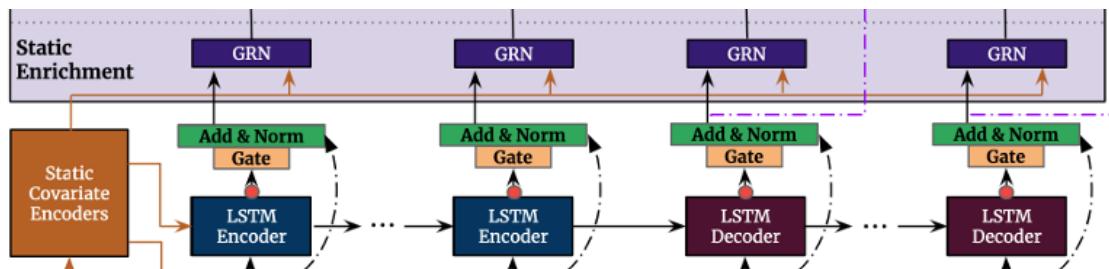
לשכבה ה-*Static Enrichment* יש תפקיד חשוב באינקורפרציה של מידע סטטי אינפורמטיבי שmagיע מה-SCE בין מידע דינامي שmagיע כפלט שכבות ה-Gating לאות גולת ה-LSTMs. השכבה מקבלת כקלט את $\tilde{\phi}(t, n, \tau_{max} - k)$ עבור כל תות יחידת LSTM מהשכבה התחנותונה בנפרד. כמו כן את c_e מה-SCE.

פורמלית נרצה להגדיר את המתבצע בשכבה ה-*Static Enrichment* באופן הבא:

$$\theta(t, n) = GRN_\phi(\tilde{\phi}(t, n), c_e)$$

פרט חשוב שחייב לציין הוא שהמשקלות של GRN_ϕ משותפות לאורך כל השכבה ולמשה המשמעות של כך היא שאין מספר תתי יחידות GRN בשכבה זו אלא Tat יחידה אחת שבעל כפלט $(n, \tilde{\phi})$ של תות יחידת LSTM ובנוסף את c_e ומפעילה בדיקת דיאוק את אותה הטרנספורמציה המתקבלת על ידי אותו סט משקלות נלמד. המשמעות של הדבר הוא שכל הפלטים מקבלים את אותו היחס ומכללים באותו הגורם ולכן יש מספר יתרונות:

- 1. הפחתת Overfitting ופישוט המודל
- 2. השפעת c_e במידה שווה על כל מרכיבים הדינמיים המועבדים
- 3. סיוע ביכולת ההכללה של המודל



Temporal Self Attention

לאחר שסימנו עם שכבת-h Static Enrichment אנו למשה יוצאים עם $\tau_{max} + k$ פלטימ המוגדרים $[\theta(t, -k) \dots \theta(t, \tau_{max})]$ שמהווים עבורנו את המידע הסטטי והדינמי שעברו אינקורופורציה.

הדבר הראשון שאנו עושים בשכבת-h Temporal Self Attention הוא איחוד של כל (i, θ) כדי מטריצה אחת המוגדרת $[\theta(t, -k) \dots \theta(t, \tau_{max})]$. הקונסולידציה הזאת היא קונסולידציה סמנטיבית שמאפשרת איחוד של המידע תחת תבנית משותפת יחידה. הדבר דומה לאופן בו במודל שפה כל טוקן מוגדר כוקטור מסווני אך כאשר מכנים אותו אל עבר שכבת-h Attention במודל השפה מתבצעת קונסולידציה של הווקטורים המסווניים לכדי מטריצה משתנה בעקבות הפעולות הלינאריות בקומפוננטה.

$\theta(t)$ בתורה נכנסת אל עבר שכבת Masked Interpretable Multi Head Attention באופן בו $(\theta(t))$ למעשה מהוות את מטריצת-h Embedding ו מבחינה פורמלית נקבע:

$$\mathbf{B}(t) = \text{InterpretableMultiHead}(\theta(t), \theta(t), \theta(t))$$

נרצה לשים לב ל-2 נקודות משמעותיות:

1. מתבצע Masking ל- $(\theta(t))$ באופן בו מידע עתיק לא יכול להשפיע בצורה חד-ערכית ישירה על מידע עתיק. למשל בהינתן ובעור $[\theta(t, -k) \dots \theta(t, \tau_{max})]$ לדוגמה מצויים לנו הזרים הבאים: $[\theta(t+2), \theta(t-1), \theta(t), \theta(t+1)]$ לאחר ה-Masking המיקוד timestamps של ה-Attention יהיה באופן בו לכל (n, θ) מתקיים כי ישנה השפעה רק של ה- $\theta(t)$ שקדמו לו באותו timestamp. כך למשל עבור $(\theta(t))$ הזרים שיופיעו הם $(1, \theta(t), 2, \theta(t-2))$.
2. אמם תיארנו את המידע הנוכחי כזיה המועוכד בצורה מטריצונית אך בפועל הפלט $(\mathbf{B}(t))$ יבוא לידי ביטוי כפלט ווקטורי עבור כל אחד מהציגים הווקטורים בנפרד (עמודות המטריצה) ו מבחינה פורמלית מוגדר באופן הבא:

$$\mathbf{B}(t) = [\beta(t, -k) \dots \beta(t, \tau_{max})]$$

סך הכל נראה כי אנחנו נצא עם $k + \tau_{max}$ עמודות במטריצה באופן בו כל אחד מהפלטים הללו עובר עיבוד נוספת בשכבת-h-Attention Temporal Self Attention באמצעות Gating Add & Norm ושכבת-h המוגדרת פורמלית באופן הבא:

$$\delta(t, n) = \text{LayerNorm}(\theta(t, n) + \text{GLU}_\delta(\beta(t, n)))$$

- נשים לב כי למעשה קלט שכבת הnormalization מורכב מ-2 קלטים:
1. $(\theta(t, n) - \text{קלט}-\text{Residual})$ שמקורו במידע המגיע מלפני שכבת-h Attention
 2. $(\text{GLU}_\delta(\beta(t, n)))$ – קלט המגיע כפלט שכבת-h Gating לאחר Attention

Position Wise Feed Forward

השלב האחרון ב-*Temporal Fusion Decoder* הוא שכבה נוספת של GRNs המוגדרת באופן הבא:

$$\psi(t, n) = GRN_{\psi}(\delta(t, n))$$

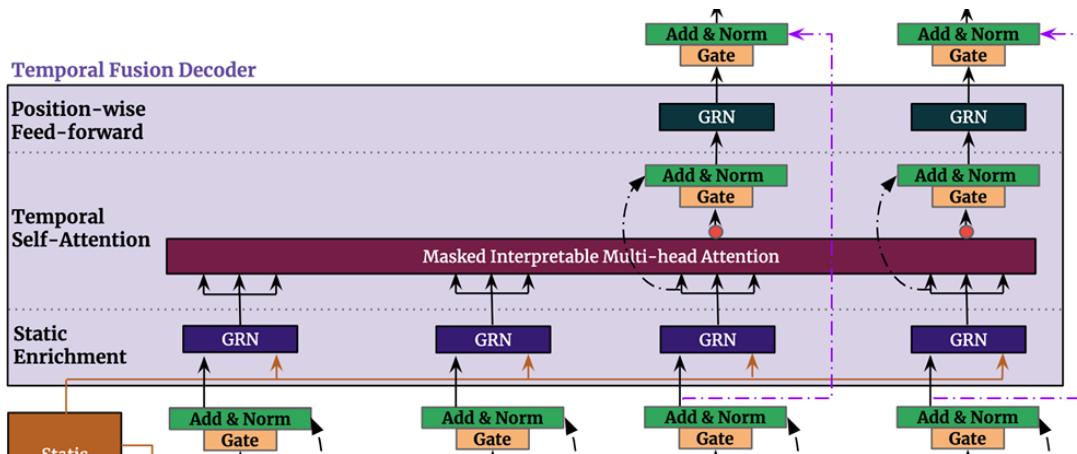
ניתן לראות כי השכבה מקבלת את הקלט (n, t) משכבה Temporal Self Attention ללא כל שכבת residual. כמו כן גם כאן GRNs פועלים כיחידת פונקציונלית הפעלת על כל timestamp בצורה עצמאית.

דבר חשוב שיש לציין הוא שגם יש שיתוף משלוות בין ה-GRNs באופן בו לכל ψ מתקיים כי סט המשקלות זהה, מתעדכן בצורה אחידה לאור תהליכי האימון וזהה עבור כל GRN במהלך תהליכי ה-*Inference*.

לבסוף ישנה שכבה Add & Norm & Position Wise Feed Forward נוספת המוגדרת מוחז ל-*Temporal Fusion Decoder*:

$$\tilde{\psi}(t, n) = LayerNorm \left(\tilde{\phi}(t, n) + GLU_{\tilde{\psi}}(\psi(t, n)) \right)$$

נשים לב כי הקלט של השכבה הוא שילוב של הפלט שmagiu מה-*Temporal Fusion Decoder* בשילוב עם קלט שmagiu ממוקם כשכבה האחרונה של *Temporal Fusion Decoder*, מוקדם יחסית בארQUITקטורה, לפני ה-*Temporal Fusion Decoder* (הקו הסגול).



Quantile and Quantile Loss – הסבר כללי

כפי שצווין, הפלט של המודל הוא לא ייחד אלא מורכב ממספר quantiles שכל אחד בתורו מגדר רמת ביטחון מסוימת לפרדייקציית המודל עבור אותה רמת בטיחון. כך המודל יכול לספק עבור quantile=0.1 פרדייקציה של 220 ועבור quantile=0.9 פרדייקציה של 270.

מהחר וanedנו מדברים על מודל שהמטרה שלו היא לספק וודאות דזוקא בתנאי אי וודאות כמו כן תפקידנו לעזרו לנו לנוהל סיכונים, במקום שהפלט יהיה ייחיד ויהוות את "הניחס הטוב ביותר ביוטר" (קרי פרדייקציה) אנו רוצים שהפלט יהיה טוח תוצאות שייעזרו לנו להעריך את הסיכון ולנהל אותו בצורה הטובה ביותר. לדוגמא, נניח וננהל מפעל שרצח להעיר מה צריכה החשמל במפעל. ככל הנראה שמה שעניינו אותנו זה מה הערך המקסימלי שיוכל להיות עבור כל יום לומור 0.99 quantile שכך הפרזה מצורכת החשמל תוביל לחריגה והעמסה על השנאים שעוללה לגרום לנזק ולהשבית את כל המכונות. لكن, דזוקא במקרה מסופת היא מנדע רחוב ערך הפרדייקציה עבור כל ה-quantiles עלולה להעיד על תנודותיות שgam יכול לעזור לנו בניהול הסיכונים בקבלה החלטות.

המשמעות המילולית של ה-quantiles

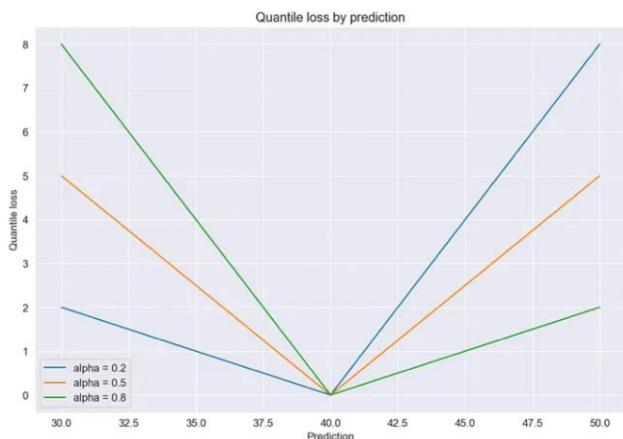
של 10th quantile – למשל עבור פרדייקציה של 220, המשמעות של הדבר הוא שעל פי דעת המודל, קיימת סבירות של 10% שהתחזית רוח האמיטית תהיה 10% ומטה מהערך שמוחזר עבור quantile זה.

של 90th quantile – למשל עבור פרדייקציה של 270, המשמעות של הדבר הוא שעל פי דעת המודל, קיימת סבירות של 10% שהתחזית רוח האמיטית תהיה 10% ומעלה מהערך שמוחזר עבור quantile זה.

הסביר כללי – Quantile loss

Quantile Loss מחושב באופן הבא:

$$\text{Quantile Loss} = \sum_{i=1}^n (q \cdot \max(y_i - \hat{y}_i, 0) + (1 - q) \cdot \max(\hat{y}_i - y_i, 0))$$



אנו יכולים לראות כי באופן חישוב ה-loss הוא פרופורציוני עבור כל quantile כתלות בהאם הוא quantile עלין תחתון או מרכז. מהסתכלות בתמונה נראה כי למשל עבור ערך ground truth של 40, מהתמקדות בגרף ה-quantile נראה כי מאחר ומדובר ב-quantile תחתון איזה הוא מעוניין בצורה מינורית יחסית פרדייקציות הקטנות מערך h-ground truth. ובקרה חזקה יחסית פרדייקציות הגדלות מהערך. באופן דומה עבור הגרף הירוק יוכל לראות תסונת מראה לתופעה ועבור הגרף הירוק הכתום המיציג quantile מרכז, הענישה היא בדומה שווה. מהסתכלות בנוסחה המתמטית של loss נוכל לראות את הדפו שתייארנו ואת העבודה ששה-loss הוא אסימטרי ומעוניין בצורה הערכת יתר והערכת חוסר

TFT – Quantile and Quantile Loss

האופן בו ה-Quantiles באים לידי ביטוי ב-TFT דומים מאוד להסביר הכללי שניתן. על מנת להבהיר את התמונה, נסתכל על מה שיש לנו כתת ביד. לאחר בלוק ה-h-Gating Decoder ולאחר שכבת Add & Norm gating וה-Quantile Loss מושגנו אנו מחזיקים ב- k ערכים וקטוריים שככל אחד מהם מציג מהוות ייצוג שuber digestion עבור כל אחד מה-timestamps. כמו כן, אנו רוצים לספק פרדייקציות רק עבור timestamps עתידיים ולכן ה-timestampים אותם מבחן מתן הפרדייקציות הם רק אלו: $\tau \in \{1, \dots, \tau_{max}\}$.

כל אחד מהייצוגים הוקטוריים הללו עבור $\tau \in \{1, \dots, \tau_{max}\}$ עובר מכפלת dot-product בשכבה Dense לינארית המוגדרת באופן הבא:

$$\hat{y}(q, t, \tau) = W_q \tilde{\psi}(t, n) + b_q$$

באופן בו $W_q \in \mathbb{R}^{1 \times d}$ וכמו כן $\tilde{\psi}(t, n) \in \mathbb{R}^{d \times 1}$ וכן מכפלתם שווה במספר יחיד המהווה את ערך הפרדייקציה עבור ה-quantile הספציפי. לכל quantile סט משקלות bias ייעודי עבורו.

מבחן פורמלית ה-objective של ארכיטקטורה מיוצגת על ידי פונקציית loss ה- Q הbhava:

$$\mathcal{L}(\Omega, W) = \sum_{y_t \in \Omega} \sum_{q \in Q} \sum_{\tau=1}^{\tau_{max}} \frac{QL(y_t, \hat{y}(q, t - \tau, \tau), q)}{M_{\tau_{max}}}$$

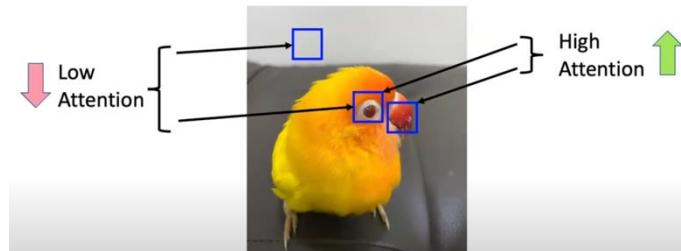
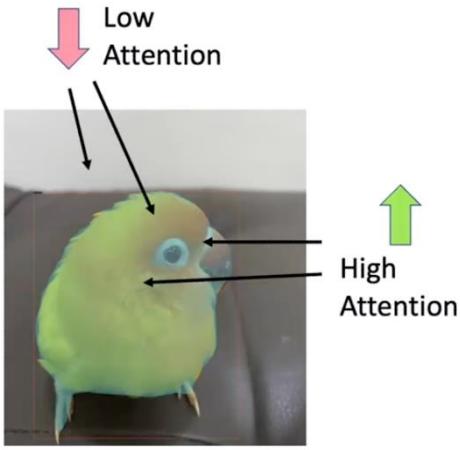
באופן בו QL מחושב כפי שהוצג בהסביר הכללי כך שישנה עניישה פרופורציונלית ביחס לסוג ה- Q -loss והאם הפרדייקציה היא מעל או מתחת ל- Q -quantile.

כמו כן, Q מייצג את דומין סט האימון או ה-Batch הרלוונטי המכיל M דוגמאות באופן בו ה-loss מחושב על ידי מיצוע של השגיאה עבור כל אחת מהדagemות. Q מייצג את אסופה ה- Q -quantiles timestamps במאמר, ה- Q -quantiles בהם התמקדו הם: $0.1, 0.5, 0.9, \dots, \tau_{max}$. מייצג את העורם אנו רוצים לספק פרדייקציות המתיחסים באופן עקרוני רק לפרדייקציות עתיד אותן אנו רוצים לבצע עבור timestamps עתידיים.

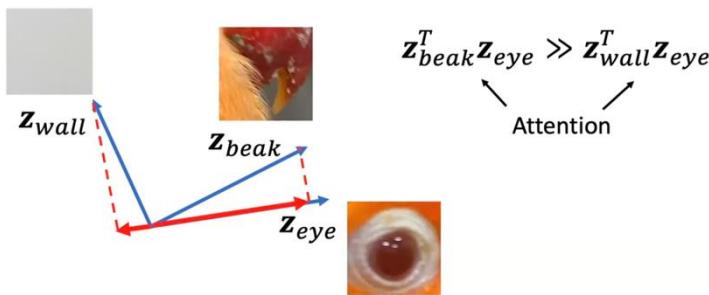
Vision Transformer – ViT

[Video Link](#)

עד כה דיברנו על טרנספורמר ארכיטקטורה שביסודה יודעת למצוא תלויות בין חלקים שונים בDATA בדgesch על Sequential Data. ViT הוא מודל מבוסס טרנספורמר המסתכל על תמונה כסדרת פאצ'ים (patches), כאשר כל פאץ' מתורגם לווקטור. המודל לומד את היחסים בין הפאצ'ים השונים דרך Attention. שכי שדבר מאפיין את ארכיטקטורת הטרנספורמר. כך, בדומה לעיבוד של טקסט, גם בעיבוד תמונות ניתן ללמידה תלויות מרוחניות ולהבין טוב יותר את הקשר בין החלקים השונים של התמונה על ידי הסתכלות עליה כולל Sequential Data.



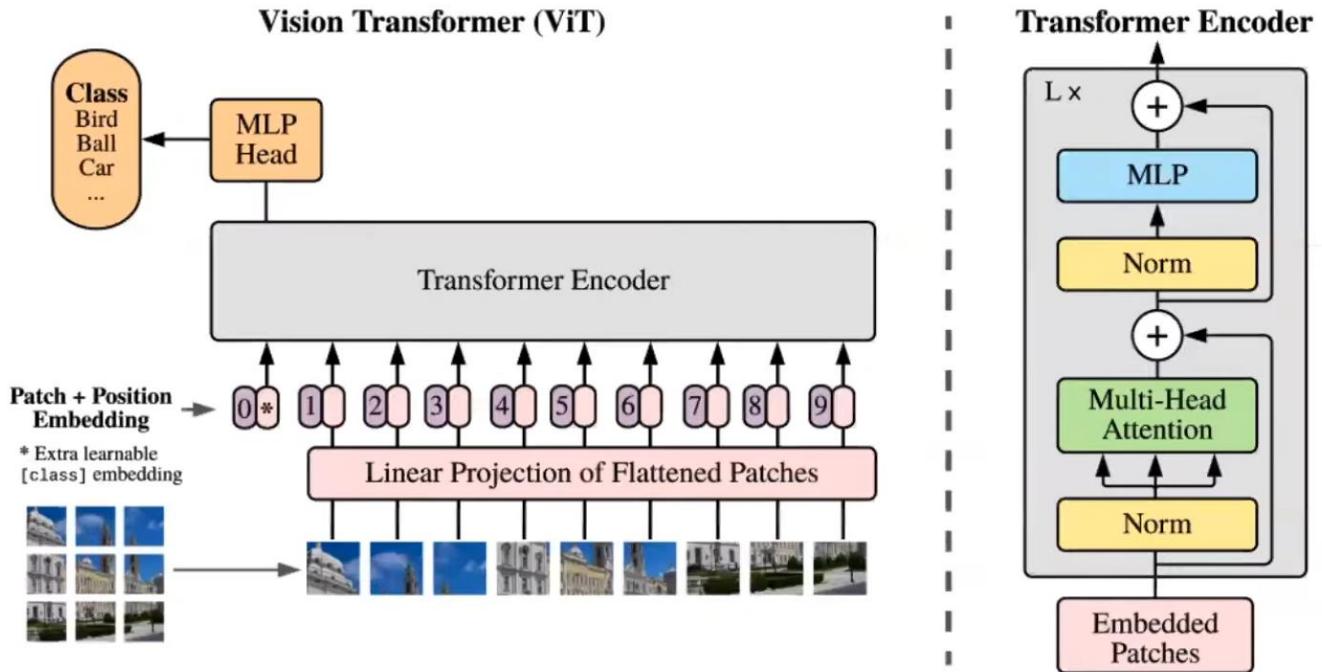
הבסיס הרעיוני לאופן פעולות הארכיטקטורה היא הפרדה בין עיקר לתפל בכל הקשרה להתחמק באופן בו אנו המודל. לשם כך, נרצה תחיללה להתחמק באופן בו אנו תופסים תמונה בתור יוצרים תבונאים. בתמונה משמאלי נוכל לראות תמונה של ציפור, בתור בני אדם נוכל לומר בפה מלא כי מה התבוננות בתמונה אנו יכולים להסיק כי מדובר בציור מהסיבה שהיא מכילה מאפיינים ציפוריים והסיבה לכך היא שאנו מסוגלים להבדיל את הציור מהרקע שלו ובתור האזוריים שבהם הציור נמצא ניתן לנו יכולות להבחין באובייקטים שמתארים את הציור כמו עין, מקור ונוספות. כמו כן אנו מקשרים בין האובייקטים השונים למשל בין המקור לבין העין ולפי זה יודעים לומר כי אכן ציפור הוא האובייקט כלומר יש תלות גבואה (Attention) בין שני האובייקטים הללו כדי שנסיק כי אכן מדובר בציור. מיידך, בין הנוצאות בין הרקע בקיר אין תלות גבואה וכן התלות זו לא עוזרת לנו להבחין כי מדובר בציור. זו ההנחה שnbracha שהמודל שלנו יפגין ואלו הרפרנציציות שאנו רוצים שהוא ילמד. למעשה הרעיון הוא לחלק את התמונה למספר פאצ'ים כאשר אנו רוצים שהמודול יסיק לגבי אופי התמונה בהתאם לתלות בין הפאצ'ים השונים והסיפור שהם מספרים.



אך כפי שצינו, בהינתן וחילקו את התמונה לפאצ'ים באופן בו אנו מודדים Attention בין הפאצ'ים השונים, נאמר כי פאצ'ים בעלי dot-product אחד לשני יתנו ערךใหญה תלות גדול יותר דבר שייעיד על כך שיש בהם תלות (למשל כמו בין עין למקור). לעומת זאת בהינתן ערך ה-dot-product הוא קטן יחסית אליו נוכל להסיק כי לא קיימת תלות בין שני הפאצ'ים (למשל כמו בין עין לקיר).

ViT – Architecture Overview

נרצה לתרגם את ארכיטקטורת ה-ViT המוצגת בתמונה הבאה:

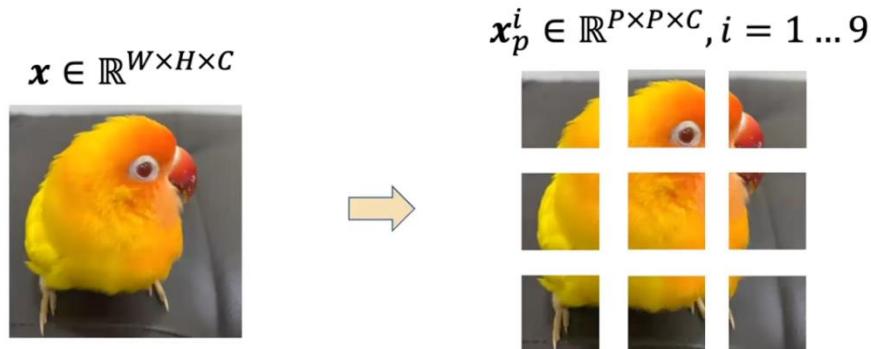


1. ניתן לראות כי הארכיטקטורה מורכבת ברובה מkomponentot שאותן כבר מכירים אותן נרצה לסקורCut: Images To Patches – בשלב זה אנו רוצים להמיר את התמונה לפאצ'ים קטנים יותר שיחד משילימים את התמונה. על מנת שהמודול יוכל ללמידה רפרזנטציות מעמיקות לגבי התמונה הוא צריך תחילה להבין את התalianות בין חלקים התמונה השונים ולשם כך צריך ליצור חלקיים אלו.
2. Patches To Features – לאחר שייצרנו את הפאצ'ים אנו רוצים להפוך אותם לייצוגים ווקטוריים מספריים שהמודול יוכל להבין ולעבד.
3. Positional Encoding – בשלב זה כפי שמקובל במודלים רבים מסוג טרנספורמר נרצה לקודד את מיקום הפאצ'ים בתמונה בוקטור המציג את הפאץ' לצורך למידה טובה יותר של המודול.
4. Transformer Encoder – החלק הלומד בארכיטקטורה שתפקידו לעכל את המידע וללמידה רפרזנטציות. כך הכל מדובר ב-Encoder סטנדרטי שכבר דובר הרבה darüber על הטרנספורמר.
5. MLP – שכבה נוספת נוספה המכילה לבסוף שכבה Dense שתפקידה ליצור לוגיטים שלבסון ישלו ל-Softmax שיבואו לבסוף לקבל Probability Distribution על כל המחלקות.

פרט שחשוב לציין הוא ש-ViT בצוותו הפשוטה כפי שאנו מדברים כאן עת הוא מודל שהמטרה שלו ללמידה רפרזנטציות על התמונה ולבסוף לשירות כל תמונה ל-Class כלשהו מבין סך כל-he-Classes. שילוי הוא נותן פרדיקציה. מספר-he-Classes הוא רב ויכול להגיע לאלפים ואף לעשרות אלפיים. כמו כן, ניתן לבצע מניפולציות מתקדיםות יותר כמו שימוש הארכיטקטורה עם GPTs באמצעות שיטות של transfer learning באופן בו הרפרזנטציות הוקטוריות שנלמדו מועברות למודל שפה לצורך הפקת מידע טקסטואלי אורך רצף עבור התמונה.

Images To Patches

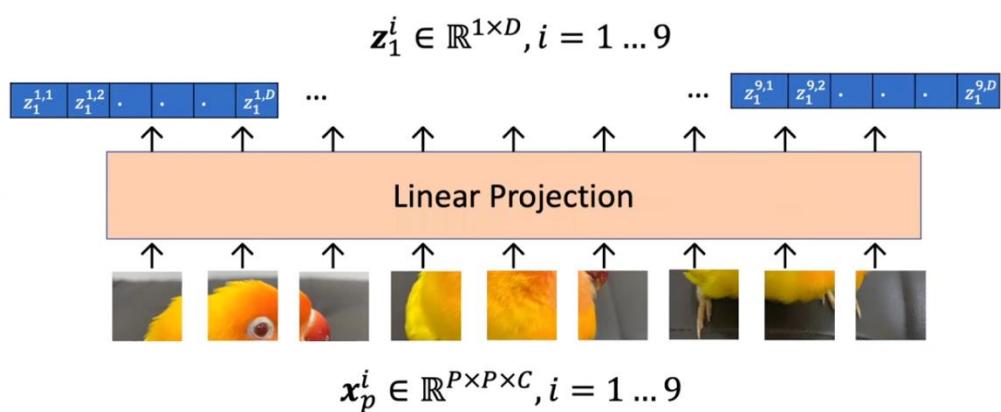
מבחן פורמלית נרצה לתאר את תהליך המעבר בין Image ל-Patches באופן הבא:
 בהינתן תמונה הIGINITALIA x נרצה לחלק אותה למספר פאצ'ים x_p^i ובקרה שלנו x_p^i כך ש מבחינה הממדים נקבל כי H, W מייצגים את הגובה והרוחב של התמונה המקורי ו- P מייצג את הגובה והרוחב של כל פאץ' המוגדר במרקחה זה להיות ריבועי. C בשני המקרים מייצג את מספר הערכים של התמונה, בהינתן ומדובר על RGB אזי $C=3$.



Patches to Features

לאחר שביצענו חלוקה של התמונה לפאצ'ים, אנו רוצים להמיר את המידע שמצוי בעות ביצוג תמונה אל ייצוג וקטורי שמייצג רפרנסטציה של הפאץ' בעני המודל. מבחינה פורמלית כל פאץ' x_p^i מומר לייצוג וקטורי z_p^i באופן בו המימד של הווקטור זה הוא D . כל מערכת כחול ביצור מייצג וקטור אחר כלשהו z_p^i ממימד D .

ההמרה מתבצעת באמצעות שכבה Linear Projection שממפה כל ייצוג פאץ' x_p^i לייצוג וקטורי z_p^i .



Linear Projection

נרצה לפרט בצורה פורמלית את התהיליך המתבצע ב-Linear Projection:

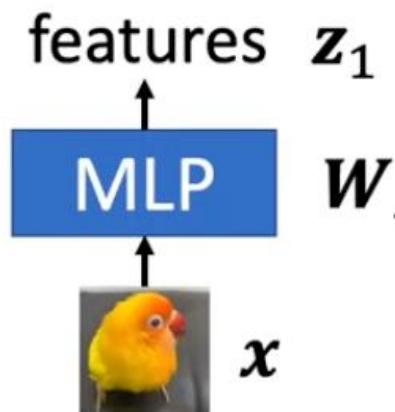
$$\mathbf{x}_p^i \in \mathbb{R}^{P \times P \times C} \rightarrow \mathbf{z}_1^i \in \mathbb{R}^{1 \times D}$$

- $\mathbf{x}_p^i \in \mathbb{R}^{P \times P \times C} \rightarrow \mathbf{x}_p^i \in \mathbb{R}^{1 \times P^2 C} \rightarrow \text{Reshape}$
- $\mathbf{x}_p^i \in \mathbb{R}^{1 \times P^2 C} \cdot \mathbf{W} \in \mathbb{R}^{P^2 C \times D} = \mathbf{x}_p^i \mathbf{W} = \mathbf{z}_1^i \in \mathbb{R}^{1 \times D} \rightarrow \text{Patch Linear Project}$
- $\mathbf{x} \in \mathbb{R}^{N \times P^2 C} \cdot \mathbf{W} \in \mathbb{R}^{P^2 C \times D} = \mathbf{x} \mathbf{W} = \mathbf{z}_1 \in \mathbb{R}^{N \times D} \rightarrow \text{Image Linear Project}$

- ניתן להבחן במספר שלבים עיקריים:
1. Reshape – למעשה מדובר ב-flatten של הטזוזר לווקטור חד מימדי. המעבר מתבצע בין הטזוזר המוגדר כ- $\mathbb{R}^{1 \times P^2 C}$ ל- $\mathbb{R}^{P \times P \times C}$.
 2. Linear Projection – בשלב זה יש מכפלת הווקטור לאחר ביצוע ה-flatten עם סט משקלות \mathbf{W} ללא bias בו ישנה התאמת של המימדים $\mathbb{R}^{P^2 C \times D}$ עם $\mathbb{R}^{1 \times P^2 C}$ באופן בו הווקטור המתkeletal הוא ווקטור חד מימי שמיידי המ $\mathbf{x}_p^i \mathbf{W} = \mathbf{z}_p^i \in \mathbb{R}^{1 \times D}$.

כפי שמקובל לא מעט בתחום, המכפלה לרוב לא מתבצעת בצורה ווקטורית ייחדנית אל בצורה מטריצונית. לשם כך, לא מתייחסים לווקטור \mathbf{x}_p^i בצורה ייחדנית אלא עורמים את הווקטורים הללו אחד על השני ליצירת מטריצה המוגדרת $\mathbb{R}^{N \times P^2 C}$ המציגת את מספר הפעצים (וואוקטורים) שערכנו זה מעל זה. לפיכך הפלט שיתקבל לאחר מכפלה עם $\mathbf{W} \in \mathbb{R}^{P^2 C \times D}$ יהיה $\mathbf{z} \in \mathbb{R}^{N \times D} = \mathbf{x} \mathbf{W}$.

מבחן ציורי נוכל להסתכל על התהיליך באופן הבא:



Conv2D – Alternative of Linear Projection

גישה אלטרנטיבית לשימוש ב-`Linear Projection` היא שימוש בקונבולוציה כדרך לייצרת ייצוג וווקטור עברור כל פazzi. למעשה כל קרNEL למד רפרנציאות במהלך תהליכי האימון שבאמצעותן הוא מוביל לכך לייצרת ערך סקלארי שמתווסף אל הווקטור.

$$\mathbf{x}_p^i \in \mathbb{R}^{P \times P \times C} \rightarrow \mathbf{z}_1^i \in \mathbb{R}^{1 \times D}$$

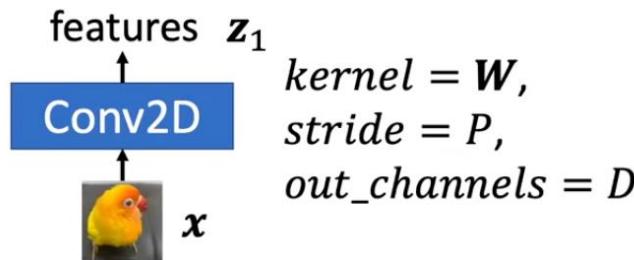
- $\mathbf{x}_p^i \in \mathbb{R}^{P \times P \times C} * \mathbf{W}_1 \in \mathbb{R}^{P \times P \times C} = \mathbf{x}_p^i * \mathbf{W}_0 = \mathbf{z}_1^{i,1} \in \mathbb{R}$
- ...
- $\mathbf{x}_p^i \in \mathbb{R}^{P \times P \times C} * \mathbf{W}_D \in \mathbb{R}^{P \times P \times C} = \mathbf{x}_p^i * \mathbf{W}_D = \mathbf{z}_1^{i,D} \in \mathbb{R}$

}
Number of filters

$\mathbf{z}_1^{i,1}$
 $\mathbf{z}_1^{i,2}$
 $\mathbf{z}_1^{i,3}$
 \vdots
 $\mathbf{z}_1^{i,D}$

- גם כאן נרצה לסקור את אופן הפעולה כenza המורכב ממספר שלבים ייחודיים מובילים לקבלת הווקטור:
1. קונבולוציה – לכל פazzi מתבצעת הקונבולוציה $\mathbb{R} \in \mathbf{z}_p^i = \mathbf{W}_p^i \mathbf{x}$. נשים לב כי גודל הפazzi מוגדר להיות $\mathbb{R}^{P \times P \times C}$ וכמו כן גודל הקרNEL הוא $\mathbb{R}^{P \times P \times C}$. עם זאת, נראה כי $\mathbb{R} \in \mathbf{z}_p^i$ קלומר סקלר ומכאן ניתן להסיק כי מתבצעת מכפלת dot-product בין כל האלמנטים בקרNEL לכל האלמנטים בפazzi לקבלת ערך בודד.
 2. Concatenation – מהסיבה שככל מכפלה בין קרNEL לבין פazzi נוונת ערך בודד למעשנה נרצה להחזיק D קרNELים $\mathbf{W}_D \dots \mathbf{W}_0$ עבור כל פazzi לקבלת סך הכל D ערכים להם נרצה לבצע Concatenation לקבלת וווקטור בגודל D.

סך הכל הפלט הסופי יהיה N וווקטורים $\mathbf{z}_p^N \dots \mathbf{z}_p^1$ (N מייצג את מספר הפazziים) כאשר כל אחד מהווקטורים הוא בגודל D לפי כמות הקרNELים המוחזקים המייצגים את המימד הסופי.



חשוב לציין כי השיטה שתוארה על שלל שלבייה לצורך לקבלת הייצוגים הווקטוריים היא שיטה אחת אופציונלית עבורה מתקיים כי עבור כל קרNEL מתקבל ערך סקלארי יחיד. עם זאת אפשר לעשות מניפולציות רבות שיכולות להוביל להעמקת הרפרנציאציה המתקבלת מביצוע הקונבולוציה כמו שימוש בערכי Padding, Stride, מיצעים, Pooling ושלל מניפולציות נוספות.

Positional Encoding



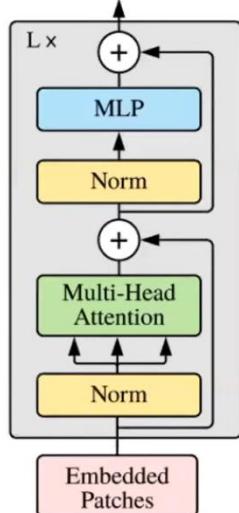
לאחר שביצענו Linear Projection או לחילופין קונבולוציה לקבלת ייצוגים וקטוריים עבור כל אחד מהפacciים, נרצה להוסיף לכל אחד מהייצוגים הוקטוריים אלמנט שיציג את מיקום הפacci בתמונה. זו הסיבה והמטרה לכך לככל Positional Encoding לתמונה באופן למידת הרפרנציאציה על ידי המודל. מסתכלות על 2 התמונות ניתן לראות כי התמונה העליונה היא ציפור והתמונה התחתונה מורכבת מערבוב הפacciים של התמונה העליונה בצורה רנדומית. כמו כן, אולי ניתן להסיק כי התמונה התחתונה היא גם ציפור אבל בהינתן והפacciים היו קטעים יותר לא בטוח שזו הייתה ממשימה קלה כל כך לאדם. עם זאת, עבור מודל זו משימה קלה מהסיבה שהוא לא תופס רפרנציאציות באופן בו אנו רוצים לככל קר הדבר אלא אוסף פacciים רנדומלית. לשם כך אנו רוצים לצלול Positional Encoding שיוביל לכך שניצור סדר בין הפacciים השונים באופן בו הרפרנציאציות שיילמדו על ידי המודל יהיו תואמות לרפרנציאציות שאנו לומדים בתור בני אדם כאשר אנו מתבוננים בתמונה.

<i>Patch</i>									
<i>Position</i>	1	2	3	4	5	6	7	8	9

- לטובת ביצוע ה-Positional Encoding ניתן להשתמש במספר שיטות:
1. Sinusoidal Encoding – כפי שהוצג במאמר המקורי של הטרנספורמר, שימוש בשיטה זו היא סטטistica ודטרמיניסטית באופן בו מוסיפים ערכיהם קבועים בהתאם לפרמטרים מוכתבים מראש. שיטה זו תוארה בהרחבה בשלב מוקדם יותר בסיכום כאשר דיברנו על הטרנספורמר.
 2. Learnable Encoding – שיטה בה אנו מתחילה במספר סטי משקלות נלמדות המתווספות לרוב בצורה של element-wise element-already-over-for כל אחד מהאלמנטים בוקטור עבור כל אחד מהפacciים. למעשה מסptr סטי המשקלות הוא כמספר הפacciים המקורי. במהלך תהליכי האימון סטי המשקלות הללו מתעדכנות בתהליך-backpropagation והמטרה שלהן היא למדוד בצורה דינמית מה האופן הנכון לבצע Positional Encoding עבור כל פacci' ועבור כל סט משקלות במיקום הפacci' מתבצעת הוספה element-wise element-already-over-for כל אחד מהפacciים. הוקטור המקורי המעודכן עבור הפacci' שלוקח בחשבון את מיקומו בתמונה.
 3. Rotary Encoding – שיטה שהובילה לשיפור שימושותי במלוטות NLP. על קצה המזלג ובוביל לצலול יותר מיידי לעומק, בהינתן וקטור המיציג פacci' המוגדר C_{1-t} נרצה להפעיל עליו פונקציה המוגדרת $(\theta, f_1, k, t) = \psi$ באופן בו ψ מוגדר להיות המיקום של הפacci' בתמונה המקורי ותטא מוגדרת להיות זווית הסיבוב. שם השיטה, אנו למעשה מושבבים בכל פעם את הוקטור באופן בו שומרים על התלות והיחסיות המרחביות שלו באופן בו ככל שהפacci' ממוקם יותר מוקדם ברצף הסיבוב שלו יהיה מינור יותר וככל שהפacci' יהיה מאוחר יותר ברצף הסיבוב שלו יהיה משמעותית יותר. יחס הגומלין בין הוקטוריים כתלות במיקומם בתמונה והיצוג שלהם במרחב נשמרים באופנה זו לקבלת Positional Encoding אפקטיבי.

Transformer Encoder

Transformer Encoder

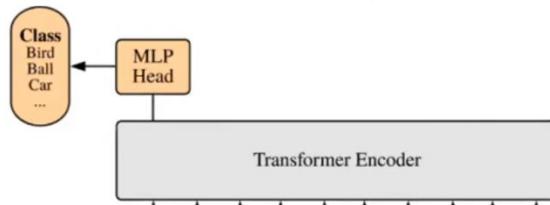


האמת היא שאין כאן איזשהו חידוש מיוחד בהשוואה ל-Encoder שראינו בטרנספורמר הרגיל. למעשה גם כאן מספר Encoders-al הוא L באופן בו-ה-*Embedded Patches* נכנסים אל-*Encoder* הראשון, עוברים דרך *Attention Multi Head Attention* שבתורה מכילה מספר ראיי *Add & Norm* ישנו לטובות מיצאת תלויות בין הפאצ'ים. לאחר מכן מוכן *MLP* לומד נוספת וועבר הפלט אל עבר-*Encoder* הבא בתור.

כפי שצוין, מטרת *Attention* בארכיטקטורה היא למצוא דפוסים בין הפאצ'ים השונים בתמונה שיביאו לכך שהמודל יוכל לקשר ביניהם וכתוצאה מכך ללמידה את הרפרזנטציה ולהשליך על תוכן התמונה לצורך מתן פרדיקציה לגבי סוג המחלקה אליה התמונה.

Prediction Head

הkomponentה האחרונה בארכיטקטורה היא *MLP* בה אנו מבצעים מכפלה מטריציונית בין סט משקלות *Linear Dense Layer* המיצרת לפלט *Encoder* האחרון באופן בו לבסוף יש מכפלה-ב-*Softmax* על כל המחלקות בDATA.



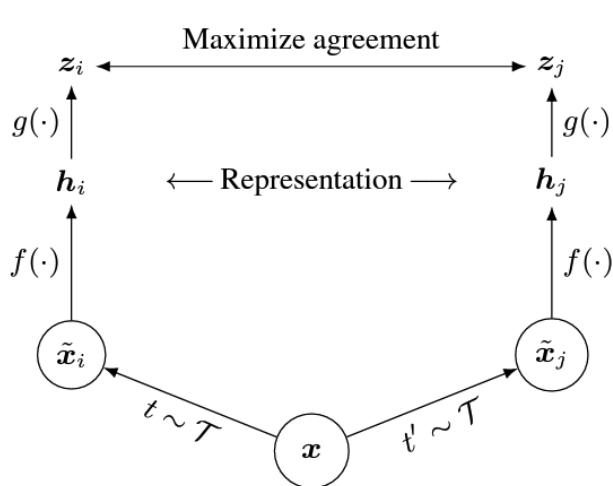
לסיום, ניתן לומר כי *TiVi* הוא בסופו של דבר מודל לטובות קלסיפיקציה של תמונות לכמונות מחלקות גדולות באופן יחס. 2 סוגים נתונים גדולים בהם ניתן לבצע שימוש:
 1. *JFT300M* – 300M תמונות המחלקות ל-18K מחלוקת.
 2. *ImageNet21K* – 14M תמונות המחלקות ל-21K מחלוקת.

באופן עקרוני ניתן לומר שדרך פעולה אפשרית היא ביצוע Pre-Train על נתונים גדולים ולאחר מכן לבצע *Fine Tune* עבור נתונים ייעודי. סך הכל השימוש המהותי-ב-*TiVi* הוא כמודל Encoder שהצליח לקודד בתוכו את רפרזנטציית התמונות באופן בו המטרה הייתה סיווג התמונות למחלקות ופעפו גראדיינטים במשקלות הרשות לטובות מיידית הדימויים של התמונות בעני המודל. לאחר הטמעת הידע ניתן להשתמש-ב-*TiVi* ביחד עם ארכיטקטורות נוספות *transfer learning*.

SimCLR

SimCLR היא ארכיטקטורה שלא עושה שימוש בטרנספורמר אך הסיבה שהיא מופיעה כאן היא מכיוון שבהמשך נרצה לתאר ארכיטקטורה בשם CLIP שעושה שימוש בטרנספורמר, ובפרט ב-ViT המבוססת על פונקציית loss בשם Contrastive Loss שhabava לכדי שימוש ב-SimCLR. מעבר לכך SimCLR היא ארכיטקטורה חשובה ולכל מבל' שום קשר היה חשוב לי לכלול אותה בסיכום.

SimCLR היא ארכיטקטורת Self-Supervised שפורסמה בשנת 2020 המאפשרת למודל למדוד רפרזנטציות על דאטא חסר תיוג באופן בו תחילתה נבעו לו אוגמנטציה ולאחר מכן נשווה בין האוגמנטציות השונות עבור כל דגימה בדאטא ומכך נלמד רפרזנטציות עבור כל הדגימות.

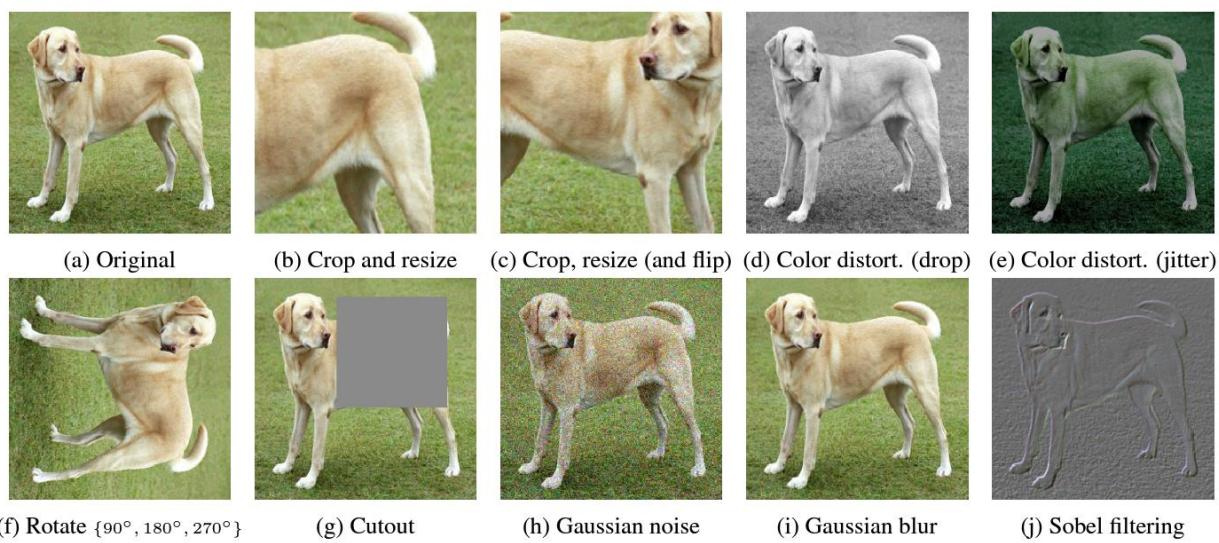


בתמונה ניתן לראות כי אנו מתחילה מدادטא המקורי
כלשהו x שלו אנו מבצעים 2 סוג אוגמנטציות:

1. $T \sim T$ לקבלת \tilde{x}_i
2. $T' \sim T'$ לקבלת \tilde{x}_j

כל אחת מהאגמנטציות \tilde{x}_i , \tilde{x}_j הן מעבירים דרך שכבה f לקבלת רפרזנטציות מרחב לטנטי המוגדרות כ- h_i , h_j לאחר מכן דרך שכבה נוספת g לקבלת z_i , z_j .
למעשה, המודל מנסה למציא רפרזנטציות z_i , z_j המתאזרת את הדאטא המקורי תחת ההנחה כי מיקסום ההסכם (Maximize Agreement) השקול למינימום ערך ה-Contrastive Loss בין z_i ל- z_j מUID על כך שהבינהן דאטא המקורי שמקורו באויה דגימה אך עם אוגמנטציה שונה יוביל לדימוי דומה בעניין המודל.

האגמנטציות כפי שתוארו יכולות להיות למשל כallo הלוקוחות מהאפשרויות הבאות:



SimCLR – פונקציית ה-loss ב-Contrastive Loss

אמנם Contrastive Loss היא לא פונקציית loss שהומצאה בשנת 2020 עבור SimCLR אך היא הובאה לכך שימוש בארכיטקטורה. מבחן רטוריקה, ב-SimCLR יש לנו דוגימות חיוביות ודוגימות שליליות. הדוגימות החיוביות הן הדוגימות הלקוחות מאותה Sample שהגדרנו אותה x והביאה לקבלת \tilde{x}_i , \tilde{x}_j ובהמשך לקבלת z_i , z_j . הדוגימות השליליות הן הדוגימות שמקורן לא באותו x . דוגמא לכך היא בהינתן DATAהמכיל תמונה כלב אחת ו-4 תמונות חתולים (פרק הכל 5 תמונות) אז במקורה וכעת אנו רוצים לחשב loss עבור תמונה הכלב, אז הדוגימות החיוביות זה בין האגמננטציות של הכלב, והשליליות בין אגמננטציות תמונה הכלב לבין אגמננטציות כל דוגימות החתולים. למעשה ב-Contrastive Loss כשמה כן היא, אנו רוצים למקסם את הנראות עבור הדוגימות החיוביות ולמנמן ככל הניתן את הנראות בין הדוגימות השליליות.

מבחן פורמלית נוכל לתאר את פונקציית ה-Contrastive Loss באופן הבא:

$$\ell_{i,j} = -\log \frac{e^{\frac{\langle z_i, z_j \rangle}{\tau}}}{\sum_{k=1}^{2N} \mathbb{I}_{[k \neq i]} e^{\frac{\langle z_i, z_k \rangle}{\tau}}}$$

באופן בו $\langle z_j, z_i \rangle$ מוגדר להיות Cosine Similarity באופן: $\langle z_i, z_j \rangle = \frac{z_i \cdot z_j}{\|z_i\| \|z_j\|}$

כמו כן $\mathbb{I}_{[k \neq i]}$ באופן בו לכל זוג סדרה המקיים $i \neq k$ מתקבל הערך 1 ולזוג הסדרה המקיים $i = k$ מתקבל הערך 0 כלומר יש החסירה של הערך מאחר והוא מוכפל ב-0.

מבחן מילולית, נראה כי בסופו של דבר המונה מכיל אך ורק את ערך ה-Cosine Similarity של הזוג הסדרה הלקוח מאותה דוגמה x שהוביל לשתי האגמננטציות \tilde{x}_i , \tilde{x}_j ולאחר מכן לרפזרנטציות z_i , z_j ואילו המכנה מכיל את כל שאר הזוגות הסדריים (כלב וחתול על כל פרמטריזציה).

נוכל להבחן כי על מנת להגיע ל-*Maximum Agreement* נרצה ערך מונה מקסימלי וערך מכנה מינימלי דבר שיעיד על קר שוואקטור z_i קרוב לווקטור z_j (מונה), כמו כן שוואקטור z_i רחוק מאוד מכך ווקטור אחר z_k (מכנה). כאמור לאחר ודבר ב-log – אז מונה גדול ומכנה קטן יוביל לערך קטן.

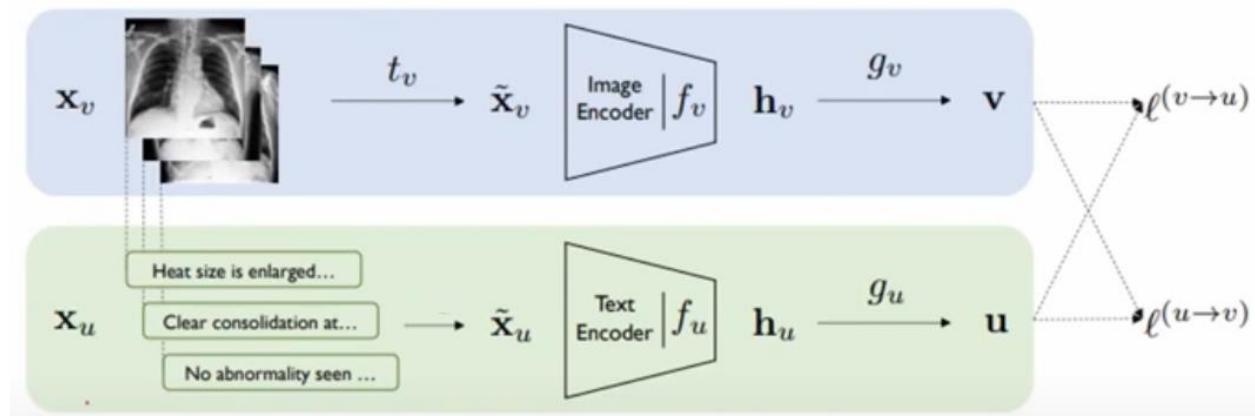
הקדמה ל-CLIP - ConVIRT

היה חשוב לי לכלול הקדמה ל-CLIP דוגמא נוספת לשימוש ב-Contrastive Loss שהתבצעה במאמר שפורסם בשנת 2020 והמטרה שלו היא לקשר בין רפרנציות של תמונות צילומי חזה לבין מידע טקסטואלי שמתאר את התמונות (דומה למה שנראה בקרוב ש-CLIP עשו).

שם המאמר:

Contrastive Learning Of Medical Visual Representations From Paired Images And Text
שם המודל: ConVIRT

ונכל להסתכל על התהילה שביצעו החוקרים בתרשים הגרפי הבא:



בדומה ל-SimCLR, אנו מתחילהים עם DATA בגודל M עבורו יש לנו M צילומי חזה ו- M תיאורים טקסטואליים. עבור צילומי החזה, אנו מבצעים אוגמנטציה באמצעות פונקציית אוגמנטציה כלשהי המוגדרת t_v כך שדגימות שעברו אוגמנטציה ומקורן אותה דגימת מקור מקבלות את אותו מידע טקסטואלי כתיאוג. כך הכל נקלט כי לאחר האוגמנטציה יהיו לנו N דגימות המורכבות מתמונה צילום החזה ומהמידע הטקסטואלי הרלוונטי לגבייה.

לאחר מכן, גם המידע הטקסטואלי וגם התמונות שעברו אוגמנטציה מועברות דרך Encoders לקבלת יציגים וקטוריים המוגדרים h_v, h_u בהתאם. לאחר מכן יש הכפלה בסטי משקלות נלמדות המוגדרות להיות g_v, g_u לקבלת v, u המייצגים את רפרנציות המודל עבור כל זוג סדור של תמונה וטקסט.

בשלב הבא נרצה בשלב האימון לחשב loss בין כל הזוגות הסדורים באמצעות Contrastive Loss. כך הכל נראה כי אנו מקבלים מטריצה מגודל N על N עבורה מתקיים כי האלכסון מייצג את הזוג הסדור של הדגימות החיוביות עבור כל דגימה ודגימה, לעומת זאת, שאר המטריצה מייצגת את הדגימות השליליות עבור כל אחד מהזוגות הסדורים.

ConVIRT – פונקציית ה-loss ב-Contrastive Loss

כפי שצוי, נרצה בשלב האימון לחשב loss בין כל הזוגות הסדריים באמצעות Contrastive Loss. לשם קר, נרצה להגדיר בצורה פורמלית את loss בו נרצה להשתמש ככזה המנסה למנמם את המרחק בין הדגימות החיוביות וממקסם את המרחק בין הדגימות השליליות. על אף שה Cosine Similarity היא מטריקת הערכה סימטרית, על פי הנאמר במאמר ולצורך רובוטטיות, המטריקה שנלקחה מוגדרת להיות דו-כיוונית. כאשר מדובר CLIP נראה בצורה ויזואלית כי $v \rightarrow u$ מתייחס לערך עבור השורות וכך $u \rightarrow v$ מתייחס לערך עבור העמודות. פורמלית נרצה לתאר באופן הבא:

$$\ell_i^{(v \rightarrow u)} = -\log \frac{e^{\frac{\langle v_i, u_i \rangle}{\tau}}}{\sum_{k=1}^N e^{\frac{\langle v_i, u_k \rangle}{\tau}}}$$

$$\ell_i^{(u \rightarrow v)} = -\log \frac{e^{\frac{\langle u_i, v_i \rangle}{\tau}}}{\sum_{k=1}^N e^{\frac{\langle u_i, v_k \rangle}{\tau}}}$$

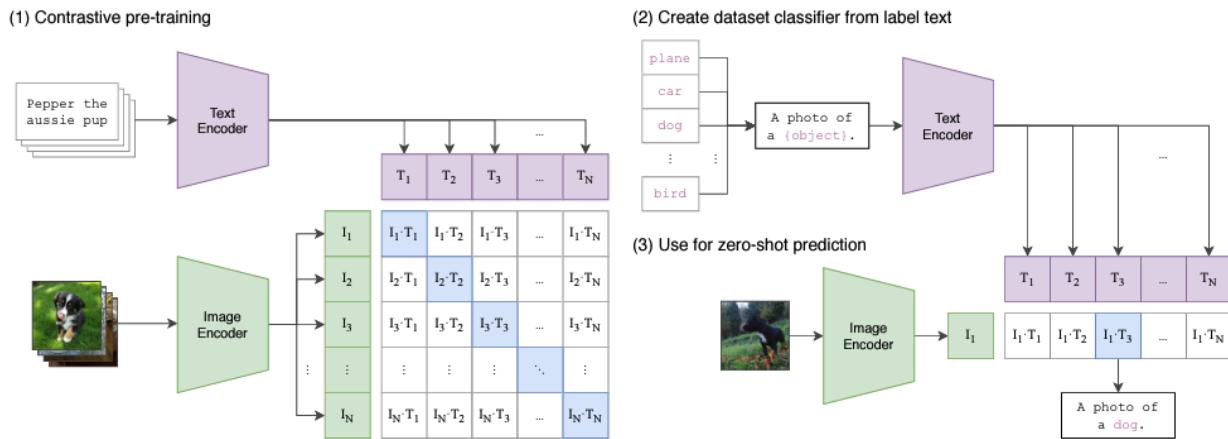
סך הכל נקבל כי ה-objective הכללי הוא:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (\lambda \ell_i^{(v \rightarrow u)} + (1 - \lambda) \ell_i^{(u \rightarrow v)})$$

למעשה נראה כי ה-objective Cosine Similarity מזיצית העבור כל זוג אלכסוני עבור כל שורה וכל עמודה בנפרד ולאחר מכן ממשק את הערכיהם הללו באמצעות היפרפרמטר λ לקבלת ה-objective. אנו נראה כי בדיקת Objective זה יהיה ה-objective שיישמש אותנו ב-CLIP.

CLIP

CLIP היא ארכיטקטורה מבית OpenAI שמטרתה למדוד רפרזנטציות ויזואליות בין תמונות לבין טקסטים המתאימים זה לזה. כוחותבים על זה גם MLP או CNN או כל ארכיטקטורה אחרת המשמשת לקולסיפיקציית תמונות לומדת רפרזנטציות בין התמונות לבין הטקסט (תיג) אך המיחודה ב-CLIP הוא שההטלות הzo נלמדות מטור ייצוגי Embedding של התמונות והtekst. כמו כן, ב-CLIP לא מתחייב כי התיאוג יהיה בדיד למשל חתול, כלב וכו' אלא יכול להיות טקסט ארוך.



לפני שנצלול לתיאור הארכיטקטורה בהתאם לשילוב נרצה תחילה לתאר שני מרכיבים מוחותיים בארכיטקטורה:

1. Text Encoder
2. Image Encoder

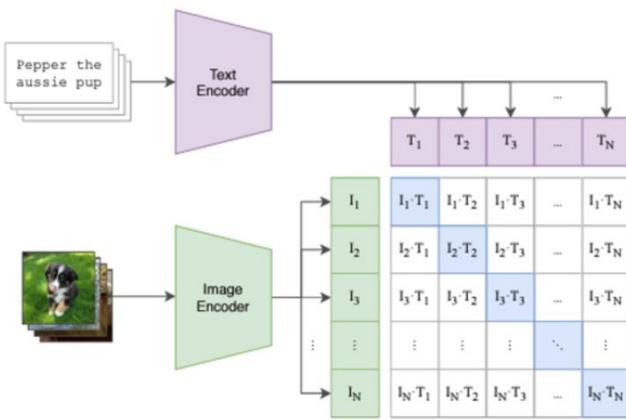
Text Encoder

למעשה מדובר ברכיב שמטרתו לבצע Embedding לtekst. הרכיב הזה יכול להיות מודל Word2Vec או GLOVe או אפילו שנהג כיום, GPT המכיל Embedding Matrix שיודיעו לקבל קולט מילה או מספר מילים ולהחזיר כפלט ווקטור המיצג את הטקסט ממייד כלשהו d.

Image Encoder

מטרתו של רכיב זה הוא לבצע Embedding לתמונה. הרכיב הזה יכול להיות ResNet, VGG או כל ארכיטקטורה אחרת שמודעת למדוד רפרזנטציות של תמונות בהתאם לתיאוג כלשהו. כיום נהוג להשתמש ב-TiVi או ארכיטקטורה דומה לה להיות זו שמבצעת Embedding לתמונות ומחזירה כפלט ווקטור ממימד d המיצג את התמונה.

שלב האימון – CLIP



בשלב האימון לוקחים Batch בגודל N המורכב מ-N תמונות ו-N רצפי טקסט המתארים אותם. כל אחד מרצפי הטקסט T_N ... עבר T_1 עבר Embedding וכנ"ל עבור כל אחת מהתמונות I_N ... I_1 . לאחר שעבור כל Embedding אחד מהתמונות/tekst בוצע Cosine Similarity לקבלת ייצוג וקטורי, מחשבים בין כל זוג סדרה $\{N \dots 1\} i \in j$, עבור כל טקסט וכל תמונה לקבלת המטריצה המתוארת בתמונה.

סך הכל נקבל כי כל שורה מתארת את הציון עבור תמונה או עם כל תיאור טקסטואלי ב-Batch. כמו כן, כל עמודה מתארת את הציון עבור כל טקסט או עם כל תמונה ב-Batch.

מבחן גראפית ומילולית נראה כי כאשר אנו מסתכלים על ה-Objective הוא מוגדר להיות כפי שדיברנו עד כה – Contrastive Loss באופן בו אנו רוצים למקם את ערך הקובייה הכחולה עבור כל שורה וכל עמודה ופורמלית:

	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	\ddots	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$...	$I_N \cdot T_N$

$$\ell_i^{(I \rightarrow T)} = -\log \frac{e^{\frac{\langle I_i, T_i \rangle}{\tau}}}{\sum_{k=1}^N e^{\frac{\langle I_i, T_k \rangle}{\tau}}}$$

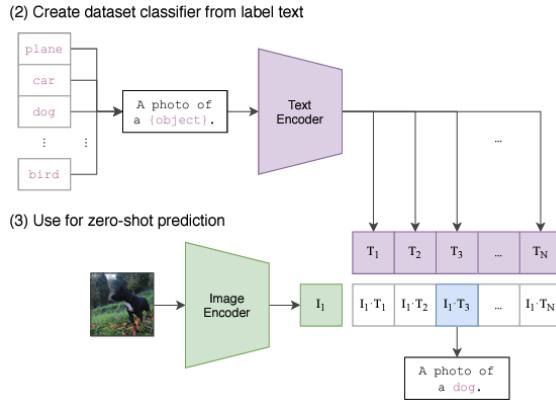
	T_1	T_2	T_3	...	T_N
I_1	$I_1 \cdot T_1$	$I_1 \cdot T_2$	$I_1 \cdot T_3$...	$I_1 \cdot T_N$
I_2	$I_2 \cdot T_1$	$I_2 \cdot T_2$	$I_2 \cdot T_3$...	$I_2 \cdot T_N$
I_3	$I_3 \cdot T_1$	$I_3 \cdot T_2$	$I_3 \cdot T_3$...	$I_3 \cdot T_N$
⋮	⋮	⋮	⋮	\ddots	⋮
I_N	$I_N \cdot T_1$	$I_N \cdot T_2$	$I_N \cdot T_3$...	$I_N \cdot T_N$

$$\ell_i^{(I \rightarrow T)} = -\log \frac{e^{\frac{\langle I_i, T_i \rangle}{\tau}}}{\sum_{k=1}^N e^{\frac{\langle I_i, T_k \rangle}{\tau}}}$$

משמעות הדבר היא שעבור כל עמודה וכל שורה אנו רוצים שערך הקובייה הכחולה יהיה מקסימלי דבר שיעיד על כך שקיים תלות הדוקה בין התמונה לתיאור שלה. לבסוף נקבל כי בדומה ל-ConViRT-URR ה-Objective הסופי הוא:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (\lambda \ell_i^{(I \rightarrow T)} + (1 - \lambda) \ell_i^{(T \rightarrow I)})$$

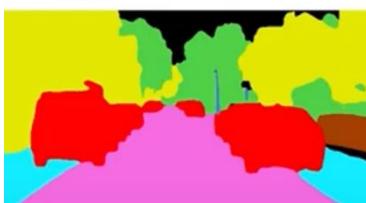
שלב ה-Inference – CLIP



בשלב ה-Inference לוקחים את כל המחלקות שיש לנו בDataset המוגדרות להיות $\{N\} \dots 1$ ומבצעים להן Embedding לקבלת $\{T_1 \dots T_N\}$.
 כתה בהינתן תמונה חדשה שנסמנתה I_1 נבצע חישוב של Cosine Similarity בין כל הייצוגים הווקטוריים $\{T_1 \dots T_N\}$ ל- $\{I_1 T_1 \dots I_1 T_N\}$.
 הטקסטואלים נראים כערוך ערך על ערך המקיים מהרר כי המיקסimal הוא זה שיוגדר להיות תיוג התמונה ולפיכך המידע הטקסטואלי שמכפלתו בייצוג הווקטורי של התמונה הוא הגודל ביותר והוא המידע הטקסטואלי שמתאים במידה רבה ביותר להיות תיאור התמונה.

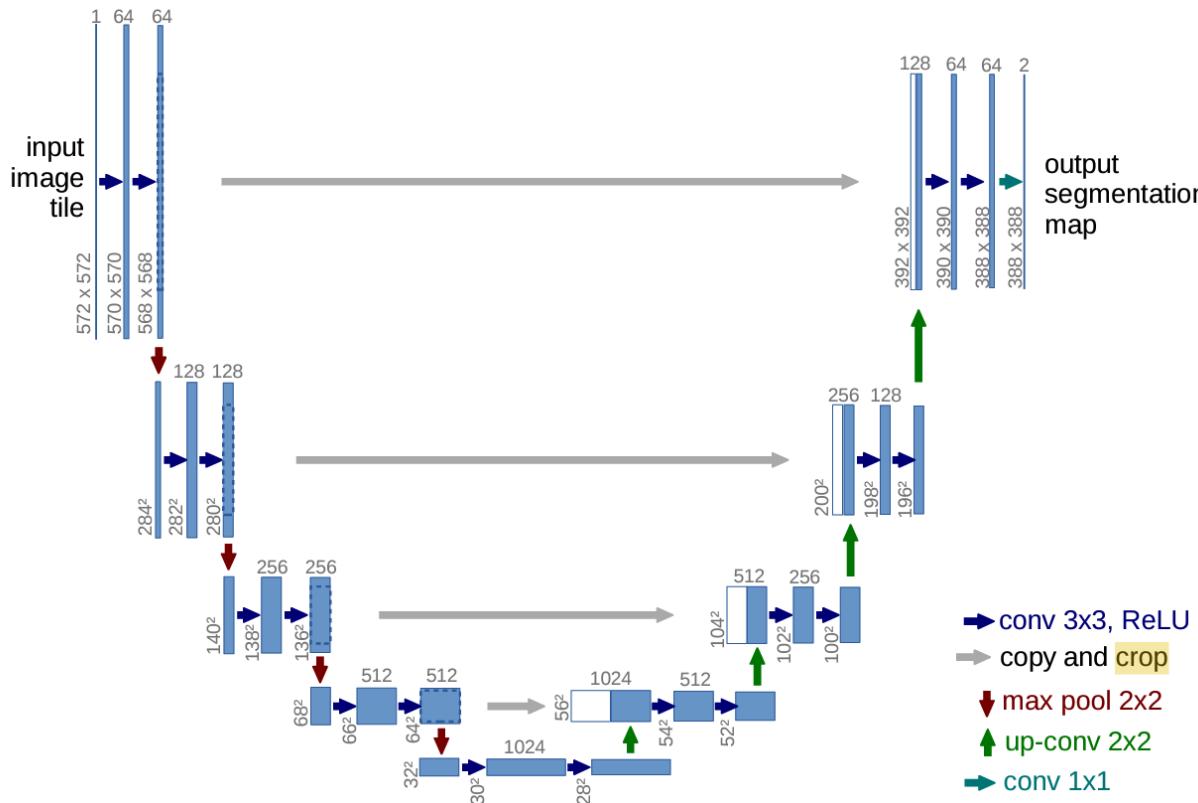
U-Net

U-Net היא ארכיטקטורה מבוססת קונבולוציה שהתפרסמה בשנת 2015 על ידי מספר חוקרים מגרמניה. במקור, מטרת הארכיטקטורה הייתה לטובות סגמנטציה תמונה ביולוגיות אך במרוצת השנים, נמצאו לה שימושים מעוניינים בתחום ה-AI Generative, בכלל ובמודלי דיפיזיה עליהם נדבר בהמשך בפרט.



על מנת להבין מה הבעיה המקוריוות אוטם רצו החוקרים לפתור נסתכל בתמונה ונראה כי בתמונה שלפנינו קיימים מספר אלמנטים: רכבים, כביש, מדרכה, עצים, שמיים גדר. המטרה של החוקרים הייתה ליצור ארכיטקטורה שבהינתן תמונה כלשהי תיווצר מפה המפה כל פיקסל בתמונה למחלקה המיצינית את הסיגוג שלו. בתמונה ניתן לראות כי מדובר בתמונה של רחוב עם אלמנטים מחי היום יומם אך המטרה של החוקרים הייתה לעשות את הסגמנטציה על נתונים ביולוגיים המייצגים רקמות המכילות תאים ביולוגיים, והמטרה הייתה לבצע סגמנטציה וחלוקת של התאים לסוגים על סמך התפקוד הפונקציונלי של התאים על סמך למידת רפרזנטציות של התאים בעני ה-U-Net.

הארQUITקטורה מזכירה מאוד AutoEncoders באופן בו אנחנו מתחילה מקלט כלשהו ולאחר סדרה של מניפולציות אנו מגעים לכדי ייצוג ווקטורי לטני של הקטל ההתחלתי. מאותה נקודה והלאה אנו מבצעים מניפולציות נוספות לכדי קבלה של פלט שדומה לקטל המקורי. מבחינה גרפית הארכיטקטורה נראה כך:



מאפייני הארכיטקטורה

שכבות קונבולוציה

בעוד شب-AutoEncoders השכבות הינו MLPs CANN, כאן מדובר בקונבולוציות בדומה ל-CNN. פרט חשוב שnitן לשים לב אליו הוא שבין כל שכבה קונבולוציה אנו למשה ממצאים Max Pooling Max מגודל 2×2 מה ששוביל לכך שאנו חותכים את גודל הפלט. עם זאת, אנו מוסיפים קרנלים שלומדים רפרנציאיות נוספות לכך למשל ניתן לראות כי הפלט של רמת הקונבולוציה הראשונה הוא $568 \times 568 \times 64$, לאחר הקונבולוציה הפלט הופך להיות $284 \times 284 \times 64$ (נכח פי 2 עבור רוחב וגובה) אך עם זאת מתבצעת קונבולוציה עם מספר קרנלים רב יותר לקלט פלט מגודל $282 \times 282 \times 128$ וכך חוזר חלילה עם כל רמת קונבולוציה.

יצוג לטנטי וקטורי

בדומה ל-AutoEncoders גם כאן אנו מגיעים לייצוג לטנטי שמנצח את ה-Essense של התמונה. בדוגמא שהוצאה במאמר הגודל של היצוג הלטנטי הוא $1024 \times 28 \times 28$.

Skip Connections

ניתן לראות כי לאורך הארכיטקטורה מצויים חיצים שחורים המהווים Residuals או בذرוגן המאמר נקראים Skip Connections שמטרתם כפי שמצוגת פעמים רבות בספרות:

1. מניעה של Vanishing Gradient
2. מתן יציבות מבנית של הקלטים – אמם ככל שמתקדמיים בארכיטקטורה לעבר הייצוג הלטנטי אנו מוצאים את מהות התמונה במהלך תהליך של למידה אך עם זאת אנו מאבדים את המבנה המרחבי של התמונה ובמקומו מוצאים ייצוגים וקטוריים חסרי תלות מרחבית. לשם כך אנו מבצעים הוספה של ערך הרמה המקבילה ב-Encoder בצורה ישירה ל-Decoder תוך כדי שאנו עושים Crop קטן כדי לוודא שבאמת הגדים מתאימים (מסומן בקוו מקווקו כחול באיזור).
3. החלקה של הגרADING למניעת התכנסות למינימום לוקליים – ניתן לקרוא עוד צאן סך הכל נראה כי מתבצעת הוספה ישירה $x + f(x)$ בין ה-Encoder ל-Decoder בהתאם להואם להו הכהול המקווקו.

יצוג הפלט

בשונה מ-AutoEncoders שם ה-loss חושב בתור Reconstruction Loss בין הפלט המקורי המשוחזר נשים לב כי במקרה זהה כפי שמופיע במאמר, הפלט הוא למעשה מיפוי בין כל פיקסל לבין האובייקט אליו הוא שייך. לפיכך נוכל לבדוק אם-2 אלמנטים מהותיים השונים בין הפלט לפלאט:

1. גודל התמונה עצמה – נובעת מ-Padding בפלט לטובת ההתאמה למודל
2. מספר העורצים בפלט – ניתן לראות באיזור כי מדובר ב-2 וכאן ניתן להסביר כי קיימות 2 מחלקות בארכיטקטורות הדיפוזיציה שנראה בהמשך חשוב לציין דוקא שהיצוג מזכיר במידה רבה יתר את המתறחש ב-Auto Encoders באופן בו אנו משתמשים ב-Net-U על מנת לעשות Denosing בין כל שכבת רעש במודל הדיפוזיציה.

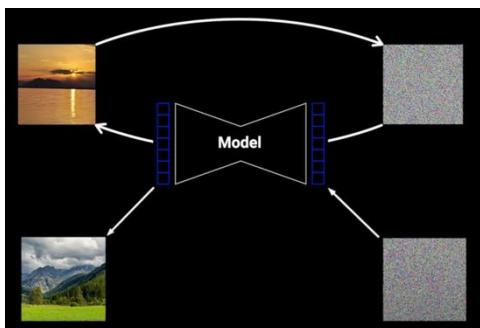
Difussion Model

[Video Link](#)

מודלי דיפיזיה התפתחו במהלך השנים האחרונות ופתחו את השער לייצור תמונות בצורה גנרטיבית. כיום, השחקנים המובילים בתחום הם AI, OpenAI, StabilityAI, Midjourney, และ Noots. כמו כן מגוון רחב של הוספות ושיפורים הושפו למודלים במשך השנים והובילו לכך התאמות המודל לייצור תוכן בצורה מותאמת אישית, מהירה ואיכותית יותר.

התקדמות הרבה בתחום הינה תודות למספר מאמרם בולטם כאשר הבולטים שביניהם הם:

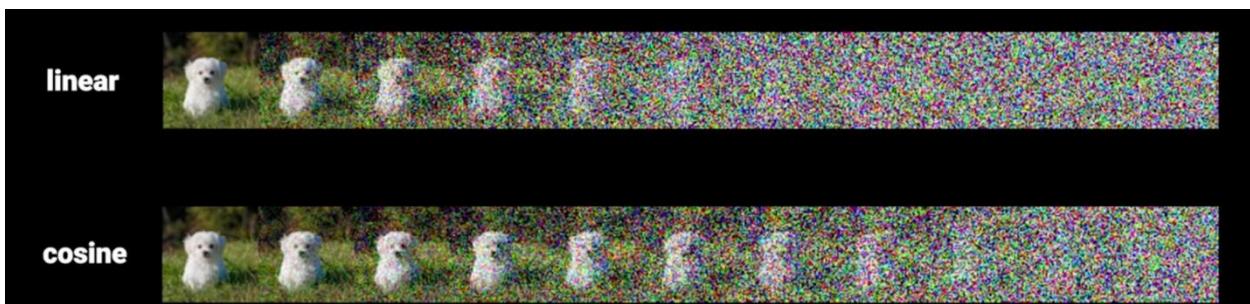
1. [Deep Unsupervised Learning using Nonequilibrium Thermodynamics](#)
2. [Denoising Diffusion Probabilistic Models](#)
3. [Improved Denoising Diffusion Probabilistic Models](#)
4. [Diffusion Models Beat GANs on Image Synthesis](#)



המאמר הראשון התפרסם בשנת 2015 על ידי מספר חוקרים מסטנפורד ובו חקרו לראשונה את רעיון הדיפיזיה. כפי שטענו, הרעיון המרכזי מאחורי מודל הדיפיזיה הוא בצורה איטית להרווש את מבנה התפלגות הדאטא בצורה איטרטיבית, לאחר מכן בתהילך הפור נרצה לשחזר את הרס מבנה התפלגות הדאטא באופן בו המודל יוכל ללמוד פרזנטציות שיובילו אותו להיות כל-גנרטיבי שיוודע לייצר דאטא מותך להתפלגות יש מאין.

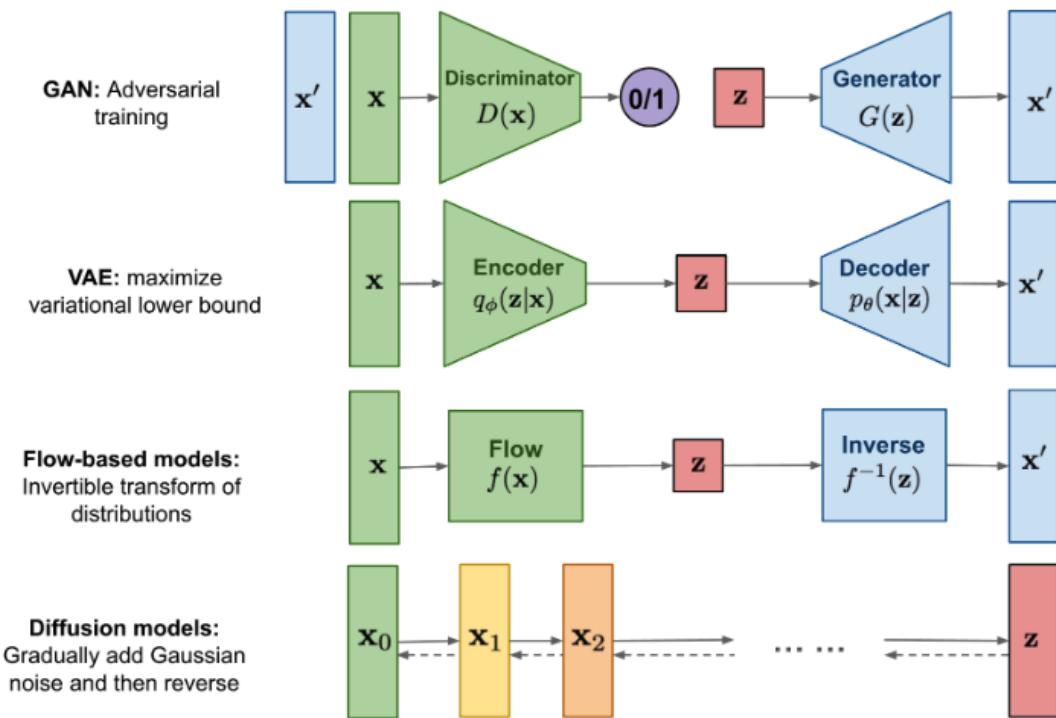
מספר שנים מאוחר יותר בשנת 2020 מספר חוקרים מברקלי שכללו את הרעיון והביאו אותו לכדי כדי יישם לייצור תמונה. החידוש המהותי של החוקרים מברקלי היה שימוש ב-U-Net ארכיטקטורתה שמודעת לבצע Denoising ובכך להוריד את הרעש בין כל שכבה לשכבה העוקבת אחרת. למעשה בינו היתר הרפזנטציה שנלמדת על ידי הארכיטקטורה היא רפזנטציית Denoising על ידי ה-U-Net שבצורה איטרטיבית יודעת איך להוריד רעש.

שנה מאוחר יותר בשנת 2021 פורטו שני חוקרים מ-OpenAI 2 מאמרם בהם שכללו עוד יותר את הארכיטקטורה במה שיהפוך לימים להיות DALLE. למעשה 2 שינויים מהותיים הבדילו את החוקרים הללו מעבודות שהתרחשו לפנייהם. הראשונה היא שלא רק ממוצע התפלגות הגaussינית נלמד לטובות ההרעשה ולאחר מכן בתהילך denoising אלא גם שונות התפלגות (החוקרים מברקלי קיבעו את השונות). הדבר השני הוא שתהילך ההרעשה היה הדרמטי יותר יותר דבר שהוביל את המודל לביצועים טובים יותר. הדבר השני של ברקלי ו-Cosine-Linear זה של OpenAI זה של Cosine-Linear.



סקירה מודלים גנרטיביים מובילים עד כה

עד כה כפי שדיברנו היו מספר מודלים גנרטיביים שהפגינו הצלחות מסוימות ביצירת תמונות אך כל אחד מהם בתורו נתקל בקשיים והגבלוות שמנעו ממנו להתקדם. חלק מהם ניתן לראות בתרשים הבא:



נרצה לתאר חלק מהקשיים והגבלוות שהמודלים הגנרטיביים הללו נתקלו בהם:

GAN

1. גינרט מוצמצם – הארכיטקטורה לא הצלחה לייצר מונע רחוב של סוג תמונות על אף סט רחב של וקטורים לטנטים z שהוכנסו לרכיב הגנרטיבי בארכיטקטורה **CatKing**.
2. קושי בירידה לפרטים – על אף שחילוקים מהמודלים הצלחו לייצר תמונות, לעיתים התמונות היו ליקות בחסר והיו לא עיקיות ולא ריאליסטיות כפי שהיוינו מצפים

VAE

1. טשטוש וגירעון תמונות – בין טبع-hat Loss Recounstruction של ה-VAE לעיתים מתkowski מצב בו התמונות המיוצרות מוטשטשות מהסיבה שפיקסלים קרובים נוטים לקבל ערך דומה
2. הכללה – בדומה ל-GAN גם כאן הארכיטקטורה מתקשה להקליל וליצור תמונות ממנע רחב

Flow-Based Models

1. זמני ריצה גדולים – לרוב המודלים הללו מערכיהם פועלות חישוביות גדולות שמובילות לזמן ריצה גבוהים וקושי בהתקנות המודל

מאפייני ארכיטקטורת ה-Diffusion

כפי שתיארנו עד כה, אופן פועלת הדיפוזיה הוא באופן בו אנו מוסיפים רעש הנדגם מהתפלגות נורמלית בצורה איטרטיבית ויסטמטית לתמונה המקורית כדי לקבל תמונה מורעשת במקצת בכל איטרציה. כתע, נרצה להיכנס קצת יותר לעומק ונוחיב על האופן שבו התהיליך מתבצע לkrarat הצלילה להסביר המסתם.

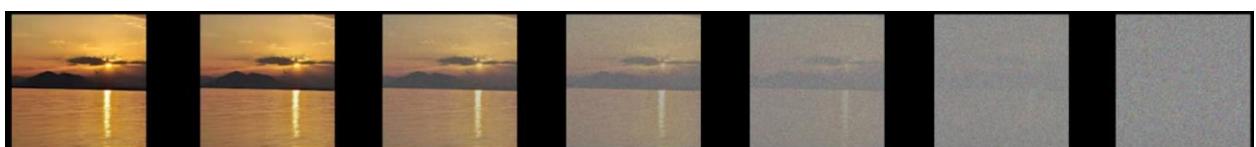
כפי שצין מספר ארכיטקטורות התפתחו במהלך השנים אך חשוב לציין כי בהסבר ובתייאור הארכיטקטורה נרצה לתאר חיצית אחידה של האופן בו ארכיטקטורת הדיפוזיה מתוארת באופן כללי מבל' להיזמד בצורה הדוקה לומר משהו צה או אחר.

ניתן לבדוק אם אכן מתחילה מהתמונה x_0 המוגדרת להיות תמונה רגילה שנלקחה זה עתה מהדאטסט ולא מכילה אף רעש. לאחר מכן, נרצה לדוגם רעש מהתפלגות גausיינית המוגדרת באופן הבא: (σ^2, μ) . את הדגימה נבצע לכל פיקסל בנפרד כך שכך הכל נקבל למשה מטריצה בגודל התמונה המכילה ערכים רנדומיים שנגדמו מתוך התפלגות הגausיינית.

בשלב הבא נבצע הוספה של המטריצה שנגדמה לתמונה המקורית x_0 לקבלת תמונה מורעשת במקצת שנסמנה x_1 . נמשיך ונחזור על התהיליך בצורה איטרטיבית עד שנקבל תמונה מורעשת לגמרי שנסמנה x או כפי שלפעמים מופיעה בספרות T באופן בו T מוגדר להיות האיטרציה הסופית והאחרונה באנו מבצעים הרעשה.



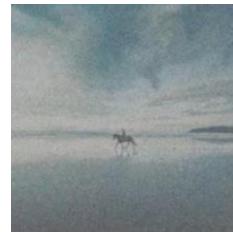
מבחן ויזואלית נראה כי אכן מתחילה מהתמונה ללא רעש כלל (צד שמאל) ולאורך התהיליך מבצעים הרעשה עד שמקבלים תמונה מורעשת לגמרי (צד ימין). המטרה היא לאמן U-Net Denoising שividע לעשות את הקפיצה ובהינתן תמונה מורעשת לעשות אותה מורעשת מעט פחות מהסיבה שהוא למד במהלך תהליך האימון רפרזנטציות שישו לו בתהיליך הורדת הרעשים.



תיאור מתמטי של ארכיטקטורת ה-הסויור

בוטציות

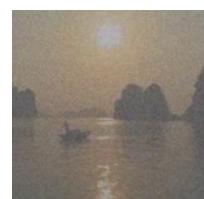
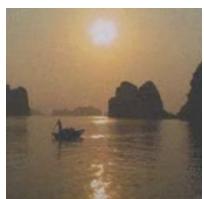
על מנת לתאר את הארכיטקטורה מבחינה פורמלית ומתמטית נרצה תחילת להתחיל בנותציות: נסמן בתור x_0 את התמונה ההתחלתית שעדיין לא התווסף אליה אף רעש. כמו כן, בכל פעם נדגם עבור כל פיקסל רעש מהתפלגות גאוסיינית המוגדרת (σ^2, μ) כך שבහינתם תמונה x_t תהיה x_{t+1} מוגננת ממנה מרווחת מעט יותר. כך למשל ניתן לראות בדוגמה ש- x_{42} מוגננת יותר מ- x_0 וכן כן ש- x_T המיצגת את התמונה המוגננת האחרונה, היא המוגננת ביותר ולא ניתן להבחין כלל בתחום התמונה מפאת הרעשיות שהתווסףו לאורך הדרך.

 x_0 x_{42} x_T 

כמו כן נרצה להגיד שני פונקציות נוספות:

1. $(x_t | x_{t-1})q$ – הפונקציה המיצגת את הרעשה.
2. $(x_t | x_{t-1})p$ – הפונקציה המיצגת את הורדת הרעש.

כפי שצוין x_{t-1} היא התמונה המוגננת פחות ו- x_t היא התמונה המוגננת יותר. הפונקציה q מבצעת את העבודה כתוצאה מדגימה מהתפלגות גאוסיינית עבורה כל פיקסל ואילו הפונקציה p מבצעת את העבודה באמצעות מודל הדיפיזיה.

 $p(x_{t-1} | x_t)$ $q(x_t | x_{t-1})$ 

הרעשה

imbachina formula nraha ci ponkzit ha'reusa mogdrat ba'open ha'ava:

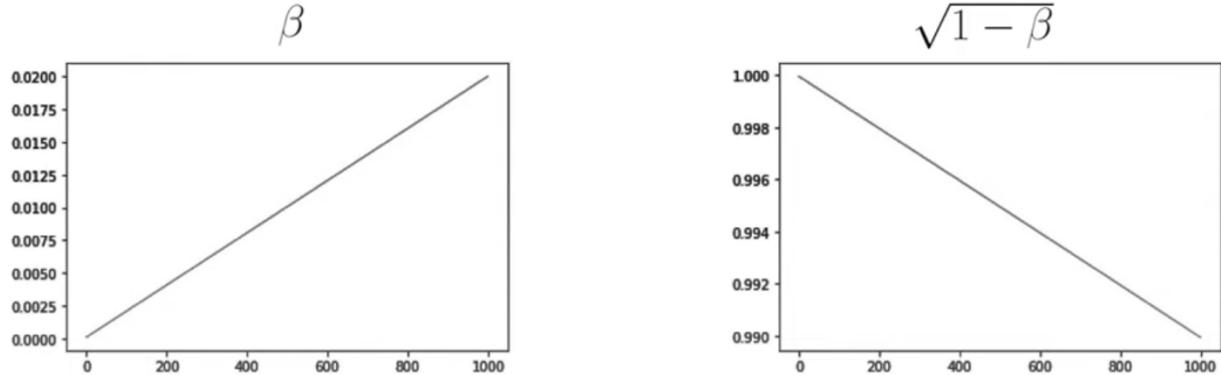
$$q(x_t | x_{t-1}) = \mathcal{N}(x_t, \sqrt{1 - \beta_t} x_{t-1}, \beta_t I)$$

במשואה, x_t מייצג את ה- Output_{t-1} , $\sqrt{1 - \beta_t} x_{t-1}$ מייצג את שונות התפלגות. כמו כן, נשים לב כי β הוא פרמטר שימושה עבור כל אחת מהאיטרציות. הערך ההתחלתי β_0 מתקבל מוגדר ב敞开ן יחסית ושונות קטנה יחסית כלומר הרעשה מסובית ודומה בא'open יחס' בין כל אחד מהפיקסלים. לעומת זאת, ככל שמתקרדים באיטרציות הרעשה נלקחת מתוך התפלגות עם ממוצע קטן ושונות גדולה כלומר יותר חלשה אך עם זאת יותר סטוכסטיות.

כך למשל נראה הגרפים עבורה:

$$\beta_{start} = 0.001$$

$$\beta_{end} = 0.02$$



az ziyino ci p'ulot ha'reusa matb'utzut be'zora a'itrat'iyet caser be'k'l a'itrat'ya mosipim cmot k'tuna shel russh ba'open bo amo match'ilim β_0 ve'zora anikr'mentalit v'be'adim k'tanim mag'ayim β_T ar le'mun ha'amta y'shnu tr'ek k'tan sh'anchnu y'kolim lab'uz shai'afshar b'iz'u p'ulot ha'reusa b'odot. ha'reusa b'voddot otteh na'racha lab'uz l'me'usa m'cila batocah at s'f ha'reusa she'znu ro'zim lab'uz batheil' a'itrat'ivi. ha'trik ha'za matap'shar mahsiba she'ospeh shel ur'k ha'reussh le'dgima hia p'ula li'na'riyot, ha'rcuba shel p'ulot li'na'riyot hia p'ula li'na'riyot chaluzma v'lo'k nitun la'hercib at kol ha'p'ulot lin'na'riyot h'�לן. achat ul ha'shni'a lcdi p'ula achot y'hida shatatar at ha'reusa ha'kollet.

מבחן מתמטית נגדיר:

$$\alpha_t = 1 - \beta_t$$

נזכיר את הערך בביטוי המתאר את ההרעשה ונשתמש ב-Trick Reparameterization ונקבל:

$$q(x_t | x_{t-1}) = \mathcal{N}(x_t, \sqrt{1 - \beta_t} x_{t-1}, \beta_t I) =$$

$$\sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon =$$

$$\sqrt{\alpha_t} x_{t-1} + \sqrt{1 - \alpha_t} \epsilon =$$

$$\sqrt{\alpha_t \alpha_{t-1}} x_{t-1} + \sqrt{1 - \alpha_t \alpha_{t-1}} \epsilon =$$

$$\sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2}} x_{t-1} + \sqrt{1 - \alpha_t \alpha_{t-1} \alpha_{t-2}} \epsilon =$$

ובהכללה לכל $t \in \{0 \dots T\}$:

$$\sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1 \alpha_0} x_{t-1} + \sqrt{1 - \alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1 \alpha_0} \epsilon$$

נראה כי ϵ מייצג דגימה מתוך התפלגות נורמלית $(I, 0)$ ולאחר מכן ישנה מכפלה במשמעות ההתפלגות Reparameterization Trick .

cut נרצה לסמן:

$$\bar{a}_t = \prod_{s=1}^t a_s$$

נשים לב כי בעקבות האנטציה זו נוכל לייצג את הביטוי:

$$\sqrt{\alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1 \alpha_0} x_{t-1} + \sqrt{1 - \alpha_t \alpha_{t-1} \alpha_{t-2} \dots \alpha_1 \alpha_0} \epsilon$$

בתווך:

$$\sqrt{\bar{a}_t} x_{t-1} + \sqrt{1 - \bar{a}_t} \epsilon$$

ומכאן נוכל להגדיר את פועלות ההרעשה בתווך:

$$q(x_T | x_0) = \mathcal{N}(x_T, \sqrt{\bar{a}_t} x_0, (1 - \bar{a}_t) I)$$

סך הכל נקבל כי אין עוד צורך במעבר איטרטיבי בהרעשה $x_{t-1} \rightarrow x_t$ אלא פשוט אפשר לנوع x_0 אל עבר x_T בצעד הרעשה בודד שמתככל בתוכו את כל צעדי ההרעשה האיטרטיביים.

Denoising

כאמור, אמנו שנשתמש ב-Net-U כרכיב בארכיטקטורה שיעוד לבצע את ה-Denoising בצורה אינטואטיבית. נרצה כעת לתאר בצורה מתמטית את הפעולה שמתבצעת בעת ה-Denoising, דבר שיבוליל אותנו בהמשך להגדרת ה-Objective.

از מבחן מתמטית, נרצה לתאר את הפעולה באופן הבא:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

מהתבוננות במשואה נוכל לבדוק במספר אלמנטים חשובים. למעשה בהינתן תמונה מורעשת x אנו רוצים ליחס תמונה מורעשת מעט פחות x_{t-1} -Output. בהרעשה, למעשה הגדרנו מפה בגודל התמונה אשר מוסיפה לכל פיקסל רעש רנדומלי בכל איטרציה. אכן, אנו רוצים לעשות בדיק את הדבר ההפוך וזה להשתמש ב-Net-U על מנת ליצור מפה בגודל התמונה אשר לכל פיקסל נוסף ערך שיבוליל למעשה Denoising שלו.

עקרונית הינו רוצים שה-Net-U ילמד את ההתפלגות על ממוצעה ועל שונותה אך מכיוון ובדוגמה זו השונות מקבעת, נרצה שההתפלגות שתלמיד תהיה מורכבת רק מערך ממוצע ההתפלגות ותהיה צוזה שהופכית לפעולות הרעשה. הסימון שייצג את ה-Net-U שלומד את ממוצע ההתפלגות הוא (x_t, t) וניתן לשים לב כי הוא מקבל כפלט את x ואת מספר האיטרציה t .

לאחר שהגדכנו את אופן פעולה המודל, נרצה כעת לדבר על ה-Objective שלו. עקרונית הינו רוצים שה-Objective יתואר כפונקציה של \hat{x}_0 ושל x_0 קלומר מעין Reconstruction Loss שיעוד למדוד את המרחק בין התמונה המשוחזרת \hat{x}_0 לבין התמונה המקורית x_0 שאנו הינו רוצים לסמן בתור:

$$-\log(p_\theta(x_0)) = -\log(p_\theta(x_0, \hat{x}_0))$$

הבעיה עם הביטוי הזה היא שאולי זה מהו שnitן לביצוע בתיאוריה אך בפועל מאוחר ובודכו הרבה מavoid פעולות Denoising לאורך הדרך לא ניתן לסתה זאת בפרקטייה. למעשה אנו תלויים במספר רב של פעולות לאורך הדרך:

$$x_0 x_1 x_2 x_3 \dots x_{T-2} x_{T-1} x_T \tilde{x}_{T-1} \tilde{x}_{T-2} \dots \tilde{x}_3 \tilde{x}_2 \tilde{x}_1 \tilde{x}_0$$

לשם כך נרצה לכלול רכיב נוסף שיבטא את הפעולות שבוצעו לאורך הדרך ויתן ביטוי מספק שטומן בחובו את פעולות Denoising. הרכיב הזה למעשה הוא KL Divergence שמנסה לקרב בין התפליגיות הרעשה לבין התפליגיות Denoising.

Variational Lower Bound – Using KL Divergence

כאמור, אמרנו שאנו לא יכולים לשירות להשתמש בביטוי המתואר כ- $\log(p_\theta(x_0))$ – מהסיבה שלא ניתן לבצע קפיצה ישירה בין התמונה המקורי לבין התמונה שנוצרה לאחר denoising והדבר אינו אינו ישים פרקטית. לפיכך נרצה להוסיף רכיב KL divergence למשואה שמטרתו לקרב את התפלגות ההרעשה להתפלגות denoising ומתואר באופן הבא:

$$-\log(p_\theta(x_0)) \leq -\log(p_\theta(x_0)) + D_{KL}(q(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0))$$

הסיבה שאנו יכולים לבצע בצורה שכך את רכיב KL divergence היא קונספט שנקרא **Variational Lower Bound**. עליו נרצה להרחיב את הדיון הנוכחי. כפי שידוע, אחת הפרדיגמות הבסיסיות ב-ML וביעות אופטימייזציה הוא להביא לכדי מינימום פונקציית loss המתארת את objective. לאורך השנים פותחו שיטות רבות לטובת ביצוע האופטימייזציה הזה אך כולן מבוססות בצורה זו או אחרת על ההנחה כי אנו מחשבים נגזרת של פונקציית loss ואז עושים צעד על גבי הגרדיינט בכיוון המנוגד לנגדת loss. המשמעות של Variational Lower Bound היא כך שהיא מנסה למצוא פונקציה חסומה על ידי פונקציית objective המקורית עבורו מתקיים כי: $\hat{y} \geq y : g(x) \in \hat{y}, f(x) \in y$.



בתמונה ניתן לבדוק כי ישנן שתי פונקציות, $f(x)$, $g(x)$ עברו מתקיים כי $f(x)$ נמצא מעל $g(x)$ לכל נקודה במרחב. אם נזכיר במדויקו שהרי היא מינימום loss על ידי צעד בכיוון המנוגד לכיוון הגרדיינט שמתקיים כתוצאה מחישוב נגזרת loss ונשים לב כי מאחר $-\hat{y} \geq y : g(x) \in \hat{y}, f(x) \in y$ אז קיימים סדר מילוי בין כל נקודה $f(x)$ לכל נקודה ב- $g(x)$ ומכאן הדבר משליך גם על הנגזרות ועל צעד הגרדיינט. מכאן מתחייב עברו מינימום של objective על גבי הגרדיינט $(x) g$ יקדם אותו אל f .

וכעת נחזור בחזרה לנקודה שלנו, במקרה שבו מופיע מאחר- $D_{KL}(q(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0))$ בהכרח מחזיר ערך חיובי אזי בהכרח: משווה חיובי + $(x_0) \log(p_\theta(x_0)) - \log(p_\theta(x_0))$. מכאן, הרעיון שמצאו פונקציה המקיים את אותויחס סדר ומכאן ניתן להסיק כי:

$$\forall y \in -\log(p_\theta(x_0)), \hat{y} \in -\log(p_\theta(x_0)) + D_{KL}(q(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0)) : y \leq \hat{y}$$

ולכן אנו יכולים להשתמש בו בתור objective תחת ההנחה שמיינמייזיה של objective החדש שקולה למינימיזיה של objective המקורי. מה שכן נשים לב כי עדין לא נפרטנו מהביטוי של $(x_0) \log(p_\theta(x_0))$ והוא עדין קיימב objective כך שעלה פניו אנחנו עדין לא יכולים להתקדם ועדין הדבר אינו ישים פרקטית. עם זאת, כעת כשמצוי לנו רכיב objective KL divergence נציג מספר פשוטים מתמטיים שיובילו לכך שנותן לפשט את הביטוי לביטוי עימיו יוכל להתקדם.

לצורך הפישוטים המתמטיים נסתכל על הגדרת ביטוי ה-KL Divergence המוגדר באופן הבא:

$$D_{KL}(q(x_{1:T}|x_0) \parallel p_\theta(x_{1:T}|x_0)) = \log \left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{1:T}|x_0)} \right)$$

cut נרצה להסתקל על המכנה ולפתחו אותו באמצעות Bayes Rule

$$p_\theta(x_{1:T}|x_0) = \frac{p_\theta(x_0|x_{1:T})p_\theta(x_{1:T})}{p_\theta(x_0)}$$

cut נרצה לתאר את הביטוי שהתקבל כתוצאה מ-Joint Probability -> Bayes Rule

$$\frac{p_\theta(x_0|x_{1:T})p_\theta(x_{1:T})}{p_\theta(x_0)} \rightarrow \frac{p_\theta(x_0, x_{1:T})}{p_\theta(x_0)} = \frac{p_\theta(x_{0:T})}{p_\theta(x_0)}$$

cut נציב שוב פעם את תוצאה פישוט המכנה בהגדרת KL Divergence ונקבל:

$$\log \left(\frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}} \right)$$

마חר - $\log \left(\frac{x}{y} \right) = \log(x) - \log(y)$

$$\log \left(\frac{q(x_{1:T}|x_0)}{\frac{p_\theta(x_{0:T})}{p_\theta(x_0)}} \right) = \log \left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right) + \log(p_\theta(x_0))$$

כך שscr הכל עבור ה-objective נקבל:

$$-\log(p_\theta(x_0)) \leq -\log(p_\theta(x_0)) + \log \left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right) + \log(p_\theta(x_0))$$

$$-\log(p_\theta(x_0)) \leq \log \left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})} \right)$$

ובכך מצאנו פונקציה חדשה שמהווה עבורה Variational Lower Bound אותה אנו כן יכולים לחשב. בשלבים הבאים מה שנרצה לעשות זה המשיך ולפשט את הביטויים המתמטיים לטובות הגעה ל-objective ממצה יותר עבור הבעיה.

נסתכל כעת על הביטוי $\log\left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right)$ ונרצה לתרו לפי אופן הגדרתו באופן הבא:

$$\log\left(\frac{q(x_{1:T}|x_0)}{p_\theta(x_{0:T})}\right) = \log\left(\frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p(x_T) \prod_{t=1}^T (p_\theta(x_{t-1}|x_t))}\right)$$

נשים לב כי במקרה זה המונה מתאר את פעולות ההרעשה והמכנה מתאר את פעולות denoising. המשמעות של $(x_T)p$ היא הדגימה מtower ההתפלגות המקוריית ולאחר מכן יש במכנה תיאור המבטה את כל פעולות denosing המתרחשות לאחר התהלייר.

ובוצע עוד מספר פישוטים מתמטיים ונקבל:

$$\log\left(\frac{\prod_{t=1}^T q(x_t|x_{t-1})}{p(x_T) \prod_{t=1}^T (p_\theta(x_{t-1}|x_t))}\right) = -\log(p(x_T)) + \log\left(\frac{\prod_{t=1}^T q(x_t|x_{t-1})}{\prod_{t=1}^T (p_\theta(x_{t-1}|x_t))}\right)$$

נכניס את $-\log$ פנימה לטובות קבלת ביטוי סכימה על פניהם מכפלה:

$$-\log(p(x_T)) + \log\left(\frac{\prod_{t=1}^T q(x_t|x_{t-1})}{\prod_{t=1}^T (p_\theta(x_{t-1}|x_t))}\right) = -\log(p(x_T)) + \sum_{t=1}^T \log\left(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right)$$

נרצה כעת לבצע הוצאה החוצה של הביטוי הראשון בסכימה ונקבל:

$$-\log(p(x_T)) + \sum_{t=1}^T \log\left(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right) = -\log(p(x_T)) + \sum_{t=2}^T \log\left(\frac{q(x_t|x_{t-1})}{p_\theta(x_{t-1}|x_t)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

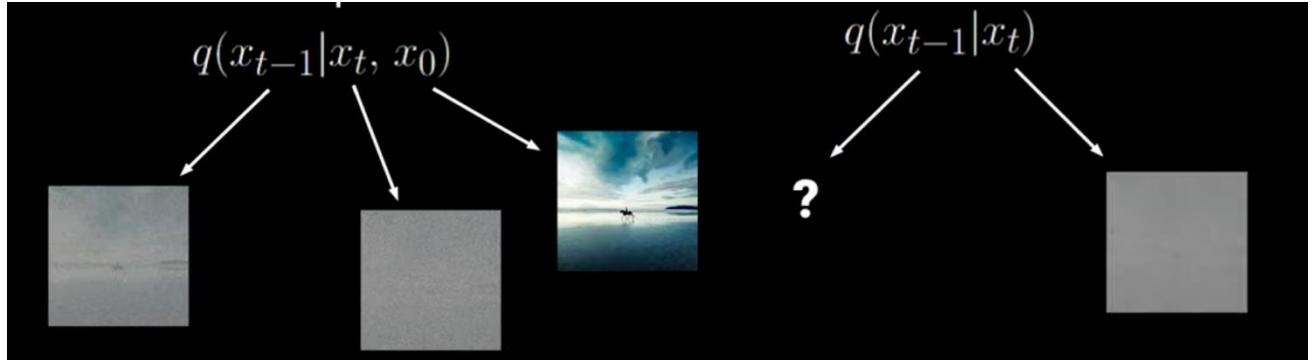
נרצה כעת להסתכל על הביטוי: $(x_{t-1}|x_t)q$. נשים לב כי אנו יכולים לייצגו באמצעות Bayes Rule באופן הבא:

$$q(x_t|x_{t-1}) = \frac{q(x_{t-1}|x_t)q(x_t)}{q(x_{t-1})}$$

עם זאת, קיימת בעיות קלה מהסיבה שיש לו מעט שונות בקריב כל אחד מהאלמנטים בביטוי. אם חושבים על זה, לכל שלב באמצעות בו אנו רוצים להריעש את x_{t-1} בהינתן x_t , יכולות להיות לא מעט אפשרויות וזו מהסיבה שאנו לא יודעים מה התמונה המקורית ממנה יצאנו. לפיכך נרצה לכלול מתמטית את התמונה המקורית ממנה יצאנו וכן נקבל את הביטוי:

$$\frac{q(x_{t-1}|x_t)q(x_t)}{q(x_{t-1})} \rightarrow \frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}$$

כעת, מכשכלנו את התלות בתמונה המקורית אנו מצמצמים באופן שימושותי את המרחב עליו פרוסות אפשרויות ההרעשה ולפיכך אנו מקבלים וודאות הרבה יותר ופחות שונות.



מהתבוננות באירועים ניתן לראות תצוגה ויזואלית לנאמרCut. בצד ימין ניתן לראות כי בהינתן x בלבד, קשה לנו להשליך על x_{t-1} וזו מהסיבה שלא דועה לנו התמונה הIGINITALית והתמונה המעודשת יותר הבאה תהיה קשה לגילוי מהסיבה שיש כל כך הרבה תמונות שיכללו להוות תמונות הIGINITALיות ולכך השונות גבורה. מאידך, מצד שמאל ניתן לראות כי בהינתן התמונה הIGINITALית x_0 , אנו יודעים מאי זו התמונה הIGINITALית יצאנו וכן השונות יורדת משמעותית כי אנו יודעים בצורה איטרטיבית מה התמונה הבאה שצפוייה להיות עד כדי הוסף רעש גausian כלשהו.

אם נעשה פлаг לנוסחא שפיתחנו אל נוסחת Objective-Aotta אנו מנסים לפתח Cut נΚבל:

$$-\log(p(x_T)) + \sum_{t=2}^T \log\left(\frac{q(x_{t-1}|x_t, x_0)q(x_t|x_0)}{p_\theta(x_{t-1}|x_t)q(x_{t-1}|x_0)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

על מנת להמשיך ולהתקדם נרצה לפרק את הסכימה ל-2 סכימות באופן הבא:

$$-\log(p(x_T)) + \sum_{t=2}^T \log\left(\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right) + \sum_{t=2}^T \log\left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

על מנת לקבל מוטיבציה מה הסיבה שבגינה רצינו לפרק את הסכימה ל-2 נסתכל על הדוגמא הבאה בה למשה ביצענו 4 צעדי הרעשה. עבור $\sum_{t=2}^4$ נΚבל:

$$\sum_{t=2}^T \log\left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right) = \log\left(\prod_{t=2}^4 \left(\frac{q(x_t|x_0)}{q(x_{t-1}|x_0)}\right)\right) = \textcolor{blue}{\log\left(\frac{q(x_2|x_0)q(x_3|x_0)q(x_4|x_0)}{q(x_1|x_0)q(x_2|x_0)q(x_3|x_0)}\right)}$$

נשים לב כי ישנו צמצום מסויימי של כל ביטוי הבינים עבור הרעשה הביניים באופן בו הביטוי שמתפרק מכיל רק את הרעשה האחרונה בהינתן התמונה המקורית במונה וכמו כן עבור המכנה הביטוי שמתפרק מכיל רק את ההרעשה הראשונה בהינתן התמונה המקורית. Cut נשים לב כי הביטוי המתפרק נראה באופן הבא:

$$-\log(p(x_T)) + \sum_{t=2}^T \log\left(\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right) + \textcolor{blue}{\log\left(\frac{q(x_T|x_0)}{q(x_1|x_0)}\right)} + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$$

מסיבה ש- $\log\left(\frac{q(x_T|x_0)}{q(x_1|x_0)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right)$ נגיא לפישוט הבא:

$$\log\left(\frac{q(x_T|x_0)}{q(x_1|x_0)}\right) + \log\left(\frac{q(x_1|x_0)}{p_\theta(x_0|x_1)}\right) = \log(q(x_T|x_0)) - \log(q(x_1|x_0)) + \log(q(x_1|x_0)) - \log(p_\theta(x_0|x_1))$$

נשים לב כי שני הביטויים האדומים מצמצמים זה את זה ולפיכך נישאר רק עם הביטוי הבא:

$$\log(q(x_T|x_0)) - \log(p_\theta(x_0|x_1))$$

עבור הביטוי הירוק: $\log(q(x_T|x_0))$, נאחדו עם הביטוי $(\log(p(x_T)))$ המופיע בתחילת ה-objective. כך שכעת objective יתואר באופן הבא: ונקבל $\log\left(\frac{(q(x_T|x_0))}{(p(x_T))}\right)$

$$\log\left(\frac{(q(x_T|x_0))}{(p(x_T))}\right) + \sum_{t=2}^T \log\left(\frac{q(x_{t-1}|x_t, x_0)}{p_\theta(x_{t-1}|x_t)}\right) - \log(p_\theta(x_0|x_1))$$

עבור שני הרכיבים הראשונים בביטוי, לאחר ומודובר ב-log בין יחסים נרצה כעת ליצגם בתצורה של KL Divergence ולפיכך נקבל את הביטוי הבא:

$$D_{KL}(q(x_T|x_0) \| p(x_T)) + \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1))$$

כמו כן, נשים לב כי עבור הביטוי $(\log(p(x_T)))$ $D_{KL}(q(x_T|x_0) \| p(x_T))$ מופיע בסוף הכל דוגימה של רוש מtower התפלגות גאוסינית (דגימת תמונה הרעשה ההתחלתית ממנה נתחיל לבצע Denoising), כמו כן, $(q(x_0|x_T))$ היא סדרה של פעולות הרעשה הלקוחות מtower התפלגות גאוסינית. נוכל להסביר כי עבור שתי ההתפלגות המרחק יהיה קטן באופן יחסי ולפיכך מגדד ה-KL Divergence $p(x_T)$, $q(x_T|x_0)$ גם כן יהיה קטן יחסית ומכאן $0 \approx D_{KL}(q(x_T|x_0) \| p(x_T))$ ולפיכך ניתן להזניחו תחת ההנחה שההתפלגות זהות במידה מה. סך הכל נישאר עם הביטוי הבא בלבד:

$$\sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) - \log(p_\theta(x_0|x_1))$$

$$\text{עבור } \sum_{t=2}^T D_{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)) \text{ נראה כי למשה אנו רוצים לקרב שתי התפלגיות:}$$

$$\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} \cdot 1. \quad .2$$

מהתבוננות ב- $(x_{t-1}|x_t)$ אנו יכולים לראות כי כפי שכבר הגדרנו, היא מוגדרת באופן הבא:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}, \mu_\theta(x_t, t), \beta I)$$

בקצרה, בהינתן x ואיטרציה t , רשות ניירונים המוגדרת μ מנוסה על ידי דוגימה מהתפלגות גausianית (בפועל נועה על ידי U-Net) לעשות denoising לקבלת x_{t-1} . חשוב לציין כי ישנן וריאציות בהן גם השונות היא ערך נלמד וערכו מתתקבל כתוצאה מהמעבר הקלט דרך רשות ניירונים, אך כאן קיבענו את ערכו להיות βI .

באשר ל- $(x_{t-1}|x_t, x_0)$ נרצה להגדיר את הפעולה באופן הבא:

$$q(x_{t-1}|x_t, x_0) = \mathcal{N}(x_{t-1}, \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I)$$

גם כאן נראה כי מדובר הרעשה התלוייה בממוצע התפלגות ובשונות. נרצה לתת ביטוי מסומי המתאר את ערך הממוצע ואת השונות שהתקבלה כתוצאה מסדרה של פישוטים מתמטיים שלאו דווקא נחוצים לצורך ההבנה הכללית של אופי הפעולה ולפיכך לא נבצע אותם אלא רק נפростם כאן כעת ונתקיים מנקודת מבקודה זו:

$$\begin{aligned} \tilde{\mu}_t(x_t, x_0) &= \frac{\sqrt{a_t} (1 - \bar{a}_{t-1})}{1 - \bar{a}_t} x_t + \frac{\sqrt{\bar{a}_{t-1}} \beta_t}{1 - \bar{a}_t} x_0 \\ \tilde{\beta}_t &= \frac{1 - \bar{a}_{t-1}}{1 - \bar{a}_t} \beta_t \end{aligned}$$

נרצה להתקדם רק עם ערך הממוצע (x_t, x_0) עבורי נמשיך בפישוטים. נסתכל כעת על הביטוי המוגדר באופן הבא: $\epsilon = \sqrt{1 - \bar{a}_t} x_0 + \sqrt{\bar{a}_t} x_t$. נוכל לראות כי הביטוי $-x_t$ מוגדר להיות כזה בו אנו מבצעים Reparameterization Trick בו אנו לוקחים את התמונה המקורית x_0 ומכפילים אותה ב- $\sqrt{\bar{a}_t}$ ומוסיפים לה $\epsilon = \sqrt{1 - \bar{a}_t} \epsilon$ כאשר ϵ היא דוגימה מתוך התפלגות נורמלית לקבלת התמונה המורעשת באיטרציה t . ממשחק בין האגפים ופישוט מתמטי קיבל את הביטוי הבא:

$$x_0 = \frac{1}{\sqrt{\bar{a}_t}} (x_t + \sqrt{1 - \bar{a}_t} \epsilon)$$

נרצה כעת לעשות פלאג לביטוי שמצאנו עבור x_0 המוגדר: (ϵ) אל תוך הביטוי
הגדול: $\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{a_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t} x_t + \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t} x_0$ ונקבל:

$$\tilde{\mu}_t = \frac{\sqrt{a_t}(1-\bar{a}_{t-1})}{1-\bar{a}_t} x_t + \frac{\sqrt{\bar{a}_{t-1}}\beta_t}{1-\bar{a}_t} \frac{1}{\sqrt{\bar{a}_t}} (x_0 + \sqrt{1-\bar{a}_t} \epsilon)$$

ולאחר פישוט מתמטי נקבל את ערך הביטוי המוגדר באופן הבא:

$$\tilde{\mu}_t = \frac{1}{\sqrt{a_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \epsilon \right)$$

סך הכל קיבלנו ביטוי מתמטי המתאר את $\tilde{\mu}$ כלומר את ערך ממוצע ההתפלגות של פועלות ההרעשה.
מהסיבה שאנו רצים לקבל מدد עבור ה-objective objectiveoss בין אופן ההרעשה לבין יכולת
ה-Denoising באופן שנוכל לכמת, נסתכל על MSE המוגדר באופן הבא:

$$L_t = \frac{1}{2\sigma_t^2} \| \tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \|^2$$

נרצה לתאר את μ_θ באופן הבא:

$$\mu_\theta(x_t, t) = \frac{1}{\sqrt{a_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \epsilon_\theta(x_t, t) \right)$$

נרצה כעת לעשות פלאג למשואה גם L_t וגם $\tilde{\mu}_t$ עבור הביטויים אליום הגענו ולקבל:

$$L_t = \frac{1}{2\sigma_t^2} \| \frac{1}{\sqrt{a_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \epsilon \right) - \frac{1}{\sqrt{a_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{a}_t}} \epsilon_\theta(x_t, t) \right) \|^2$$

בסיומו של דבר את כל הביטוי הגדול הזה אנו יכולים לפשט לביטוי הבא:

$$L_t = \frac{\beta_t^2}{2\sigma_t^2 a_t (1-\bar{a}_t)} \| \epsilon - \epsilon_\theta(x_t, t) \|^2$$

כמו כן נשים לב כי $\frac{\beta_t^2}{2\sigma_t^2 a_t (1-\bar{a}_t)}$ הוא כפל קבוע ולכן ניתן להתייחס ל- L_t באופן הבא:

$$L_t = \| \epsilon - \epsilon_\theta(x_t, t) \|^2$$

מהתבוננות ב- $\|x_t, t) - \epsilon\|$ נשים לב כי בסך הכל, כל מהות הארכיטקטורה היא למצוא קירוב בגין האופן שבו הzbacעה ההרעשה מתוך התפלגות ϵ אל מול האופן בו מתבצע הניקוי באמצעות U-Net-U מתוך התפלגות נלמדת $(x_t, t)\epsilon$. את המהלך בין שתי התפלגות הללו אנו רוצים לצמצם כך שבעור כל איטרציה בדרך אופן ה-Denoising ייה קרוב ככל הניתן לאופן ההרעשה מה שיביאו אותנו ליצירת דוגמאות כמה שיותר קרובות למקור מצד אחד ומצד שני מכך ואנו מדברים על התפלגות נלמדת במהלך תהליך Denoising, סתוקואה שהמודול יוכל להקליל את התפלגות הזו ויביא לכך שבעתה תהליך-h-Inference, Denoising יוביל ליצירת תמונה חדשה יש מאין הלוקוה מאותה התפלגות.

כעת כאשר אנו רוצים לתאר את שלב denoising באמצעות שימוש ב-Reparametrization Trick, נוכל לתארו באופן הבא:

$$x_{t-1} = \frac{1}{\sqrt{a_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{a}_t}} \epsilon_\theta(x_t, t) \right) + \sqrt{\beta_t} \epsilon$$

נוכל לשים לב כי ערך התמונה x_{t-1} מוכתבת על פי התמונה x_t וכמו כן ϵ מתארת את הפעולה המבוצעת באמצעות U-Net-U כפיעולות Denoising. כמו כן, ϵ מתארת רעש הנדגם כתוצאה מהתפלגות נורמלית רגילה.

סך הכל נוכל לתאר את ה-objective הכלל עבור הארכיטקטורה באופן הבא:

$$L = \mathbb{E}_{t, x_0, \epsilon} \| \epsilon - \epsilon_\theta(\sqrt{\bar{a}_t} x_0 + \sqrt{1 - \bar{a}_t} \epsilon, t) \|^2$$

מבחן מילולית נראה כי ϵ מתאר את הרעש הנדגם כתוצאה מהתפלגות נורמלית. $\sqrt{\bar{a}_t} x_0$ הוא ביטוי בו $\sqrt{\bar{a}_t}$ מתאר את פרמטר ההרעשה המורכב בסיסו β_t - x_0 מייצגת את התמונה ההתחלתית. $\sqrt{1 - \bar{a}_t} \epsilon$ מייצג את הרעש המתווסף כתוצאה מהתפלגות נורמלית כך שסך הכל עבור איטרציה t אנו רוצים לחזות את תוחלת הרעש שהתווסף בהשוויה לרעש שבאמת התווסף וביצע מינימיזציה על הביטוי הזה כך שהמודול יוכל להקליל את התפלגות הללו באופן בו denoising יוביל לניקוי מיטבי בשלב האימון וליצירה של תונות חדשות יש מאין בשלב h-Inference.

תיאור האלגוריתם

כעת נרצה לכנס את כל הדברים שנאמרו על מהות הארכיטקטורה לכדי Pseudo Code שמתמצאת את אופן הפעולה. ה-Pseudo Code שנציג יהיה עבור שלב האימון בו מלמדים את הרשת ללמידה רפרנציאיות עבור ניקוי הרעשים, ועבור שלב ה-Sampling שהוא השלב בה אנו מתחילה מהתמונה כלשהי מושפעת לחילוטין ומנסים להגיע לתמונה שיצרה הרשת שלמה את התפלגות הדאטא.

שלב האימון

בשלב זה נרצה לדגום תמונות מtower Batch שבתו כולם יציג דוגמה p.i. מtower הדאטאスト באופן בו בצורה איטרטיבית נדגים תמונה יחידה, נוסיף לה רעש וננסה באמצעות h-Net-U לחזות את הרעש שההוויס בכל איטרציה. למעשה כפ' שצין מטרת h-Net-U הוא לבצע Denoising והאופן בו אנו מלמדים לעשות זאת היא לחשוף אותו לכמויות גדולות של תמונות ולכמויות רבה של סוג רעש כך שכך הכל נקווה שהרשת תדע להקליל ולבצע את הורדת הרעש בצורה מספקת עבור מספר סוג תמונות עבור מספר סוג איטרציות ועבור מספר סוג רעשים.

Algorithm 1 Training

```

1: repeat
2:    $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ 
3:    $t \sim \text{Uniform}(\{1, \dots, T\})$ 
4:    $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
5:   Take gradient descent step on
        
$$\nabla_{\theta} \|\epsilon - \epsilon_{\theta}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, t)\|^2$$

6: until converged

```

1. נróż בולאה עבור כל התמונות ב-Batch או בדאטאסט
2. נדגם בצורה שרירותית α להיות תמונה ממנה נרצה להתחילה
3. נדגם מספר איטרציות כלשהו T מבין {1 ... T }
4. נדגם רעש מtower התפלגות גאוסינית שהוועה את הרעש אליו נרעיש את התמונה \mathbf{x}_0
5. על פ' ה-e-Objective נמדד loss בין הרעש שהוספנו לבין הרעש שאנו לומדים באמצעות שימוש ב-Net-U. לאחר חישוב loss נבצע נגזרת, ובאמצעות SGD ו-Backpropagation או כל אופטמייזר אחר נפערע את הגראדיינט לאורך הרשת.
6. נמשיך את סדרת הפעולות עד להתקנסות עבור כל התמונות ב-Batch או בדאטאסט

שלב ה-Sampling

- בשלב ה-Sampling אנו משתמשים פעמיים:
1. כאשר אנו רוצים לדגום תמונה מורעתת לגמרי x אנו רוצים לנகوت בשלב האימון ולהשווות את אופן ניקוי הרעש לעומת התמונה המקורי, למדוד loss ולעדכן את משקלות הרשת בהתאם לכך.
 2. בעת ביצוע Inference כאשר אנו מתחילה מרשע מוחלט ומבצעים סדרת Denoising לקבלת תמונה הלוקחה מתוך התפלגות.

אף על פי שהשלב משתמש לשני דברים שונים, האופן בו הוא מתבצע בכל אחד מהמרקם הוא זהה. בכל אחד מהמרקם אנו דוגמים תמונה מורעתת לגמרי x_T ולאחר מכן סדרה של צעדים אנו שואפים להגיע לתמונה בעלת משמעות x_0 .

Algorithm 2 Sampling

```

1:  $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 
2: for  $t = T, \dots, 1$  do
3:    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$  if  $t > 1$ , else  $\mathbf{z} = \mathbf{0}$ 
4:    $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left( \mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\alpha_t}} \epsilon_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$ 
5: end for
6: return  $\mathbf{x}_0$ 

```

1. דוגם תמונה מורעתת לגמרי x_T מתוך התפלגות נורמלית
2. נróż בולולאה עבור מספר האיטרציות שהוגדר T
3. דוגם רעש מתוך התפלגות נורמלית שייצג את הערך שאנו מוסיפים עבור השונות (כאמור מקבועה). נשים לב כי הדגימה מתבצעת עבור כל איטרציה $1 < t$ מהסיבה שעבור $t = 1$ אנו לא רוצים להוסיף רעש כי הדבר מושל לכך שכבר הגענו לתמונה שברצוננו להחזיר
4. באמצעות Reparameterization Trick נבצע השמה לערך של x_{t-1} כתלות בערך של x_t ובאמצעות הרפרזנטציות שנלמדו עד כה על ידי ϵ_θ
5. נמשיך בביצוע הפעולה לכל אחת מהאייטרציות $\{1 \dots T\}$
6. נבצע החזרה של x_0 המוגדרת להיות התמונה הנקייה ובעלת המשמעות הלוקחה מתוך ההתפלגות שנלמדה.

Stable Diffusion

[Video Link](#)

Stable Diffusion היא ארכיטקטורה מבוססת Diffusion שהתפרסמה בשנת 2022 על ידי מספר חוקרים מאוניברסיטת מינכן במאמר בשם:

High-Resolution Image Synthesis with Latent Diffusion Models
Latent Diffusion Model או LDM שעמוד עבור Latent Diffusion Model

לארכיטקטורה מספר חידושים משמעותיים ומשמעותיים עליהם נרחב בעמודים הקרובים אך החידוש המשמעותי ביותר הוא שאמם עד כה כאשר דיברנו על ביצוע תהליך הדיפיזיה כזזה המופעל על התמונה באופן ישיר, כתעת ישנה העברה של התמונה למרחב לטני ותהליך הדיפיזיה מתרחש למרחב הלטני. אחת היתרונות בשימוש בשיטה זו הוא חסכו עצום בזמן ריצה. בעוד שבתמונה יש לא מעט פיקסלים ולכן גודל התמונה יחסית גדול דבר שmobiel למכפלות מטירציאניות כבודות ומסובכות, המרחב הלטני קטן בהרבה ולכן החישובים מהירים יותר. כמו כן, חלק מהפיקסלים לא מהותיים ולא מבאים עימם אינפורמציה קונקרטית שמעידה על ההתפלגות ולכן דזוקא יציג הלטני קומפקטי וחזקק יותר.

כפי שתארו החוקרים במאמר המקורי, מרחב התמונה המקורי נקרא מרחב הפיקסלים בו באמת כל תמונה מרכיבת פיקסלים שככלותם מביאים לכדי ייצרת התמונה הכוללת.
באמצעות שימוש ב-Encoder המקבל כקלט את התמונה מרחב הפיקסל אנו מקבלים ייצוג לטני של התמונה באופן בו היציג הלטני מייצג את מהות התמונה על סמך הרפרנציות שלמד ה-Encoder לאורך תהליך האימון.

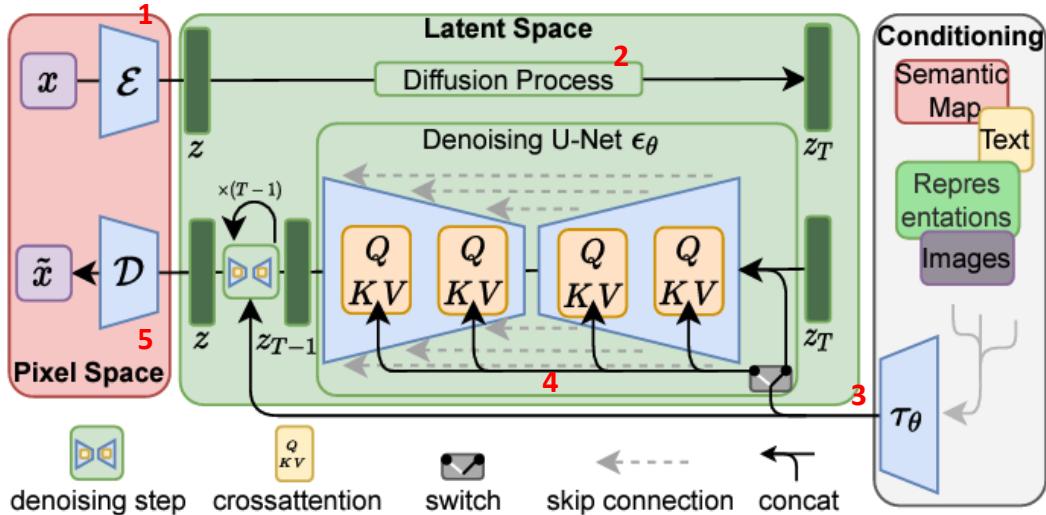
לאחר מכן, מתבצע תהליך של דיפיזיה ותהליכי נוספים שנייכנס אליום בהרחבה על היציג הלטני של התמונה ולבסוף היציג הלטני שעבר את המניפולציות הרבות לאורך הדרך, נכנס ל-Decoder שבתו ממיר את היציג הלטני בחזרה לתמונה למרחב הפיקסל.

צר הכל יכול לומר שהאררכיטקטורה מכילה הרבה אלמנטים וקומפוננטות נלמדות שדיברנו עליהם בהרחבה כבר לאורך הסיכום. למעשה אפשר לחשב על LDM כעל ארכיטקטורת על שבאמצעות הרכבה נכונה של קומפוננטות וארכיטקטורות ייחסית עליות מובילה לכדי למידה רפרנציות מדויקות שמובילות לכדי התוכנות.

כמו כן, אחד החידושים המהותיים ב-LDM היא האפשרות לייצר תמונות מתוך טקסט או מכל יציג Embedding כלשהו של המזויות. עד כה כאשר דיברנו על מודלי דיפיזיה כאמור הדיבור היה על כך שהרפרנציה הנלמדת שקופה להתפלגות הדאטאטס ולפיכך התמונות שיוכלו להיווצר בעת גינרטט תליות באופן בלעדי בדאטאטס שעלי נאמן את המודל. כאן ישנה אפשרות לכלול גם פרומפט שנייכנס כקלט עבור המודל ועל סמך רפרנציות משולבות שיילמדו על סמך התמונות ועל סמך הטקסט יוכל בעית Inference להכניס כקלט טקסט ולקבל תמונה מבוססת על הטקסט באופן בו המודל לוקח בחשבון את היציג הטקסטואלי כאשר הוא מייצר את התמונה.

תיאור הארכיטקטורה

נרצה להסתכל על תמונה שמתארת את רכיבי הארכיטקטורה:



1. E מייצג את הממיר את קלט התמונה x ממירחב הפיקסלים לטובות קלט מהמרחב הלטנטי z . כאמור הוא רכיב לומד הוליך רפרנסנציות בשלב האימון כך שמאוחר יותר בשלב ה-Inference ישנה המירה לייצוג לטנטי המזקק את מהות התמונה.
2. בשלב ה-Diffusion Process מתבצע תהליך הרעשה כפי שדווח שמתבצע במודל דיפיזיה. למשה מוגרל $\{T \dots 1\}$ שמנדר את מספר צעדי הרעשה הנילקה מתוך התפלגות גאוסיינית. מבחינה נוטיצה אנו מתחילה מוקטור הייצוג הלטנטי z ולאחר צעד הרעשה אנו מקבלים את הווקטור z_T שמהווה את הייצוג הלטנטי המורעש לגמר.
3. שלב זה הוא שלב Conditioning בו אנו נתקלים בפעם הראשונה. למשה אחד החידושים במודל Stable Diffusion היא היכולת לבצע אינוקרפציה למידע נוסף אנו רוצים שהמודל ילמד בשלב האימון וישפיע על הפרדי-קיציות בתהליכי ההසקה. הקלט ל-Conditioning יכול להיות כל קלט שנלך מכל התפלגות מסוימת למשל קלט תМОונתי, סמנטי, סיגנאלי כאשר הדוגמא האינטואיטיבית ביותר בשלב התפתחויות בשנים האחרונות הוא טקסט המאפשר ייצור של תמונות מתוך טקסט כמו שרווח במודלים כמו DALL-E ואחרים.
4. לאחר ביצוע של Preprocess על קלט ה-Conditioning ניתן לראות כי יש מכפלה שלו בסרט משקולות נלמד θ ולאחר מכן יש שילוב של המידע הזה בתהליך Denoising-U-Net ביחד עם הווקטורים הלטנטיים המייצגים את מהות התמונה המקורית.
5. בתהליך Denoising-U-Net ניתן לבדוק אם עושים שימוש ב-Attention-Net-U כפי שמקובל לעשות במודל דיפיזיה אך חידוש נוסף שנitin לראות כאן הוא שימוש ב-Attention. למעשה התהליך חוזר במשך T פעמים באופן בו בכל פעם אנו מתקדים אל עבר ייצוג לטנטי מורעש פחות.
6. D מייצג את ה-Decoder הממיר את הקלט מהמרחב הלטנטי z חוזרת אל תמונה ממירחב הפיקסל \tilde{x} המייצגת את התמונה לאחר ה-conditioning כפי שהמודל לימד.

Encoder

כפי שכבר דובר בארכיטקטורות קודמות גם בארכיטקטורה הנוכחית מטרת ה-Encoder היא להמיר את הקלט המקורי ממרחב הפיקסלים אל המרחב הלטני. כאמור, ההמרה מתבצעת לוטבת עילוות חיובית באופן בו למעשה יש זיקוק של המידע ממרחב הפיקסלים אל עבר תצורה מצומצמת יותר שמקפקת את המהות בדאטא. מביננה פורמלית נרצה להגדיר את אופן פעולה ה-Encoder באופן הבא:

$$x \in \mathbb{R}^{H \times W \times 3} \rightarrow \mathcal{E}(x) = z \in \mathbb{R}^{h \times w \times c}$$

ניתן לראות כי אנו מתחילה במרחב הפיקסלים עם תמונה בגודל $\mathbb{R}^{H \times W \times 3}$ ולאחר הזיקוק באמצעות ה-Encoder אנו מקבלים פלט מהמרחב הלטני בגודל $\mathbb{R}^{h \times w \times c}$. חשוב לציין כי היחס בין h/H כמו כן היחס בין w/W מוגדר להיות $N \in m : f = 2^m$.

Decoder

עבור ה-Decoder נרצה לבצע את הפעולה ההפוכה לפעולות ה-Encoder באופן בו נרצה לקחת כל דגימה מהמרחב הלטני הכלול את השינויים שנגרמו כתוצאה מהפרצנטציות שנלמדו על ידי המודל בקומפוננטות השונות ולהמיר אותה לדגימה ממרחב הפיקסלים. מביננה פורמלית ההמרה על ידי ה-Decoder מוגדרת באופן הבא:

$$\mathcal{E}(x) = z \in \mathbb{R}^{h \times w \times c} \rightarrow \mathcal{D}(z) = \tilde{x} \in \mathbb{R}^{H \times W \times 3}$$

Encoder – Decoder Objective

ה-Objective של Encoder-Decoder דומה מאוד לזה של VAE והוא חלק מ-VAE גודל יותר עבור ה-Autoencoder-Stable Diffusion-Shitumar בהמשך. עם זאת, ה-Objective של Encoder-Decoder מוגדר להיות כזה הכלול בתוכו ריכיב המתאר את חלק ה-Reconstruction Loss- מצד אחד ומצד שני מגדר קירוב התפלגות הדגימות מהמרחב הלטני המתקבלות כתוצר ה-Encoder אל התפלגות נורמלית. פורמלית:

$$L_{rec}(x, \mathcal{D}(\mathcal{E}(x))) = MSE(x, \mathcal{D}(\mathcal{E}(x))) = MSE(x, \tilde{x})$$

$$L_{reg}(x; \mathcal{E}, \mathcal{D}) = KL\left((\mathcal{E}(x)), \mathcal{N}(0, I)\right)$$

נוכל לראות כי L_{rec} מייצג את ה-Reconstruction Loss בין התמונה המקורית לבין התמונה שהתקבלה לאחר תהיליך ייצור התמונה. במקרה זה מוגדרת להיות MSE בין שתי התמונות אך בפועל יכולה להיות מיוצגת על ידי כל מטרייקה אחרת המסוגלת למדוד מרחק בין שתי התמונות. מצד שני, L_{reg} מוגדרת להיות קירוב בין התפלגות המרחב הלטני לבין התפלגות נורמלית כפי שיכלנו לראות ב-VAE. בפועל קירוב התפלגות מוביל לכך למידת פרצנטציות טובות יותר של המודל בתהיליך האימון, ויכולת דגימה טובה יותר מתווך רעש אסיאני בתהיליך ה-Inference.

תהליך ה-Diffusion (הרעשה)

כפי שדיברנו בעבר מודל דיפיזיה רגילים, תהליך הרעשה מוגדר באופן הבא:

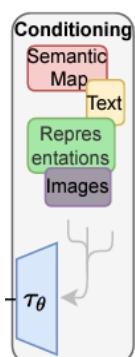
$$q(z_t | z_{t-1}) = \mathcal{N}(z_t, \sqrt{1 - \beta_t} z_{t-1}, \beta_t I)$$

ונכל לראות כי השוני המהוות ממודל הדיפיזיה המקורי היא שכן הרעשה מתבצעת על הייצוגים מהמרחב הלטנטי ולא על ההיצוגים ממרחב הפיקסלים.

לטובות התזכרות, נציין כי בכלל פעם נדגום עבור כל פיקסל (מהמרחב הלטנטי) רעש מהתפלגות אגוסינית המוגדרת (σ^2, μ) \mathcal{N} כך שבইנטן ייצוג לטנטי z_t הייצוג הלטנטי z_{t+1} יהיה מושפע מעט יותר. במשווה, z_t מייצג את ה-Output, $\sqrt{1 - \beta_t} z_{t-1} + \beta_t z_{t-1}$ מייצג את ערך ממוצע התפלגות $-I_t$ מייצג את שונות התפלגות. כמו כן, נשים לב כי β הוא פרמטר שמשתנה עבור כל אחת מהאייטרציות. הערך ההתחלתי $-\beta$ מקבל הוא קטן והערך הסופי $-\beta$ מקבל הוא גדול באופן יחסית. דבר המוביל לכך שתחילה הרעשה מתחילה ממוצע התפלגות גבוהה יחסית ושונות קטנה יחסית ככל מרעה מסובבית ודומה באופן יחסית בין כל אחד מהפיקסלים. לעומת זאת, ככל שמתקדמיים באיטרציות הרעשה נלקחת מתוך התפלגות עם ממוצע קטן ושונות גדולה ככל מרעה יותר חלה אר-עם זאת יותר סטוכסית.

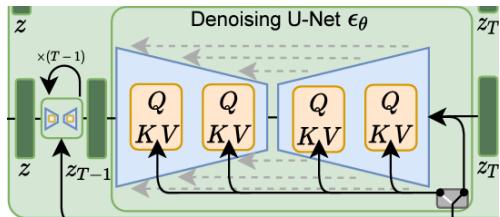
תהליך ה-Conditioning

כאמור, תהליך Conditioning הוא תהליך בו אנו מבאים לכך רפרזנטציה איחידה את המידע הנוסף על מנת שיוכל להשתלב ביחיד עם הייצוג הלטנטי המתאר את התמונה. על המידע האחד הזה נרצה בשלב ה-Denoising לבצע ניקוי כך שהפלט שיתקבל יתכלל בתוכו את הניקוי שהתקבל גם מעיבוד המידע הנוסף.



לטובות עיבוד המידע המקדמים אנו עושים שימוש ב- τ_θ המהווה קומפוננטה לומדת או קומפוננטה שעבירה כבר Pre-Train במידה והמידע הנוסף שלנו הוא מידע עבורו כבר נלמדו רפרזנטציות על ידי מודל כלשהו. למשל עבור טקסט, נוכל לify אט τ_θ באמצעות GPT באופן בו ניקח את ייצוג המשפט שנלקח כתוצאה מניפויים על הייצוגים הוקטוריים הנלקחים מתוך ה-Embedding Matrix לטובות קבלת ייצוג מזוקק של המידע. במידה והמידע הנוסף הוא לא מידע שכיח ואין ארכיטקטורת Train Pre-Train שיכולה לספק לנו ייצוגים וקטוריים מסוימים, נרצה ליצור אותם בעצמינו באמצעות שימוש בקומפוננטות המסוגלות לייצר ייצוגים וקטוריים כאלה כמו MLP, CNN וארქיטקטורות נוספות מבוססות למידה עמוקה המסוגלות לספק לנו Hidden Representation בדמות וקטור המציג את המידע הנוסף בתוצוגה וקטוריית מספרית המגלמת בתוכה את מהות המידע.

תהליך Denoising

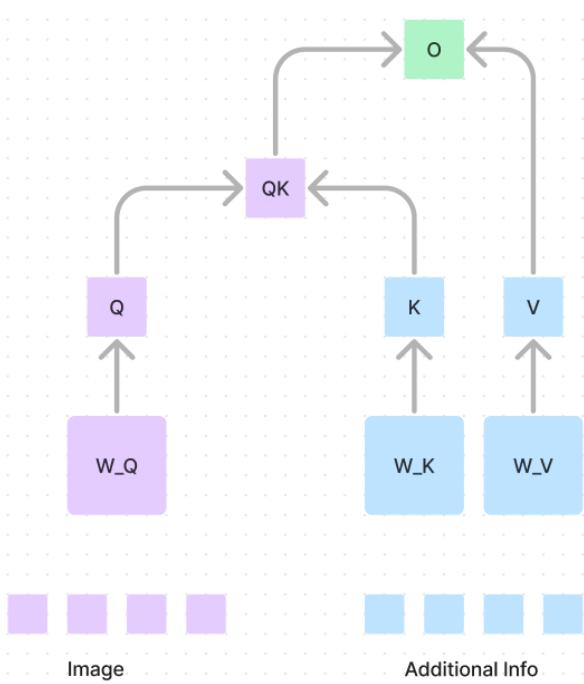


מהתבוננות בתהליך Denoising ב-h-denoising ניתן לראות כי הוא מורכב משני תהליכי עיקריים השזורים זה בזה:
 1. Cross Attention.
 2. U-Net.
 השימוש ב-U-Net היה כבר מוכר לנו ממודלי דיפיזיה רגילים אך-cut ניתן לראות כי ישנו שימוש גם ב-Cross Attention לטובות למידת רפרנציאיות נרחבות יותר המגלמות גם את התלות של הייצוג הלטנטי במידע הנוסף המגיע מה-Conditioning.

מפהת החלוקה הסמנטית נרצה גם לתאר את התהליך כאשר נתיחס לביצוע denoising. ולאחר מכן נראה איך הוא משתלב ביחד עם ה-U-Net.

התהליך שיתואר עבור Cross Attention ועבור denoising הוא התהליך המתבצע בכל איטרציה בעת denoising. למשהו כל שלב, יש אינקורפרציה של המידע המגיע מתוך הייצוג הלטנטי של התמונה בין Info וה-U-Net.

Cross Attention



מבחינת Cross Attention ניתן להבחין כי הקלט מחולק:

1. Image Embedding Representation
2. Additional Info

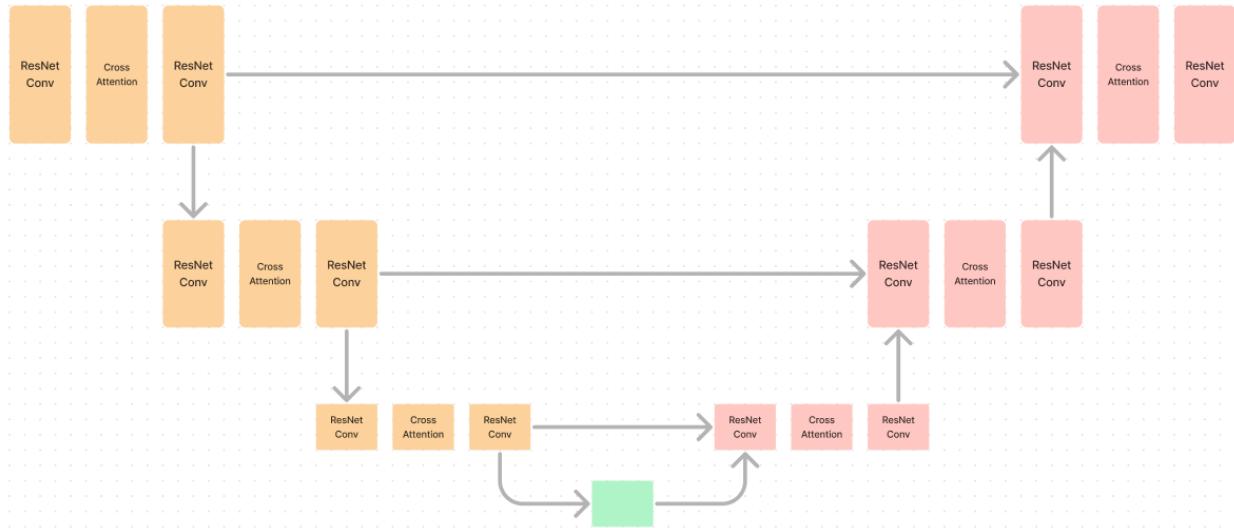
ניתן לראות כי התמונה עוברת Embedding לטובות קבלת ייצוגים וקטוריים ולאחר מכן עוברת מכפלה עם סט משקولات נלמד לקבלת Q. לצורך פשטות ההסביר נניח כי Additional Info הוא מידע טקטים או שער Embedding לקבלת ייצוגים וקטוריים. הטקסט בתורו עובר מכפלה עם סטי משקولات נלמדים לקבלת V, K. לאחר מכן מתבצעת מכפלה בין Q-L-K כיאה למנגנון Cross Attention ולאחר מכן מכפלה עם V לקבלת הייצוג שמתאר את פלט שכבת denoising.

נשים לב כי למעשה באופן ביצוע הפעולה, אנו מבצעים אינקורפרציה בין Additional Info לבין הייצוג הלטנטי של התמונה. באופן זה אנו קושרים בין השניים ומאפשרים למודל למדוד רפרנציאיות משותפות עבור שניהם.

חשוב לציין כי המידע המגיע מתוך Additional Info הקלט המקורי שעובר מניפולציה על ידי ϵ_θ בשלב ההתחלתי ובכל פעם חדש נכנס קלט אל סטי המשקولات הנלמדים W_V, W_K, W_Q . לעומת זאת, הקלט ל- W_Q הוא בכל פעם הפלט של השכבה הקודמת ב-U-Net שעובר אל עבר השכבה הנוכחית.

U-Net

במהלך הבדיקה, ניתן לראות את אופן ביצוע ה-U-Net-U המגלם בתוכו את ה-Cross Attention ארכיטקטוריית. ניתן לראות את אופן ביצוע ה-U-Net-U המגלם בתוכו את ה-Cross Attention ארכיטקטוריית.



לאחר מכן, כפי שתואר גם בארכיטקטורת h-Net-U המקורית ישנה קונבולוציה נוספת שמובילה להקטנה נוספת של המימדים. למשה תיאור זה מהוות תיאור עבור שכבה יחידת h-Net-U אך יש לציין כי אופי הפעולה הוא חזרתי בין השכבות באופן בו הקונבולוציות וה-*Cross Attention* מתרחשות מספר פעמים ולא פעם אחת.

חשוב לציין כי כפי שתואר בארכיטקטורה המקורי, גם כאן יש skip connections שמטרתם להוביל להחלה הגדודית ולהוביל לכך הכללה טובה יותר של המודל על ידי הימנעות ממיינימוםים לוקאלים לאורור תחילי האימון.

סך הכל נראה כי פלט h-Net-U מתאר איטרציה אחת של LDM בו מתבצעת Denoising בודדת לקבלת ייצוג לטני מורעש פחות המתקרב מעט אל עבר מה שייהי בסופו של דבר התמונה הסופית. התחילה היזה חוזר בזרה איטרטיבית במסך T פעמים עד לכדי קבלת ייצוג לטני של התמונה המקורית שנכננו אל ה-Decoder לטובת קבלת התמונה הסופית המציגת את התמונה המג'ונרטת.

Stable Diffusion Objective

ניתן לחוש על ארכיטקטורת Stable Diffusion כחזו המרכיבת למשה מ-2 ארכיטקטורות – הראשונה היא-h Autoencoder בה אנו ממיררים ממרחב הפיקסלים למרחב הלטני ולהפוך, השנייה היא פעולה הדיפיזיה וה-Denoising. לכן, הולכה למשה אנו מאמנים את המודל ב-2 חלקים באופן בו החלק הראשון מתיחס לאימון ה-Autoencoder בנפרד והחלק השני מתייחס לאימון מודל הדיפיזיה בנפרד. לכן, כתעת נרצה לפרט את 2 ה-Objectives ונתחיל ב-objective של ה-Autoencoder.

Autoencoder Objective

ה-objective של ה-Autoencoder מורכב ממספר קומפוננטות שכלי אחד בתורה מתארת רצון ללמידה אספקט ייחודי שיביל את המודל ללמידה פרזנטציית טבות יותר. נרצה לתאר מבחינה פורמלית את ה-objective של הארכיטקטורה ולאחר מכן לתאר כל רכיב בנפרד ובצורה עצמאית. פורמלית ה-objective מוגדר באופן הבא:

$$L_{\text{Autoencoder}} = \min_{\mathcal{E}, \mathcal{D}} \max_{\psi} \left(L_{\text{rec}} \left(x, \mathcal{D}(\mathcal{E}(x)) \right) - L_{\text{adv}} \left(\mathcal{D}(\mathcal{E}(x)) \right) + \log D_{\psi}(x) + L_{\text{reg}}(x; \mathcal{E}, \mathcal{D}) \right)$$

רכיב 1 – $L_{\text{rec}} \left(x, \mathcal{D}(\mathcal{E}(x)) \right)$

רכיב המתאר את ה-Reconstruction Loss בין התמונה המקורית לבין התמונה שייצרת על ידי המודל ומוגדר באופן הבא:

$$L_{\text{rec}} \left(x, \mathcal{D}(\mathcal{E}(x)) \right) = \text{MSE} \left(x, \mathcal{D}(\mathcal{E}(x)) \right) = \text{MSE}(x, \tilde{x})$$

ונכל לראות כי במקרה זה מוגדרת להיות MSE בין שתי התמונות אך בפועל יכולה להיות מייצגת על ידי כל מטריקה אחרת המסוגלת למדוד מרחק בין שתי התמונות. מטריקה נוספת נספפת נפוצה להערכת ה-PSNR היא Reconstruction Loss המוגדרת באופן הבא:

$$\text{PSNR} = 20 \cdot \log_{10}(MAX_i) - 10 \cdot \log_{10}(\text{MSE})$$

ונכל לראות כי למעשה גם ב-PSNR אנו מבצעים שימוש ב-MSE שמודד את המרחק בין שתי התמונות. ערך PSNR גבוה יותר מעיד על שונות קטנת יותר וההתאמה גדולה יותר בין שתי התמונות.

רכיב 2 – $L_{\text{reg}}(x; \mathcal{E}, \mathcal{D})$

רכיב נוסף הוא הרכיב שמטרתו לבצע קירוב התפלגיות בין התפלגות הוקטור הלטני לבין התפלגות נורמלית. פורמלית נראה כי הרכיב מתואר באופן הבא:

$$L_{\text{reg}}(x; \mathcal{E}, \mathcal{D}) = \text{KL} \left((\mathcal{E}(x)), \mathcal{N}(0, I) \right)$$

בפועל קירוב ההתפלגיות מוביל לכך למידת פרצנטזיות טובות יותר של המודל בתהיליך האימון, ויכולת דגימה טובה יותר מتوزר רעש גאוסייני בתהיליך ה-Inference.

רכיב 3 - Adversarial

ב-Objective של Autoencoder ישנים 2 רכיבים נוספים המוגדרים להיות המתארים את האופי ה-Adversarial של הארכיטקטורה. כפי שדיברנו ב-GAN, גם כאן יש שימוש בגינרטור ובדיסקרימינטור באופן בו ה-Loss המוגדר הוא Adversarial. כך שהגינרטור מטרתו לייצר דוגמאות הדומות ככל הניתן לדוגמאות שנראו/cailed נלקחו מההתפלגות האמיתית ואילו תפקיד הדיסקרימינטור הוא להזות זאת. הגינרטור במקורה שלנו מוגדר על ידי $D(\mathcal{E}(x)) = G$ כך שהגינרטור למעשה מיצג על ידי ה-Encoder ה-Decoder בividamente מהוים תחת יחידה פונקציונלית היודעת בהינתן x לייצר באופן גינרטיבי \hat{x} .

רכיב 3 – $\log D_\psi(x)$

כאשר דיברנו על GAN, תיארנו את ה-Objective של הדיסקרימינטור כ- Cross Entropy באופן בו אנו למשה מתמודדים עם בעיית קולסיפיקציה ביןארית. מבחינה פורמלית ניתן לתאר באופן הבא את ה-Loss עבור הדיסקרימינטור:

$$\text{Cross Entropy} = L(\hat{y}, y) = [y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

$$\text{Discriminator Objective} = \max_D \log(D(x)) + \log(1 - D(G(x)))$$

סך הכל נראה כי אנו רוצים ב-Objective של הדיסקרימינטור למקסם את הניראות של הדאטא כתלות בדאטא האמיתית ובדאטא המג'ונרטט. אנו רוצים שהדיסקרימינטור יהיה מiomן ביותר וידע לומר נכון הדוגמאות שנלקחו ישרות מההתפלגות האמיתית הן אמיתיות וכל הדוגמאות שייצרו על ידי הגינרטור מזויפות. (להעמקה ניתן לחזור לעמוד 19 בסיכום).

רכיב 4 – $L_{adv}(\mathcal{D}(\mathcal{E}(x)))$

רכיב זה מהווה את ה-Loss המוגדר על הגינרטור ופורמלית מוגדר באופן הבא:

$$\text{Cross Entropy} = L(\hat{y}, y) = [y \log \hat{y} + (1 - y) \log (1 - \hat{y})]$$

$$\min_G \log(D(x)) + \log(1 - D(G(x)))$$

גם כאן ניתן לראות כי אנו מדברים על Cross Entropy רק שהפעם מتوزר הסתכילות על כיוון הגינרטור נוכל לראות כי דוקא מטרת הגינרטור היא להתבל בדיסקרימינטור וליצור תമונות כל כך טובות עד שהדיסקרימינטור, מiomן ככל שהיא, יתבלבל ויחשוב שמקורן של התמונות בהתפלגות האמיתית ולא בהתפלגות שמקורתה על ידי הגינרטור.

LDM Loss

כאשר דיברנו על ה-Loss של מודלי דיפיזיה רגילים (שאינם לטנטיטים) הגדרנו את תוחלת ה-Loss ואת ה-objective כזאת המוגדר באופן הבא:

$$L_{DM} = \mathbb{E}_{x, \epsilon \sim \mathcal{N}(0, I), t} \| \epsilon - \epsilon_\theta(x_t, t) \|_2^2$$

נשים לב כי כפי שתואר, כאן ב-DM רגילים ה-objective מוגדר מעל למרחב הפיקסלים באופן בכלל איטרציה בתהיליך denoising אנו מנסים לחזות את הרעש התווסף במהלך ההרעשה.

עם זאת, כאשר אנו מדברים על LDM אנו רוצים שה-objective יהיה כזו המופעל על המרחב הלטנטי ולא על מרחב הפיקסלים ולכן פורמלית נרצה להגדירו באופן הבא:

$$L_{LDM} = \mathbb{E}_{\mathcal{E}(x), \epsilon \sim \mathcal{N}(0, I), t} \| \epsilon - \epsilon_\theta(z_t, t) \|_2^2$$

ונכל להבחין כי בעת ה-objective היא לא על דגימה הלקוחה מתוך מרחב הפיקסלים x אלא על דגימה הלקוחה מתוך המרחב הלטנטי $z = (x)$. הסיבה לכך היא שכי שתוואר גם ההרעשה מצד שני בתחום denoising לא מתבצע מעל מרחב הפיקסלים אלא מעל המרחב הלטנטי.

הדבר האחרון שנרצה לכלול ב-objective שאם יתאר באופן סופי את ה-objective Conditioning-over-arcticuttoura הוא את האלמנט שככלל את ה-loss ביחסוב ה-objective. פורמלית מתואר באופן הבא:

$$L_{LDM} (\text{with conditioning}) = \mathbb{E}_{\mathcal{E}(x), \epsilon \sim \mathcal{N}(0, I), t} \| \epsilon - \epsilon_\theta(z_t, t, \tau_\theta) \|_2^2$$

ונכל לראות כי במשוואת ה-objective אנו כוללים גם רכיב המתאר את τ_θ . הסיבה לכך היא ש- τ_θ היא רשת לומדת שמהווה ארכיטקטורה שלמה בפני עצמה שאליה נרצה גם כן לדוחוף גראדיינטים בעת ביצוע תהליך הלמידה לטובת עדכון המשקولات.

סוף דבר ה-objective

از כי שתוואר, למשה ה-objective של LDM מוגדרת על ידי 2 Objectives שונים המתארים את שתי תתי הארכיטקטורות שמשלים אחת את השניה לכדי ארכיטקטורת על אחת. כי שצוי שני הארכיטקטורות הללו המורכבות את המודל מתאמנות בנפרד באופן בו הרפרזנטציות הנלמדות על ידיהן מוכתבות בצורה נפרדת על סמך תהליכי אימון נפרדים.