דוח תרגיל בית 2 – VQA

תוצאת הדיוק הכי טובה על evaluation – 48.2908

תהליך העבודה:

ארכיטקטורת מודל: המודל מכיל שלושה תתי מודלים- V model, Q model, Attention model ומודל נוסף שמאחד אותם: VQA model.

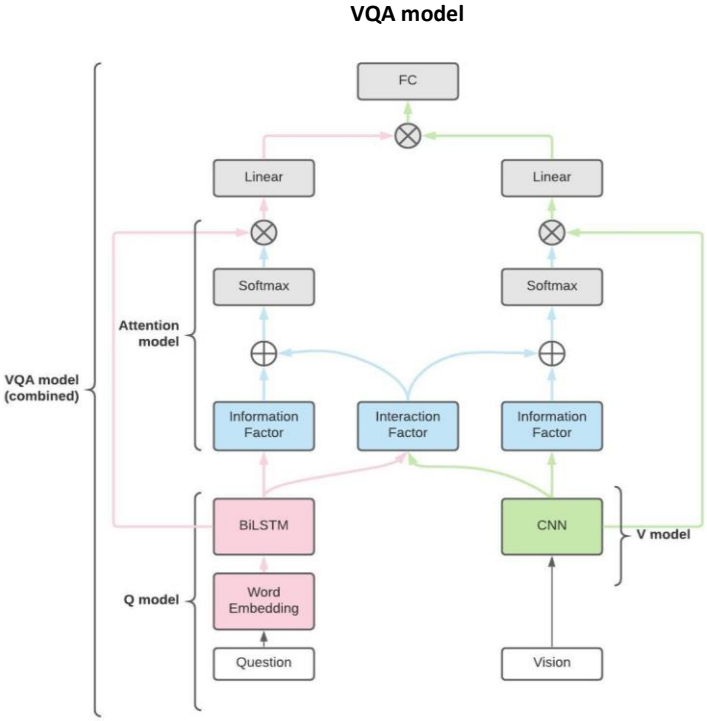
V model: מודל שלומד את התמונות. מכיל שכבות קונבולוציה שביניהן אקטיבציה, pooling ונרמול. הפלט הוא ייצוג לכל region בתמונה. המשקולות ההתחלתיות של המודל נלקחו ממודל autoencoder שאימנו מראש על משימת reconstruction בעזרת שכבות קונבולוציה ושכבות דה-קונבולוציה.

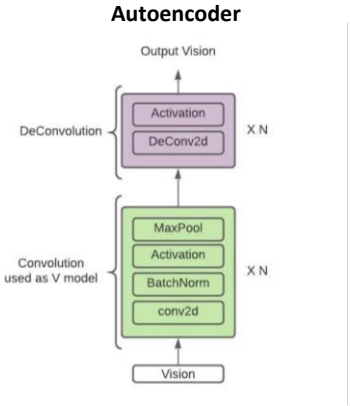
Q model: מודל שלומד את השאלות. מכיל שכבת embedding וBiLSTM דו שכבתי. הפלט הוא הייצוג מהשכבה האחרונה עבור כל מילה בשאלה.

Attention model: כפי שלמדנו בהרצאה 7. מכיל אינטרקציות בין V לV, Q לQ ובין V לQ. הפלט הוא ייצוגי V ו-Q.

VQA model: מריץ את שלוש המודלים לעיל, זה אחר זה. לאחר מכן מעביר את מכפלת הייצוגים המוכנים של Q ושל V בשכבת FC. הפלט הוא וקטור הסתברויות לכל אחת מהתשובות האופציונליות (אלו שעברו סינון).

אילוסטרציה:



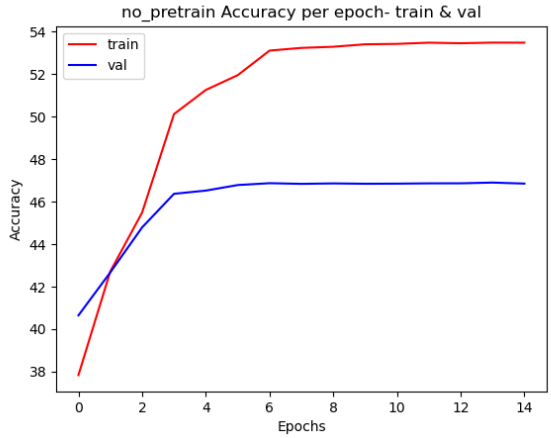


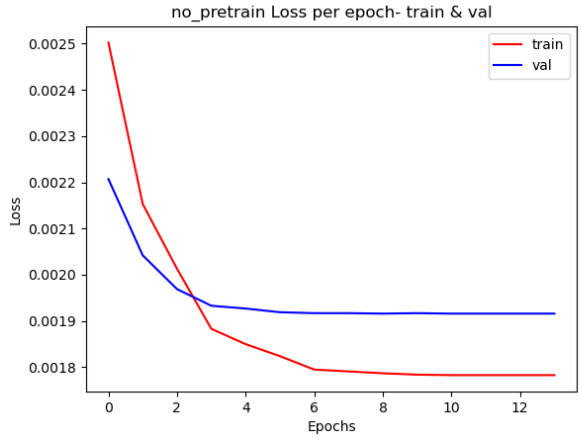
Ensemble: אימנו בנפרד ובאופן בלתי תלוי 3 מודלי VQA, הנבדלים בכמות שכבות הקונבולוציה ובשימוש או אי שימוש בautoencoder ליצירת משקולות ראשוניות למודל V:

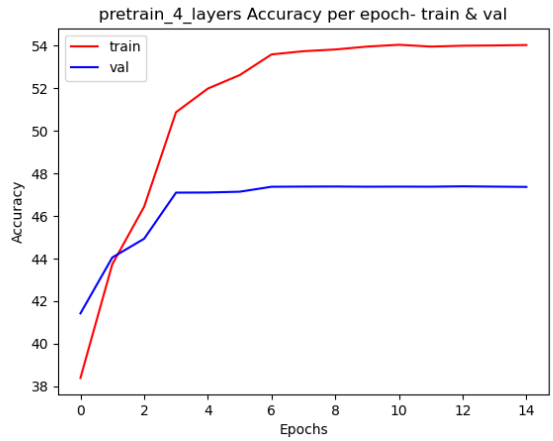
1. מודל שאינו אומן מראש, בעל 8 שכבות קונבולוציה
2. מודל מאומן מראש, עם 4 שכבות קונבולוציה
3. מודל מאומן מראש, עם 8 שכבות קונבולוציה (אותן שכבות ממודל 1 ואותה ארכיטקטורת autoencoder של מודל 2)

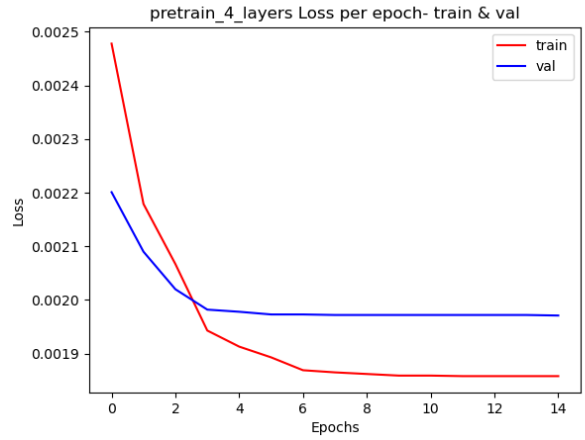
בשלב הevaluation, הכנסנו כל דגימה לכל אחד מהמודלים בנפרד, ומיצענו את שלוש וקטורי ההסתברויות שהתקבלו, עם שקלול של 1 לשני המודלים המאומנים מראש ו-0.6 למודל 1. התוצאה הממוצעת היא הפלט הסופי של המודל שלנו, ועבורה חישבנו את מדד הsoft accuracy.

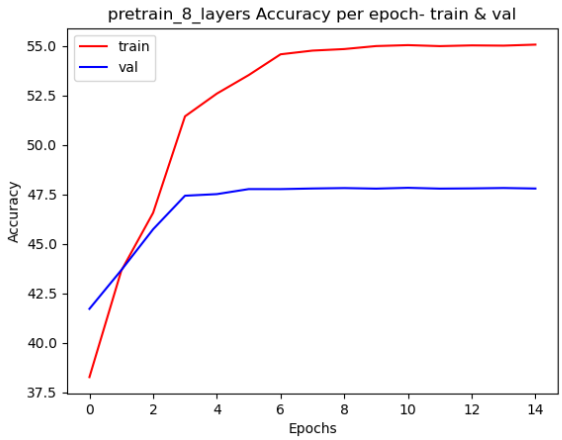
גרפי התכנסות: נציג את גרפי ההתכנסות עבור 3 המודלים בהם השתמשנו (שם המודל בכותרת הגרף)

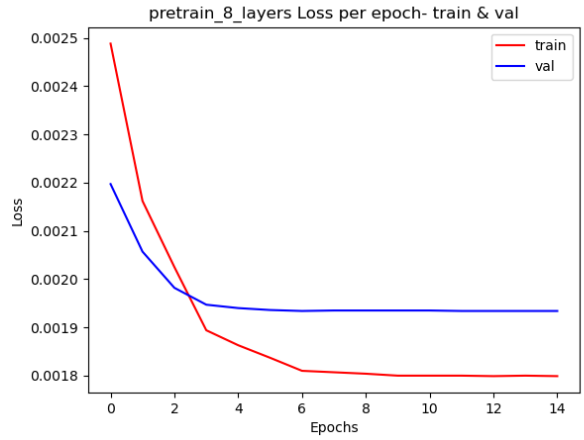












תהליך האימון:

Preprocess: ביצענו הליך מקדים בעת יצירת הdatasetים, עבור השאלות, התשובות והתמונות.

<u>התמונות</u>: הומרו לפורמט RGB, נורמלו, עברו resize והועברו לtensors.

<u>השאלות</u>: עברו preprocess לפי הscript שניתן בתרגיל, padding לפי אורך השאלה הכי ארוכה בדטא, ועברו encoding.

<u>התשובות</u>: עברו preprocess לפי הscript שניתן בתרגיל ולאחר מכן עברו סינון כך שהתשובות היחידות שאופציונליות הן רק אלו שהיוו תשובה ללפחות 9 שאלות בTrain. סה"כ קיבלנו 2410 תשובות אפשריות. הליך זה הותיר שאלות עבורן אף תשובה (מבין ה10) אינה עוברת את הסף. הטיפול בכך הוא שמחקנו דגימות כאלו מהtrain (לא נרצה ללמוד מהן) וטעינו עבור דגימות כאלו מהval.

היפר פרמטרים:

Train: num_epochs=15 batch_size=64 lr=1e-3 lr_gamma=0.1 lr_step_size=3

Dataset: resize=224*224 filter_answers=9

Q model: vocab_size=13278 emb_dim=100 hid_dim=512 n_layers=1 dropout=0.3

V model: dims=[3, 32, 32, 64, 64, 128, 128, 256, 256] OR [3, 32, 64, 128, 256] kernel_size=3 padding=1 pool=2 activation=ReLU

Attention model: projected_dim=500

VQA model: activation=ReLU dropout=0.3 out_dim=2410 is_concat=False

פרטי אופטימיזציה: השתמשנו בBCEwithLogist בתור פונקציית loss ובADAM כoptimizer. ביצענו 15 epochs ובחרנו את המודל מהepoch שהניב דיוק val מקסימלי.

סיכום ניסיונות ומסקנות:

<u>סוגי ארכיטקטורות שניסינו</u>:

- מודל בסיסי של VQA כפי שמתואר בהנחיות התרגיל.
- מודל VQA עם attention כפי שמתואר בהרצאה 7: high-order-attention models for visual question answering.
- מודל VQA (בסיסי/ attention) שכולל מודל V שאימנו מראש בעזרת convolutional autoencoder:
בתרגיל נאסר להשתמש במודלים מאומנים מראש, ולכן השתמשנו בautoencoder שלמדנו בהרצאה בשביל אימון מודל תמונות על משימה של reconstruction ולקיחת המשקולות המאומנות למודל הVQA, לfine tuning. בנינו מודל שכולל encoder שהוא שכבות conv2s (משמש בהמשך כמודל V שלנו) ו-decoder שהוא שכבות deconvolution. למעשה, הencoder מצמצם את התמונה וה-decoder בונה אותה בחזרה. הליך האימון התבצע עם MSEloss ושמרנו את המודל בepoch בו חשבנו שכבר הייתה התכנסות בערכי הloss.
- Ensemble של שלושה מודלי VQA הנבדלים בשימוש/ אי שימוש במודל שאימנו מראש ובמספר שכבות הקונבולוציה של מודל V. המוטיבציה לשימוש בשיטה זו היא שזו שיטה ידועה כמוצלחת ומעלה דיוק בתחרויות בסגנון Kaggle. לאחר אימון שלוש המודלים (בנפרד ובאופן בלתי תלוי) על משימת הVQA, בעת הevaluation הכנסנו כל קלט לכל אחד מהמודלים, ומיצענו את הפלט שלהם כממוצע המשוקלל שהניב את התוצאה הגבוהה ביותר- משקל 0.6 למודל שאינו מאומן מראש, ומשקל 1 לשני המודלים הנוספים. הערך הממוצע היווה את הפלט הסופי של המערכת- איתו חישבנו accuracy.

<u>שינויי פרמטרים במודלים (V, Q, vqa (combined ובאופן כללי</u>:

- מודל V (גם במקרה הpre-trained וגם במקרה הרגיל):
  - שינוי מספר שכבות: ניסינו שילובים שונים של כמות שכבות וכמות filters, למשל: [3, 16, 32], [3, 16, 32, 64, 64, 128, 128], [3, 32, 32, 64, 64, 128, 128, 256, 512, 1024]. גילינו שהוספת שכבות מעלה דיוק, אולם ישנו trade off בכך שהגדלת שכבות דורשת ביצוע פחות pooling כדי למנוע מצב של גודל תמונה קטן מדי. לכן, כאשר הגדלנו את מס' השכבות מעל 4, ביצענו pooling כל שכבה שנייה ובשכבה האחרונה.
  - שינויי הפרמטרים של מודל הקונבולוציה: שילובים שונים של kernel_size, padding, stride, maxpooling. הרצנו עם פרמטרים שונים ובחרנו את השילובים שהניבו תוצאות דיוק גבוהות יותר.
  - ניסינו להשתמש במודל conv לא מאומן ובמודל שאימנו מראש בעזרת autoencoder. המודל המאומן מראש הניב דיוק גדול יותר, בערך ב2-1 נקודות דיוק.
  - ניסינו לבצע שילובים שונים של מודלים בensemble: כל אחד משלוש המודלים סיפק תוצאה בין 46.89 לבין 47.81. כאשר שילבנו את שני המודלים המאומנים מראש קיבלנו דיוק של 48.07 וכאשר הוספנו את המודל

השלישי שלא אומן מראש קיבלנו 48.24 (למרות שכביכול ראינו שמודל לא מאומן מראש מניב דיוק נמוך יותר מהמאומנים מראש). השערה לתוצאה זו היא שישנה שונות גדולה יותר בין תוצאות מודל מאומן מראש לעומת לא מאומן מראש והשונות תורמת למודל משוקלל ומחזקת אותו.

- שינוי משקולות המודלים בensemble: כאשר שינינו את המשקולות בעת מיצוע שלוש המודלים הצלחנו לעלות מ48.24 ל48.29.

- מודל Q:
  - ניסינו שילובים שונים של מספר שכבות ושל bidirectional. דו כיווני תרם לנו, מס' שכבות לא השפיע הרבה.
  - ניסינו להשתמש בפונקציה pack_padded_sequence אשר מתמודד עם padded sequences. לדעתנו דרך זו מדויקת יותר, שכן היא עוזרת לLSTM להבין מה אורך המשפט האמיתי כך שהוא מתעלם מהpadding.
  - ניסינו להשתמש במודל attention (מהרצאה 7) עבורו מודל Q החזיר את הייצוג LSTM עבור כל אחת מהמילים, לעומת מודל ללא attention בו הLSTM החזיר ייצוג של המילה האחרונה כמייצגת את המשפט. המודל שכלל attention הניב תוצאות דיוק גבוהות יותר.
  - שינוי גודל מימד הembedding.
- מודל משולב (מקבל את הפלטים של מודלים V ו-Q ומחזיר פרדיקציה):
  - השוונו Concat לפלטי המודלים V ו-Q לעומת Multiply לפלטי המודלים V ו-Q. המכפלה הייתה עדיפה.
  - שינוי הLoss: NLLLoss עם הmajority_class הוביל לתוצאות נמוכות יותר מאשר BCEWithLogitsLoss על הוקטור A כולו, בפער של כ-3 נקודות דיוק, מדיוק של 41 לדיוק של 44.
- כללי:
  - גודל batch: בחירת גודל 16 הניב תוצאות נמוכות יותר מאשר גודל 64.
  - Lr step size: לאחר מספר הרצות בהן הרגשנו שהמודל נתקע במינימום מקומי (עולה במשך כמה epochs ואז נתקע באותו ערך ליתר הepochs), ניסינו להגדיל את גודל הstep size בlr_scheduler . הופתענו לגלות ששינוי זה העלה אותנו ב3 נקודות- מדיוק של 44 לדיוק של 47, כבר בepoch 4.