



دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

پروژه تحقیقاتی درس شبکه‌های عصبی

ماشین تورینگ عصبی

نگارش

علیرضا مازوچی

استاد درس

دکتر رضا صفابخش

تیر ۱۴۰۱

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

چکیده

در این قسمت چکیده پایان نامه نوشته می‌شود. چکیده باید جامع و بیان‌کننده خلاصه‌ای از اقدامات انجام‌شده باشد. در چکیده باید از ارجاع به مرجع و ذکر روابط ریاضی، بیان تاریخچه و تعریف مسئله خودداری شود.

واژه‌های کلیدی:

کلیدواژه اول، ...، کلیدواژه پنجم (نوشتن سه تا پنج واژه کلیدی ضروری است)

فهرست مطالب

صفحه

عنوان

۲	ماشین تورینگ عصبی	۲
۳	۱-۲ ساختار و محاسبات ماشین تورینگ عصبی	۳
۵	۲-۲ وظایف یادگیری ترتیبی	۵
۶	۳ افزونه‌ها	۶
۷	۱-۳ ماشین تورینگ عصبی تکاملی	۷
۸	۲-۳ ماشین تورینگ عصبی ابر تکاملی	۸
۱۲	۳-۳ ماشین تورینگ عصبی متوجه	۱۲
۱۳	۴-۳ ماشین تورینگ عصبی پویا	۱۳
۱۴	۴ کاربردها	۱۴
۱۵	۱-۴ تخمین عمر مفید باقی‌مانده	۱۵
۱۷	۵ پیاده‌سازی و نتایج	۱۷
۱۸	۱-۵ تخمین عمر مفید باقی‌مانده	۱۸
۱۹	۶ جمع‌بندی و نتیجه‌گیری	۱۹
۲۰	منابع و مراجع	۲۰

شکل	فهرست اشکال	صفحه
۱-۳	نحوه کارکرد شبکه تات [۳]	۱۰
۲-۳	بستر طراحی شده برای شبکه تات در مدل مائع ابرتکاملی [۳]	۱۱
۱-۴	معماری پیشنهادی برای کاربرد تخمین عمر مفید در کار تحقیقاتی فالکن و همکاران .	۱۶

صفحه	فهرست جداول	جدول
۱۸	۱-۵	نتایج کار تحقیقاتی فالکن و همکاران برای تخمین عمر مفید باقی مانده [۲]

فصل اول

مقدمه

فصل دوم

ماشین تورینگ عصبی

۱-۲ ساختار و محاسبات ماشین تورینگ عصبی

ماتع شامل دو جز است:

۱. کنترل گر شبکه که می تواند یک شبکه عصبی جلورو یا یک شبکه عصبی بازگشتی باشد.
 ۲. یک واحد حافظه خارجی که یک ماتریس حافظه $N * W$ است. N تعداد واحدهای حافظه و W ابعاد هر سلول حافظه را نمایش می دهد. [۱]
- فارغ از آنکه کنترل گر بازگشتی باشد یا خیر، کل معماری بازگشتی محسوب می شود چراکه ماتریس حافظه در طول زمان نگهداری می شود. کنترل گر سرهای خوانده و نوشتن دارد که به ماتریس حافظه دسترسی دارد. تاثیر یک عمل خواندن یا نوشتن روی یک سلول حافظه خاص با مکانیسم توجه نرم^۱ وزن دهی می شود. این مکانیسم آدرس دهی مشابه مکانیسم توجه استفاده شده در یادگیری ماشین عصبی است به جز آنکه آدرس دهی وابسته به موقعیت را با آدرس دهی وابسته به محتوای موجود در مکانیسم توجه را ترکیب می کند. [۱]

به طور خاص برای یک ماتع در هر گام زمانی t برای هر سر خواندن و نوشتن کنترل گر یک تعدادی پارامتر را به عنوان خروجی می دهد. این پارامترها برای محاسبه وزن w_t بر روی N خانه حافظه در ماتریس حافظه M_t استفاده می شوند. نحوه محاسبه w_t در رابطه ۱-۲ آورده شده است. [۱]

$$w_t^c(i) \leftarrow \frac{\exp(\beta_t K[k_t, M_t(i)])}{\sum_{j=0}^{N-1} \exp(\beta_t K[k_t, M_t(j)])} \quad (1-2)$$

در رابطه ۱-۲ برخی از پارامترها دارای محدودیت هایی هستند: [۱]

- $\beta_t \leq 0$ •
- $g_t \in [0, 1]$ •
- $\sum_k s_t(k) = 1$ •
- $\forall_k s_t(k) \leq 0$ •
- $\gamma_t \leq 1$ •

در رابطه ۱-۲ w_t^c آدرس دهی وابسته به محتوا را فراهم می کند. k_t یک کلید جستجو در حافظه را نشان می دهد و K مطابق رابطه ۲-۲ یک معیار شباهت مانند شباهت کسینوسی است. [۱]

^۱Soft Attention Mechanism

$$K[u, v] = \frac{uv}{||u|| \cdot ||v||} \quad (2-2)$$

با یک سری از محاسبات مطابق روابط ۲-۳، ۲-۴ و ۲-۵ ماترها امکان تکرار بر روی وزن‌های حافظه فعلی و قبلاً محاسبه‌شده را خواهند داشت. رابطه ۲-۳ به شبکه اجازه می‌دهد تا بین بردار وزن قبلی یا فعلی انتخاب کند که از کدام استفاده کند. رابطه ۲-۴ امکان تکرار از طریق حافظه با عمل کانولوشن وزن فعلی و یک کرنل کانولوشنی جابجایی^۲ یک بعدی را فراهم می‌کند. رابطه ۲-۵ رخداد تارشدن^۳ که به واسطه عمل کانولوشن رخ داده است را اصلاح می‌کند. [۱]

$$w_t^g \leftarrow g_t w_t^c + (1g_t)w_{t1} \quad (3-2)$$

$$\tilde{w}_t(i) \leftarrow \sum_{j=0}^{N-1} w_t^g(j) s_t(ij) \quad (4-2)$$

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_{j=0}^{N-1} \tilde{w}_t(j)^{\gamma_t}} \quad (5-2)$$

سپس بردار r_t مطابق رابطه ۲-۶ به وسیله‌ی یک سر^۴ خواندن خاص در زمان t محاسبه می‌گردد. [۱]

$$r_t \leftarrow \sum_{i=0}^{N-1} w_t(i) M_t(i) \quad (6-2)$$

نهایتاً مطابق رابطه ۲-۷ و ۲-۸ هر سر نوشتن ماتریس حافظه را در گام t با محاسبه بردارهای جانبی پاک کردن یعنی e_t و جمع کردن یعنی a_t تغییر می‌دهد. [۱]

²Shift

³Blurring

⁴Head

$$\tilde{M}_t(i) \leftarrow M_{t-1}(i)[1 - w_t(i)e_t] \quad (۷-۲)$$

$$M_t(i) \leftarrow \tilde{M}_t(i) + w_t(i)a_t \quad (۸-۲)$$

۲-۲ وظایف یادگیری ترتیبی

برای ماتع‌ها چندین وظیفه مصنوعی در نظر گرفته شده است که تمام آن‌ها از نوع مسئله یادگیری ترتیبی^۵ است؛ زمینه‌ای که آن‌ها در آن توانمند هستند. این وظایف عبارت‌اند از:

- رونوشت‌گیری^۶: برای وظیفه رونوشت‌گیری یک دنباله تصادفی از بردارهای بیت با یک نماد خاص به عنوان پایان دنباله به شبکه داده می‌شود. این وظیفه نیاز دارد تا شبکه دنباله ورودی را نگه دارد و سپس آن را از حافظه برگرداند.
- رونوشت‌گیری تکرارشونده^۷: مشابه وظیفه رونوشت‌گیری دنباله‌ای از بردارهای بیتی تصادفی به شبکه داده می‌شود. برخلاف وظیفه رونوشت‌گیری بعد از دنباله یک عدد که نشان دهنده تعداد دفعاتی است که باید دنباله در خروجی ظاهر شود به شبکه داده می‌شود.
- یادآوری انجمنی^۸: این وظیفه نیز یک مسئله یادگیری دنباله با دنباله‌های متشکل از بردارهای بیتی تصادفی است. در این مورد ورودی به چندین عنصر تقسیم می‌شود که هر کدام شامل بردارهای ۶×۳ بعدی است. بعد از آنکه یک دنباله از آیتم‌ها و نماد پایانی دنباله به شبکه داده می‌شود. خروجی صحیح عنصر بعدی دنباله ورودی بعد عنصر کوثری است. [۱]

^۵Sequence Learning

^۶Copy

^۷Repeat Copy

^۸Associative Recall

فصل سوم

افزونه‌ها

۱-۳ ماشین تورینگ عصبی تکاملی

تکامل عصبی توپولوژی‌های تقویت‌کننده^۱ که در ادامه آن را به اختصار تکتوت می‌نامیم با یک جمعیت^۲ از شبکه‌های عصبی ساده شروع می‌کند و سپس آن‌ها را در طی نسل‌ها^۳ با افزودن رئوس جدید و اتصالات به کمک جهش^۴ پیچیده‌تر می‌کند. با تکامل شبکه‌ها از این را لازم نیست توپولوژی شبکه‌ها از پیش دانسته شده باشد. تکتوت به طرز فزاینده‌ای در شبکه‌های پیچیده جستجو می‌کند تا یک سطح مناسب از پیچیدگی را پیدا کند. ویژگی مهم تکتوت این است که هم توپولوژی و هم وزن‌های یک شبکه را تکامل می‌دهد. چراکه به سادگی و تدریجی پیچیدگی را افزایش می‌دهد و این باعث می‌شود که یک شبکه مناسب با اندازه مینیمال حاصل شود.^[۳]

بر پایه ماتع و با استفاده از تکتوت مدل ماشین تورینگ عصبی تکاملی^۵ که در ادامه آن را به طور اختصار ماتع تکاملی می‌نامیم معرفی شده است. در این روش توپولوژی و وزن‌های شبکه عصبی کنترل‌گر با کمک تکتوت یاد گرفته می‌شود. بنابراین برخلاف ماتع استاندارد نیاز به دانش پیشین^۶ نیست و شبکه می‌تواند با توجه به پیچیدگی وظیفه رشد پیدا کند. ماتع تکاملی اغلب توپولوژی‌های فشرده برای حل یک وظیفه خاص پیدا می‌کند؛ در نتیجه جلوی جستجوی غیرضروری در فضای با ابعاد بالا گرفته می‌شود. به علاوه ماتع تکاملی قادر به حل مسائل یادگیری مستمر پیچیده است. چراکه شبکه از مشتق استفاده نمی‌کند و می‌تواند از توجه سخت^۷ و مکانیسم جابجایی^۸ استفاده کند که امکان تعمیم خوب برای دنباله‌های بلند در وظیفه رونوشت‌گیری را فراهم می‌کند. به علاوه یک نوار^۹ پویا و از نظر تئوری با اندازه بی‌نهایت قابل استفاده است.^[۳]

ماتع تکاملی یک سر تکی ترکیبی خواندن/نوشتن دارد. این شبکه بردار نوشتن w با اندازه M ، ورودی کنترل درون‌یابی^{۱۰} نوشتن i ، ورودی کنترل پرش محتوا j و سه ورودی کنترل جابجایی s_l (جابجایی چپ)، s_0 (بدون جابجایی) و s_r (جابجایی راست) را خروجی می‌دهد. اندازه بردار نوشتن M اندازه هر خانه حافظه بر روی نوار را مشخص می‌کند. جزء درون‌یابی نوشتن امکان مخلوط کردن مقادیر فعلی نوار و بردار نوشتن در موقعیت نوشتن را فراهم می‌کند. $M_h(t)$ محتوای نوار در موقعیت سر فعلی h در زمان t ، درون‌یابی نوشتن و w_t بردار نوشتن است. برای تمام این‌ها در زمان t رابطه ۱-۳ را خواهیم

¹Neuroevolution of Augmenting Topologies (NEAT)²Population³Generations⁴Mutation⁵Evolvable Neural Turing Machine⁶Prior Knowledge⁷Hard Attention⁸Shift Mechanism⁹Tape¹⁰Interpolation

داشت. [۳]

$$M_h(t) = M_h(t1)(1i_t) + w_t i_t \quad (۱-۳)$$

پرش محتوا مشخص می‌کند که آیا سر باید به موقعیتی در حافظه حرکت کند که بیشترین شباهت را به بردار نوشتن دارد یا نه. یک پرش محتوا انجام می‌شود اگر مقدار ورودی کنترل از ۵.۰ بیشتر شود. شباهت بین بردار نوشتن w و بردار حافظه m مطابق با رابطه ۲-۳ حساب می‌شود. [۳]

$$s(w, m) = \frac{\sum_{i=1}^M |w_i m_i|}{M} \quad (۲-۳)$$

در گام زمانی t اقدامات زیر به ترتیب انجام می‌شود:

۱. بردار نوشتن w_t برای موقعیت فعلی h بدست می‌آید. این بردار با محتوای موجود باتوجه به درونیابی نوشتن i_t درونیابی می‌شود.
۲. اگر ورودی کنترل پرش محتوا j_t بزرگ‌تر از ۵.۰ شود، سر به مکانی در نوار که بیشترین شباهت به بردار نوشتن w_t دارد حرکت می‌کند.
۳. سر به یک موقعیت چپ‌تر، راست‌تر روی نوار حرکت می‌کند یا در همان جا ثابت می‌ماند که این وابسته به مقادیر ورودی کنترل جابجایی s_l ، s_0 و s_r است.
۴. مقادیر نوار را در موقعیت جدید سر می‌خواند و بر می‌گرداند. [۳]

۲-۳ ماشین تورینگ عصبی ابر تکاملی

در کدگذاری‌های مستقیم مانند تکتوت هر بخش از نمایش جواب به یک تکه کوچک از ساختار نهایی جواب نگاشت می‌شود. عیب مهم این روش آن است که بخش‌های مختلف راه‌حل که به یک‌دیگر شبیه هستند نیز باید کد شوند و جداگانه کشف شوند. این ابراد با کدگذاری غیرمستقیم تا حد زیادی قابل حل است؛ در کدگذاری غیرمستقیم راه‌حل به شکل فشرده توصیف می‌شود و حجم اطلاعات کدشده می‌تواند کاهش بیابد. در کدگذاری غیرمستقیم به دلیل آنکه یک راه‌حل به شکل الگویی از پارامترها و نه تمام پارامترها نمایش پیدا می‌کند قدرت‌مند است. [۳]

روش ابر تکامل عصبی توپولوژی‌های تقویت‌کننده^{۱۱} که در ادامه به اختصار آن را ابرتوت می‌نامیم

^{۱۱}Hyper Neuroevolution of Augmenting Topologies(HyperNEAT)

یک افزونه از تکتوت است. در این افزونه به جای کدگذاری غیرمستقیم از کدگذاری مستقیم استفاده می‌شود. در تکتوت از شبکه‌های عصبی معمولی استفاده می‌شود درحالی که در ابرتوت از شبکه‌های تولید الگوی ترکیبی^{۱۲} که در ادامه آن را به اختصار شبکه تات می‌نامیم استفاده شده است. شبکه تات برای کدگذاری ترکیب توابع طراحی شده‌اند که هر تابع در ترکیب مرتبط با یک منظم‌سازی^{۱۳} است.^[۳]

حسن شبکه تات آن است که به الگوهای مکانی اجازه می‌دهد که به عنوان شبکه‌هایی از توابع ساده نمایش پیدا کنند. این یعنی تکتوت می‌تواند با شبکه تات مانند شبکه‌های عصبی تکامل پیدا کند. شبکه‌های تات مشابه شبکه‌های عصبی هستند با این تفاوت که آن‌ها متکی بر بیشتر از یک تابع فعال‌سازی هستند. کدگذاری غیرمستقیم شبکه تات می‌تواند به طور فشرده الگوها با نظم‌هایی نظیر تقارن^{۱۴}، تکرار^{۱۵} و تکرار با تغییر^{۱۶} را کد کنند. به عنوان مثال با انتخاب یک تابع گاوسین که خاصیت تقارن دارد الگوی خروجی نیز به سادگی متقارن خواهد شد. انتخاب یک تابع دوره‌ای مانند سینوس در حین تکرار قطعه‌سازی انجام می‌دهد. نهایتاً تکرار با تغییر به سادگی با ترکیب یک تابع منظم (مانند سینوس یا گاوسین) با یک تابع نامنظم (مانند محور x نامتقارن) بدست می‌آید.^[۳]

ایده اصلی ابرتوت آن است که شبکه‌های تات می‌تواند اتصال الگوها را کد کند. بدین طریق یک تکتوت می‌تواند یک شبکه تات که شبکه‌های عصبی بزرگ با منظم‌سازی‌ها و تقارن‌های خود نمایش می‌دهد را تکامل دهد.^[۳]

عملکرد شبکه تات در تصویر ۱-۳ آورده شده است. شبکه‌های تات سنتی توابع هندسی هستند که الگوهای اتصال خروجی آن رئوسی در n بعد است که n تعداد ابعاد در فضای کارترین است. یک شبکه تات که چهار ورودی با برچسب‌های x_1, y_1, x_2, y_2 را دریافت کند و به عنوان خروجی مشخص می‌کند که اتصال بین نقاط دوبعدی (x_1, y_1) و (x_2, y_2) چه میزان است. بنابراین با داشتن یک شبکه تات آموزش یافته می‌توان با ارسال یک کوئری شامل هر دو راس در شبکه عصبی مقدار اتصال آن را بدست آورد و شبکه عصبی را ایجاد کرد.^[۳]

همانطور که در توضیحات قبل بدان اشاره شد باید تمام گره‌های شبکه عصبی در یک بستر^{۱۷} قرار داده شوند. یعنی مشخص شود که هر گره در شبکه عصبی در چه مختصاتی در فضا باید قرار بگیرد. مشابه چیزی که در سمت چپ تصویر ۱-۳ قابل مشاهده است. ایده اصلی در مانتع ابرتکاملی پیشنهاد یک بستر برای بهره‌مندی از شبکه تات در مانتع تکاملی بوده است.

¹²Compositional Pattern Producing Networks (CPPN)

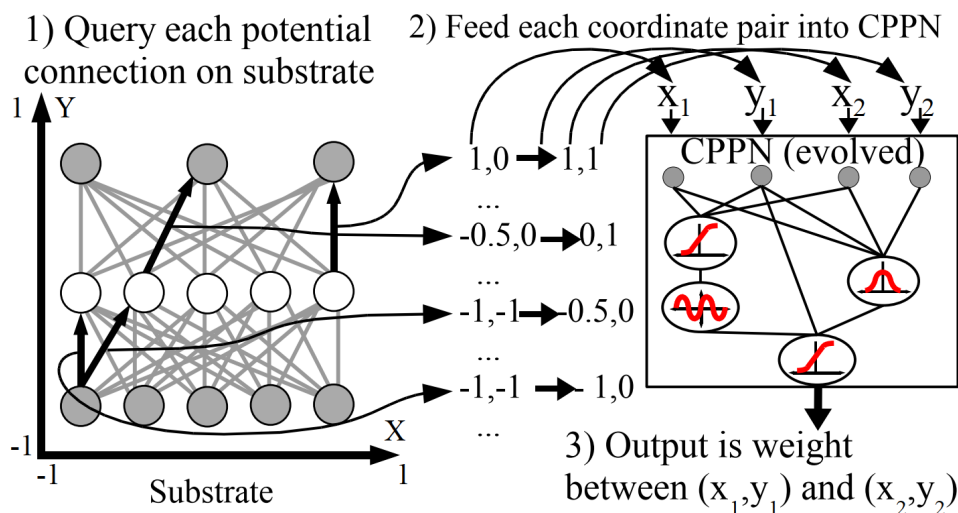
¹³Regularity

¹⁴Symmetry

¹⁵Repetition

¹⁶Repetition with Variation

¹⁷Substrate



شکل ۳-۱: نحوه کارکرد شبکه تات [۳]

مجموعه‌ای از گره‌ها به مختصات بین -1 تا $+1$ در تمام ابعاد نظیر می‌شوند. (۱): هر اتصال ممکن در یک شبکه عصبی کوثری زده می‌شود تا مجاورت و وزن آن مشخص شود. خطوط جهت‌دار تیره نمایش داده شده در تصویر یک نمونه از اتصالاتی است که کوثری زده شده است. (۲): در درون یک شبکه تات یک گراف است که مشخص می‌کند کدام توابع فعال‌سازی به یکدیگر متصل هستند. همان‌طور که در شبکه عصبی اتصالات وزن‌دهی می‌شوند که خروجی یک تابع با چه وزنی به طرف دیگر اتصال برود، برای هر کوثری ارسال شده به شبکه تات جایگاه دو سر اتصال را به عنوان ورودی می‌گیرد و وزن اتصال را به عنوان خروجی می‌دهد. (۳): بنابراین شبکه تات می‌تواند الگوهای منظم از اتصالات در فضا را تولید کند.

در مدل مائع ابر تکاملی شبکه تات نه تنها اتصال ورودی‌ها و خروجی‌های شبکه عصبی مرتبط با وظیفه را مشخص می‌کند بلکه اینکه اطلاعات آمده از حافظه چگونه باید در شبکه ادغام شوند و چگونه اطلاعات در حافظه نوشته شوند را هم مشخص می‌کند. زیرا ابرتوت که می‌تواند هندسه یک وظیفه را یاد بگیرد قانداً باید بتواند الگوی هندسی اطلاعات خوانده شده از و نوشته شده در حافظه را نیز یاد بگیرد. [۳]

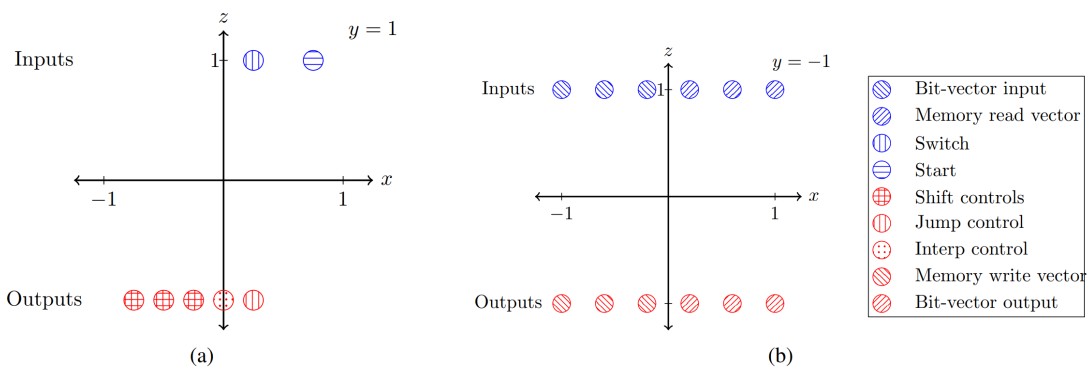
شبکه مائع تکاملی ورودی‌های زیر را دارد:

- شروع: ورودی که هرگاه فعال می‌شود، ذخیره اعداد شروع می‌شود.
- تعویض: ورودی که هرگاه فعال شود، ذخیره اطلاعات خاتمه می‌یابد و شبکه باید مقادیر به خاطر سپرده شده را به یاد بیاورد.
- ورودی بردار بیتی: بردار بیتی که شبکه به عنوان ورودی می‌گیرد. توجه کنید قبل از آنکه ورودی تعویض فعال شود رنج این ورودی با بیت‌هایی که بعداً می‌خواند فعال می‌شود.
- ورودی خواندن حافظه: بردار حافظه که ماشین تورینگ در گام قبل خوانده است. [۳]

این شبکه خروجی‌های زیر را هم دارد:

- خروجی بردار بیتی: بردار بیتی که شبکه به عنوان خروجی می‌دهد. توجه کنید در حین دریافت ورودی این خروجی نادیده گرفته می‌شود.
- خروجی نوشتن حافظه: بردار حافظه‌ای که باید در حافظه نوشته شود.
- کنترل گره‌های ماشین تورینگ: خروجی‌های کنترل مخصوص ماشین تورینگ یعنی پرش، درون‌یابی و سه کنترل جابجایی (چپ، راست و توقف) [۳]

در ادامه بستر طراحی شده برای وظیفه رونوشت‌گیری آورده می‌شود. این بستر در تصویر ۲-۳ نشان داده شده است. این بستر طراحی شده است که گره‌های ورودی بردار بیتی مختصات x را با گره‌های نوشتن بردار حافظه به اشتراک بگذارد و بالعکس با گره‌های خواندن بردار حافظه و گره‌های خروجی بردار بیتی. به علاوه ورودی تعویض مختصات x اش را با خروجی پرش به اشتراک می‌گذارد بنابراین شبکه را می‌تواند وادار به پرش به حافظه‌ای کند که خواندن را از آن شروع کرده است. در این مقاله اندازه بردارهای حافظه برابر با اندازه بردار بیتی است. به علاوه هیچ یک از بسترها شامل گره‌های مخفی مانند آن چیزی که نشان داده شده است و ممکن است مسائل با اندازه‌های بزرگ‌تر بدون گره مخفی را حل کند نیست. [۳]



شکل ۲-۳: بستر طراحی شده برای شبکه تات در مدل ماتع ابرتکاملی [۳]

تمام ورودی‌ها در $z = 1$ و تمام خروجی‌ها در $z = -1$ هستند. قسمت الف تمام گره‌ها در $y = 1$ را نشان می‌دهد که ورودی‌های شروع، تعویض و کنترل گره‌های ماشین تورینگ هستند. لازم به ذکر است که مختصات x برای ورودی تعویض و خروجی کنترل گر پرش یکسان است. قسمت ب گره‌ها را در $y = -1$ نشان می‌دهد که ورودی و خروجی‌های بردار حافظه و بردار بیتی را نشان می‌دهد. گره‌های ورودی بردار بیتی مختصات x را با گره‌های نوشتن بردار حافظه به اشتراک می‌گذارد، درحالی‌که گره‌های خواندن بردار حافظه مختصات x را با گره‌های خروجی بردار بیتی به اشتراک می‌گذارد. [۳]

در کنار خروجی شبکه تات که وزن هر اتصال را مشخص می‌کند، هر شبکه تات یک خروجی تابع

قدم^{۱۸} اضافه دارد که خروجی بیان پیوند نامیده می‌شود. این خروجی مشخص می‌کند که آیا یک اتصال باید بیان شود یا خیر. اتصالات بالقوه برای هر ورودی در لایه‌های $y = 1$ و $y = -1$ به هر خروجی در لایه‌های $y = 1$ و $y = -1$ کوئری زده می‌شود. تعداد ورودی‌ها و خروجی‌ها در لایه $y = -1$ مطابق قسمت ب تصویر ۳-۲ وابسته به اندازه بردار بینی وظیفه رونوشت‌گیری است، که در مثال نشان داده شده اندازه بردار بیتی برابر با ۳ است. نوروها به شکل یکنواخت در بازه‌های -1 تا 2.0 در مختصات x برای ورودی‌های بردار بیتی و بردار نوشتن حافظه استفاده می‌شود و بازه 2.0 تا 1 برای بردار نوشتن حافظه و خروجی بردار بیتی استفاده می‌شود.^[۳]

نهایتاً برای تعیین میزان بایاس یک گره باید مختصات همان گره را هم به عنوان گره مبدا و هم به عنوان گره مقصد به شبکه تات کوئری زد.^[۳]

۳-۳ ماشین تورینگ عصبی متوجه

در مدل ماتع متوجه حافظه با یک مکانیسم آدرس‌دهی وابسته به محتوا بازیابی می‌شود. آدرس‌دهی برپایه محتوا به صورت ضروری یک گام محاسبه شباهت میان بردار خروجی کنترل‌گر C_t و بردارهای حافظه موجود M_t است. وزن توجه در خواندن با رابطه ۳-۳ تولید می‌شود. در رابطه مذکور متغیر β می‌تواند دقت تمرکز را افزایش یا کاهش دهد و معیار شباهت استفاده‌شده معیار شباهت کسینوسی است.^[۴]

$$w_t^r(i) = \frac{\exp(\beta \cdot \text{sim}(C_t, M_t(i)))}{\sum_j (\exp(\beta \cdot \text{sim}(C_t, M_t(j))))} \quad (3-3)$$

در حالت کلی کنترل‌گر می‌تواند با هر شبکه عصبی مصنوعی طراحی شود. در ماتع متوجه از یک لایه توجه چندسرها^{۱۹} استفاده کرده‌اند تا روابط بین دنباله‌های ورودی و خروجی را مدل کنند.^[۴]

رابطه مربوط به لایه توجه در رابطه ۳-۴ آورده شده است. در این رابطه مطابق معمول K ، Q و V به ترتیب ماتریس کوئری، کلید و مقدار است و n تعداد ابعاد است. توجه بر دنباله ورودی I اعمال می‌شود تا یک جمع وزن‌دار برای هر i_t بر پایه i_1 تا i_t اعمال شود که t زمان فعلی است.^[۴]

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{n}}\right) \quad (4-3)$$

¹⁸Step Function

¹⁹Multi-Headed Attention

به عنوان خروجی مدل یک بردار وزن مرتبط با هر تلاش را یاد می‌گیرد و بردار وزن یادگرفته‌شده برای محاسبه پیش‌بینی خروجی i_t استفاده می‌شود. از تابع خطا میانگین مربعات خطا مطابق رابطه ۳-۵ استفاده می‌شود. در این رابطه a_t^k مقدار پیش‌بینی، gt_t^k مقدار واقعی برچسب، b اندازه دسته ^{۲۰}، k طول دنباله و t زمان است. [۴]

$$L = \sum_b \sum_{k=1}^n \sum_{t=1}^T MSE(a_t^k, gt_t^k) \quad (۵-۳)$$

در ماتع متوجه حافظه با مکانیسم آدرس‌دهی برپایه موقعیت بروز می‌شود. مکانیسم آدرس‌دهی برپایه موقعیت برای چندین گام طراحی شده است تا تکرار بر روی خانه‌های حافظه و پرش‌های دسترسی تصادفی را امکان‌پذیر کند. گام اول محاسبه درون‌یابی بین بردار وزن نوشتن پیشین w_{t-1} و بردار وزن محتوا تولیدشده توسط مکانیسم آدرس‌دهی محتوا w_t^c در گام زمانی فعلی با استفاده از دروازه ^{۲۱} درون‌یابی w_t^g مطابق رابطه ۳-۶ است. [۴]

$$w_t^g = g_t w_t^c + (1g_t)w_{t1} \quad (۶-۳)$$

بعد از درون‌یابی یک وزن‌دهی جابجایی s_t بر ماتریس وزن دروازه‌دار با یک کانولوشن دایره‌ای برای تنظیم حافظه مطابق رابطه ۳-۷ اعمال می‌شود. در این رابطه N برابر با اندازه حافظه است. نهایتاً مطابق رابطه ۳-۸ عمل تیزکردن ^{۲۲} برای نرمال‌سازی استفاده می‌شود. [۴]

$$w_t^{ro} = \sum_{j=0}^{N-1} w_t^g(j) s_t(i-j) \quad (۷-۳)$$

$$w_t(i) = \frac{w_t^{ro}(i)}{\sum_j w_t^{ro}(j)} \quad (۸-۳)$$

۴-۳ ماشین تورینگ عصبی پویا

²⁰Batch

²¹Gate

²²Sharpen

فصل چهارم

کاربردها

ماتع و افزونه‌هایش در سال‌های اخیر کاربردهای گوناگونی پیدا کرده‌اند. در این بخش قصد داریم به تعدادی از کاربردهای آن بپردازیم.

۱-۴ تخمین عمر مفید باقی‌مانده

فالکن^۱ و همکاران [۲] در تحقیقی عمر مفید باقی‌مانده وسایل مکانیکی استفاده شده در حوزه درمان و بهداشت را بررسی کرده‌اند. تخمین عمر مفید یک وسیله مکانیکی یکی از مسائل مهم در حوزه مدیریت سلامت و پیشگیری است. توانایی تخمین قابل اطمینان بودن آن منجر به بهبود در برنامه‌ریزی نگهداری و کاهش هزینه‌های مرتبط با آن می‌شود. در دسترس بودن سنسورهای با کیفیت بالا که چندین جنبه از اجزا را می‌سنجد این امکان را فراهم می‌کند که حجم زیادی از داده‌ها جمع‌آوری شود که این داده‌ها می‌تواند در تنظیم کردن مدل‌های برپایه داده^۲ استفاده شود. [۲]

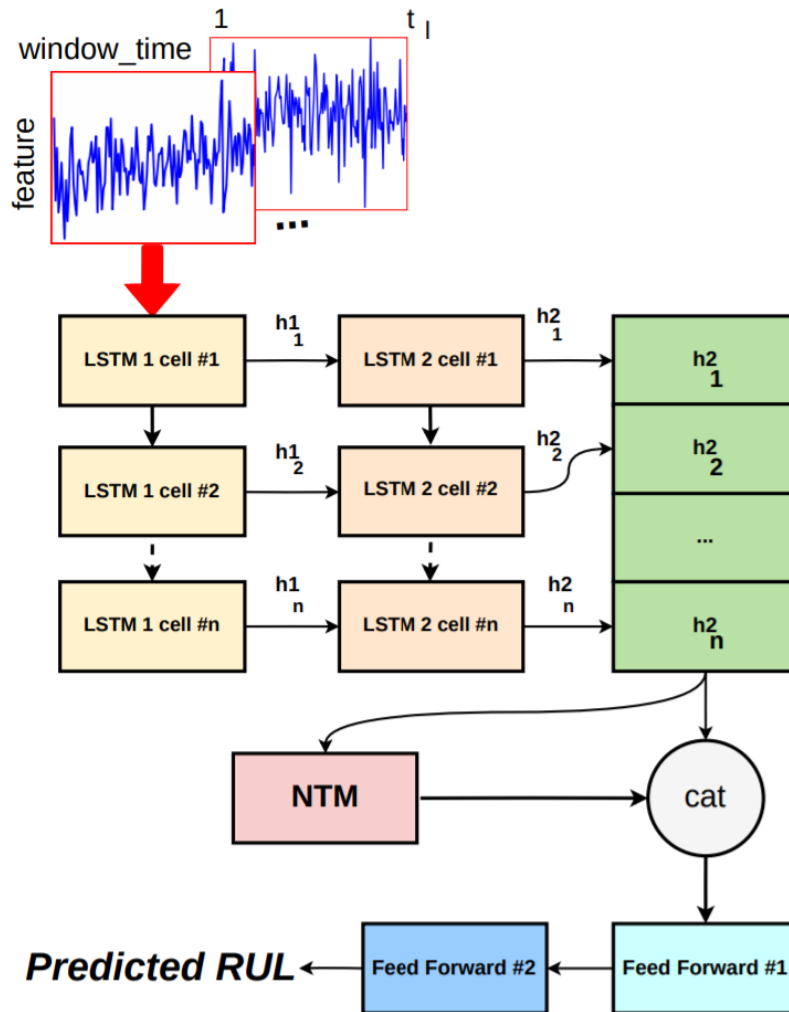
معماری مدل استفاده‌شده در تصویر ۱-۴ آورده شده است. داده‌ای که در حوزه بهداشت و پیشگیری استفاده می‌شود معمولاً مقادیر اندازه‌گیری‌شده طولانی‌مدت سری‌های زمانی^۳ حس‌گرها هستند. سری زمانی‌های خام ورودی با پیش‌پردازش تبدیل به پنجره سری‌زمانی‌های کوچک‌تر می‌شوند. هر کدام از این پنجره‌ها به عنوان ورودی به یک شبکه از دو لایه پشته‌شده LSTM به عنوان ورودی داده می‌شود و یک دنباله از ویژگی‌های استخراج‌شده حاصل می‌گردد. این ویژگی‌ها با ویژگی‌های خروجی یک ماتع الحاق می‌شود. نتیجه نهایی بعد از عبور از یک شبکه جلورو دو لایه پشته‌شده نهایی حاصل می‌گردد. [۲]

فالکن و همکاران معتقدند که وجود یک ماتع می‌تواند کمک به فهم بهتر الگوهای مخفی در داده‌ها و ذخیره‌سازی آن شود. در پژوهش آن‌ها کنترل‌گر ماتع را از نوع شبکه‌های جلورو برگزیدند. [۲] بهبود نتایج آن‌ها که در بخش پنجم به آن اشاره خواهد شد اثبات‌کننده نقش مثبت ماتع در مقاله آنان است.

¹Falcon

²data-driven

³Time Series



شکل ۴-۱: معماری پیشنهادی برای کاربرد تخمین عمر مفید در کار تحقیقاتی فالکن و همکاران
 سری‌های زمانی ابتدا به پنجره‌های کوچک‌تر می‌شکنند سپس به عنوان ورودی به شبکه داده می‌شوند. ورودی از دو لایه LSTM پشته‌شده می‌گذرد و سپس با خروجی یک مانتع الحاق می‌شود. در نهایت شبکه جلورو پشته‌شده مقدار تخمین عمر مفید را ارائه می‌دهند.

فصل پنجم

پیاده‌سازی و نتایج

جدول ۵-۱: نتایج کار تحقیقاتی فالکن و همکاران برای تخمین عمر مفید باقی‌مانده [۲]

مدل	امتیاز	ریشه میانگین مربعات خطا
LSTM	۳۳۹	۱۶/۱۶
LSTM + ماتع	۲۴۲	۱۲/۵۰

۵-۱ تخمین عمر مفید باقی‌مانده

در جدول ۵-۱ بخشی از نتایج مربوط به کار تحقیقاتی فالکن و همکاران که مدلی بر پایه ماتع برای تخمین عمر مفید باقی‌مانده وسایل مکانیکی پیشنهاد داده بودند آورده شده است. آن‌ها برای ارزیابی مدل از یک تابع امتیاز که در مقاله‌ای دیگر معرفی شده بود و معیار خطای ریشه میانگین مربعات خطا^۱ استفاده کرده بودند. هر چه این دو معیار برای یک مدل پایین‌تر باشد آن مدل کارتر است. همانگونه که مشخص است استفاده از ماتع در کنار LSTM منجر به کاهش حدود ۲۲ درصدی خطا شده است.

^۱Rooted Mean Square Error (RMSE)

فصل ششم

جمع‌بندی و نتیجه‌گیری

منابع و مراجع

- [1] Collier, Mark and Beel, Joeran. Implementing neural turing machines. In International Conference on Artificial Neural Networks, pages 94–104. Springer, 2018.
- [2] Falcon, Alex, D’Agostino, Giovanni, Serra, Giuseppe, Brajnik, Giorgio, and Tasso, Carlo. A neural turing machine-based approach to remaining useful life estimation. In 2020 IEEE International Conference on Prognostics and Health Management (ICPHM), pages 1–8. IEEE, 2020.
- [3] Merrild, Jakob, Rasmussen, Mikkel Angaju, and Risi, Sebastian. Hyperentm: evolving scalable neural turing machines through hyperneat. In International Conference on the Applications of Evolutionary Computation, pages 750–766. Springer, 2018.
- [4] Zhao, Jinjin, Bhatt, Shreyansh, Thille, Candace, Gattani, Neelesh, and Zimmaro, Dawn. Cold start knowledge tracing with attentive neural turing machine. In Proceedings of the Seventh ACM Conference on Learning@ Scale, pages 333–336, 2020.