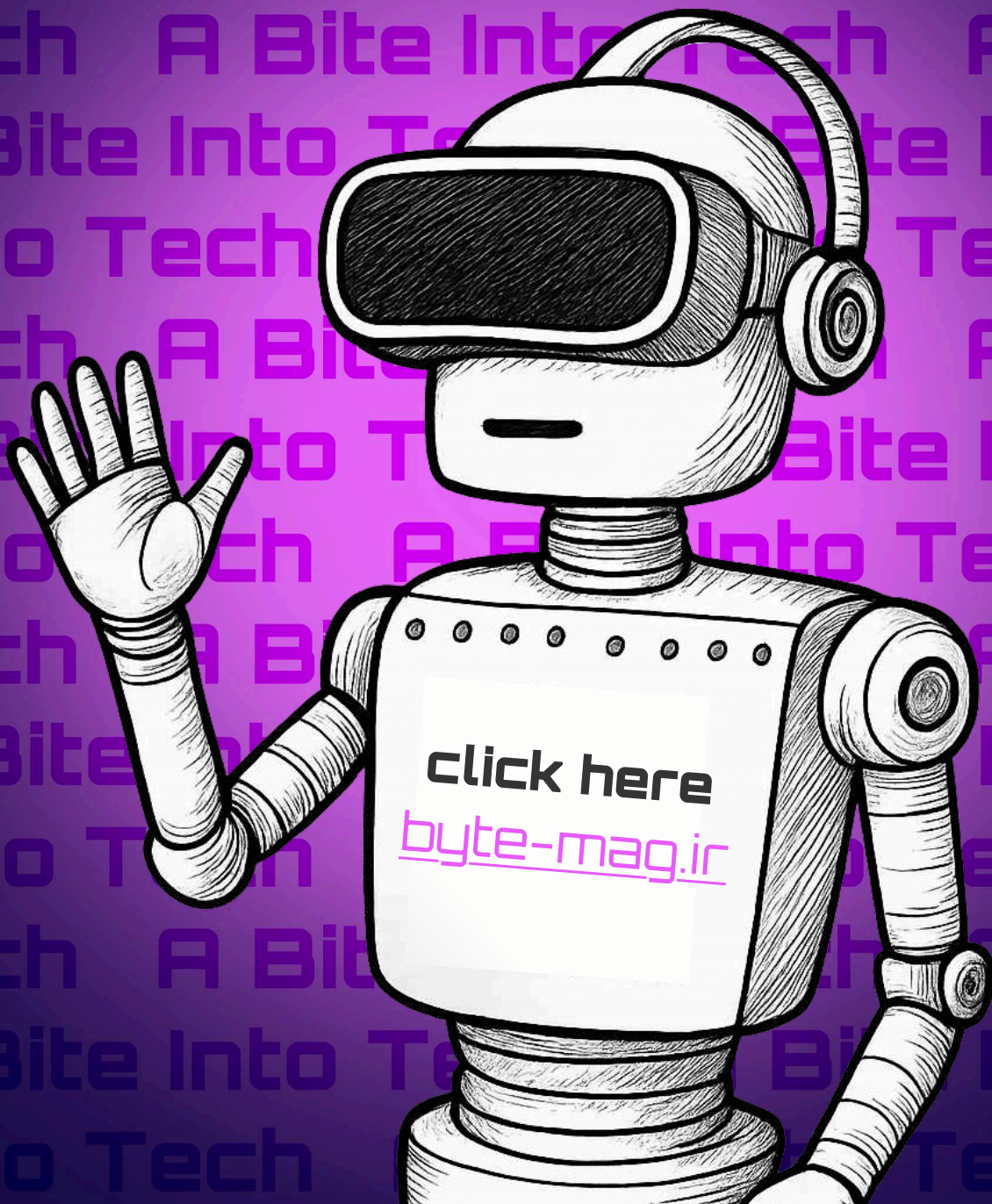


byte

مهر ۱۴۰۴



فهرست

- آرش شاه حسینی ۲ دقیقه ۱
نقطه سر خط ۳
- بزدان بهادری منش ۵ دقیقه ۱
پروتکل XDP: یک قدم قبل از کرنل ۴
- آروین بقال اصل ۷ دقیقه ۱
هنگامی که نوروں و سیایکون به هم می‌رسند! ۶
- سارا یونسی ۸ دقیقه ۱
وقتی داده‌ها مهاجرت نمی‌کنند: روایت یادگیری فدرال ۱۰
- علی قمری - آیدا جباری ۵ دقیقه ۱
کارخانه‌ای با مغز دیجیتال ۱۴

صاحب امتیاز: انجمن علمی دانشکده مهندسی کامپیوتر (SSC)

مدیرمسئول: امیرحسین شهیدی

سرمدیر: آرش شاه حسینی

صفحه آرایی و گرافیک: آرش شاه حسینی، امیررضا اینانلو، سید احمد

موسوی و معین آعلی

ویراستاران ادبی: امیرحسین صوری، متین غیاثی، صهیب صادقی و

امیرمهدی نامجو

تحریریه: آرمان طهماسبی، معین آعلی، مهدی علی‌نژاد، آروین طاهری،

امیرحسین محمدزاده، آروین بقال اصلی و امیرحسین شایان



نقطه سر خط.

سرمقاله

می گویند پایان ها ملموس ترین آغاز هستند؛ آغازی در سکوت، پیش از نوشتن سطری تازه. نقطه ای کوچک که شاید در ظاهر تمام کننده است، اما در دل خود بذر ادامه را پنهان نگاه داشته است. پایان فصل اول «بایت» نیز چنین است؛ یک نقطه در انتهای سطر، پیش از سرخطی دیگر. در هر شماره، حلقه ای از دستورها اجرا شدند، این تکرار راه دستیابی به دقت، عمق و کارایی بیشتر بود. همزمان، خطی از این کد جمعی نگاشته شد؛ متنی آمیخته با درک، کنکاش و شور پژوهش. اما اکنون، پس از طی سه گام تغییر، خطا و آینده، برنامه در نقطه ای ایستاده است که برای گشودن صفحه ای تازه نیازمند بازنویسی است. در فرآیند توسعه نرم افزار، پیش از انتشار نسخه ای جدید، تیم ها نشست های بازیابی برگزار می کنند، پردازش های پس زمینه را متوقف می سازند و حافظه موقت را پاک می کنند تا محیط آزمون برای وارد کردن بهبودها آماده شود. همین وقفه سازمان یافته است که زمینه آزمون ایده های تازه و تولید نسخه های اصلاح شده را فراهم می آورد. بایت نیز اکنون به چنین مقطع سازمان یافته ای رسیده است؛ فاصله ای میان خاتمه یک چرخه و آغاز چرخه ای نو؛ فاصله ای که فرصت بازاندیشی و بازآرایی را فراهم می کند. ما در این مسیر فقط از فناوری نگفتم، از انسان در دل فناوری سخن آوردیم. از اراده ای که در میان صفر و یک ها به دنبال معنا می گردد. از ذهن هایی که در میان الگوریتم ها هنوز رویا می بینند. هر خطی که در این مجله نوشته شد، تلاشی بود برای به روزرسانی نگاه، برای نزدیک شدن به لبه دانش، برای یافتن آن جایی که انسان و ماشین، درک متقابل پیدا می کنند. این فصل با مشارکت گروهی تیم بایت و همراهی شما دانشجویان علاقه مند شکل گرفت. کوشیدیم پنجره ای گشوده باشیم به سوی افق های فناوری؛ افق هایی که داده ها در آن ها جاری اند؛ اما تحلیل دقیق و پرسشگری خلاق، همچنان مهم ترین منبع پیش روی ماست. بیت های به ظاهر کم اهمیت، سنگ بنای برنامه های بزرگترند و اگر توانسته ایم حتی ذره ای از دانشی نو را به شما برسانیم، خوشحالیم که سهمی کوچک در مسیر یادگیری تان داشتیم.

در پایان این فصل، نقطه ای می گذاریم نه به معنای پایان. ما با زدن کلید Enter، به سطر بعدی می رویم... جایی که فصل دوم «بایت» در حال بارگذاری است.





چگونه کار می‌کند؟

در عمل، برنامه‌های XDP کدهایی هستند که با زیرمجموعه‌ای از زبان C نوشته شده و به بایت‌کد eBPF کامپایل می‌شوند. این کدها در یک محیط ایزوله و امن در کرنل به نام ماشین مجازی eBPF اجرا می‌شوند. تا این‌جا چندین بار به این‌که XDP از eBPF استفاده می‌کند اشاره شد و سؤال این‌جاست که eBPF چیست؟

eBPF: مغز متفکر عملیات

eBPF یک تکنولوژی قدرتمند در لینوکس است که اجازه می‌دهد برنامه‌های کوچکی را داخل کرنل اجرا کنیم، بدون نیاز به این‌که کدهای کرنل را تغییر بدهیم. قبل از بارگذاری هر برنامه، تأییدکننده^{۱۳} کد را به دقت بررسی و تست می‌کند تا مطمئن شود که کد نوشته‌شده، هیچ کار خطرناکی مانند حلقه‌های بی‌نهایت یا دسترسی غیرمجاز به حافظه انجام نمی‌دهد. این سازوکار ایمنی، انقلابی در برنامه‌نویسی کرنل است؛ زیرا به ما اجازه می‌دهد بدون خطر نوشتن ماژول‌های پیچیده، کدهای سفارشی را در قلب سیستم‌عامل و کرنل اجرا کنیم. دنیای کاربردهای eBPF بسیار گسترده است و هر روز هم وسعت بیشتری پیدا می‌کند. پس بیا ببینیم چند مورد از کاربردهای مهم eBPF به جز پروتکل XDP را بررسی کنیم:

مشاهده و ردیابی^{۱۳} بی‌درنگ

با eBPF می‌توانیم نقاط حساس کرنل و یوزر اسپیس را بی‌درنگ اندازه‌گیری و دنبال کنیم: تأخیر سیستمی، فراخوانی توابع، مقدار I/O، تعداد پردازنده‌ها و غیره. با استفاده از ابزارهایی مثل bpftrace (ابزاری برای اسکریپت‌های tracing با eBPF) داده‌ها را جمع‌آوری می‌کنند و امکان ساخت متریک‌ها^{۱۴} و پارامترهای دقیق دیگری را می‌دهند، بدون این‌که برنامه را متوقف کنیم یا عملکرد را به‌طور قابل‌توجهی خراب کنیم. همچنین خروجی‌ها را می‌توان به ابزارهای مانیتورینگ و tracing مثل Prometheus فرستاد تا تحلیل و ارتباط بین سرویس‌ها امکان‌پذیر شود.

یک راه‌حل رادیکال

نقطه قوت XDP دقیقاً همین‌جاست. این پروتکل، یک قلاب^۶ استراتژیک در درایور کارت شبکه، یعنی قبل از آن‌که کرنل بخواهد ساختار پرهزینه sk_buff را ایجاد کند، قرار می‌دهد. عملکرد XDP صرفاً سرعت بالاتر نیست، بلکه برای حذف کارهای غیرضروری است. برای هر بسته‌ای که یک برنامه XDP تصمیم به رها کردن^۷ یا هدایت مجدد^۸ آن می‌گیرد، کرنل از تمام هزینه مربوط به ایجاد و مدیریت یک بافر سوکت خلاص می‌شود. به همین دلیل است که XDP می‌تواند ده‌ها میلیون بسته در ثانیه را روی یک کرنل پردازنده مدیریت کند. این یک جهش عملکردی عظیم در مقایسه با فیلتر کردن بسته‌ها در لایه‌های بالاتر مانند iptables است؛ به‌ویژه در سناریوهایی مانند حملات DDoS که سرنوشت اکثر بسته‌ها، دور ریخته شدن است.

از آزمایش تا نهایت سرعت

یک برنامه XDP می‌تواند در سه حالت مختلف اجرا شود که هر کدام تعادلی بین عملکرد و سازگاری با سخت‌افزار ارائه می‌دهند:

- XDP بومی^۹: سریع‌ترین حالت که در آن برنامه مستقیماً در درایور کارت شبکه بارگذاری می‌شود و نیازمند پشتیبانی درایور است.
- XDP عمومی^{۱۰}: یک حالت نرم‌افزاری جایگزین که روی هر سخت‌افزاری کار می‌کند. این حالت برای توسعه و آزمایش ایده‌آل است، زیرا به توسعه‌دهندگان اجازه می‌دهد منطق خود را حتی روی لپ‌تاپ‌شان بنویسند و بعداً روی سرورهای قدرتمند مستقر کنند.
- XDP تخلیه‌شده^{۱۱}: نهایت عملکرد، جایی که برنامه eBPF مستقیماً روی سخت‌افزار کارت شبکه اجرا می‌شود و بسته‌ها را بدون درگیر کردن پردازنده اصلی پردازش می‌کند. این حالت نیازمند کارت‌های شبکه تخصصی است.

تا به حال به این فکر کرده‌اید که گول‌های فناوری مانند کلافلر^۱ چگونه حملات گسترده DDoS را در نطفه خفه می‌کنند یا شرکت‌هایی مانند گوگل و متا چطور میلیاردها درخواست کاربر را بدون لحظه‌ای وقفه مدیریت می‌کنند؟ پاسخ، اغلب در اعماق کرنل لینوکس (منظور از کرنل لینوکس، کرنل گنو/لینوکس است) و در دل فناوری قدرتمندی به نام مسیر داده سریع^۲ نهفته است.

XDP یک پروتکل مهم در دنیای شبکه‌های مدرن است که با عملکرد فوق‌العاده بالا، به برنامه‌ها اجازه می‌دهد بسته‌های شبکه را در اولین نقطه ممکن، یعنی درست در درایور کارت شبکه، مدیریت کنند.

این یعنی دستیابی به سرعتی نزدیک به سخت‌افزار، بدون نیاز به دور زدن کامل کرنل. XDP با استفاده از eBPF، به توسعه‌دهندگان این قدرت را می‌دهد که منطق سفارشی خود را برای مقابله با حملات، متعادل‌سازی بار^۳ و فایروال، با کمترین سربار ممکن پیاده‌سازی کنند.

در این نوشته ابتدا می‌بینیم که مشکل استک شبکه سنتی لینوکس چیست، سپس تا حدودی با معماری هوشمندانه XDP آشنا می‌شویم و در نهایت، نگاهی به کاربرد واقعی آن در پروژه‌های مانند Cilium^۴ می‌اندازیم که از XDP برای ساخت نسل جدیدی از متعادل‌کننده‌های بار خود استفاده می‌کند.

مشکل کجاست؟

برای درک ارزش XDP، ابتدا باید با مسیر یک بسته در استک شبکه سنتی لینوکس آشنا شویم. وقتی یک بسته به اینترنتیس کارت شبکه^۵ می‌رسد، کرنل برای مدیریت آن یک داده‌ساختار به نام sk_buff (بافر سوکت) ایجاد می‌کند.

این ساختار، مثل یک کوله‌پشتی، اطلاعات زیادی را در خود جای می‌دهد و همین اطلاعات زیاد، باعث سنگین و پرهزینه شدن آن شده است. در حجم ترافیک بالا، مثلاً در مقیاس میلیون‌ها بسته در ثانیه، فرایند تخصیص و مدیریت حافظه برای این «کوله‌پشتی‌ها» به یک گلوگاه^۵ جدی تبدیل می‌شود و عملکرد درست را مختل می‌کند.



با استفاده از eBPF می‌توان مصرف CPU، حافظه، I/O و شبکه را دقیق‌تر از قبل اندازه‌گیری و کنترل کرد. مثلاً برای اجرای سیاست‌های QoS^۲ (اولویت‌بندی و تضمین کیفیت سرویس) یا تضمین SLO^۳ (هدف کمی برای کیفیت سرویس) می‌توانیم ترافیک یا کارهای یک گروه پردازش را اولویت‌بندی و محدود کنیم. ترکیب eBPF با cgroups و کنترلرهای دیگر به شما اجازه می‌دهد رفتار زمان‌بندی و مصرف منابع را بر اساس قواعد پویا تغییر بدهید.

بالانسرها و مسیریابی هوشمند^۴

با eBPF این امکان را داریم که ترافیک شبکه را بر اساس اطلاعات سطح اپلیکیشن یا متریک‌های runtime هدایت کنیم؛ این یعنی لودبالانسرهایی که از وضعیت واقعی هر نود آگاه هستند و تصمیمات هوشمندانه‌تری می‌گیرند.

پروژه متن‌باز Cilium، قدرت‌گرفته از XDP

Cilium یک پروژه شبکه ابری پیشرو برای Kubernetes است که از eBPF برای شبکه، امنیت و مشاهده‌پذیری استفاده می‌کند. Cilium به‌طور هوشمندانه از XDP برای شتاب‌دهی ترافیک ورودی و خروجی از کلاستر (ترافیک شمال-جنوب) در متعادل‌کننده بار^۴ خود بهره می‌برد.

این معماری به Cilium اجازه می‌دهد تا قابلیت‌های پیشرفته‌ای مانند بازگشت مستقیم سرور^۵ را پیاده‌سازی کند. در این حالت، سرور پشتیبان پاسخ را مستقیماً برای کاربر ارسال می‌کند و متعادل‌کننده، بار را در مسیر بازگشت دور می‌زند که باعث کاهش تأخیر و حفظ آدرس IP اصلی کاربر می‌شود.

داستان موفقیت Seznam.cz

شرکت فناوری Seznam، یکی از شرکت‌های فناوری بزرگ و موتور جستجوی جمهوری چک، متعادل‌کننده بار قدیمی خود (مبتنی بر IPVS) را با راه‌حل L4LB مستقل Cilium که از XDP استفاده می‌کند، جایگزین کرد. حالا چالش کجاست؟

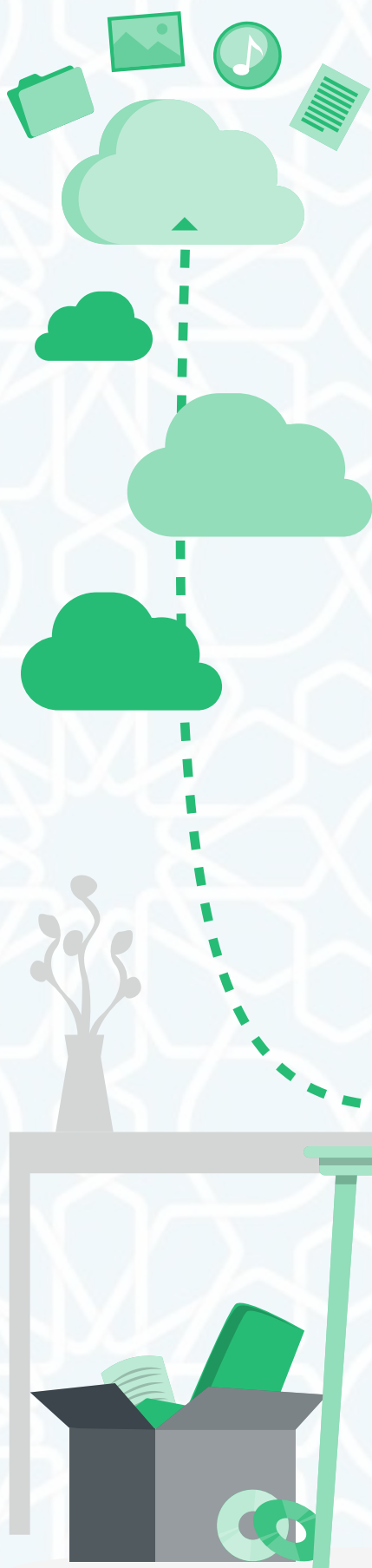
راه‌حل قبلی آن‌ها در ترافیک سنگین (تا ۲۰ میلیون اتصال همزمان) هسته‌های پردازنده را به مرز ظرفیت خود می‌رساند. در نتیجه تصمیم گرفتند به Cilium مهاجرت کنند.

آن‌ها شاهد کاهش ۷۲ برابری در مصرف پردازنده بودند! مصرف پردازنده از ۳۶ هسته کامل به تنها نصف یک هسته کاهش یافت، در حالی‌که همان حجم ترافیک را مدیریت می‌کردند.

آینده‌ای که می‌تواند بسازد

XDP فقط یک ابزار بهینه‌سازی نیست؛ بلکه یک بلوک ساختمانی بنیادی برای نسل بعدی زیرساخت‌های شبکه است. با ترکیب عملکرد خام سخت‌افزار و انعطاف‌پذیری نرم‌افزار، XDP به توسعه‌دهندگان اجازه می‌دهد راه‌حل‌های نوآورانه‌ای برای چالش‌های بزرگ دنیای شبکه، از امنیت و متعادل‌سازی بار گرفته تا مشاهده‌پذیری، خلق کنند. این فناوری که در سکوت، در قلب لینوکس کار می‌کند، بدون شک سنگ‌بنای بسیاری از زیرساخت‌های نرم‌افزارمحور در آینده خواهد بود. باید در نظر داشته باشیم همچنان اکثر شرکت‌های بزرگ و کوچک از این تکنولوژی استفاده نمی‌کنند و در آینده احتمالاً زیاد قرار است اسم XDP را بشنویم!

در واقع XDP هنوز در مراحل پذیرش اولیه قرار دارد و خیلی از شرکت‌ها به تازگی سراغ آن رفته‌اند، ولی روند استفاده از آن رو به رشد است. با بیشتر شدن پشتیبانی ابزارها و تجربه جامعه کاربری، این پروتکل می‌تواند به یکی از اجزای عادی و جدی زیرساخت‌های شبکه در سال‌های آینده تبدیل شود.





مقدمه

سوژهٔ این نوشتار با تیتیر «ملاقات نورون و سیلیکون»، در واقع تلاشی است در جهت تلفیق سامانه‌های زیستی و سخت‌افزاری. تلاشی که نتیجهٔ نهایی آن می‌تواند ما را برای دستیابی به اهداف بسیاری یاری کند؛ اهدافی مانند: استفاده از این سامانه‌ها در زمینهٔ هوش مصنوعی، درک بهتر از نحوهٔ کارکردهای شناختی و عصی پیچیدهٔ ما، پیشی گرفتن از محدودیت‌هایی که ساختارهای سیلیکونی سنتی ایجاد می‌کنند و بسیاری از کاربردهای دیگر.

همهٔ این کاربردها نشان‌دهندهٔ گریزهای متنوع این سوژه به شاخه‌هایی از قبیل زیست-رایانش^۱، هوش مصنوعی^۲، علوم اعصاب^۳، مهندسی زیستی^۴ و محاسبات عصی^۵ هستند که نشان‌دهندهٔ گسترهٔ علمی وسیعی است که درگیر این فناوری نوظهور و جذاب است. شاید یکی از دلایل جذابیت بالای این موضوع، پرسشی عمیق باشد که اگر ما برای دستیابی به هوش، به جای استفاده از ساختارهایی صرفاً سیلیکونی، به سراغ نورون‌ها (که اتفاقاً واحدهای سازندهٔ مغز ما و منشأ تمام پیچیدگی‌های شناختی ما هستند) برویم؛ آیا همچنان می‌توان آن را هوش مصنوعی نامید، یا این‌که می‌توان به آن هوش طبیعی گفت؟!

اتفاقاً، کاربردی که در این نوشتار هم آن را به عنوان موضوع اصلی قرار دادیم، هوش و یادگیری است. از نظر تئوری، به احتمال بالایی، ما به هوش زیستی ساختگی^۶ جامع، قبل از هوش جامع مصنوعی^۷ دست خواهیم یافت.

انگیزهٔ اصلی من برای نوشتن این متن، بعد از آشنایی با کارهای Cortical Labs - که یکی از شرکت‌های پیش‌گام در حوزهٔ تکنولوژی زیستی مستقر در ملبورن استرالیاست - به وجود آمد. به همین خاطر، در ادامه، برای بیان کردن نمونه‌ها و برخی مفاهیم تخصصی‌تر از پروژه‌ها و مقاله‌های همین شرکت بهره برده‌ام.

سلول‌های مغز بر روی بشقاب

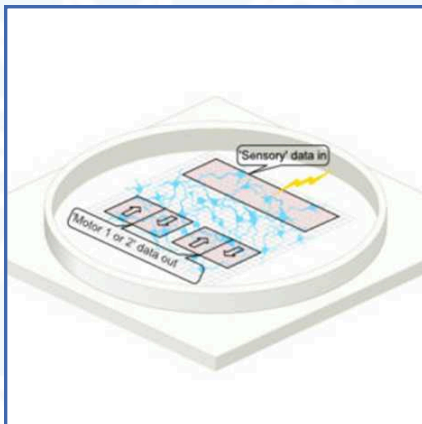
شبکه‌های عصی در ما (و دیگر موجودات) و سیستم‌های الکترونیکی، هر دو،

از سیگنال‌های الکتریکی برای جابه‌جایی داده‌ها از بخشی به بخش دیگر استفاده می‌کنند. در واقع، در مغز و بدن ما، یون‌ها هستند که به عنوان میراث تکاملی ما بر پایهٔ کربن، وظیفهٔ این جابه‌جایی را از یک سلول به سلول دیگر دارند. ژن‌هایی که مسئول ارتباطات الکتریکی سلول‌های عصی هستند، برای اولین بار در اجداد کهن دریایی ما فرگشت یافته‌اند که وجود این یون‌ها را به خوبی توجیه می‌کند. منظور از «بر پایهٔ کربن بودن» در جمله‌های قبلی، وجود و نقش چشم‌گیر عنصر کربن و ترکیبات آن در تمام گونه‌های زیستی‌ای است که در زمین مشاهده شده‌اند. در مقابل، سامانه‌های سخت‌افزاری، بر پایهٔ عنصر سیلیسیم ساخته شده‌اند و این الکترون‌های آزاد هستند که در آن‌ها نقش انتقال اطلاعات را بازی می‌کنند. شاید جالب باشد که بدانید در نظر گرفتن فرگشت بر پایهٔ سیلیسیم از نظر تئوری ممکن است؛ البته که در واقعیت با موانع بسیاری روبه‌روست و تا به حال مشاهده نشده‌است. برخی تحقیقات بیان می‌کنند که تحقق فرگشت سیلیسمی، نیاز به نقطهٔ شروع بسیار متفاوت‌تری نسبت به دریاها و زمین دارد. در هر حال، همین «زبان مشترک» یا پالس‌های الکتریکی که در این سامانه‌ها دیده می‌شود، برای ساختن پلی بین سخت‌افزارهای ساختگی کامپیوتری و نورون‌های زیستی کافی است.

در این پل، نقش سخت‌افزاری را آرایه‌های میکروالکترونی^۸ که برای اندازه‌گیری فعالیت الکتریکی سلول‌های عصی و انتقال سیگنال‌های الکتریکی به آن‌ها استفاده می‌شوند، بازی می‌کنند. نورون‌ها نیز بر روی یک پتری دیش^۹ کشت داده می‌شوند. با استفاده از همین چینش ساده، می‌توان به سامانه‌ای حلقه‌باز^{۱۰} دست یافت که در آن خروجی نورون‌ها تأثیری بر روی ورودی آن‌ها ندارد. در صورتی که اگر این حلقه را ببندیم، به گونه‌ای که خروجی‌های نورون همزمان بر روی ورودی‌های آن تأثیرگذار باشند، فوراً متوجه توانایی‌هایی این سیستم برای هوشمندسازی می‌شویم که ما را دچار پرسش‌هایی خواهد کرد. «آیا می‌توانیم به این نورون‌ها یاد بدهیم که در قبال ورودی‌ها به طور مشخصی رفتار کنند؟» یا این‌که «آیا می‌توانیم به آن‌ها بیاموزیم که تحرکی را پیش‌بینی کنند که هنوز اتفاق نیفتاده، یا پاسخی جدید و متفاوت تولید کنند؟»

در نهایت این پرسش‌ها ما را به سوی

مسئلهٔ بنیادین هوش سوق می‌دهند.



طرح‌واره‌ای از پیکربندی ورودی/خروجی DishBrain بر روی یک آرایهٔ چندالکترونی با تراکم بالا همراه با نورون‌ها

هوش

هوش، مفهوم شگفت‌انگیزی است که از دیرباز مورد پرسش بشریت بوده‌است. پرسشی که امروزه هم تقریباً جوابی نهایی به آن داده نشده‌است و همچنان از عوامل کشمکش بین برخی از فلاسفه و متخصصین علوم اعصاب است. نقل قولی معروف از Emerson M. Pugh، پژوهشگر آمریکایی، وجود دارد که می‌گوید: «اگر مغز انسان آن قدر ساده می‌بود که می‌توانستیم آن را بفهمیم، آن‌گاه ما آن قدر ساده می‌بودیم که نمی‌توانستیم!».

به نظر می‌رسد که استفاده از روش‌های تکنولوژی زیستی و زیست-رایانش، ما را بیش‌تر از همیشه به درک مفهوم هوش و چگونگی کارکرد آن امیدوار کرده‌است. همان‌طور که پیش‌تر هم گفته شد، بنا بر بهینگی ذاتی و تکاملی سیستم‌های زیستی، هوش زیستی سنتزی جامع احتمالاً زودتر از هوش جامع مصنوعی از راه خواهد رسید. البته که تلاش برای فهم چگونگی کارکرد هوش حیوانی و انسانی که ساختار چندلایه‌ای دارند، کار بسیار پیچیده‌ای است و شاید ایزوله کردن تک‌تک این فرایندها برای بررسی آن‌ها به راحتی ممکن نباشد.

یکی از پژوهش‌های جالب در همین زمینه، یادگیری سامانهٔ DishBrain (همان ساختاری که بالاتر راجع به آن صحبت شد) برای انجام بازی پونگ است. جالب است که بدانید در این پژوهش از دو نوع نورون متعلق به انسان و همچنین جوندگانی مانند موش استفاده شد

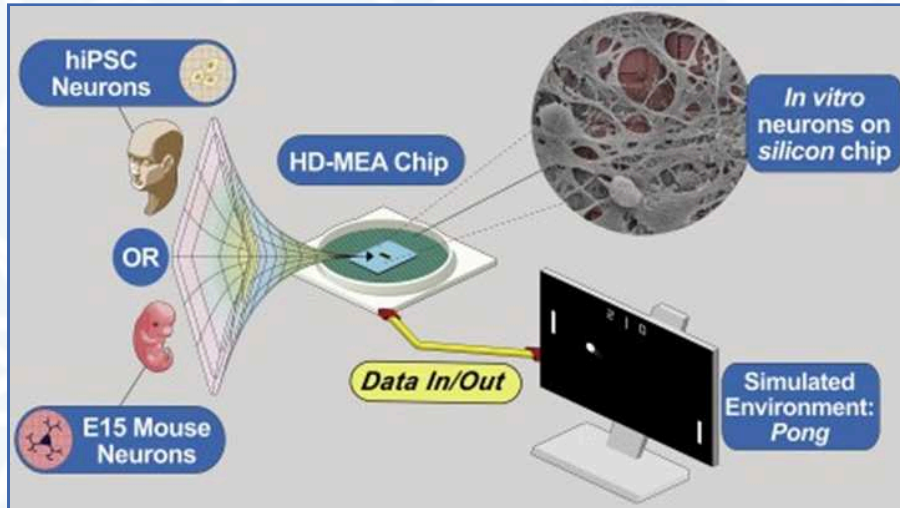


و همان‌طور که احتمالاً حدس می‌زنید، سلول‌های انسانی عملکرد بهتری داشتند.

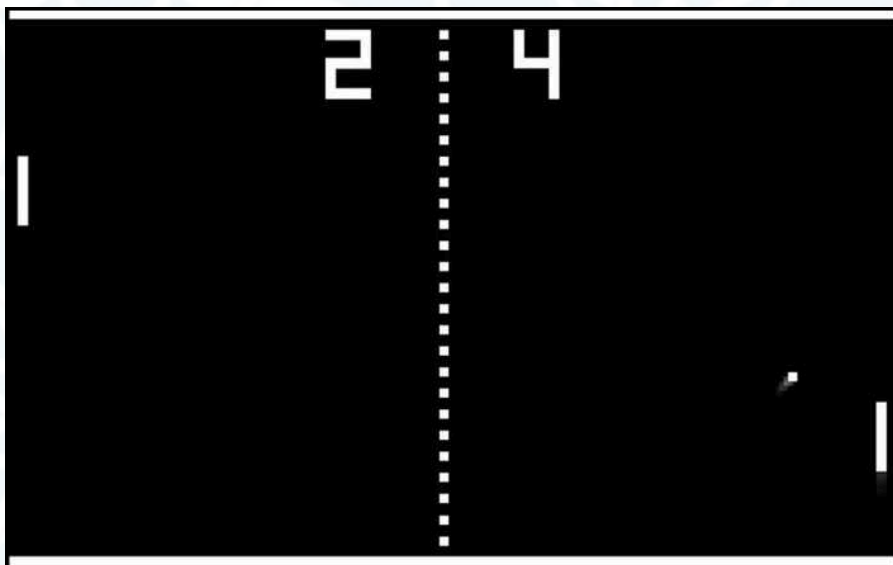
بازی پونگ

بسیاری از شما با بازی پونگ آشنایی دارید. پونگ به‌عنوان یکی از اولین بازی‌های ویدیویی، یک بازی آرکید و ساده به شمار می‌رود که شامل دو تخته و یک توپ است. برای انجام این پژوهش، بازی به شکل ساده‌تر شبیه‌سازی شده و تنها یک تخته دارد که باید جابه‌جا شود و توپ هر دفعه به صورت تصادفی به سمت آن پرتاب می‌شود. آرایه الکترونی‌ای که در این آزمایش‌ها استفاده شده، هشت الکتروود بسیار ریز دارد که می‌توان گفت وظیفه «نوشتن» و «خواندن» بر نورون‌ها را به عهده دارند. اما چگونه به این سیستم ورودی داده می‌شود؟ یا به بیان دیگر، نورون‌ها از کجا می‌فهمند که توپ در هر لحظه کجا قرار دارد؟ برای ورودی دادن، هر الکتروود نماینده نوار به عرض یک هشتم عرض کل صفحه خواهد بود (این نوارها در حقیقت هشت مستطیل بلند افقی هستند که روی هم قرار گرفته‌اند) و همزمان اگر توپ در آن نوار باشد، پالس می‌زند. با این کار می‌توان مختصات y توپ را در هر لحظه به سیستم ورودی داد و نقشه‌ای با ساختار خام توپوگرافی از توپ ایجاد کرد. شاید این کار کمی ساده به نظر برسد، اما در مغزهای واقعی هم ناحیه‌هایی که به آن‌ها sensory map گفته می‌شود (مثلاً برای بینایی یا لامسه) از ساختارهای توپوگرافی کاملاً مشابهی استفاده می‌کنند. از آنجایی که تخته تنها در راستای y حرکت می‌کند می‌توان حس کرد که مؤلفه x برای ما آورده مکانی خاصی ندارد. در نتیجه لازم نیست نگران داده نشدن این مؤلفه به مدل باشیم.

حالا نوبت به اعمال تصمیم نورون‌ها بر روی محیط شبیه‌ساز می‌رسد. برای این‌که بالا بردن و پایین آوردن تخته معنی پیدا کند، می‌توانیم الکتروودها را به دو گروه تقسیم کنیم و آن‌ها را «ناحیه یک» و «ناحیه دو» بنامیم. یکی از آن‌ها «موتور» بالا بردن و دیگری «موتور» پایین آوردن تخته خواهد بود. در نهایت الکتروودها در هر لحظه فعالیت‌های کشت‌های نورونی را رصد می‌کنند و تخته را به همان سمتی حرکت می‌دهند که نورون‌های ناحیه آن شلیک بیشتری داشته‌اند. این کار همان دریافت پاسخ نورون‌ها خواهد بود. تنها چیزی که باقی می‌ماند، در نظر گرفتن یک پاداش یا تنبیه مناسب برای نورون‌ها در صورت برخورد کردن یا نکردن توپ با تخته است. برای این کار به اصلی مهم در عصب‌شناسی و علوم شناختی به نام اصل انرژی آزاد تکیه می‌شود. این اصل بیان می‌کند که



نمای نموداری از چیدمان سیستم Dish Brain که در پژوهش‌ها از آن استفاده شده است.



بازی ویدیویی پونگ، یک بازی دو بعدی که به تنیس روی میز شباهت دارد.

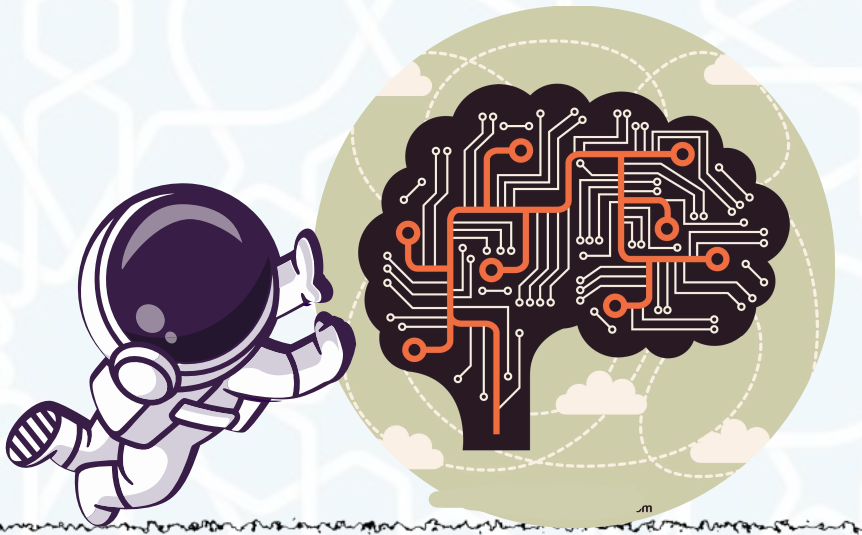
هستند. مغز ما با داشتن تعداد نورون‌های بسیار بیشتر، وجود تنظیم‌کننده‌های عصبی مانند دوپامین، دارا بودن عناصری همچون حافظه بلندمدت و وجود ادراکات مختلف در بدن، در یادگیری، بسیار پیچیده‌تر عمل می‌کند. با این حال، اصل انرژی آزاد، همچنان به‌عنوان قاعده‌ای بنیادین در مغز نیز رعایت می‌شود؛ بدین معنی که نورون‌های مغز در هر لحظه پیش‌بینی‌ای از ورودی آینده خود دارند و اگر این ورودی با پیش‌بینی آن‌ها تفاوت زیادی داشته باشد، سعی می‌کنند خود را اصلاح کنند. به عبارتی، مغز همواره تلاش می‌کند حالت بعدی هر اتفاق را به درستی پیش‌بینی کند، فرایندی که موجب یادگیری در ما می‌شود. البته این آزمایش ریزه‌کاری‌های بسیاری برای دقیق بودن نتایج نهایی دارد. به‌عنوان مثال، برای این‌که مطمئن باشند هندسه نواحی انتخاب‌شده، سوگیری‌ای در یادگیری ایجاد نمی‌کند، آزمایش را هر دفعه

همه سامانه‌های زیستی از جمله مغز و خود نورون‌ها تلاش می‌کنند که «انرژی آزاد» خود را کمینه کنند تا بقای خود را تضمین کنند و با محیط اطراف‌شان هماهنگ شوند. ساده‌تر این است که بگوییم در این‌جا بیشتر بودن «انرژی آزاد» با غیرمنتظره بودن ورودی نورون‌ها ارتباط مستقیم دارد و این یعنی نورون‌ها از دریافت ورودی‌های غیرمنتظره گریزان هستند. برای همین هم به‌عنوان پاداش برای گرفتن توپ، به آن‌ها پالس‌های یکنواخت و بدون نویز داده می‌شود. در صورتی که توپ از دست برود، الکتروودها پالس‌های غیرمنتظره و نویزدار تولید می‌کنند. حال اگر در مورد مکانیزم تنبیه و پاداش در مغز ما و تفاوت آن با DishBrain کنج‌کاو باشید، باید گفت که این سامانه ساده‌تر از آن عمل می‌کند که بخواهد با فرایندهای مغزی مقایسه شود، اما هر دوی این سامانه‌ها در برخی از جنبه‌های بنیادین مانند یکدیگر

با نواحی موتوری متفاوتی اجرا می‌کنند. همچنین حالت‌های کنترل مختلفی برای این‌که تضمین کنیم نتیجه نهایی سامانه به دلیل تلاش خود نوروهاست، در نظر گرفته می‌شوند. در آخر می‌بینیم که نوروها به صورت حقیقی در طی زمان یاد می‌گیرند که چگونه رفتار کنند تا بهترین نتیجه را داشته باشند! اگر بازه‌های زمانی ۵ دقیقه اول و ۱۵ دقیقه آخر آزمایش را در نظر بگیریم، افزایش معناداری در تعداد توپ‌هایی که پشت سر هم توسط آن‌ها به درستی دریافت شده مشاهده می‌شود. جالب است بدانید که در ابتدای آزمایش، نوروهای موشی بهتر از نوروهای انسانی عمل کردند، ولی در نهایت نوروهای انسانی با اختلاف نسبتاً کمی توانستند مشخص کنند که سریع‌تر و بهتر عمل می‌کنند.

چرا؟

اما پس از همه این صحبت‌ها، چرا ما باید واقعاً به سوی سیستم‌های هیبریدی زیستی هوشمند قدم برداریم؟ اگر برخی از دلایل ریز و درشتی که تا به این‌جا کار بیان شد را نادیده بگیریم، یکی از مهم‌ترین انگیزه‌ها برای این کار می‌تواند مصرف بهینه انرژی باشد. امروزه اجرای الگوریتم‌های یادگیری ماشین به مقدار قابل توجهی (تا میلیون‌ها وات) توان نیاز دارد، در حالی‌که سیستم‌های عصبی زیستی می‌توانند در آب قند به کار خود بپردازند! برای روشن‌تر شدن، میزان انرژی مورد نیاز در انتقال‌های سیناپسی و طبیعی از مرتبه پیکوژول است ولی همین میزان در همتاهای مصنوعی و سیلیکونی آن‌ها از مرتبه نانوژول است. مصرف بالای برق به آسیب‌های جبران‌ناپذیر زیست‌محیطی منجر می‌شود. همچنین این میزان از مصرف انرژی تقریباً دسترسی گسترده مصرف‌کنندگان به این فناوری را در سطح شخصی غیرممکن می‌سازد. انگیزه‌های بسیار دیگری نیز وجود دارند، یکی دیگر از آن‌ها توانایی رشد و یادگیری سریع هوش زیستی با تعداد نمونه‌های محدودتر در قیاس با نیاز الگوریتم‌های فعلی یادگیری ماشین است. سامانه‌های زیستی انعطاف‌پذیری بالاتری دارند و در سناریوهای پیچیده خوش‌فکتر عمل می‌کنند، چیزی که چندان واضح در مدل‌های امروزی دیده نمی‌شود. در نهایت همه این تلاش‌ها ممکن است بتوانند به پرسش‌های اساسی و فلسفی ما در مورد هوش زیستی پاسخ بدهند. البته که این سامانه‌ها خالی از عیب نیستند. یکی از مشکلات اصلی‌ای که فعالان این حوزه اکنون با آن دست‌وپنجه نرم می‌کنند، تلاش برای زنده نگه داشتن نوروهاست. کشت‌های



آن‌قدر هم جدید نیست و مربوط به سه تا چهار سال پیش است. اگر همین حالا به وبسایت Cortical Labs سری بزنید، می‌بینید که این شرکت از اولین کامپیوتر زیستی خود به نام CL1 و حتی فضای ابری ویژه‌ای رونمایی کرده‌است! همه این‌ها نمونه‌های حاضری هستند که احتمالات بی‌شماری را که در مسیر این حوزه وجود دارند، برای ما روشن‌تر می‌سازند.

در نهایت، دنبال کردن پژوهش‌های مربوطه و حتی متصور شدن آینده این سیستم‌های هیبریدی زیستی (که تا حد زیادی به فیلم‌های علمی-تخیلی شباهت دارد!) می‌تواند برای افراد علاقه‌مند به حوزه‌های هوش مصنوعی و کامپیوتر به همراه کمی چاشنی از زیست و عصب‌شناسی بسیار جذاب باشد و آن‌ها را به فکر وادارد.

نورونی، حتی اگر در بهترین شرایط هم نگهداری شوند، کم‌تر از یک سال زنده می‌مانند که شرایط را سخت می‌کند. این سیستم‌ها مقیاس‌پذیری پایین‌تری هم دارند. برخی از مزایا و معایبی که در این بخش بیان شدند، هر کدام منحصر به یکی از سامانه‌های زیستی و یا سیلیکونی بودند که معمولاً هم در تقابل با یکدیگر قرار داشتند. همین امر، منجر می‌شود که اگر این سیستم‌ها به درستی در چیدمان با یکدیگر قرار بگیرند، بتوانند ایرادهای یکدیگر را جبران کنند.

پایان

پژوهش بازی پونگ که از آن به عنوان نمونه‌ای بنیادین در این نوشتار استفاده کردم،



byte

Our world is built on biology and once we begin to understand it, it then becomes technology.

Ryan Bethencourt



If you think that the Internet has changed your life, think again. The Internet of Things is about to change it all over again!

Brendan O'Brien

Deep Neural Networks are responsible for some of the greatest advances in modern computer science.

Yann LeCun





مقدمه

یادگیری تقویتی یا Reinforcement Learning

در سال‌های اخیر، هوش مصنوعی با رشد سریع خود، نقش مهمی در تحلیل داده‌های واقعی ایفا می‌کند. یکی از رویکردهای نوین در این زمینه، یادگیری فدرال^۱ است؛ مدلی که به جای جمع‌آوری داده‌ها در یک سرور مرکزی، آموزش را به صورت توزیع‌شده میان دستگاه‌های مختلف انجام می‌دهد.

در FL، هر دستگاه مدل را با داده‌های محلی خود آموزش می‌دهد و به‌طور دوره‌ای پارامترها را با سرور مرکزی همگام می‌سازد. این رویکرد نه تنها از توان محاسباتی توزیع‌شده بهره می‌برد، بلکه افق‌های جدیدی برای توسعه سیستم‌های هوشمند فراهم می‌کند. در ادامه این نوشتار، مفاهیم پایه، معماری‌های رایج، مزایا و روش‌های بهبود عملکرد در FL بررسی خواهند شد.

تاریخچه

ایده اصلی یادگیری فدرال، یعنی آموزش مدل‌های یادگیری ماشین به صورت توزیع‌شده و بدون انتقال داده‌ها، ریشه در دهه‌های گذشته و تحقیقات سیستم‌های توزیع‌شده دارد. اما مفهوم «یادگیری فدرال» به عنوان یک حوزه مستقل و مشخص، برای نخستین بار در سال ۲۰۱۶ توسط محققان شرکت گوگل معرفی شد.

در مقاله‌ای با عنوان «Communication-» Efficient Learning of Deep Networks from Decentralized Data»، الگوریتم معروف میانگین‌گیری مشارکتی^۲ معرفی شد. این الگوریتم، امکان آموزش مدل‌های یادگیری عمیق^۳ را با استفاده از داده‌های پراکنده روی هزاران دستگاه (مانند گوشی‌های هوشمند) فراهم می‌کند.

از آن زمان، یادگیری فدرال به سرعت مورد توجه جامعه تحقیقاتی قرار گرفت و در حوزه‌هایی مانند بینایی ماشین^۴، پردازش زبان طبیعی^۵، سلامت، اینترنت اشیا^۶ و صنایع مالی^۷ به کار گرفته شد.

مزایای یادگیری فدرال

یادگیری فدرال مدل‌ها را مستقیماً روی دستگاه‌های مختلف آموزش می‌دهد. در این شیوه، داده‌ها در محل تولید خود باقی می‌مانند و مدل‌ها به صورت توزیع‌شده روی دستگاه‌های مختلف آموزش می‌بینند. این روش مزایای مهمی دارد:

یادگیری پیوسته و به روز

مدل‌ها می‌توانند به صورت مداوم و بی‌درنگ از داده‌های جدید روی دستگاه‌ها یاد بگیرند، بدون نیاز به جمع‌آوری مجدد داده‌ها.

کاهش وابستگی به ارتباط دائم با سرور

با انجام بیشتر محاسبات در سمت کلاینت، نیاز به ارسال حجم بالای داده به سرور کاهش یافته و کارایی در شرایط شبکه محدود افزایش می‌یابد.

استفاده از قدرت پردازش به صورت توزیع‌شده

یادگیری فدرال با استفاده از منابع پردازشی دستگاه‌های متعدد (مانند گوشی‌ها یا سنسورها) به شکلی مؤثر، مقیاس‌پذیری را افزایش می‌دهد.

پشتیبانی از داده‌های ناهمگون

در بسیاری از کاربردها، داده‌های کاربران متنوع و غیرهمگن^۸ اند. یادگیری فدرال به خوبی می‌تواند با این تنوع داده سازگار شود.

اجزای کلیدی و فناوری‌های پشتیبان اینترنت اشیا صنعتی

یادگیری فدرال یک فرایند مشارکتی بین سرور مرکزی و مجموعه‌ای از کلاینت‌ها (مانند گوشی‌های هوشمند یا رایانه‌های شخصی) است که بدون انتقال داده‌های خام، یک مدل یادگیری را به صورت توزیع‌شده آموزش می‌دهند. این فرایند شامل مراحل زیر است:

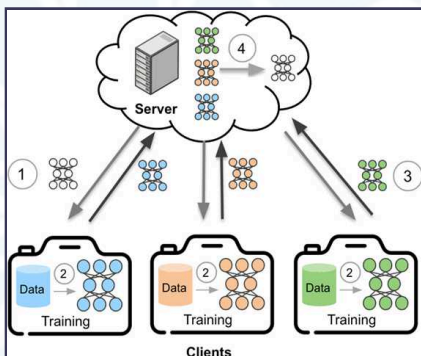
۱. ارسال مدل اولیه به کلاینت‌ها: سرور مرکزی یک مدل پایه شامل وزن‌ها و پارامترهای اولیه را به کلاینت‌های منتخب ارسال می‌کند.

۲. آموزش محلی مدل در هر کلاینت: کلاینت‌ها مدل دریافتی را با استفاده از داده‌های محلی خود آموزش می‌دهند. این مرحله معمولاً طی چند دور^۹ انجام می‌شود.

۳. بازگرداندن پارامترهای به‌روزشده به سرور: پس از آموزش، هر کلاینت تنها پارامترهای به‌روزشده را به سرور بازمی‌گرداند. داده‌های خام کاربران هرگز ارسال نمی‌شوند.

۴. تجمیع پارامترها و ساخت مدل جدید: سرور با استفاده از روش‌هایی مانند FedAvg، پارامترهای دریافتی را ترکیب کرده و یک مدل به‌روزرسانی‌شده می‌سازد. این مدل جدید سپس به کلاینت‌ها ارسال می‌شود.

۵. تکرار چرخه تا رسیدن به دقت مطلوب: مراحل بالا به طور تکراری انجام می‌شوند تا زمانی که مدل نهایی از نظر دقت و عملکرد به سطح مطلوب برسد. در شکل زیر نیز مراحل اصلی نمایش داده شده‌اند.



مراحل اصلی یادگیری فدرال

مثال عملی: یادگیری فدرال در عمل

بیابید یک مثال ساده اما واقع‌بینانه از یادگیری فدرال را با استفاده از فریم‌ورک Flower^{۱۰} بررسی کنیم. این مثال، که از یک دوره آموزشی DeepLearning.AI در زمینه FL اقتباس شده است، نشان می‌دهد چگونه چندین کلاینت می‌توانند بدون اشتراک‌گذاری داده‌های خام، به صورت مشارکتی یک مدل را آموزش دهند. در این‌جا از دیتاست کلاسیک MNIST^{۱۱} استفاده می‌کنیم، آن را بین کلاینت‌های مجازی تقسیم می‌کنیم و چند دور آموزش فدرال با الگوریتم FedAvg را اجرا خواهیم کرد.

گام ۱: بارگذاری کتابخانه‌ها


```
def get_weights(net):
    return [val.cpu().numpy() for _, val
in net.state_dict().items()]
```

این توابع وزن‌های مدل PyTorch را به آرایه‌های NumPy و برعکس تبدیل می‌کنند، تا فریم‌ورک Flower بتواند آن‌ها را در طول دوره‌های یادگیری فدرال رد و بدل کند.

گام ۴: تعریف کلاینت Flower

```
class FlowerClient(NumPyClient):
    def __init__(self, net, trainset,
testset):
        self.net = net
        self.trainset = trainset
        self.testset = testset

    def fit(self, parameters, config):
        set_weights(self.net, parameters)
        train_model(self.net,
self.trainset)
        return get_weights(self.net),
len(self.trainset), {}

    def evaluate(self, parameters,
config):
        set_weights(self.net, parameters)
        loss, accuracy =
evaluate_model(self.net, self.testset)
        return loss, len(self.testset),
{"accuracy": accuracy}
```

هر کلاینت آموزش و ارزیابی محلی را انجام می‌دهد و سپس وزن‌های به‌روزشده را بازمی‌گرداند. مدیریت ارتباطات در پشت صحنه توسط Flower انجام می‌شود.

گام ۵: تعریف تابع مقداردهی اولیه کلاینت

```
def client_fn(context: Context):
    net = SimpleModel()
    partition_id =
int(context.node_config["partition-id"])
    return FlowerClient(net,
train_sets[partition_id,
testset]).to_client()
```

این تابع یک کلاینت جدید برای شبیه‌سازی ایجاد می‌کند و به هر کلاینت یک بخش مشخص از داده‌ها را اختصاص می‌دهد.

گام ۶: تابع ارزیابی در سمت سرور

```
def evaluate(server_round, parameters,
config):
    net = SimpleModel()
    set_weights(net, parameters)
```

```
from flwr.client import NumPyClient, ClientApp
from flwr.common import
ndarrays_to_parameters, Context
from flwr.server import ServerApp,
ServerConfig, ServerAppComponents
from flwr.server.strategy import FedAvg
from flwr.simulation import run_simulation
from utils2 import *
```

گام ۲: آماده‌سازی دیتاست MNIST

```
trainset = datasets.MNIST("./MNIST_data/",
download=True, train=True,
transform=transform)

total_length = len(trainset)
split_size = total_length // 3
torch.manual_seed(42)
part1, part2, part3 = random_split(trainset,
[split_size] * 3)

part1 = exclude_digits(part1,
excluded_digits=[1, 3, 7])
part2 = exclude_digits(part2,
excluded_digits=[2, 5, 8])
part3 = exclude_digits(part3,
excluded_digits=[4, 6, 9])

train_sets = [part1, part2, part3]
```

ما داده‌های آموزشی را بین سه کلاینت تقسیم می‌کنیم. به هر کلاینت یک زیرمجموعه غیرهم‌پوشان از ارقام اختصاص داده می‌شود تا داده‌های IID غیر (یعنی داده‌های مستقل و یکسان توزیع‌نشده) شبیه‌سازی شود - چیزی که در بسیاری از سناریوهای واقعی یادگیری فدرال رایج است. سپس داده‌های تست را آماده می‌کنیم:

```
testset = datasets.MNIST("./MNIST_data/",
download=True, train=False,
transform=transform)
testset_137 = include_digits(testset, [1,
3, 7])
testset_258 = include_digits(testset, [2,
5, 8])
testset_469 = include_digits(testset, [4,
6, 9])
```

گام ۳: تعریف هندلرهای پارامترهای مدل

```
def set_weights(net, parameters):
    params_dict =
zip(net.state_dict().keys(), parameters)
    state_dict = OrderedDict({k:
torch.tensor(v) for k, v in params_dict})
    net.load_state_dict(state_dict,
strict=True)
```

```
server = ServerApp(server_fn=server_fn)
client = ClientApp(client_fn=client_fn)

run_simulation(
    server_app=server,
    client_app=client,
    num_supernodes=3,
    backend_config=backend_setup,
)
```

معماری‌های متداول در یادگیری فدرال

یادگیری فدرال (FL) به عنوان رویکردی نوین در آموزش مدل‌های یادگیری ماشین روی داده‌های توزیع‌شده، دارای معماری‌های مختلفی است که متناسب با نوع سیستم، نوع داده و سطح کنترل طراحی می‌شوند. در ادامه، سه معماری رایج در این حوزه را معرفی می‌کنیم:

▪ معماری متمرکز^۱

در این ساختار، یک سرور مرکزی وظیفه هماهنگی کل فرآیند یادگیری را بر عهده دارد. ابتدا مدل پایه را به کلاينت‌ها (مانند گوشی‌ها یا لپتاپ‌ها) ارسال می‌کند، سپس پارامترهای آموزش‌دیده را جمع‌آوری کرده و آن‌ها را ترکیب می‌کند. این معماری، ساده و محبوب‌ترین نوع در کاربردهای صنعتی است، اما به سرور مرکزی وابسته است و آسیب‌پذیری نقطه تکی شکست^۲ دارد.

▪ معماری غیرمتمرکز^۳

در این رویکرد، هیچ سرور مرکزی‌ای وجود ندارد. کلاينت‌ها به صورت مستقیم با یکدیگر ارتباط برقرار می‌کنند و پارامترهای مدل را با هم به اشتراک می‌گذارند. این معماری معمولاً با ساختارهایی مانند شبکه‌های همتا به همتا^۴ یا گراف ارتباطی پیاده‌سازی می‌شود. این معماری برای محیط‌هایی با اعتماد کمتر یا ساختار توزیع‌شده‌تر مناسب است، اما مدیریت و همگرایی مدل در آن چالش‌برانگیزتر است.

▪ معماری ترکیبی^۵

ایده این معماری، استفاده از چندین سطح پردازشی است. برای مثال، ابتدا مدل‌ها در سطح دستگاه‌ها آموزش می‌بینند، سپس به گره‌های^۶ میانی (مانند گیت‌وی‌ها^۷ یا سرورهای لبه) منتقل می‌شوند و در نهایت خروجی آن‌ها به سرور مرکزی فرستاده می‌شود. این مدل بین بهره‌وری، مقیاس‌پذیری و پایداری تعادل برقرار می‌کند و برای شبکه‌های بزرگ یا محیط‌های چندلایه^۸ مناسب است.

تنظیمات پشت صحنه:

نقش پارامترها و ابرپارامترها^۹ در یادگیری فدرال

در یادگیری فدرال، پارامترهای مدل مانند وزن‌ها و بایاس‌ها به صورت محلی روی داده‌های هر کلاينت آموزش دیده و سپس برای تجمیع به سرور مرکزی ارسال می‌شوند، بدون آن‌که نیاز به انتقال داده‌های خام باشد. برخلاف یادگیری متمرکز، یادگیری فدرال با چالش‌هایی مانند ناهمگونی داده‌ها روبه‌روست که نیازمند الگوریتم‌های خاصی مانند FedAvg، FedProx یا FedNova برای تجمیع مؤثر پارامترهاست. همگرایی این پارامترها و وزن‌دهی به کلاينت‌ها نقشی کلیدی در دقت و تعمیم‌پذیری مدل جهانی دارد.

در کنار آن، ابرپارامترها مانند نرخ یادگیری، تعداد دوره‌های محلی، اندازه دسته^{۱۰} و تعداد کلاينت‌های فعال، باید پیش از آموزش تعیین شوند. این مقادیر تأثیری مستقیم بر کیفیت یادگیری، میزان بیش‌برازش^{۱۱} محلی، سرعت همگرایی و مصرف منابع دارند. همچنین، نحوه انتخاب

```
_, accuracy = evaluate_model(net,
testset)
_, acc137 = evaluate_model(net,
testset_137)
_, acc258 = evaluate_model(net,
testset_258)
_, acc469 = evaluate_model(net,
testset_469)
```

```
log(INFO, "Accuracy (All): %.4f",
accuracy)
log(INFO, "Accuracy [1,3,7]: %.4f",
acc137)
log(INFO, "Accuracy [2,5,8]: %.4f",
acc258)
log(INFO, "Accuracy [4,6,9]: %.4f",
acc469)
```

```
if server_round == 3:
    cm =
compute_confusion_matrix(net, testset)
plot_confusion_matrix(cm, "Final
Global Model")
```

این تابع مدل سراسری را پس از هر دور ارزیابی می‌کند و معیارهای عملکرد را به صورت جزئی برای زیرمجموعه‌های ارقام ثبت می‌کند.

گام ۷: پی‌کربندی استراتژی FedAvg

```
net = SimpleModel()
params =
ndarrays_to_parameters(get_weights(net))

def server_fn(context: Context):
    strategy = FedAvg(
        fraction_fit=1.0,
        fraction_evaluate=0.0,
        initial_parameters=params,
        evaluate_fn=evaluate,
    )
    config = ServerConfig(num_rounds=3)
    return
ServerAppComponents(strategy=strategy,
config=config)
```

در این‌جا از FedAvg استفاده می‌کنیم که رایج‌ترین استراتژی در یادگیری فدرال است.

این روش وزن‌های همه کلاينت‌ها را میانگین‌گیری می‌کند تا مدل سراسری به روزرسانی شود.

گام ۸: اجرای شبیه‌سازی

Peer-To-Peer (P2P)^{۱۲}
Multi-Tier^{۱۳}

Decentralized FL^{۱۴}
Gateway^{۱۵}
Overfitting^{۱۶}

Single Point of Failure^{۱۷}
Node^{۱۸}
Batch Size^{۱۹}

Centralized FL^{۲۰}
Hierarchical/Hybrid FL^{۲۱}
Hyperparameters^{۲۲}

کلاینت‌ها و تناوب ارتباطات بین کلاینت و سرور بر بهره‌وری انرژی و پهنای باند مؤثر است. در نتیجه، تنظیم بهینهٔ ابرپارامترها برای پایداری، مقیاس‌پذیری و دقت نهایی مدل در یادگیری فدرال ضروری است.

نقش ارتباطات شبکه

در یادگیری فدرال، ارتباطات شبکه نقشی اساسی در عملکرد و پایداری سیستم ایفا می‌کنند. برخلاف روش‌های متمرکز که داده‌ها به سرور منتقل می‌شوند، در یادگیری فدرال فقط پارامترهای مدل یا گرادینت‌ها میان کلاینت و سرور جابه‌جا می‌شوند. این ویژگی، گرچه حریم خصوصی را حفظ می‌کند، اما سیستم را وابسته به زیرساخت ارتباطی قابل اعتماد می‌سازد.

عواملی مانند پهنای باند، تأخیر شبکه، فرکانس تبادل پارامترها و ناهمگنی کانال‌های ارتباطی می‌توانند بر کیفیت آموزش تأثیر بگذارند. مثلاً ارتباط ضعیف برخی کلاینت‌ها ممکن است باعث تأخیر یا از دست رفتن به‌روزرسانی‌ها شود که در نتیجهٔ آن، فرایند تجمع مدل مختل می‌شود.

برای حل این چالش‌ها، از راهکارهایی مانند انتخاب تطبیقی کلاینت‌ها، فشرده‌سازی پارامترها و آموزش ناهمگام^۱ استفاده می‌شود. همچنین، الگوریتم‌هایی مانند FedAvgM و FedBuff برای مدیریت شرایط ناپایدار شبکه طراحی شده‌اند.

در نتیجه، موفقیت در یادگیری فدرال تنها به الگوریتم‌های یادگیری وابسته نیست، بلکه به طراحی هوشمندانهٔ شبکه و مدیریت ارتباطات بین کلاینت‌ها و سرور نیز بستگی دارد.

از Gboard تا خودروهای خودران

یادگیری فدرال کاربردهای گسترده‌ای در دنیای واقعی دارد، از جمله در صفحه‌کلید هوشمند گوگل^۲؛ که با آموزش مدل‌ها به صورت محلی روی دستگاه‌های کاربران و ارسال فقط به‌روزرسانی‌های مدل به سرور، هم عملکرد بهتری ارائه می‌دهد و هم حریم خصوصی حفظ می‌شود.

در خودروهای خودران، یادگیری فدرال به مدل‌ها اجازه می‌دهد تا با داده‌های محیطی مثل تصاویر و سنسورها به صورت توزیع‌شده آموزش ببینند و تجربهٔ رانندگی را به اشتراک بگذارند، بدون نیاز به انتقال حجم بالای داده‌ها. این روش مصرف پهنای باند را کاهش داده و مدل‌ها را بهتر با شرایط واقعی جاده سازگار می‌کند.

در حوزهٔ سلامت دیجیتال، FL به بیمارستان‌ها،

مراکز تصویربرداری و دستگاه‌های پوشیدنی امکان می‌دهد مدل‌های تشخیص و پیش‌بینی پزشکی را روی داده‌های محلی و حساس بیمارانش آموزش دهند و بدون انتقال دادهٔ خام، این مدل‌ها را با مدل مرکزی تلفیق کنند.

این فرایند موجب افزایش دقت تشخیص، پوشش گسترده‌تر داده‌های متنوع و رعایت قوانین حفاظت از داده‌ها مانند HIPAA و GDPR می‌شود.

از توسعه تا استقرار: فریم‌ورک‌های مدرن یادگیری فدرال در یک نگاه

یادگیری فدرال با چالش‌های ذاتی مانند هماهنگی پیچیده بین چندین گره محاسباتی، مدیریت داده‌های ناهمگون و تضمین مقیاس‌پذیری، نیازمند فریم‌ورک‌های نرم‌افزاری تخصصی است که توسعه، ارزیابی و پیاده‌سازی الگوریتم‌ها را تسهیل کنند.

در سال‌های اخیر، چندین فریم‌ورک متن‌باز پیشرفته معرفی شده‌اند که هر یک ویژگی‌ها و امکانات منحصر به فردی در حوزهٔ آموزش توزیع‌شده را فراهم می‌آورند.

TensorFlow Federated

به عنوان شاخص‌ترین پلتفرم توسعهٔ یادگیری فدرال، ساختاری ماژولار مبتنی بر TensorFlow ارائه می‌کند که ضمن بهره‌گیری از API‌های سطح بالا، امکان شبیه‌سازی فرایندهای آموزش توزیع‌شده، تجمع وزن‌ها و مدیریت چرخهٔ زندگی مدل را فراهم می‌سازد. TFF به ویژه برای پژوهشگران و مهندسانی که به دنبال توسعهٔ مدل‌های مقیاس‌پذیر و بهینه در محیط‌های واقعی هستند، ابزاری قدرتمند به شمار می‌آید.

PySyft

در نقطهٔ مقابل، فریم‌ورکی پیشرفته بر بستر PyTorch است که علاوه بر پشتیبانی از یادگیری فدرال، به صورت تخصصی بر حفظ حریم خصوصی تمرکز دارد. این فریم‌ورک با فراهم کردن تکنیک‌هایی نظیر محاسبات چندجانبهٔ امن^۳ و یادگیری تفاضلی^۴، گزینه‌ای ایده‌آل برای پروژه‌هایی با الزامات امنیتی و قانونی پیچیده است.

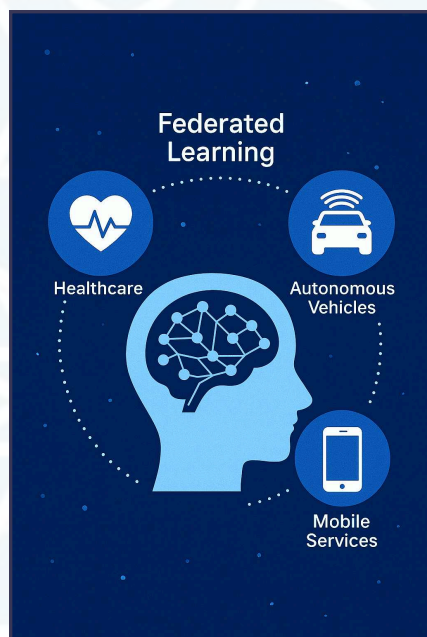
Flower

فریم‌ورک متن‌باز Flower به دلیل معماری سبک، سازگاری با کتابخانه‌هایی مانند TensorFlow، PyTorch و JAX، و تمرکز بر انعطاف‌پذیری و توسعه‌پذیری، به سرعت به انتخاب اول برای پیاده‌سازی سیستم‌های یادگیری فدرال در مقیاس صنعتی تبدیل شده است. این ابزار امکان اجرای مدل‌های توزیع‌شده در محیط‌های چندگانه با الگوریتم‌های مختلف تجمع را فراهم می‌سازد.

همچنین فریم‌ورک‌هایی مانند FedML، PaddleFL و LEAF نیز با قابلیت‌هایی نظیر شبیه‌سازی شرایط واقعی، مدیریت داده‌های ناهمگون و ساده‌سازی توسعه، نقش مهمی در پیشبرد تحقیقات و کاربردهای یادگیری فدرال دارند. انتخاب میان آن‌ها به نیاز پروژه، زبان برنامه‌نویسی و محیط اجرایی بستگی دارد.

جمع‌بندی

یادگیری فدرال به عنوان رویکردی نوین در هوش مصنوعی، با آموزش مدل‌ها روی داده‌های محلی و بدون جابه‌جایی اطلاعات خام، توانسته است همزمان به دو دغدغهٔ اساسی، یعنی حفظ حریم خصوصی و مقیاس‌پذیری، پاسخ دهد و در حوزه‌هایی مانند سلامت دیجیتال، خودروهای خودران و سرویس‌های موبایلی به کار گرفته شود. این روش با بهره‌گیری از قدرت محاسباتی توزیع‌شده و سازگاری با داده‌های ناهمگون، جایگاه ویژه‌ای یافته است؛ هرچند که چالش‌هایی نظیر هم‌گرایی مدل، ناهمگونی شدید داده‌ها و محدودیت‌های ارتباطی همچنان چابرجاست. آیندهٔ این حوزه در گرو توسعهٔ الگوریتم‌های مقاوم‌تر، کاهش هزینه‌های ارتباطی، بهره‌گیری از معماری‌های ترکیبی و به کارگیری فناوری‌های امنیتی پیشرفته، همچون حریم خصوصی تفاضلی^۵ و رمزنگاری^۶ است. یادگیری فدرال دیگر صرفاً یک تکنیک نیست، بلکه به ستون فقرات هوش مصنوعی توزیع‌شده بدل می‌شود؛ مسیری که نوید همکاری گستردهٔ دستگاه‌ها بدون نقض محرمانگی داده‌ها را می‌دهد.





علی قمری

کارشناسی ۱۴۰۳
مهندسی شیمی اراک

کارخانه‌ای با مغز دیجیتال



آیداجباری

کارشناسی ۱۴۰۳
مهندسی شیمی اراک

در دنیای امروز صنعت، رقابت اصلی نه صرفاً در توانایی بالا بردن نرخ تولید، بلکه در میزان بهره‌گیری صنایع از فناوری‌های نوین است. یکی از جذاب‌ترین این فناوری‌ها، اینترنت اشیا صنعتی (IIoT)^۱ است که از پایه‌های انقلاب صنعتی چهارم^۲ محسوب می‌شود. در این رویکرد، هدف صرفاً خودکارسازی^۳ نیست، بلکه حرکت به سوی هوشمندسازی^۴ کل زنجیره تولید است.

در چنین فضایی، دیگر نمی‌توان صنعت را صرفاً مجموعه‌ای از ماشین‌آلات و خطوط تولید دانست، بلکه هر واحد صنعتی به یک سامانه پیچیده هوشمند تبدیل شده است که بر پایه داده، تحلیل و بهینه‌سازی فعالیت می‌کند. بنابراین، شناخت دقیق اجزاء این اکوسیستم و نحوه عملکرد آن، برای دانشجویان و متخصصان آینده صنعت، نه یک انتخاب، بلکه یک ضرورت علمی و حرفه‌ای محسوب می‌شود.

کاربردهای عملی IIoT در کارخانه‌های پیشرفته و صنایع بزرگ

اینترنت اشیا صنعتی صرفاً یک واژه تئوری نیست، بلکه به یک ابزار قدرتمند برای افزایش کیفیت محصول، ارتقاء ایمنی و کاهش هزینه‌های عملیاتی در دنیای واقعی صنعت تبدیل شده است.

برای مثال، شرکت Toray Plastics در ژاپن که تولیدکننده فیلم‌های پلیمری پیشرفته برای کاربردهای غذایی است، با استفاده از تکنولوژی GE Plant Applications، داده‌های بی‌درنگ را از دستگاه‌های لبه^۵ جمع‌آوری کرده و آنالیز می‌کند. این سیستم به اپراتورها اجازه می‌دهد تا خطاهای تولیدی را در همان لحظه تشخیص داده و ضایعات محصول را کاهش دهند.

در حوزه انرژی، شرکت GE Digital با توسعه سامانه مدیریت عملکرد دارایی‌ها^۶، از IIoT برای پیش‌بینی خرابی تجهیزات و انجام تعمیرات پیش‌گیرانه در نیروگاه‌ها استفاده می‌کند. در این سامانه با استفاده از اطلاعات صدها حسگر نصب‌شده بر تجهیزات و تحلیل داده‌ها، می‌توان زمان دقیق نیاز به تعمیر یا تعویض قطعه را پیش‌بینی کرد و از توقف‌های پرهزینه جلوگیری کرد.

در بخش پالایش و صنایع سنگین هم نمونه‌های موفق در دنیا مشاهده می‌شوند. پالایشگاه Ergon Refining در ایالات متحده، با استفاده از حسگرهای صوتی بی‌سیم در شیرهای بخار، توانسته عملکرد غیرعادی شیرها را شناسایی کند و در کمتر از پنج ماه، هزینه پیاپی‌سازی این فناوری را جبران کند و علاوه بر آن، سالانه میلیون‌ها دلار صرفه جویی اقتصادی داشته باشد.

اینترنت اشیا صنعتی چیست؟

اینترنت اشیا صنعتی، به شبکه‌ای هوشمند از حسگرها، ابزارهای دقیق، تجهیزات کنترلی و نرم‌افزارها گفته می‌شود که به منظور جمع‌آوری، پردازش و تحلیل داده‌ها در محیط‌های صنعتی به کار گرفته می‌شوند. در صنایع شیمیایی، این فناوری نقشی حیاتی در پایش بی‌درنگ پارامترهای فرایندی مانند دما، فشار، جریان، سطح مایع، ترکیب شیمیایی، رطوبت و ارتعاشات مکانیکی و حتی افزایش ایمنی عملیاتی و بهینه‌سازی بهره‌برداری از واحدهای تولیدی دارد.

برخلاف اینترنت اشیا عمومی^۷ که بیشتر در کاربردهای روزمره مانند خانه‌های هوشمند یا دستگاه‌های پوشیدنی استفاده می‌شود، IIoT با تمرکز بر محیط‌های صنعتی پیچیده، حساس و پرریسک مانند پالایشگاه‌ها، واحدهای پتروشیمی و کارخانه‌های تولید مواد شیمیایی طراحی شده است. این تفاوت‌ها، خود را در الزامات فنی مانند پایداری سیستم، امنیت سایبری، قابلیت اطمینان و سرعت پاسخ‌دهی نشان می‌دهند.

برای مثال، در یک رآکتور شیمیایی پیوسته^۸، استفاده از حسگرهای دمایی و فشاری متصل به شبکه IIoT می‌تواند به سیستم کنترل اجازه دهد تا تغییرات جزئی در شرایط واکنش را شناسایی کرده و به صورت خودکار، عملگرهایی مانند شیرها یا مبدل‌های حرارتی را تنظیم کند. این سطح از کنترل هوشمند، نه تنها باعث بهبود کیفیت محصول و کاهش ضایعات می‌شود، بلکه نقش مهمی در پیشگیری از حوادث ایفا می‌کند. در صنایع شیمیایی که ماهیت فرایندها عمدتاً

پیوسته، خطرناک و وابسته به کنترل دقیق است، اینترنت اشیا صنعتی به عنوان یک ابزار انقلابی برای خودکارسازی پیشرفته، پیش‌بینی خرابی‌ها و تصمیم‌گیری مبتنی بر داده‌های بی‌درنگ شناخته می‌شود.

اجزاء کلیدی و فناوری‌های پشتیبان اینترنت اشیا صنعتی

اکوسیستم منسجم اجزاء سخت‌افزاری و نرم‌افزاری برای عملکرد درست اینترنت اشیا صنعتی در محیط‌های پیچیده‌ای مثل واحدهای تولید مواد شیمیایی، دارویی یا حتی پتروشیمی بسیار حیاتی است. اجزاء کلیدی این اکوسیستم عبارتند از:

حسگرهای هوشمند^۹

در صنایع شیمیایی، حسگرهایی برای پایش دما، فشار، PH، ترکیب شیمیایی و حتی میزان خوردگی تجهیزات به کار می‌روند. حسگرهای نسل جدید با قابلیت کالیبراسیون خودکار و تشخیص خطا به سیستم اجازه می‌دهند تا در صورت بروز انحراف، به سرعت و قبل از وقوع حادثه به مهندس فرایند هشدار دهند.

درگاه‌های ارتباطی صنعتی^{۱۰}

داده‌ها از طریق درگاه‌هایی امن و مقاوم، از حسگرها به سمت پلتفرم‌های ابری یا سرورهای داخلی منتقل می‌شوند. این انتقال می‌تواند با پروتکل‌هایی مانند Modbus TCP یا MQTT، OPC UA صورت گیرد که برای صنایع حساس طراحی شده‌اند.

رایانش لبه^{۱۱}

در بسیاری از فرایندهای شیمیایی، زمان پاسخ باید در حد میلی‌ثانیه باشد. به همین دلیل، تحلیل اولیه داده‌ها در لبه و نزدیک به دستگاه‌ها انجام می‌شود تا سیستم بتواند بدون تأخیر به تغییرات واکنش نشان دهد؛ تغییراتی مانند تنظیم فوری دبی جریان در زمان وجود واکنش‌گرهای حساس داخل سامانه.



پلتفرم‌های تحلیلی ابری^۱

پلتفرم‌هایی مانند GE-Predix و Siemens-MindSphere و ABB-Ability امکان تحلیل داده‌های انبوه، تشخیص الگوها، پیش‌بینی خرابی‌ها و بهینه‌سازی کلی عملکرد را فراهم می‌کنند. این پلتفرم‌ها نقش مهمی در کاهش مصرف انرژی و حفظ کیفیت بالای محصول دارند. اتلاف انرژی و کاهش نوسانات کیفیت محصول از بزرگترین چالش‌های مهندسان فرایند در طراحی فرایندهای شیمیایی است.

امنیت سایبری صنعتی^۲

حفاظت از داده‌ها در واحدهای صنعتی بزرگ مانند کارخانه‌های مهم صنعتی، پالایشگاه‌ها و پتروشیمی‌ها از مهم‌ترین مباحث اینترنت اشیا صنعتی است. استفاده از فایروال‌های^۳ صنعتی، رمزگذاری داده‌ها و سیستم‌های شناسایی تهدیدات نیز از اجزاء حیاتی زیرساخت IIoT محسوب می‌شوند.

مهم‌ترین چالش‌ها و موانع پیاده‌سازی IIoT

ناسازگاری با سیستم‌های قدیمی^۴

بیشتر واحدهای شیمیایی موجود از تجهیزات کنترلی سنتی، PLC‌های قدیمی و نرم‌افزارهای محدودکننده استفاده می‌کنند. ادغام این سیستم‌ها با زیرساخت‌های جدید IIoT، نیازمند بازطراحی معماری کنترلی و سرمایه‌گذاری قابل توجه است.

مقاومت سازمانی و فرهنگی

در برخی شرکت‌ها، کارکنان عملیات یا مهندسان باتجربه، به دلایل گوناگون مانند ترس از جایگزینی نیروی انسانی یا ناآشنایی با فناوری‌های جدید در برابر هوشمندسازی واحد مقاومت می‌کنند. غلبه بر این مقاومت نیازمند آموزش هدفمند و مشارکت فعال منابع انسانی است.

امنیت سایبری در محیط‌های بحرانی

اتصال تجهیزات حساس شیمیایی به شبکه‌های صنعتی، اگر بدون راهکارهای امنیتی مناسب انجام شود، می‌تواند منجر به افشای اطلاعات فرایندی، اختلال در عملیات یا حتی تهدیدهای فیزیکی خطرناک شود. پیاده‌سازی فایروال‌های صنعتی، پروتکل‌های رمزنگاری و مدیریت دسترسی باید به عنوان یک اولویت دیده شوند، نه یک گزینه جانبی و انتخابی.

هزینه‌های اولیه و عدم بازگشت سریع سرمایه

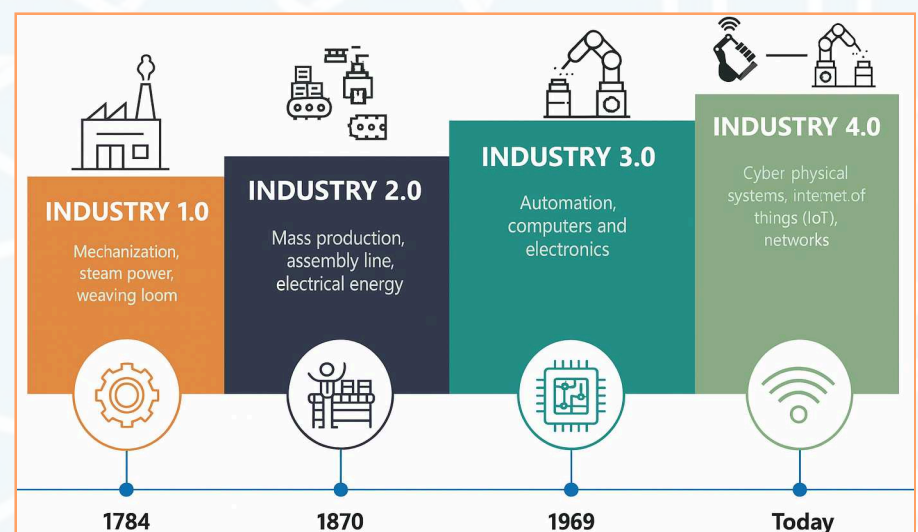
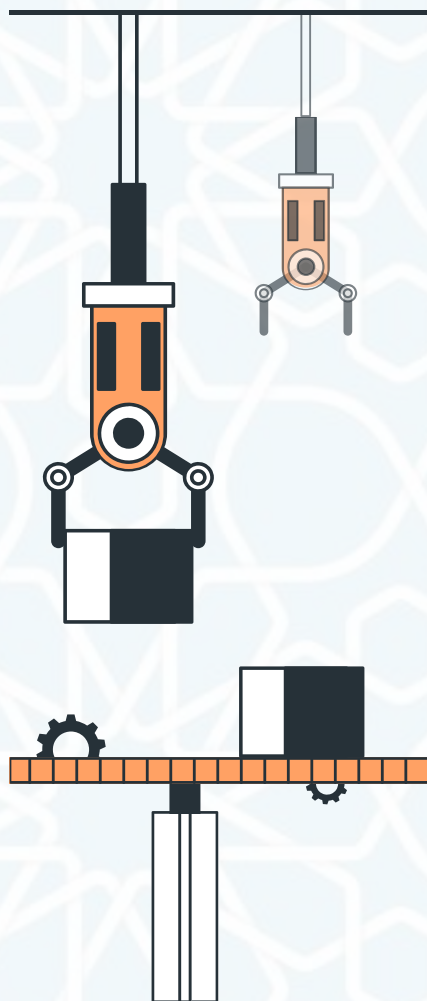
در بسیاری از واحدهای شیمیایی، مدیران مالی یا ارشد، انتظار بازگشت سریع سرمایه^۵ دارند؛ در حالی که مزایای IIoT معمولاً در میان‌مدت یا بلندمدت آشکار می‌شوند، مانند کاهش خرابی تجهیزات یا بهینه‌سازی مصرف انرژی.

پیچیدگی تحلیل داده و نیاز به نیروی متخصص

صرف داشتن داده کافی نیست! تحلیل صحیح، مصورسازی کاربردی و استخراج الگوهای معنادار از داده‌های بزرگ فرایندی، نیاز به تیم‌هایی با مهارت ترکیبی در مهندسی شیمی، داده‌کاوی و هوش مصنوعی دارند؛ نیروهایی که در بسیاری از شرکت‌ها هنوز به صورت ساختاریافته وجود ندارند.

آیندهای در حال شکل‌گیری

اینترنت اشیا صنعتی، صرفاً یک ابزار نیست؛ بلکه دست‌رنج نسلی از مهندسان خواهد بود که با درک عمیق فرایندهای شیمیایی و تسلط بر داده، صنایع شیمیایی را به سطحی از هوشمندی می‌رسانند که تا گذشته نه‌چندان دور، صرفاً در حد پیش‌بینی و ایده‌پردازی مطرح بود. این آینده، توسط نسل تازه مهندسان در حال ساخت است.



Legacy Systems

Firewall

Industrial Cybersecurity

Cloud & Hybrid Analytics
Return on Investment (ROI)

Almost Human



زمانی چنین فرض می‌شد که روزی رایانه‌ها خواهند توانست سفينه‌ها را هدایت کنند، جراحی‌های ظریف انجام دهند و پیچیده‌ترین معادلات را حل کنند؛ اما در عین حال، برقراری ابتدایی‌ترین گفت‌وگوها برایشان دشوار خواهد بود. زبان انسان پدیده‌ای به‌غایت پیچیده است؛ چیزی فراتر از واژه‌ها و دستور زبان؛ آمیخته با تجربه، احساس و پیش‌زمینه‌هایی که به سادگی در هیچ فرهنگ لغتی نمی‌گنجند. چت‌بات‌ها با مکث‌های ساده‌ای مانند «اممم» یا با پرسش‌هایی که پاسخ صحیح ندارند، سردرگم می‌شوند؛ زیرا آن‌چه در گفت‌وگوی انسانی جریان دارد، تنها تبادل داده نیست، بلکه کنشی عمیق از فهم و نیت است.

به‌نظر می‌رسد شبیه‌سازی کامل مکالمه انسانی به چیزی بیش از افزایش حافظه و توان محاسباتی نیاز دارد و شاید به درکی از معنا و آگاهی وابسته است که هنوز تعریف روشنی برایش نداریم. اکنون که ماشین‌ها بیش از هر زمان به‌گذر از آزمون تورینگ نزدیک شده‌اند، شاید ناگزیر باشیم بار دیگر با همان پرسش بزرگ ماهیت هوشیاری سر و کله بزنیم.

نه سال از انتشار این ویدیو گذشته و در این مدت می‌توان پیشرفت سرسام‌آور هوش مصنوعی را دید، پیشرفتی که در آن زمان کم‌تر کسی تصورش را می‌کرد. تا جایی که شاید دیگر ندانید این متن توسط هوش مصنوعی نوشته شده یا نه.

انتقادات و پیشنهادهای خود را
بامادر میان بگذارید

