# Review 190: Hungry Hungry Hippos: Towards Language Modeling with State Space Models(H3)

**Paper: https://arxiv.org/abs/2212.14052v3**

https://arxiv.org/abs/2212.14052

עד עכשיו ראינו מאמרים שמימשו את ארכיטקטורת SSM בתור רכיב הזכרון של המערכת. אף אחת מהמאמרים שסקרנו לא ניסה לשלב גישה זו(SSM) יחד עם מנגוננים אחרים שמוכרים לנו מעולם של עיבוד סדרות דאטה עם רשתות נוירונים. המאמר המסוקר משלב את גישת SSM, המיושמת באמצעות מערכות דינמיות לינאריות, עם מנגנון תשומת הלב הלינארי.

דיברנו על מנגנון attention הלינארי בסקירה השלישית של המאמר: Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. המאמר הזה הציע להחליף את מנגנון תשומת הלב הרגיל עם softmax של הטרנספורמרים בחישוב לינארי: (f(k)\*f(q כאשר \* מסמן מכפלה פנימית ו- f היא פונקציה לא לינארית. המאמר מראה כי ניתן לתאר טרנספורמר עם מנגנון זה בתור RNN ולהימנע מסיבוכיות חישוב ריבועית הרגילה של הטרנספורמרים. כלומר אין צורך להתחשב בצורה מפורשת בכל פיסות הדאטה לפני טוקן i בשביל לחזות אותו אלא כל הזיכרון של הטוקנים הקודמים נדחס ושמור בשני וקטורים.

אוקיי, אבל למה צריך בעצם לשלב ארכיטקטורות מבוססת SSM עם מנגנונים אחרים? התשובה היא פשוטה - ארכיטקטורות אלה לא מספיק טובות לכמה משימות. למשל מחברי המאמר שמו לב כי במשימות כמו Induction Head שצריך לעקוב על טוקן שבא אחרי טוקן מסוים, ארכיטקטורה זו מפגינה ביצועים לא מרשימים במיוחד. כדי להתמודד עם סוגיה זו המחברים הציעו לשלב SSM עם מטריצות A מסוימות עם מנגנון תשומת הלב הלינארי.

אז איך כל הסיפור הזה עובד? בשלב הראשון מכפילים את ייצוגי הטוקנים במטריצות Q, K ו- V כמו בטרנספורמרים. בשלב השני מפעילים SSM על המפתח k (עבור כל הטוקנים) עם מטריצה A המדמה ״זיכרון של הטוקן הקודם״(בערך A\_ij=1 כאשר i - j=1 ואפס אחרת). מבחינת מנגנון תשומת הלב הלינארי זה ״מקביל״ ל (f(k למרות ש f כאן ״די לינארית״.

בשלב השלישי לוקחים q, v והתוצאה של השלב הקודם ל h חתיכות (= ״ראשים״ במנגנון ה-attention). לאחר מכן מכפילים כל חתיכה של q בחתיכה של התוצאה של השלב הקודם (עם k) ו״מעבירים״ את התוצאות דרך SSM עם מטריצה A אלכסונית. את התוצאה מכפילים ב-q, מאחדים את כל התוצאות ומכפילים במטריצה W\_O כמו שמקובל בטרנספורמרים מרובי ראשים(multi-head transformers).

בנוסף המאמר מציע מנגנון הנקרא FlashConv לחישוב חיזוי הטוקנים באופן מקבילי במהלך האימון. כמו שאתם זוכרים הקרנל קונבולוציוני שם מאוד ארוך וחישובו יכול להיות יקר גם מבחינת הזיכרון וגם מבחינת הזמן אם נעשה בצורה נאיבית. המחברים משכללים את המנגנון כאשר העיקרון המוביל הוא ניצול מקסימלי של זיכרון SRAM המהיר שיש ב-GPUs תוך מזעור של הערבות דאטה לשם (זה איטי ובד״כ מהווה צוואר בקבוק) . הזיכרון הזה לא גדול ולא ניתן לדחוף שם יותר מדי אז נדרשות שיטות מתוחכמות המפרקות את חישוב הקונבולוציה לחלקים תוך ניצול תכונות של FFT ו- IFFT. נזכיר שהחישוב הקונבולוציה מתבצע בצורה: ((c(x) = iFFT(FFT(c)\*FFT(x כאשר (c(x היא קונבולוציה על x עם קרנל c.