מבוא ללמידה עמוקה - תרגיל מספר 2

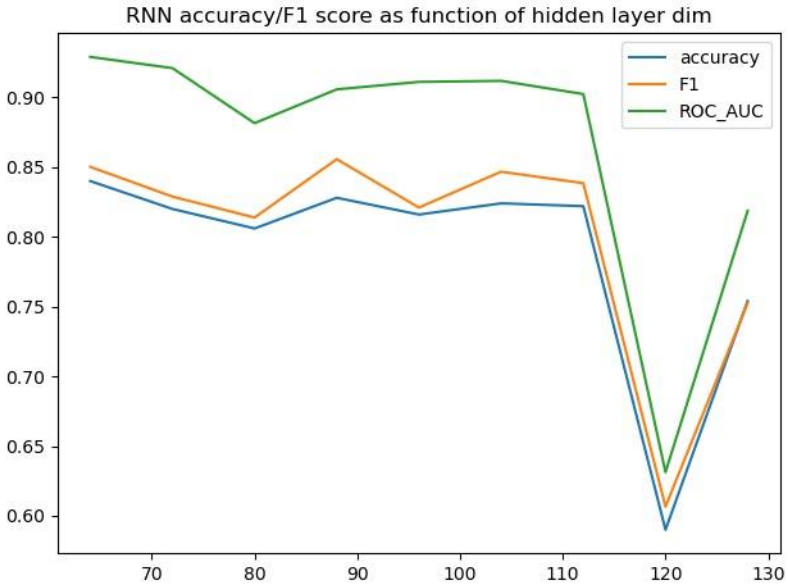
מגישות :

ירדן טל, 203730700

נטע ברק, 204635593

חלק מעשי:

1. יישמנו את רשת הRNN בעזרת הרשת הבסיסית Elman אותה למדנו. פונקציית האקטיבציה אותה בחרנו על מנת לממש את אי הלינאריות ברשת הנ״ל היא הפונקציה הנפוצה בשימוש, relu.



ניתן לראות מהגרף הנ״ל כי המודל משיג את התוצאות הטובות ביותר עבור המטריקות אותן בחנו ב-hidden layers בטווח המימדים של [112, 64]. בנוסף, ניתן לראות כי בשכבות ממימד יותר גבוה (בין 128-12) ביצועי המודל יורדים, יתכן כי מדובר במימדים גדולים מדי עבור המשימה והדאטה הנ״ל והמודל חווה תופעה של over-fitting.

עוד נוסיף, כי שמנו לב שהשונות בין ריצות שונות לא מבוטלת, ולכן ייתכנו מקרים שיראו שונה מהמתואר לעיל.

את רמת הדיוק הגבוהה ביותר קבלנו עבור hidden state בגודל 64, ואלו היו תוצאותיו-

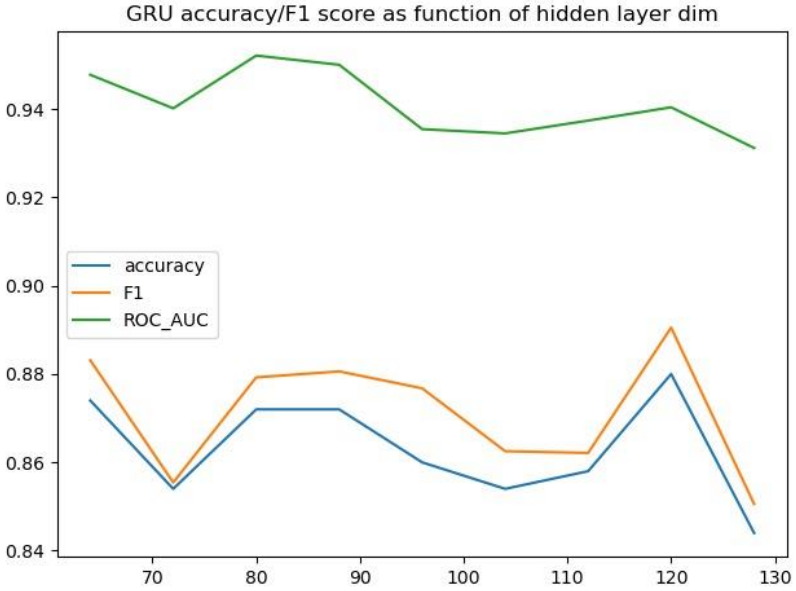
Accuracy - 0.84

F1 - 0.8501872659176031

AUC - 0.9290097538918675

יישמנו בנוסף רשת GRU כפי שלמדנו בכיתה (פונקציית אקטיבציה של sigmoid עבור z ו- r ופונקציית אקטיבציה מסוג tanh עבור h').

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$



ניתן לראות מהגרף כי סך הכל הביצועים כתלות בגודל ה-hidden state די דומים. בנקודה בה ה-hidden state ממימד 120 היא הנקודה בעלת הביצועים הטובים ביותר, אך עם זאת ניתן לראות שהנקודות הקרובות לה משני צידיה קבלו דווקא ביצועים פחות טובים וייתכן כי אם הייתי עושים יותר הרצות או הרצה מחדש היא לאו דווקא הייתה נותנ את התוצאות הטובות ביותר.

את רמת הדיוק הגבוהה ביותר קבלנו עבור hidden state בגודל 120, ואלו היו תוצאותיו-

Accuracy - 0.88
F1 - 0.8905109489051094
AUC - 0.9403946773464247

בצענו אבלואציה למודל בעזרת שלוש שיטות - Accuracy, F1, AUC כתלות במימד ה- hidden state.

ניתן לראות מהתוצאות כי המטריקות איתן בחנו את התוצאות של שתי הארכיטקטורות הנ״ל מבציעות על ביצועים טובים יותר של הGRU לעומת הRNN בכל ערכי הhidden states איתם בחנו את הרשתות. ממצא זה אכן מתיישב עם מה שלמדנו בכיתה, שרשתות GRU\LSTM מאפשרות לנו גמישות רבה יותר לגבי מידת הלמידה שלנו מהעבר ויותר מכך מאפשרות לנו skip connection

שמאפשר לנו "לקחת בחשבון" בכל זמן נתון גם את "הזכורונות" היותר רחוקים שלנו, מה שעתיד לעזור בעיקר בביקורות ארוכות יותר. באופן כללי, למשימות מסוג זה נעדיף לרב להשתמש בLSTM\GRU על פני RNN פשוט יותר.

2. בסעיף זה מימשנו שיטה אלטרנטיבית לאלו המוצעות בסעיפים קודמים ע"י הפעלה של כמה שכבות FC על כל מילה בנפרד במשפט כאשר לכל מילה הוצאה פלט יחיד שמהווה את ה- sub score שלה, ולבסוף על מנת לתת פרדיקציה למשפט כולו סמכנו על כלל ציוני המילים והעלנו פונקציית sigmoid.

כדי לבחון שיטה זו, בחנו ארכיטקטורות שונות של שכבות FC (כמות שכבות וגודל שכבות שונה).

טבלה הבאה ניתן לראות את התוצאות השונות לארכיטקטורות השונות לפי המדדים Accuracy, F1, AUC. (בעמודה המתארת את הארכיטקטורה של הרשת, גודל המערך מייצג את מספר השכבות ללא השכבה האחרונה אפשר מוציאה פלט ממימד 1, כאשר כל ערך מייצג את כמות הנוירונים בכל שכבה).

Architectures	Accuracy	F1	AUC
[50]	0.862	0.8211382113821137	0.9230572105836375
[64, 32]	0.858	0.8627450980392157	0.9370555448779551
[64, 32, 16]	0.862	0.8603773584905661	0.9357582164136075
[200, 50]	0.852	0.8710280373831776	0.9448555320648344
[32, 16]	0.86	0.8571428571428573	0.9303606893458901
[8 ,16 ,32]	0.824	0.8649706457925636	0.9321705426356588

ניתן לראות כי השונות בין הארכיטקטורות לא מאוד גבוהה (ויותר מכך, ישנה גם שונות בין ריצות שונות). בחרנו את הארכיטקטורה המסומנת בצהוב שנתנה את שקלול המדדים הגבוה ביותר.

כעט נבחן שתי ביקורות אותן כתבנו ונתנו למודל לסווגן לחיוביות או שליליות, כאשר על אחת מהן המודל טעה (על 2) ועל השנייה צדק (על 1) –

1. This movie is the best movie I have ever seen
2. Not what I excepted

להלן התוצאות של ה-prediction עבור כל משפט וה-sub score עבור כל מילה במשפט:

Prediction - Positive

this [-0.02481293]

movie [-0.07545237]

is [0.04863422]

the [0.02420379]

best [0.36183047]

movie [-0.07545237]

i [0.13949262]

have [-0.06085153]

ever [0.10517512]

seen [0.09028802]

Prediction - Positive

not [-0.12315343]

what [0.0330016]

i [0.13949262]

expected [0.04185721]

3. בחלק זה השתמשנו באותה ארכיטקטורה מהחלק הקודם, כאשר הוספנו עוד שכבת פלט לרשת שהיא שכבת משקלות עבור המילים השונות במשפט, אשר עליה בצענו נרמול בעזרת softmax ואז הכפלנו בשכבת הציונים וכך המודל נתן פרדיקציה למשפט.
ראשית נציין כי השונות בין ריצות שונות גבוהה, זה ככל הנראה נובע מאתחול רנדומלי של משקולות הרשת וכן מבחירת הדאטה בצורה רנדומלית כל ריצה מחדש. בחלק מהריצות ניתן לראות השפעה של משקול המילים השונות במשפט ובחלק מהריצות לא.
ככלל, נראה כי תוספת המשקול עבור המילים במשפט איננה שיפרה (בהרצה אותה ציינו) את התוצאות על פי המדדים השונים אותם בחנו.
להלן התוצאות עבור הארכיטקטורה אותה בחרנו בסעיף מספר 2 (16, 32, 64) עבור ריצה זו –

Accuracy - 0.848
F1 - 0.8457
AUC - 0.9391

(נציין כי גם פה בחנו את כלל הארכיטקטורות שהצגנו בסעיף הקודם וגם כאן קבלנו שארכיטקטורה זו נתנה את התוצאות הטובות ביותר.)

השיטה הנ״ל ללא הוספת המשקולות בוחנת את סנטימנט המשפט כולו ללא התייחסות לתלות בין המילים (FC על כל מילה בנפרד ולבסוף סכימה של כלל התוצאות). בשלב הזה, הוספת משקול לכל מילה (יחד עם softmax) ביחס למילים האחרות במשפט נועדה ליצור תלות בין המילים. למרות השיפור הנ״ל, **ניתן לראות כי עדיין המודל מסווג את המשפטים שבחרנו בסעיף 2 באותו אופן, כלומר מתייג ביקורת אחת כחיובית והיא אכן חיובית ואת השנייה כחיובית למרות שהיא שלילית.**

להלן התוצאות של ה-prediction עבור כל משפט וה-sub score עבור כל מילה במשפט:

Prediction - Positive

this [-1.03862349]

movie [-2.02220357]

is [0.01583018]

the [0.01580326]

best [5.44588837]

movie [-1.02220357]

i [2.07246494]

have [-0.07591059]

ever [3.10312125]

seen [4.17129166]

Prediction - Positive

not [-7.140239]

what [-0.53022045]

i [3.2505891]

expected [2.7086198]

לסיכום, לדעתנו הסיבות לכך שמשקול המילים אחת ביחס לשנייה במשפט לא מוביל לשיפור בביצועים הן –

א. אמנם כל מילה מקבלת משקול שונה במשפט, שזה צעד שאכן אמור לשפר את התוצאות, אך ה-score עצמו לא משתנה. לדוגמא, אם יש לנו את הביטוי- not good, היינו רוצים שלמילה good יינתן ציון שלילי שכן המשמעות שלה במשפט היא משמעות שלילית. אך במקרה הנ״ל, המשקלים לכל מילה משתנים, אך לא ה-score עצמו.

ב. כל מילה מתחשבת במילים אחרות אך ללא התייחסות למיקום שלהן במשפט. לדומגא – אם המילה not לא הייתה קשורה למילה good במשפט, או אם היא כן, עדיין המשקלים היו ניתנים באותה צורה.

ג. בנוסף, כל מילה מקבלת משקול התחלתי ביחס לעצמה בלבד, ורק לבסוף אנחנו מבצעים נרמול על ידי softmax. ייתכן כי הדבר מוביל לכך שהמשקול אינו מדויק ומתאים לקונטקט המשפט כולו.

4+5. בסעיף זה הוספנו שכבת attention לפני שכבות הfc מסעיף 3. שכבת הattention לא התייחסה לכל המילים במשפט הקלט אלא לחלון בגודל 5.

להלן התוצאות עבור הארכיטקטורה הנ״ל, כאשר השארנו את ארכיטקטורה שכבות ה-fc כפי שבחרנו בסעיף מספר 2 (16, 32, 64) –

Accuracy - 0.878
F1 - 0.873366511053021
AUC - 0.9418615364611103

כעט נבחן שתי ביקורות אותן כתבנו ונתנו למודל לסווגן לחיוביות או שליליות, כאשר על אחת מהן המודל טעה (על 1) ועל השנייה צדק (על 2) –

1. I really can not understand why people said this is a good movie
2. Not what I excepted

להלן התוצאות של ה-prediction עבור כל משפט וה-sub score עבור כל מילה במשפט:

Prediction - Negative

not [-7.6082687]

what [-2.31238696]

i [3.894897]

expected [-4.8289909]

Prediction – Positive

i [6.6103826]

really [1.7209157]

can [-1.1429608]

not [-4.819366]

understand [-1.2137502]

why [-1.5227742]

people [-1.8042912]

said [0.26997596]

this [1.0148298]

is [5.7207956]

a [-4.322294]

good [4.8892765]

movie [1.3485745]

ניתן לראות לפי הניסוי הנ״ל כי כפי שלמדנו בכיתה מנגנון ה-attention מאפשר למודל לקודד כל מילה במשפט בהקשר רחב יותר, במקרה שלנו בחלון בגודל 5. וכך כל מילה מקבלת ציון של הסנטימנט שלה לפי ההקשר שלה במשפט.

מהדוגמאות אותן בחנו ניתן לראות כי המודל תיקן את הפרדיקציה על משפט מספר 2 לאחר שטעה עלייה בכלל המודלים הקודמים שבחנו. המודל נותן ציונים שליליים מאוד למילים ״not״ ו-״excepted״, ומכך ניתן להסיק כי שכבת ה-attention אכן אפשרה למודל כפי שציפינו לקשר בין שתי המילים הללו ולהבין את ההקשר השלילי במשפט.

עם זאת, על דוגמא מספר 1 המודל טעה. מדובר בביקורת מורכבת יותר, כאשר המרחק בין המילה ״not״ למילה ״good״ גדול מ-5, וייתכן כי זו הסיבה שהמודל אינו מצליח להתמודד עם ביקורת זו בצורה טובה, שכן את שכבת הattention הפעלנו על חלון בגודל 5.

חלק תאורטי:

1. כלל השרשרת:

a. כתבו את הנגזרת עבור $\frac{df(x+y,2x,z)}{dx}$:

נסמן $g_1(x) = x + y, g_2(x) = 2x, g_3(x) = z$ ונקבל לפי כלל השרשרת:

$$\frac{df(x+y,2x,z)}{dx} = \frac{df(g_1(x),g_2(x),g_3(x))}{dx} =$$
$$\frac{dg_1(x)}{dx}\frac{df(g_1(x),g_2(x),g_3(x))}{dg_1(x)} + \frac{dg_2(x)}{dx}\frac{df(g_1(x),g_2(x),g_3(x))}{dg_2(x)} +$$
$$\frac{dg_3(x)}{dx}\frac{df(g_1(x),g_2(x),g_3(x))}{dg_3(x)} = 1 \cdot \frac{df(g_1(x),g_2(x),g_3(x))}{dg_1(x)} + 2 \cdot$$
$$\frac{df(g_1(x),g_2(x),g_3(x))}{dg_2(x)} + 0 \cdot \frac{df(g_1(x),g_2(x),g_3(x))}{dg_3(x)} =$$
$$\frac{df(g_1(x),g_2(x),g_3(x))}{dg_1(x)} + 2 \cdot \frac{df(g_1(x),g_2(x),g_3(x))}{dg_2(x)}$$

b. כתבו ביטוי כללי לכלל השרשרת עבור הפונקציה $f(x) = f_1(f_2(\dots f_n(x)))$:

$$\frac{df}{dx} = \frac{df_1}{df_2}\frac{df_2}{df_3}\cdots\frac{df_n}{dx}$$

כאשר הנגזרת $\frac{df_i}{df_{i+1}}$ תשוערך בנקודה $f_{i+1}(f_{i+2}(\cdots f_n(x)))$ עבור כל $i \in \{1, 2, \dots, n-1\}$, ועבור $\frac{df_n}{dx}$ היא תחושב עבור x.

c. איזו פונקציית אקטיבציה הייתן מוסיפות לכל שכבה על מנת לייצב את הנגזרת של הביטוי הנ"ל?

היינו משתמשות ב-**leaky ReLU**. ראשית, נסביר למה לא היינו משתמשות בסיגמואיד או ב-tanh: הנגזרות שלהן מחזירות ערכים בתחומים [0,0.25] ו-[0,1] בהתאמה, אבל עבור רוב הקלטים ב-$\mathbb{R}$ הן יחזירו ערכים קרובים מאוד ל-0. כשמבצעים back propagation משתמשים בכלל השרשרת ומבצעים מכפלה של הנגזרת שוב ושוב לערכים שונים, ולכן ישנה סבירות גבוהה שהגרדיאנט יתאפס – gradient vanishing. ב- ReLU לעומת זאת, הנגזרת היא 0 עבור קלט קטן מ-0 ו-1 עבור ערך גדול מ-0, ולכן הסיכוי להתאפסות הגרדיאנט קטן יותר, אבל עדיין לא מבוטל. Leaky ReLU היא האופציה המועדפת משום שהנגזרת שלו היא תמיד שונה מ-0 (ε קטן כלשהו עבור ערכים

שקטנים מ-0, ו-1 עבור ערכים שגדולים מ-0) ולכן פחות סביר שמכפלת הנגזרות של כל השכבות תחזיר ערך 0.

d. כתבו את הביטוי המתאר את הנגזרת של ההרכבה $f_1(x, f_2(x, f_3(\dots f_{n-1}(x, f_n(x)))))$:

ראשית, נתבונן בנגזרת של $f_{n-1}(x, f_n(x))$:

$$\frac{df_{n-1}(x, f_n(x))}{dx} = \frac{df_{n-1}}{d(1,0)}(x) + \frac{df_{n-1}}{d(0,1)}(f(x)) \cdot \frac{df_n}{dx}(x)$$

כעת, נתבונן בנגזרת של $f_{n-2}(x, f_{n-1}(x, f_n(x)))$:

$$\frac{df_{n-2}(x, f_{n-1}(x, f_n(x)))}{dx} = \frac{df_{n-2}}{d(1,0)}(x) + \frac{df_{n-2}}{d(0,1)}(f_{n-1}(x, f_n(x))) \cdot \frac{df_{n-1}(x, f_n(x))}{dx}$$

נמשיך להפעיל את כלל השרשרת באופן רקורסיבי ונקבל:

$$f_1\left(x, f_2\left(x, f_3\left(\dots f_{n-1}(x, f_n(x))\right)\right)\right) = \frac{df_1}{d(1,0)}(x) + \frac{df_1}{d(0,1)}\left(f_2\left(x, f_3\left(\dots f_{n-1}(x, f_n(x))\right)\right)\right) \cdot \frac{df_2\left(x, f_3\left(\dots f_{n-1}(x, f_n(x))\right)\right)}{dx}$$

e. מה היתרון של $f(x + g(x + h(x)))$ לעומת $f\left(g(h(x))\right)$?

נתבונן בנגזרות של שני הביטויים:

$$f\left(g(h(x))\right)' = f'(g(h(x))) \cdot g'(h(x)) \cdot h'(x)$$

לעומת זאת,

$$f\left(x + g(x + h(x))\right)' = f'\left(x + g(x + h(x))\right) \cdot (1 + g'(x + h(x)) \cdot (1 + h'(x)))$$

במקרה הראשון, אם $h(x)=0$ או $g(x)=0$, הערך של כל הנגזרת יתאפס. במקרה השני (שמתאר שימוש ב-*skip connections*), הערך של כל הנגזרת לאו דווקא יתאפס. לכן שימוש ב- *skip connections* יכול לסייע במניעה של *vanishing gradient*.

2. הסבירו מה סוג ארכיטקטורת הרשת המתאימה לבעיות הבאות:

a. <u>זיהוי דיבור (אודיו לטקסט)</u> - אנחנו מניחות שהקלט באודיו כבר מגיע כשהוא מחולק למילים ולכל מילה יש וקטור פיצ'רים שמתאר אותה. במידה ואנחנו מקבלות וקטורים שמייצגים דגימות של הקול לכל נקודת זמן, נשתמש בשכבות קונבולוציה על מנת לחלץ את הפיצ'רים לכל מילה. נבחין שגם לקלט וגם לפלט בבעיה יש אורך משתנה. כמו כן, ההקשר בו מופיעה מילה הוא לפעמים קריטי להבנת המשמעות שלה, אופן הכתיבה שלה והתמודדות עם רעשים. לדוגמה, המשפט I ate a cherry יכול להישמע כמו המשפט eye eight oh Jerry, אבל המשפט השני הוא רצף לא הגיוני של מילים. לכן, נבחר להשתמש ב-**RNN** מסוג **Many to Many** כשכל חוליה ברשת ה- **RNN** היא **GRU**. הרשת תקבל את הקלטים בזה אחר זה ובסוף כל מילה בקטע הקול, תפלוט וקטור הסתברויות שמתאים למילה. אם כל הקלט האודיטורי נתון מראש, אפשר להשתמש ב-RNN דו כיווני ולשקלל את התוצאות מהמעבר מההתחלה לסוף ומהסוף להתחלה, כדי לשפר את הבנת ההקשר של המשפט.

b. <u>מענה על שאלות</u> - אנחנו מניחות שהבעיה היא בעיה של מענה על שאלות מתוך טקסט קיים, למשל ערך בויקיפדיה. ניתן לפתור בעיה זו באמצעות מודל **BERT** כאשר הקלט יהיה השאלה, טוקן מפריד ואחריה הטקסט בו נרצה לחפש את התשובה. הפלט יכיל וקטור לכל טוקן בנפרד, ואז נפעיל על כל פלט כזה שכבות של **MLP** עד לקבלת פלט בגודל 3 לכל טוקן - שיכיל את ההסתברות לכך שהמילה מופיעה בתחילת התשובה, באמצע התשובה או בסוף התשובה. כדי למצוא את התשובה בטקסט, נסתכל על התוצאות עבור המילים ששייכות לטקסט (ולא לשאלה המקורית) ונחשב את המיקומים הכי סבירים להתחלה ולסוף המשפט (ניתן לעשות זאת בזמן ריצה ריבועי ע"י שקלול כל ההסתברויות לזמני ההתחלה והסוף האפשריים).

c. <u>ניתוח סנטימנט</u> - הקלט שלנו הוא בגודל משתנה, המיקום של המילים לאורך הקלט לאו דווקא מעיד על הקשר ביניהן וכדי להבין את המשפט נצטרך בהרבה מקרים לקרוא את כולו (קריאה של חלק מהקלט לאו דווקא מבטיחה

הבנה שלו, לפעמים צריך לקרוא עד הסוף כדי להבין את המשמעות של תחילת הטקסט). כדי לפתור את הבעיה נשתמש ב-**Transformer** או ב-**BERT** (ייתכן שנצטרך יותר מ-transformer אחד כדי שהרשת תהיה אקספרסיבית מספיק). בדומה למה שעשינו בתרגיל המעשי, נפעיל שכבות של **fully connected** על כל פלט בנפרד ונבצע סכימה או סכימה ממושקלת של התוצאות עבור כל המילים.

d. <u>קלסיפיקציה של תמונות</u> - נשים לב שהקלט הוא בגודל קבוע (אם התמונות לא בגודל קבוע נקטין/או נגדיל אותם לגודל הקלט שהרשת מצפה לקבל ע״י אינטרפולציה או pooling), אדיש להזזה, ולתבנית מסוימת בתמונה יש את אותה משמעות ללא קשר למיקום שלה בתמונה. בנוסף, כל פיקסל קשור לפיקסלים שקרובים אליו ולא לאלו שרחוקים ממנו, אז אין צורך שהרשת תנתח את הקשרים בין פיקסלים רחוקים זה מזה. לכן, נשתמש ב- **CNN** שמריצה פילטרים קטנים יחסית שחוזרים על עצמם עבור חלקי התמונה השונים. את הפלט של ה-CNN נכניס ל-**MLP** שיבצע חישובים נוספים (כולל אקטיבציה לא לינארית) ויחזיר פלט בגודל של מספר המחלקות האפשריות. נפעיל שכבת **softmax** ונקבל הסתברויות לקבלת כל אחת מהמחלקות.

e. <u>תרגום מילה בודדת</u> - נשים לב שהקלט שלנו הוא באורך קבוע, ומייצג יישות אחת - מילה. נקבל כקלט את השיכון של המילה בשפה א׳, ונחזיר פלט בגודל של השיכון של המילה בשפה ב׳. את הפלט הזה נעבד לוקטור הסתברות עם ערך לכל אחת מהמילים בשפה ב׳, בהתאם למרחק של השיכון שלהן (למשל הזווית) מהפלט של הרשת. נעשה זאת ע״י שימוש ב-**MLP**.

3. ארכיטקטורה המתאימה לרשת שמקבלת טקסט ומחזירה תמונה.
   a. נשתמש ברשת שמורכבת משני חלקים - **encoder** ו-**decoder**.
      i. ה- encoder יקבל כקלט את המשפט ויחזיר כפלט וקטור במרחב החבוי. ניתן לממש אותו בכמה דרכים:
         1. Transformer שיקבל את השיכונים של כל המילים במשפט וישקלל את הפלט (שיהיה בגודל מתאים למספר המילים במשפט) לכדי פלט בגודל אחיד, למשל ע״י סכום ממושקל כמו שעשינו בחלק המעשי של התרגיל.
         2. Transformer שיקבל את השיכונים של כל המילים במשפט ובנוסף להם טוקן קלט מיוחד (כמו ה-cls שראינו בשיעור), והוקטור החבוי יתבסס רק על הערך בפלט המתאים לטוקן הנ״ל.
         3. רשת LSTM שתקבל את כל המילים במשפט כקלט ותעבור עליהן אחת אחרי השנייה, והוקטור החבוי יתבסס רק על התוצאות של הפלט האחרון.

ii. ה-decoder יקבל כקלט את המשפט ויחזיר כפלט תמונה, ויתבסס על שכבות קונבולוציה ו-upsampling.

b. נוסיף בסוף ה-encoder שלנו שכבת Attention. ה-keys ו-values שלה יהיו הפלט של ה-encoder לאחר שכפלנו אותו במטריצת משקולות שונה עבור ה-keys ו-values. ה-queries שלה יהיו 4 וקטורי שאילתות קבועים, אחת לכל רביע בהתאמה. הפלט שנקבל יהיה 4 וקטורים שמתארים את ארבעת הרביעים בתמונה ואותם נעביר ל-decoder.