Simple linear attention language models balance the recall-throughput tradeoff

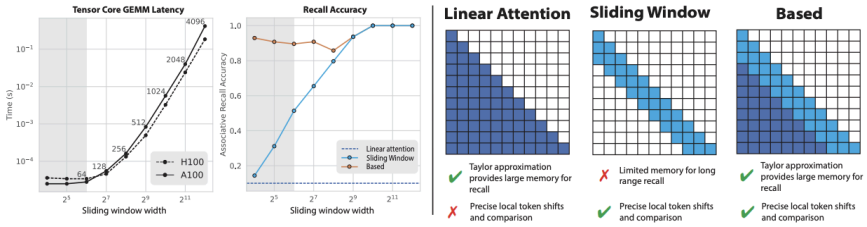

Figure 1: **Based overview.** Combining linear attention with *tiny* sliding window softmax attention (e.g., 64 or 128 tokens in width) enables improved recall accuracy with limited efficiency overhead vs. smaller tile sizes. (*Left*) Time to execute Cutlass GEMMs (y) vs. sliding window attention size (x), with batch size 512 on tensor cores. (*Center*) Model recall accuracy (y) vs. sliding window attention size (x). We compare linear attention alone (dark blue), sliding window attention alone (light blue), and their combination (BASED, orange). (*Right*) Schematic diagram of BASED illustrating how the two components complement each other.

מודלי שפה ענקיים של היום מפגינים יכולת מרשימה של למידת in-context כלומר יכולת לבצע משימות חדשות (שלא אומן עליהם באופן מפורש) בהתבסס על כמה דוגמאות המדגימות (ממחישות) את את המשימה. כמובן דוגמאות אלו מוזנות למודל שפה כפרומפט. המאמר שנסקור היום מדבר על משימת in-context ספציפית הנקראת recall. המטרה של משימה זו היא לזהות חוקיות מסומיות בפרומפט ולענות על שאלות בנוגע אליו. למשל אם פרומפט המוזן הוא "A 4 B 3 C 6 F 1 G 2". אם לאחר מכן אני מכניסים למודל שפה "B ? F ? C?" המודל צריך לענות 6 1 3 כלומר המספר בא מיד אחרי כל אות בפורמפט השאלה.

ארכיטקטורת הטרנספומרים מתמודדת בהצלחה עם משימות recall אך היא מתקשה עם אורכי הקשר (context length) מאוד ארוכים עקב מנגנון self-attention שלהם. ד״א המימושים המודרניים של מנגנון זה (כמו FlashAttention2 ו-Paged-Attention) הם בעלי סיבוכיות subquadratic במונחי אורך הסדרה אך עדיין גם הם מתקשים ״לעכל״ אורכי הקשר ממש ארוכים.

כדי לתת מענה לסוגיה זו הוצעו מספר חלופות למנגנון ה-attention כמו attention לינארי, גישות המבוססות על חלון הזז (sliding window) ובנוסף לאחרונה משפחת ארכיטקטורות ממבה (סקרתי אותן בהרחבה לפני כחודשיים).

מנגנון attention לינארי בגדול מחליף את הסופטמקס של המכפלה הפנימית של וקטורי שאילתה (Q) ווקטורי ערך (K) למכפלה הפנימית של f(Q) ו- f(K) עבור פונקציה לא לינארית f (יש לא מעט מאמרים המציעים לקחת פונקציות f שונות עבור ההחלפה הזו). אחת הדוגמאות היא לבחור f בתור כמה איברים ראשונים של פיתוח טיילור של סופטמקס.

פעולה זו מאפשרת להחליף סדר הפעולות בחישוב ה-attention ולבצע את החישוב באופן לינארי במונחי אורך הסדרה. דרך אגב החלפה זו היא כמו reparameterization trick ב- SVMs אבל בכיוון ההפוך. היא מאפשרת להיפטר מ״גרירה״ של הייצוגים של כל הטוקנים הקודמים באופן מפורש באינפרמס ומאפשרת חישוב בסגנון RNN. כלומר כל הזכרון עד טוקן n נדחס לכדי 2 וקטורים (ממליץ לקרוא על זה כאן) וכמובן זה מאפשר לחסוך במשאבי חישוב הנדרשים לביצוע אינפרנס באופן משמעותי.

מנגנון ה-attention עם החלון הזז הוא פשוט הגבלת גודל ההקשר במנגנון ה-attention כאשר יש מגוון גישות ל"איך לדחוס" את הדאטה שלא נכנסת לחלון זה (העבר). בתוך החלון ה-attention מחושב באופן רגיל כלומר הגדלה משמעותית של חלון זה משפרת את הביצועים אבל גם כרוכה בביצוע של יותר חישובים.

מצד אחד ארכיטקטורות המבוססת על attention לינארי יודעות להסתדר לא רע עם אורכי הקשר ארוכים מאוד במשימות מסוימות אבל מתקשות לספק ביצועים גבוהים לשאלות בסגנון recall. מצד שני ארכיטקטורות הממשות חלון attention זז מסתדרות יפה עם משימות recall בתוך החלון הזה אולם כדי להביא ביצועים גבוהים עם הקשר ארוך צריך להגדיל את גודל החלון שכאמור כרוך בהקצאה של יותר משאבים ואו/גם ב-latencies גבוהים יותר באינפרנס.

אוקיי דיברנו הרבה על הרקע למאמר אז הגיע הזמן לדבר על מאמר עצמו. קודם כל החמברים מוכיחים באופן תיאורטי (את הקטע הזה הכי אהבתי כאן) כי ככל שאורך הקלט למשימת recall "המודל צריך לזכור" O(N) "מידע" כאשר N הוא "אורך" של פרומפט ה-recall (זה גם נבדק אמפירית). כלומר זה תקף לכל ארכיטקטורה והשאלה היחידה איך כל מודל (למשל טרנספורמר לינארי, hyena, mamba, s3 ועוד) בונים ומנהלים את הזכרון הזה ואיך הוא משפיע על ביצועי אינפרנס.

לגבי החידוש שהמאמר מציע: המחברים שילבו את ה"טוב" שיש במנגנון ה-attention הלינארי ובגישת החלון הזז והציעו מנגנון attention חדש הנקרא Based. הם לקחו מנגנון ה-attention הלינארי החסכוני והיעיל מבחינת ניהול הזכרון והוסיפו לו חלון זז קצר יחסית הממשמש מנגנון attention רגיל של הטרנספורמים. וזה עבד להם לא רע בכלל במשימות recall שונות המצריכות חלון הקשר גדול. בנוסף גם הציעו מספר שכלולים לשיטה זו המאפשרים להריץ אותה בצורה מאוד יעילה על GPUs (למשל בחירת גודל החלון כדי שיהיה ניתן לבצע את החשובים עבור על ידי שימוש רק הזכרון המהיר של GPU.

בסך הכל מאמר די נחמד...

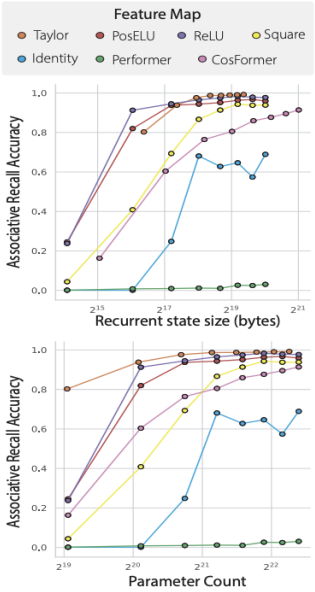

Figure 3: **Linear attention feature maps on AR.** x: state size (bytes) during generation or param. count; y: MQAR accuracy. This setting is harder than fig. 2 (256 key-value pairs).