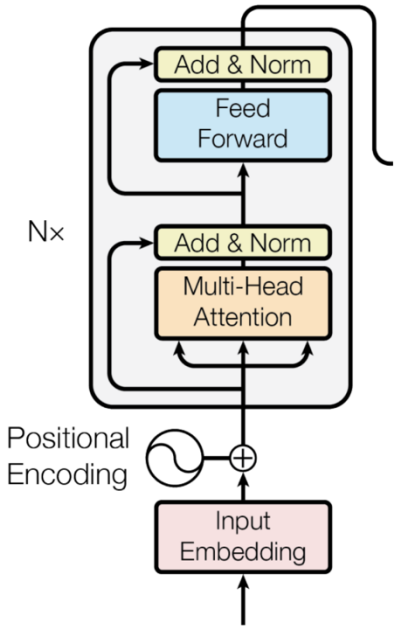
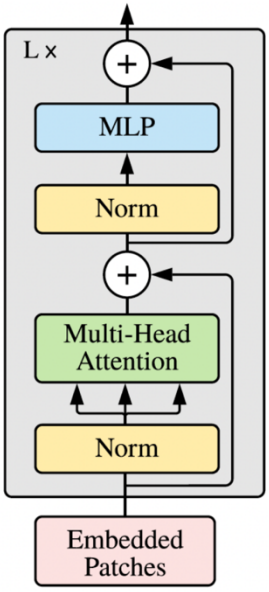
נושאים רלוונטיים לקראת מימוש המודל

Attention, Self-Attention and Transformers





הרעיון הכללי [1]

הארכיטקטורה של ViTPose משתמשת בבלוק מיוחד הנקרא טרנספורמר (באנגלית: Vision Transformer). על מנת להסביר איך הטרנספורמר עובד קודם נצטרך להסביר מה זה Attention ובפרט Self-Attention.

הרעיון של מנגנון הAttention (או בעברית: תשומת לב) הוצע לראשונה עבור משימות שהקלט שלהן הוא סדרה של קלטים באורך משתנה (למשל – משפטים באנגלית, כאשר מספר המילים במשפט אינו קבוע). מנגנון ה-Attention מאפשר למודל לבחור להתייחס באופן דינמי לחלקים ספציפיים בסדרת הקלטים ולהתעלם מאחרים. בעזרת מנגנון זה, המודל יכול להתמקד באופן יעיל במידע החשוב ביותר עבור המשימה, ולהתעלם ממידע פחות רלוונטי.

במילים אחרות, מנגנון ה-Attention מאפשר למודל לבחור מתי להתייחס לכל קלט בסדרת הקלטים ולא על כולם בבת אחת. זהו כלי עוצמתי המאפשר למודל להבין ולפענח את המידע בסדרת הקלטים באופן מדויק וברמה גבוהה יותר. המנגנון מאפשר למודלים להשיג תוצאות מדויקות יותר ולשפר את ביצועיהם.

דוגמה לשימוש ב-Attention יכולה להיות, למשל, במודל שמטרתו לתרגם משפטים מאנגלית לספרדית. המודל יקבל כקלט סדרה של מילים באנגלית ויפיק סדרה של מילים בספרדית המהוות את תרגום המשפט. ה-Attention יוכל בעת הפקה של כל מילה בספרדית, לבחור לאילו מן הקלטים להקדיש יותר תשומת לב ולאילו פחות. למשל,

בתרגום סדרת הקלטים "[we] [are] [eating] [bread]" לסדרת הפלטים "[estamos] [comiendo] [pan]", המודל יבחר (כנראה) להקדיש יותר תשומת לב לשתי הקלטים הראשונים בסדרה ([we], [are]) בעת הפקת המילה [estamos], ולהקדיש יותר תשומת לב לקלט האחרון בסדרה ([bread]) בעת הפקת המילה [pan].

חשוב להדגיש כי הרעיון ב-Attention הוא שהמודל <u>ילמד בעצמו</u> כיצד לבחור לאילו קלטים עליו להקדיש תשומת לב. אנו איננו עוזרים למודל ומגלים לו מתי הוא צריך להקדיש תשומת לב לכל קלט בסדרה – היכולת של המודל לדעת על איזה קלטים הוא צריך להסתכל ומתי, נלמדת כחלק מתהליך האימון של המודל.

נראה צורה בסיסית של מנגנון ה-Attention ולאחר מכן נכליל אותו לSelf-Attention ומשם לTransformers. נשים לב שה-Attention לא כבול רק למודלים שהקלט שלהם הוא סדרה. אפשר לקחת את הרעיון גם למשימות ראייה ממוחשבת, כאשר המודל יבחר על אילו חלקים בתמונה להסתכל ומתי ולפי זה להפיק את הקלט.

---

[1] ההסברים מבוססים על קורס "למידה עמוקה לראייה ממוחשבת" של אוני' מישיגן. ניתן למצוא את הרצאות הקורס בקישור הזה.

הקדמה – מודל בסיסי המשתמש ב-Attention

נניח שאנו בונים מודל המקבל תמונה ומפיק תיאור מילולי (caption) של התמונה. אזי למשל, אם נכניס למודל תמונה של חתול יושב, הוא יוצא כפלט את סדרת המילים "cat sitting outside". ניעזר ב-Attention על מנת לבנות מודל כזה.

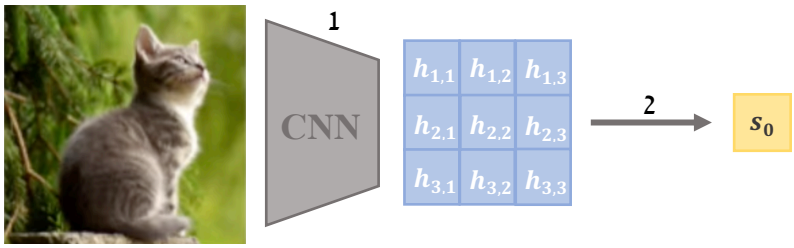
<u>הרעיון</u>: המודל ייקח את התמונה ויפיק ממנה גריד 3×3 של *feature vectors* אותם נסמן $\{h_{i,j}\}_{i,j=1}^{3}$ שמטרתם "לסכם" את התמונה – כאשר כל ווקטור מתאים לחלק שונה בתמונת הקלט. לאחר מכן, נרצה להפיק את הפלט של המודל (תיאור מילולי של התמונה) מילה אחר מילה. עבור כל מילה בנפרד, המודל יבחר בעצמו לאיזה ווקטורים מבין ה-feature vectors עליו להקדיש יותר תשומת לב. לאחר שבחר, הוא יפיק את המילה על סמך ה-feature vectors שבחר ויעבור להפקת המילה הבאה.

למשל, ככל הנראה שבעת הפקת המילה [cat] נבחר לתת יותר תשומת לב לfeature vectors שמתאימים לחתול. ובהפקת המילה [outside] נבחר לתת יותר תשומת לב לfeature vectors שמתאימים לרקע התמונה.

על סמך מה המודל יבחר לאיזה feature vectors עליו להקדיש תשומת לב בהפקת המילה הנוכחית?
נרצה להגדיר מעין פונקציה $f_{attention}$ המקבלת feature vector ומחזירה סקלר שמייצג עד כמה הווקטור הזה חשוב להפקת המילה הנוכחית (כמה תשומת לב צריך להקדיש לאותו feature vector). כך, נוכל להפעיל את פונקציה זו על כל feature vector.

אמנם, לא נרצה להשתמש באותה הפונקציה עבור כל שלב בהפקת הפלט, כיוון שכאמור כנראה שהמודל ירצה להיות מסוגל להקדיש תשומת לב לווקטורים שונים בעת הפקת מילים שונות בפלט. לכן, אנו נרצה להביא לפונקציה הזאת קלט נוסף אותו נסמן s_t, מלבד הfeature vector. הווקטור s_t נקרא *וקטור מצב פנימי* שמתיימר לשמור בתוכו את "המצב הפנימי" של המודל, ואת המילים שהפיק עד כה. כך, $f_{attention}(s_t, h_{i,j})$ מחזיר סקלר המייצג כמה תשומת לב צריך לתת ל$h_{i,j}$ בעת הפקת המילה הt.

נדגיש, כי וקטור זה לא באמת שומר את המצב הפנימי בצורה שאנחנו בוחרים בראש (למשל מערך של המילים שהפקנו עד כה). הרעיון הוא שהמודל יבחר בעצמו, בתהליך האימון כיצד להפיק את וקטור המצב הבא בכל פעם באופן שיאפשר לו להבין איפה הוא כרגע בתהליך הפקת הפלט, ולאילו feature vectors עליו להקדיש כעת תשומת לב. כיצד המודל יוכל לבחור דבר כזה בעצמו? למשל נגדיר לו פונקציה $f_W(s_t) = s_{t+1}$ כאשר W זה סט של פרמטרים שהפונקציה תלויה בהם. פרמטרים אלו ילמדו כחלק מתהליך האימון. בחירת f תלויה במימוש ולא נכנס אליה. אמנם נראה שאכן המצב הפנימי הנוכחי תלוי בכל המצבים הפנימיים שהיו עד כה, ולכן נחשוב על וקטור זה כמעין "הזיכרון הפנימי" של המודל.



השלב הראשון: הפקת feature vectors וקטור המצב הפנימי ההתחלתי

נראה כי בעזרת s_0 נבחר לאילו feature vectors לתת תשומת לב, ובעזרת הווקטורים שבחרנו לתת להם תשומת לב, נפיק את המילה הראשונה בפלט. לאחר מכן, בעזרת s_0 נפיק את המצב הפנימי הבא - s_1. בעזרתו נבחר לאילו feature vectors לתת תשומת לב הפעם, ונפיק את המילה השנייה בפלט. ואז נפיק את הווקטור הפנימי הבא - s_2 וכן הלאה.

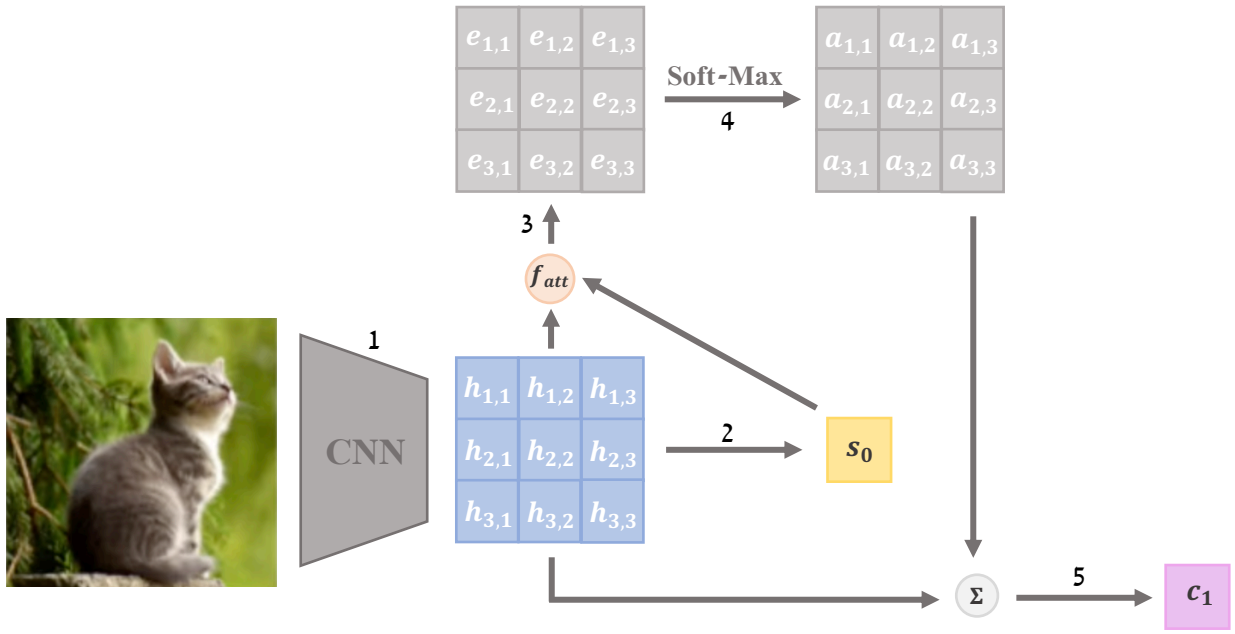
באופן פורמלי, תהיה לנו פונקציה $e_{i,j}^t = f_{attention}(s_t, h_{i,j})$ אשר מקבלת את המצב הפנימי הנוכחי, ו-feature vector יחיד, ומפיקה סקלר $e_{i,j}^t$. הסקלר בעצם מייצג את כמות תשומת הלב שיש להקדיש ל$h_{i,j}$, בעת הפקת המילה הt.

נפעיל את $f_{attention}$ עם s_0 ועם כל $h_{i,j}$ בנפרד. נקבל 9 ערכי $e_{i,j}^0$ הנקראים *Alignment Scores*.
ערכים אלו יכולים להיות כל מספר ממשי, נרצה לנרמל אותם להתפלגות, כלומר למספרים חיוביים שסכומם הוא 1. נעשה זאת על ידי פונקציית soft-max המקבלת את תשע המספרים $\{e_{1,1}^0, \dots, e_{3,3}^0\}$ ומחזירה סקלרים $\{a_{1,1}^0, \dots, a_{3,3}^0\} \subseteq [0,1]$ שסכומם 1, הנקראים *משקלי Attention*.

משקלי ה-Attention בעצם אומרים כמה תשומת לב עלינו להקדיש לכל ווקטור. למשל, אם $a_{1,2}^0$ הוא גבוה (למשל 0.95), המשמעות היא שהמודל ייתן תשומת לב רבה לווקטור $h_{1,2}$ בעת הפקת המילה הראשונה של הפלט. אם $a_{3,1}^0$ שווה 0, המשמעות היא שהמודל יתעלם לחלוטין מ$h_{3,1}$ בעת הפקת המילה הראשונה.

המשמעות המתמטית של "ייתן תשומת לב רבה/מועטה" באה לידי ביטוי כעת: ניקח ממוצע ממושקל של $\{h_{i,j}\}$ בהתאם למשקלי ה-Attention שלהם – נפיק את $c_1 = \sum_{i,j\in\{1,2,3\}} a_{ij}^0 \cdot h_{ij}$ הנקרא *וקטור הקשר*.

בעזרת המצב הפנימי s_t, יחד עם וקטור ההקשר c_t, נפיק את וקטור המצב הפנימי הבא s_{t+1}. ובעזרתו, את המילה הבאה בפלט y_t בפלט. וכך שוב: בעזרת המצב הפנימי s_{t+1} נפיק את וקטור ההקשר הבא c_{t+1} וכן הלאה. כך, עד שהמודל יבחר להוציא את המילה $y_t = [END]$ (מילה שמורה) ונסיים. לא נכנס בפירוט לאופן הפקת המילים משום שזה חורג משלב הAttention של המודל ופחות רלוונטי עבורנו.



השלב השני: שימוש במצב הפנימי הנוכחי כדי לבחור משקלי Attention לכל feature vector, ולהפיק את ווקטור ההקשר הבא.

מנגנון Attention חדש

בעולם מדעי המחשב, כאשר מוצאים רעיון שעובד טוב, הדבר הבא שננסה יהיה לנסות להכליל אותו ולהנגיש אותו למשימות נוספות, וזה מה שנעשה כאן.

נתאר כעת מנגנון Attention שלא נועד בהכרח למודלים שהפלט שלהם הוא סדרה (כמו בדוגמא שראינו עד כה). כלומר לא יהיה שימוש בוקטורי המצב הפנימי $s_0, s_1, \ldots$.

שלב ה-Attention יקבל כקלט את וקטורי הקלט אותם נסמן להיות $X \in \mathbb{R}^{N_X \times D_X}$ (כלומר N_X וקטורי קלט ממימד D_X כל אחד – אנו מסתכלים על השורות ולא על העמודות!). וקטורי הקלט יכולים להיות למשל feature vectors בדומה למנגנון הקודם. ובנוסף יקבל כקלט את *וקטורי השאילתה* שנסמן אותם להיות $Q \in \mathbb{R}^{N_Q \times D_Q}$ (כלומר N_Q וקטורי שאילתה ממימד D_Q כל אחד). וקטורי השאילתה בעצם מחליפים לנו את התפקיד של וקטורי המצב הפנימי שהוצגו במנגנון הקודם.

נחשוב על כל וקטור שאילתה בתור <u>שאלה</u> שאנו מעוניינים לשאול. וה-Attention יחשב עבור כל וקטור מוקטורי הקלט כמה הוא רלוונטי עבור התשובה לשאלה הזאת. ויחזיר לנו וקטור "תשובה" שהוא ממוצא משוקלל של וקטורי הקלט, לפי הרלוונטיות שלהם לשאילתה הנוכחית. כך נעשה עבור כל וקטור שאילתה.

באופן פורמלי, עבור השאילתה $Q_i \in \{Q_i\}_{i=1}^{N_Q}$, נחשב את משקלי ה-Attention עבור כל וקטור קלט, כלומר נקבל עבור כל $X_i \in \{X_i\}_{i=1}^{N_X}$ ערך $a_i \in [0,1]$. ולבסוף, נחזיר עבור השאילתה את הפלט $Y_i \coloneqq \sum_{i=1}^{N_X} a_i \cdot X_i$. נעשה זאת עבור כל שאילתה Q_i ונקבל את מטריצת הפלט $Y = \left(Y_1, Y_2, \dots, Y_{N_Q}\right)$. כל Y_i הוא ממוצע משוקלל של וקטורי הקלט.

נשים לב שאנו משתמשים בוקטורי הקלט פעמיים במהלך החישוב. בפעם הראשונה אנו משתמשים בהם כדי לחשב את משקלי ה-Attention (הרלוונטיות שלהם) ביחס לכל שאילתה. ובפעם השנייה אנו משתמשים בהם כדי להפיק את הפלט של ה-Attention ביחס לכל שאילתה (חישוב הממוצע הממושקל שלהם בהתאם למשקלי ה-Attention שחישבנו).

אלו בעצם שני חישובים נפרדים, ומסתבר שמקבלים ביצועים טובים יותר אם נמנעים מהשימוש ההדוק בווקטורי הקלט ומשתמשים בהכללה הבאה: נמיר את ווקטורי הקלט X לוקטורי *מפתח* ווקטורי *ערך*. וקטורי המפתח ישמשו להפקת משקלי ה-Attention, ואילו וקטורי הערך ישמשו להפקת הפלט.

באופן פורמלי, יהיו לנו שתי מטריצות. *מטריצת המפתח* אותה נסמן בתור $W_K \in \mathbb{R}^{D_X \times D_Q}$, *ומטריצת הערך* אותה נסמן בתור $W_V \in \mathbb{R}^{D_X \times D_V}$. הערכים של המטריצות הללו הם משקלים שנלמדים בתהליך האימון של המודל.

את וקטורי המפתח (K) והערך (V) נחשב בעזרת מכפלת מטריצות:

$$K \coloneqq X \cdot W_K \in \mathbb{R}^{N_X \times D_Q}, \qquad V \coloneqq X \cdot W_V \in \mathbb{R}^{N_X \times D_V}$$

נשים לב שקיבלנו N_X וקטורי מפתח ממימד D_Q כל אחד, וN_X וקטורי ערך ממימד D_V כל אחד.

כעת, לחישוב משקלי ה-Attention ניעזר בווקטורי המפתח (K) בלבד. ניקח את וקטורי השאילתות Q ואת וקטורי המפתח K ונחשב

$$e_{i,j} = f_{attention}\left(Q_i, K_j\right)$$

בדומה למנגנון הקודם, נחשוב על $Q_i \in \mathbb{R}^{D_Q}$ בתור המצב הפנימי. ולכל וקטור מפתח $K_j \in \mathbb{R}^{D_Q}$ נקבל סקלר $e_{i,j}$ (הנקרא כאמור Alignment Score) המייצג את הרלוונטיות של K_j עבור השאילתה Q_i. מסתבר, כי אין צורך להגדיר את $f_{attention}$ בתור רשת נוירונים עם משקלים שצריך ללמוד. ומספיק להגדיר אותה ככפל וקטורים פשוט, כלומר,

$$e_{i,j} = f_{attention}\left(Q_i, K_j\right) \coloneqq \frac{\langle Q_i, K_j \rangle}{\sqrt{D_Q}}$$

(כפל איבר-איבר של הווקטורים $Q_i, K_j \in \mathbb{R}^{D_Q}$, סכמת התוצאות, וחלוקה בשורש המימד - D_Q. לא נכנס לסיבה מדוע בוחרים לחלק ב$\sqrt{D_Q}$, אך חלוקה זו אמורה לשפר את ביצועי המודל ולמנוע בעיה באימון מודלים הנקראת *Vanishing Gradients*).

נשים לב כי ניתן לחשב את כל הAlignment Scores בבת אחת לקבלת *מטריצת הSimilarities*, שנסמן אותה להיות E.

$$E \coloneqq QK^T \in \mathbb{R}^{N_Q \times N_X}, \qquad E_{i,j} = e_{i,j} = \frac{\langle Q_i, K_j \rangle}{\sqrt{D_Q}}$$

בכל שורה i במטריצה E יש את ערכי הAlignment Scores של כל N_X וקטורי המפתח, עבור השאילתה Q_i.

כעת, בדומה למנגנון הקודם, נרצה להמיר את הAlignment Scores למשקלי Attention (עבור כל שאילתה בנפרד – כלומר כל שורה במטריצה E בנפרד). נעשה זאת על ידי פונקציית Soft-Max שורה-שורה לקבלת *מטריצת המשקלים*, אותה נסמן להיות A. בעצם סכום כל הערכים בכל שורה בA הוא 1.

$$A \coloneqq SoftMax(E, dim = 1) \in [0,1]^{N_Q \times N_X}$$

הערך $A_{i,j}$ הוא משקל ה-Attention עבור ווקטור המפתח K_j ביחס לשאילתה Q_i.

חישבנו את משקלי ה-Attention עבור כל שאילתה ולכל ווקטור קלט (שהמרנו אותם לוקטורי מפתח). כעת נרצה להפיק את הפלט של הAttention. כאמור עבור השלב הזה נשתמש בווקטורי הערך (V) ולא בוקטורי המפתח.

עבור כל שאילתה Q_i נפיק את וקטור הפלט Y_i שהוא ממוצע משוקלל של וקטורי הערך, ביחס למשקלי ה-Attention שחישבנו בחזרת ווקטורי המפתח. כלומר $Y_i \coloneqq \sum_{j=1}^{N_X} A_{i,j} \cdot V_j$.

נשים לב שגם כאן ניתן לחשב את כל וקטורי הפלט בבת אחת באמצעות כפל מטריצות,

$$Y = \left(Y_1, \dots, Y_{N_Q}\right)^T = A \cdot V \in \mathbb{R}^{N_Q \times D_V}$$

ואלו הפלטים של שכבת ה-Attention.

נסכם את מנגנון ה-Attention שתואר במלואו,

<u>קלטים למנגנון</u>

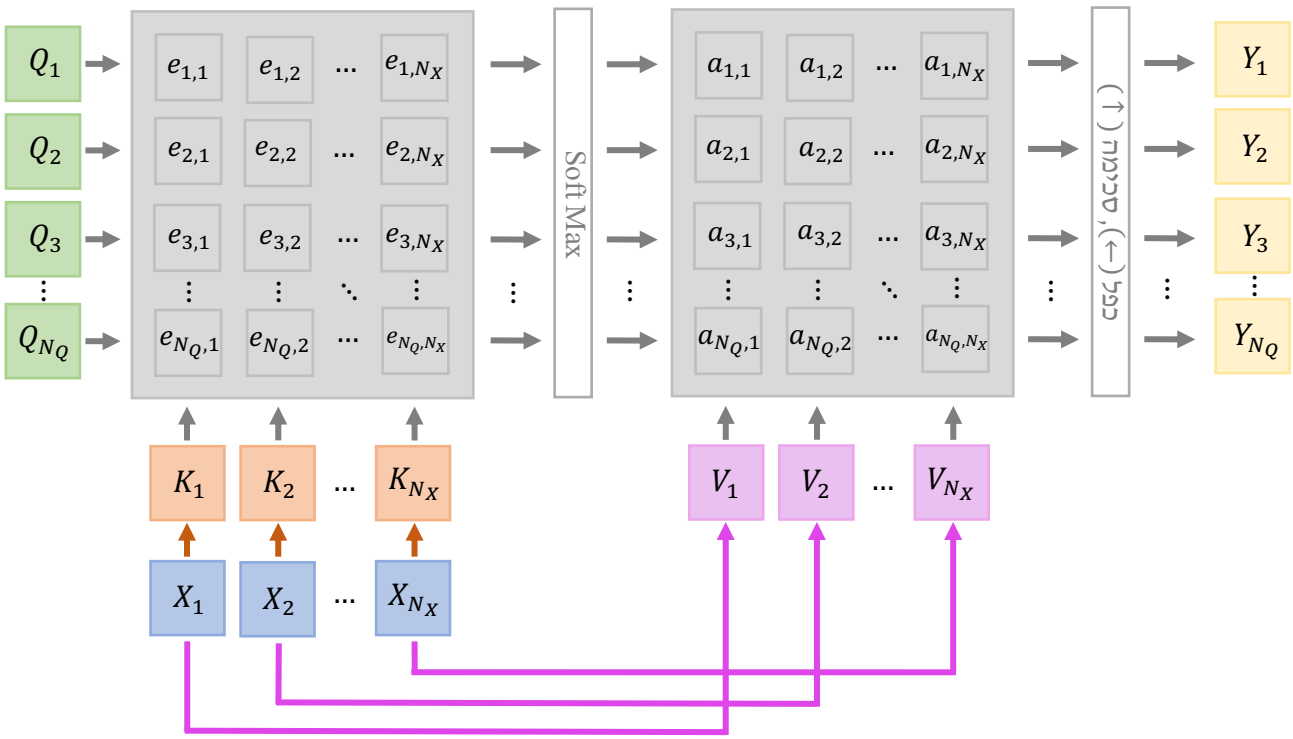
- וקטורי שאילתה $Q = \left(Q_1, \dots, Q_{N_Q}\right)^T$ (מימד $N_Q \times D_Q$)
- וקטורי קלט $X = \left(X_1, \dots, X_{N_X}\right)^T$ (מימד $N_X \times D_X$)
- מטריצת המפתח W_K (מימד $D_X \times D_Q$).

ומטריצת הערך W_V (מימד $D_X \times D_V$).

משקלי המטריצות נלמדים בתהליך האימון

<u>חישוב הפלט של המנגנון</u>

- וקטורי מפתח $K = X \cdot W_K$ (מימד $N_X \times D_Q$)
- וקטורי ערך $V = X \cdot W_V$ (מימד $N_X \times D_V$)
- מטריצת הSimilarities $E = Q \cdot K^T / \sqrt{D_Q}$ (מימד $N_Q \times N_X$)
- משקלי הAttention $A = SoftMax(E)$ (מימד $N_Q \times N_X$)
- וקטורי הפלט $Y = A \cdot V$ (מימד $N_Q \times D_V$)



איור הממחיש את שכבת ה-Attention במלואה.

ניתן לראות באיור את הפיצול של וקטורי הקלט לוקטורי מפתח וערך. וקטורי המפתח משמשים לחישוב משקלי הAttention ביחס לשאילתות, ואילו אחר כך מפיקים את הפלט על ידי ממוצע משוקלל של וקטורי הערך, עם משקלי הAttention שחושבו (עבור כל שאילתה – שורה במטריצה – בנפרד).

מנגנון ה-Self-Attention

שכבת ה-Attention שתוארה לעיל נועדה למצבים שבהם יש לנו שני סטים של דאטה (אחד שאנו חושבים עליו בתור "שאילתה" – Q, ואחד שאנו חושבים עליו בתור "קלטים" – X) ואנו מעוניינים להוציא כפלט תשובה לכל שאילתה המבוססת על (ממוצע משוקלל של) הקלטים.

מקרה ספציפי ומאוד שימושי של השכבה הזאת הוא שכבה הנקראת Self-Attention. בה יש לנו סט אחד של דאטה (רק ה-"קלטים" – X). סט זה משמש אותנו גם עבור השאילתה וגם עבור הקלטים. כלומר, כל וקטור בסט הזה מהווה גם שאילתה, שהתשובה שלה תהיה ממוצע משוקלל של הווקטורים בסט (כולל הוא עצמו). ומכאן שם השכבה – אנו לא מחשבים את הרלוונטיות של כל וקטור קלט ביחס לשאילתה כלשהי, אלא מחשבים את הרלוונטיות של כל וקטור קלט ביחס לעצמם.

כך, אנו בעצם משווים כל וקטור בסט הקלט שלנו לכל וקטור בסט הקלט שלנו. נתאר את המתמטיקה מאחורי הרעיון הזה.

השכבה נשארת כמעט זהה, אלא עם שינוי קטן: הקלט שלנו כאמור לא יכיל סט של וקטורי שאילתה. אלא נפיק את וקטורי השאילתה מתוך וקטורי הקלט. כלומר תהיה לנו *מטריצת שאילתה* אותה נסמן בתור

$$W_Q \in \mathbb{R}^{D_X \times D_Q}$$

ונוסיף שלב בו ניקח את וקטורי הקלט שלנו $X \in \mathbb{R}^{N_X \times D_X}$ (כאמור - N_X וקטורי קלט ממימד D_X כל אחד) ונפיק את וקטורי השאילתה על ידי כפל מטריצות:

$$Q = X \cdot W_Q \in \mathbb{R}^{N_X \times D_Q}$$

כלומר נקבל N_X וקטורי שאילתה (הפעם – כמספר וקטורי הקלט) שכל אחד ממימד D_Q. המימד D_Q של וקטורי השאילתה הוא היפר-פרמטר שצריך לבחור.

שאר השלבים במנגנון יישארו זהים.

נסכם את מנגנון ה-Self-Attention שתואר במלואו, נדגיש את החלקים שעברו שינוי ביחס למנגנון הקודם,

<u>קלטים למנגנון</u>

- וקטורי קלט $\boldsymbol{X} = \left(\boldsymbol{X_1}, \dots, \boldsymbol{X_{N_X}}\right)^{\boldsymbol{T}}$ (מימד $\boldsymbol{N_X} \times \boldsymbol{D_X}$)
- מטריצת המפתח $\boldsymbol{W_K}$ (מימד $\boldsymbol{D_X} \times \boldsymbol{D_Q}$).

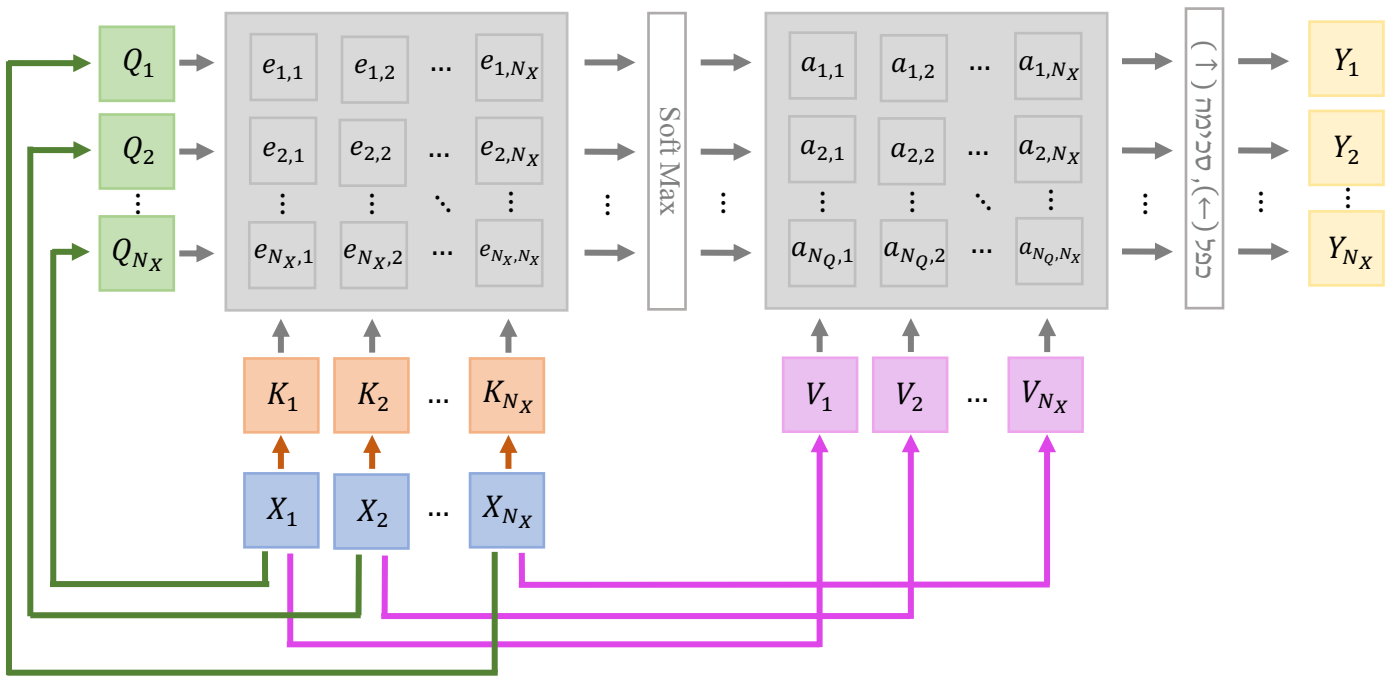
מטריצת הערך $\boldsymbol{W_V}$ (מימד $\boldsymbol{D_X} \times \boldsymbol{D_V}$).

ומטריצת השאילתה $\boldsymbol{W_Q}$ (מימד $\boldsymbol{D_X} \times \boldsymbol{D_Q}$)

משקלי המטריצות נלמדים בתהליך האימון

<u>חישוב הפלט של המנגנון</u>

- **וקטורי השאילתה $\boldsymbol{Q} = \boldsymbol{X} \cdot \boldsymbol{W_Q}$ (מימד $\boldsymbol{N_X} \times \boldsymbol{D_Q}$)**
- וקטורי מפתח $\boldsymbol{K} = \boldsymbol{X} \cdot \boldsymbol{W_K}$ (מימד $\boldsymbol{N_X} \times \boldsymbol{D_Q}$)
- וקטורי ערך $\boldsymbol{V} = \boldsymbol{X} \cdot \boldsymbol{W_V}$ (מימד $\boldsymbol{N_X} \times \boldsymbol{D_V}$)
- מטריצת הSimilarities $\boldsymbol{E} = \boldsymbol{Q} \cdot \boldsymbol{K^T} / \sqrt{\boldsymbol{D_Q}}$ (מימד $\boldsymbol{N_X} \times \boldsymbol{N_X}$)
- משקלי הAttention $\boldsymbol{A} = \boldsymbol{SoftMax}(\boldsymbol{E})$ (מימד $\boldsymbol{N_X} \times \boldsymbol{N_X}$)
- וקטורי הפלט $\boldsymbol{Y} = \boldsymbol{A} \cdot \boldsymbol{V}$ (מימד $\boldsymbol{N_X} \times \boldsymbol{D_V}$)



איור הממחיש את שכבת ה-Self-Attention במלואה. נבחין כיצד מוקטורי הקלט בלבד אנו מפיקים את וקטורי השאילתה, המפתח והערך.

לאחר הפקת המטריצות Q, K ו-V מוקטורי הקלט. נסמן את פלט ה-Self-Attention בתור $Attention(Q, K, V)$. שכמובן אפשר גם לכתוב את החישוב באופן ישיר:

$$Attention(Q, K, V) = SoftMax(Q \cdot K^T / \sqrt{D_Q}) \cdot V$$

Multi-Head Self-Attention

ההכללה האחרונה שנבצע לפני שנעבור לדבר על טרנספורמרים היא הוספה של "ראשים" לשכבת ה-Attention. מה הכוונה? ראשית נבחר מספר של ראשים, נסמן מספר זה להיות h, זהו היפר-פרמטר שצריך לבחור מראש. לאחר הפקת וקטורי השאילתה, מפתח וערך (Q, K, V – אלו המטריצות כך שווקטורי השאילתה, מפתח וערך מסודרים בשורות), נרצה לחלק כל ואחד מהווקטורים האלו ל-h וקטורים שונים.

כלומר, עבור וקטור שאילתה כלשהו $q \in \mathbb{R}^{D_Q}$, נרצה לחלק אותו ל-h וקטורים. כך גם עבור כל וקטור מפתח k וגם עבור כל ווקטור ערך v. החלוקה לh וקטורים תעשה על ידי הגדרת המטריצות $\{W_i^Q, W_i^K, W_i^V\}_{i=1}^h$. הפרמטרים של המטריצות הללו הם עוד פרמטרים שנלמדים בתהליך האימון.

עבור כל וקטור שאילתה q, נגדיר h וקטורי שאילתה חדשים $\{q_i\}_{i=1}^h$ כך ש $q_i = q \cdot W_i^Q$.
עבור כל וקטור מפתח k, נגדיר h וקטורי מפתח חדשים $\{k_i\}_{i=1}^h$ כך ש $k_i = k \cdot W_i^K$.
ועבור כל וקטור ערך v, נגדיר h וקטורי ערך חדשים $\{v_i\}_{i=1}^h$ כך ש $v_i = v \cdot W_i^K$.

נשים לב שניתן לחשב את כל הווקטורים בסט הראשון ($i = 1$) בבת אחת על ידי כפל מטריצות:

$$Q_1 = Q \cdot W_1^Q, \qquad K_1 = K \cdot W_1^K, \qquad V_1 = V \cdot W_1^V$$

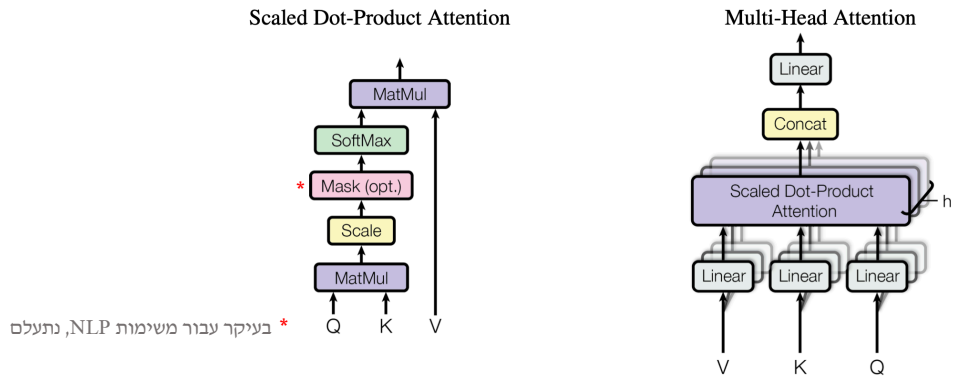
וכך גם עבור כל הווקטורים בסט השני ($i = 2$): $Q_2 = Q \cdot W_2^Q,\ K_2 = K \cdot W_2^K,\ V_2 = V \cdot W_2^V$, עד הווקטורים בסט ה-$h$.

עבור כל סט (Q_i, K_i, V_i) נחשב את ה-Attention <u>בנפרד</u>. ונקבל את הפלט $head_i$. לבסוף, נאחה (concat – כלומר ניקח את המטריצות $head_i$ "ונדביק" אותן אחת לשנייה בשורה) את כל הפלטים מכל הסטים לווקטור פלט יחיד $Concat(head_1, \dots, head_h)$. ובסוף נכפיל את הפלט שהתקבל בעוד מטריצה W^O כלשהי (שגם הפרמטרים שלה נלמדים בתהליך האימון) כדי לקבל את הפלט במימד שאנו רוצים.

פורמלית,

$$head_i = Attention(Q_i, K_i, V_i) = Attention\left(Q \cdot W_i^Q, K \cdot W_i^K, V \cdot W_i^V\right)$$
$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h) \cdot W_0$$

נסכם באיור שלקוח מהמאמר המפורסם [2] Attention Is All You Need שמסביר את הנושאים שהוצגו כאן.



משמאל, תיאור שלבי החישוב של שכבת Attention . מימין, תיאור שלבי החישוב של שכבת Attention בעלת h ראשים שמחושבים במקביל.

Vision Transformers

נציין כי המבנה של Vision Transformer הוא מעט שונה ממבנה של Transformer רגיל. נציג את המבנה של Vision Transformer (או בקיצור: ViT) כיוון שהוא האחד הרלוונטי עבורנו.

הטרנספורמר הוא בלוק נוסף שהוצג במאמר Attention Is All You Need. בלוק זה מקבל כקלט סט של וקטורים, ומשתמש בSelf-Attention (או Multi-Head Self-Attention) בתור <u>השלב היחיד</u> המשווה ומבצע אינטראקציה בין הווקטורים בסט הקלט.

בלוק הטרנספורמר יקבל כקלט סט של וקטורים $\{X_i\}_{i=1}^N$.
ראשית נעביר כל אחד מהווקטורים הללו דרך שכבת *Layer Normalization*. מטרת שכבה זו היא להבטיח שהפיצ'רים (האלמנטים) של כל וקטור יהיו בעלי תוחלת 0 ושונות 1. שכבה זו עוזרת להפחית דבר שנקרא *Internal Covariate Shift* ומקצר את זמן האימון של המודל. חישוב השכבה מתבצע על ידי חישוב הממוצע והשונות של כל וקטור, והוצאת ציון התקן שלו. באופן פורמלי, בהינתן וקטור $X_i \in \{X_i\}_{i=1}^N$ בעל מימד d, נחשב:

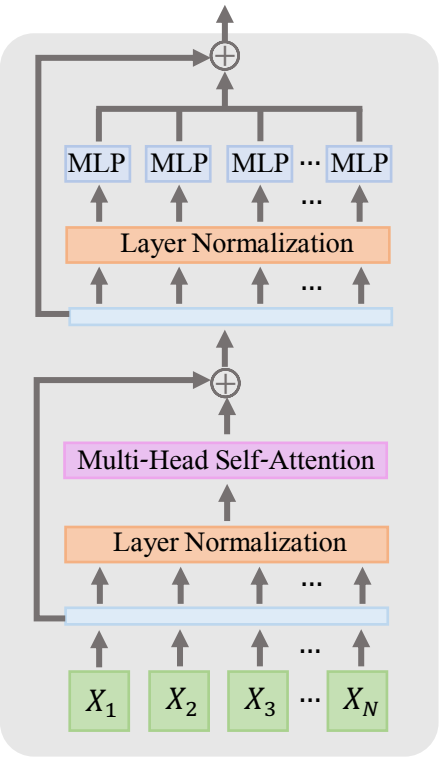
$$LN(X_i) = \gamma \cdot \frac{X_i - \mu_i}{\sigma_i} + \beta$$

כאשר $\mu_i = \sum_{j=1}^d X_{i,j}$ ו-$\sigma_i = \sqrt{\sum_{i=1}^d \left(X_{i,j} - \mu_i\right)^2}$. והסקלרים γ, β הם סקלרים נוספים שנלמדים בתהליך האימון. נשים לב שבשכבת הLN אין אינטראקציה בין הווקטורים בסט הקלט $\{X_i\}_{i=1}^N$ והשכבה מנרמלת כל וקטור באופן עצמאי לאחרים.

[2] Attention Is All You Need, Vaswani et al., NeurIPS, 2017.

השלב הבא שהוא יבצע זה להעביר את סט הווקטורים האלו דרך שכבת Self-Attention (יכולה להיות בעלת מספר ראשים). כל פלט שנקבל נשכבה זו יהיה תלוי בכל הקלטים שהוזנו אליה. רק כאן בעצם בא לידי ביטוי האינטראקציה בין ווקטורי הקלט.

לאחר שכבת הSelf-Attention נוסיף *חיבור רזידואלי (Residual Connection)* שהמשמעות של זה הוא שנוסיף את וקטורי הקלט הראשוניים, לתוצאה שהגענו אליה עד כה. כלומר, אם נסמן את הפלטים של שכבת ה-Attention להיות $\{A_i\}_{i=1}^N$, לאחר החיבור הרזידואלי, נעביר לשכבה הבאה את הסט $\{X_i + A_i\}_{i=1}^N$. נשים לב שזה בפרט דורש משכבת ה-Attention להוציא פלט מאותו המימד כמו הקלט שהיא קיבלה (כזכור אפשר לבחור את מימד הפלט). המטרה של חיבור כזה הוא להתמודד עם בעיה שנקראת Vanishing Gradient. לא נפרט על זה מעבר.



בלוק הטרנספורמר

לאחר החיבור הרזידואלי נוסיף עוד שכבת Layer Normalization, ואחריה שכבה של *Multilayer Perceptron*. כלומר רשת Fully-Connected כאשר בין שכבותיה יש פונקציית אקטיבציה כלשהי. שכבת הMLP תעבוד על כל וקטור (שיצא כפלט משכבת הNL) בנפרד ובאופן בלתי תלוי לאחרים. הפרמטרים של רשת הנוירונים הזאת נלמדים כמובן כחלק מתהליך האימון.

לאחר שכבה זו נוסיף עוד חיבור רזידואלי: ניקח את הפלטים משכבת הMLP ונוסיף אליה את הפלטים שהתקבלו לאחר החיבור הרזידואלי הקודם. כלומר, אם נסמן את הפלטים שיתקבלו משכבת הMLP להיות $\{B_i\}_{i=1}^N$ אזי לאחר החיבור הרזידואלי, נוציא כפלט את הסט $\{B_i + (X_i + A_i)\}_{i=1}^N$.

וזהו! נשים לב שמספר הווקטורים שקיבלנו כקלט שווה למספר הווקטורים שאנו מוציאים כפלט – והוא N.