به نام خدا

اعضای گروه

ایمان محمدی
آرمین ثقفیان
شایان صالحی

نیم‌سال پاییز ۱۴۰۳

خلاصه مقاله‌ی Machine Learning in Embedded Systems

چالش‌ها در یادگیری ماشین تعبیه‌شده:

- محدودیت منابع: حافظه کم، توان پردازشی محدود و مصرف انرژی پایین از مشکلات اصلی در این سیستم‌ها هستن.
- محدودیت داده: توی سیستم‌های تعبیه‌شده، جمع‌آوری و پردازش سریع داده‌ها سخت میشه چون ممکنه شبکه یا دستگاه همیشه به‌راحتی جواب نده.
- ایمنی و امنیت: یادگیری ماشین در این سیستم‌ها (مثل سیستم‌های پزشکی) باید کاملاً مطمئن و دقیق باشه تا مشکلی برای کاربر پیش نیاد.
- نیاز به پردازش بلادرنگ: خیلی از کاربردهای تعبیه‌شده نیاز دارن در لحظه پاسخ بدن، ولی خب الگوریتم‌های یادگیری ماشین خیلی سنگین هستن و سریع اجرا نمی‌شن.

راه‌حل‌ها برای یادگیری ماشین کارآمدتر:

- بهینه‌سازی مدل‌ها: از تکنیک‌هایی مثل فشرده‌سازی مدل، حذف پارامترهای اضافی، کاهش دقت و تقطیر دانش استفاده می‌کنن که این‌ها به کاهش اندازه و سبک‌تر شدن مدل کمک می‌کنن تا توی سخت‌افزارهای کوچیک بهتر اجرا بشه.
- استفاده از سخت‌افزارهای مخصوص: مثل GPUها، FPGAها و ASICها که مخصوص یادگیری ماشین طراحی شدن و سرعت کارو بالا می‌برن و مصرف انرژی رو هم پایین میارن.
- محاسبات درون حافظه: این روش کمک می‌کنه که داده‌ها به جای انتقال بین حافظه و پردازنده، همونجا توی حافظه پردازش بشن؛ این‌طوری هم سریع‌تر و هم بهینه‌تره.

معیارگذاری و ارزیابی یادگیری ماشین در سیستم‌های تعبیه‌شده:

- مقاله روی اهمیت معیارگذاری (Benchmarking) تأکید داره تا بفهمیم مدل‌ها چطور در سیستم‌های تعبیه‌شده عمل می‌کنن. معیارهایی مثل مصرف انرژی، سرعت پردازش و دقت مد نظر هستن که باعث میشن مدل مناسب‌تر انتخاب بشه.

خلاصه مقاله‌ی TinyML

کاربردهای TinyML در سیستم‌های تعبیه‌شده:

- سخت‌افزار: معمولاً این برنامه‌ها روی میکروکنترلرهای فوق‌العاده کم‌مصرف مثل 32STM، سری ARM Cortex-M و Ambiq Apollo 3 اجرا می‌شن. این میکروکنترلرها جمع‌وجور، مقرون‌به‌صرفه و از نظر مصرف انرژی بهینه هستن، واسه همین برای اجرای الگوریتم‌های یادگیری ماشین در دستگاه‌های لبه‌ای خیلی خوب جواب میدن.
- فریم‌ورک‌ها: برای اینکه مدل‌های یادگیری ماشین توی دستگاه‌های تعبیه‌شده راه‌اندازی بشن، از فریم‌ورک‌هایی مثل TensorFlow Lite و جایگزین‌های خاص TinyML مثل TinyOL برای Arduino استفاده می‌کنن. TensorFlow Lite مخصوصاً خیلی رایجه و روی سخت‌افزارهای مختلف هم پشتیبانی می‌شه.
- موارد استفاده: بعضی کاربردهای متداول شامل شناسایی کلمات کلیدی، طبقه‌بندی تصاویر، تشخیص اشیا، تشخیص ناهنجاری‌ها و شناسایی ژست‌ها هستن. مثلاً شناسایی کلمات کلیدی برای دستیارهای صوتی مثل Siri و Alexa خیلی محبوبه چون انرژی کمی لازم داره.
- دیتاست‌ها: توی برنامه‌های TinyML از دیتاست‌های قابل دسترس مثل Speech Commands گوگل برای شناسایی کلمات کلیدی و COCO برای شناسایی بصری استفاده می‌کنن. با این حال، کمبود دیتاست‌های مناسب یکی از چالش‌های رشد TinyML به حساب میاد.

ارزیابی‌ها و چالش‌های عملکرد:

- فریم‌ورک‌های ارزیابی: سیستم‌های TinyML با استفاده از ابزارهایی مثل MLPerf Tiny ارزیابی می‌شن که روی معیارهایی مثل تأخیر، دقت، مصرف انرژی و استفاده از حافظه تمرکز دارن.
- چالش‌ها در ارزیابی: چون این دستگاه‌ها محدودیت‌های حافظه و قدرت پردازشی دارن، ارزیابی‌ها باید مصرف انرژی و بار محاسباتی رو بهینه کنن. این نشون میده که نیاز به معیارهای خاصی داریم که بهینه بودن مصرف انرژی رو بیشتر از معیارهای محاسباتی سنتی در اولویت قرار بده.

چالش‌ها و مسیرهای آینده:

حوزه TinyML داره رشد می‌کنه و تحقیقات فعلی روی توسعه مدل‌های جمع‌وجور و کم‌مصرف مثل شبکه‌های عصبی کم‌دقت یا حذف‌شده تمرکز داره.

پیش‌بینی می‌شه که TinyML به زمینه‌های مختلفی مثل کشاورزی هوشمند، سلامت الکترونیک و صنعت ۴.۰ گسترش پیدا کنه و دغدغه‌های اخلاقی مثل حریم خصوصی داده و مسئولیت‌پذیری هوش مصنوعی هم توش بیشتر مطرح بشن.

An Overview of Machine Learning within Embedded and Mobile Applications

یه سری الگوریتم و مدل سبک یادگیری ماشین که برای دستگاه‌های با منابع محدود مناسب هستن بررسی شده. این مقاله به الگوریتم‌های کلاسیکی مثل ماشین بردار پشتیبان (SVM)، درخت تصمیم‌گیری و نزدیک‌ترین همسایه (KNN) پرداخته؛ الگوریتم‌هایی که به‌خاطر کارایی خوبشون روی دستگاه‌های کم‌قدرت معروفن. همچنین مدل‌های سبک یادگیری عمیق مثل MobileNet و نسخه پیشرفته‌ترش MobileNetV2، EfficientNet و مدل‌های کوچیک‌تر مثل TinyML که مخصوص کار تو محیط‌های محدود طراحی شدن رو هم بررسی کرده.

از نظر سخت‌افزاری، این مقاله بنچمارک‌هایی روی دستگاه‌های مختلف مثل پردازنده‌های ARM Cortex-M (که بیشتر تو میکروکنترلرها استفاده می‌شن)، سری Jetson انویدیا (برای مدل‌های پیچیده‌تر) و TPU گوگل Coral (که برای پردازش سریع مدل‌ها بهینه شده) انجام داده.

نتایج مقاله نشون می‌دن که مدل‌های کلاسیک مثل KNN و SVM روی دستگاه‌های کوچیک مثل میکروکنترلرها راحت پیاده‌سازی می‌شن و حتی KNN از نظر مصرف حافظه عملکرد خیلی خوبی داره. در مورد مدل‌های یادگیری عمیق هم، نسخه‌های مختلف MobileNet بهترین عملکرد رو داشتن و با کمترین مصرف انرژی و تاخیر پایین، دقت مناسبی ارائه دادن. برای مثال، MobileNetV2 روی TPU گوگل Coral تونست به‌صورت بلادرنگ پیش‌بینی‌ها رو انجام بده. EfficientNet هم تو شرایطی که دقت مهم‌تر از سرعت بود، عملکرد بهتری داشت.

در کل، مقاله نتیجه می‌گیره که هم مدل‌های کلاسیک و هم مدل‌های یادگیری عمیق سبک‌وزن می‌تونن با بهینه‌سازی مناسب، روی دستگاه‌های توکار و موبایل به‌خوبی اجرا بشن و گزینه‌های قابل قبولی برای استفاده تو سیستم‌های موبایل و اینترنت اشیا باشن.

TinyML Platforms Benchmarking

این مقاله با عنوان «ارزیابی پلتفرم‌های TinyML» به بررسی عملکرد فریم‌ورک‌های یادگیری ماشین روی میکروکنترلرها، مخصوصاً 32STM، پرداخته. تو این مقاله دو فریم‌ورک مهم بررسی شدن: اولی TensorFlow Lite Micro که نسخه‌ی سبکی از TensorFlow برای میکروکنترلرهاست، و دومی STM32Cube.AI که ابزاری از شرکت STMicroelectronics برای تبدیل شبکه‌های عصبی به کد بهینه‌شده برای 32STM محسوب میشه.

این تحقیق از دو مدل اصلی یادگیری ماشین استفاده کرده؛ شبکه‌های عصبی کاملاً متصل یا همون FCNN که بیشتر برای کارهای ساده‌ی طبقه‌بندی مناسبه، و شبکه‌های عصبی کانولوشن (CNN) که توی شناسایی تصاویر عملکرد خوبی دارن. سخت‌افزارهایی که تست کردن شامل دو پلتفرم مختلفه؛ یکی STM32-NucleoF401RE که یه برد میکروکنترلر 32STM با پردازنده 4ARM Cortex-M داره، و دیگری Arduino Nano BLE که اینم از یه پردازنده مشابه استفاده می‌کنه و برای مقایسه عملکرد آوردنش.

نتایجی که از این آزمایش‌ها گرفتن نشون میده که STM32Cube.AI روی برد STM32-NucleoF401RE زمان استنتاج سریع‌تری نسبت به TFLM داره و این یعنی STM32Cube.AI به‌خوبی برای سخت‌افزار 32STM بهینه‌شده. از نظر مصرف حافظه هم STM32Cube.AI بهتر عمل کرده و در نتیجه، بیشتر برای استفاده تو محیط‌های کم‌مصرف مناسبه. از لحاظ دقت هم هر دو فریم‌ورک تقریباً تو یه سطح قرار دارن و عملکرد بهتر STM32Cube.AI دقت مدل‌ها رو پایین نیاورده.

فریم‌ورک‌های مشابه TinyML برای سیستم‌های تعبیه‌شده

TensorFlow Lite Micro (TFLM):

نسخه‌ای از TensorFlow Lite که برای میکروکنترلرها طراحی شده تا مدل‌های یادگیری ماشین رو روی دستگاه‌هایی با حافظه و پردازش کم اجرا کنه.

Edge Impulse: یه پلتفرم برای توسعه و پیاده‌سازی مدل‌های یادگیری ماشین روی سخت‌افزار تعبیه‌شده که ابزارهایی برای جمع‌آوری داده، آموزش مدل و بهینه‌سازی فراهم می‌کنه.

Qeexo AutoML: این فریم‌ورک به‌طور خودکار مدل‌های یادگیری ماشین سبک برای دستگاه‌های تعبیه‌شده می‌سازه و برای مصرف کم و عملکرد سریع طراحی شده.

uTensor: یه فریم‌ورک متن‌باز که مخصوص میکروکنترلرها طراحی شده و برای اجرای بهینه مدل‌های یادگیری ماشین روی پلتفرم‌های تعبیه‌شده مناسبه.

CMSIS-NN: که توسط Arm توسعه داده شده و مجموعه‌ای از توابع شبکه عصبی بهینه برای پردازنده‌های Cortex-M هست که عملکرد یادگیری ماشین رو توی سیستم‌های تعبیه‌شده بهبود می‌ده.

TinyML یه حوزه‌ی جذاب و رو به رشده که می‌تونه هوشمندی رو به دستگاه‌های کوچیک و محدود از نظر منابع بیاره و این ابزارها و فریم‌ورک‌ها واقعاً کمک می‌کنن که برنامه‌های یادگیری ماشین توی این دستگاه‌ها به راحتی پیاده‌سازی بشن.

فشرده‌سازی و بهینه‌سازی مدل‌های یادگیری ماشین

تکنیک‌های فشرده‌سازی مدل یکی از موضوعات داغ در زمینه یادگیری ماشین تعبیه‌شده است. این تکنیک‌ها به کاهش حجم مدل‌ها و صرفه‌جویی در منابع کمک می‌کنند:

۱. کوآنتیزاسیون | **Quantization**: این روش داده‌های شناور ۳۲ بیتی را به اعداد ۸ بیتی تبدیل می‌کند. کوآنتیزاسیون سبب کاهش حجم مدل می‌شود و علاوه بر کاهش مصرف انرژی، سرعت پردازش را نیز افزایش می‌دهد.
۲. تقطیر دانش | **Knowledge Distillation**: این روش با آموزش یک مدل کوچک‌تر به کمک خروجی‌های مدل بزرگ‌تر، سبب انتقال دانش و کاهش پیچیدگی می‌شود.
۳. برش | **Pruning**: با حذف اتصالات غیرضروری در شبکه‌های عصبی، برش باعث کاهش تعداد پارامترها و سبک‌تر شدن مدل‌ها می‌شود، در نتیجه مدل بهینه‌تری ایجاد می‌شود که در سخت‌افزارهای تعبیه‌شده کارایی بهتری دارد.

پردازش Federated و حریم خصوصی داده‌ها

با توجه به اهمیت حفظ حریم خصوصی در یادگیری ماشین Federated Learning به یک تکنیک مؤثر برای سیستم‌های تعبیه‌شده تبدیل شده است:

- پردازش فدراسیونی امکان آموزش مدل‌ها را بدون نیاز به انتقال داده‌های حساس به سرورها فراهم می‌کند. این روش با انتقال به‌روزرسانی‌های مدل به جای داده‌های خام، امنیت داده‌ها را بهبود می‌بخشد.
- این تکنیک می‌تواند به‌ویژه در دستگاه‌های پوشیدنی و سیستم‌های پزشکی که حاوی داده‌های خصوصی هستند، مؤثر باشد.

هوش مصنوعی مبتنی بر رویداد (Event-Driven AI)

سیستم‌های تعبیه‌شده اغلب نیازمند واکنش سریع به رویدادها هستند. هوش مصنوعی مبتنی بر رویداد به کاهش پردازش و مصرف انرژی کمک می‌کند:

- این رویکرد تنها در زمانی که رویداد خاصی رخ می‌دهد، پردازش را آغاز می‌کند و به‌جای پردازش مداوم، بهینه‌سازی انرژی و سرعت پاسخ‌دهی را فراهم می‌کند.

- در سناریوهای حساس به انرژی، مانند نظارت محیطی یا دستگاه‌های پوشیدنی، این روش به عملکرد بهتر و طول عمر بیشتر دستگاه‌ها کمک می‌کند.

کاربردهای پیشرفته TinyML در صنایع مختلف

۱. کشاورزی هوشمند: سیستم‌های مبتنی بر TinyML در کشاورزی می‌توانند برای تشخیص بیماری‌های گیاهی، پیش‌بینی آفات و مانیتورینگ وضعیت رشد محصولات استفاده شوند. این سیستم‌ها قادرند اطلاعات دقیق و بلادرنگ از وضعیت محصولات فراهم کنند.
۲. سلامت الکترونیک و نظارت بر بیماران: در حوزه سلامت، TinyML می‌تواند در دستگاه‌های پایش سلامت به کار گرفته شود. تشخیص فعالیت‌های فیزیکی و وضعیت‌های غیرعادی (مانند افتادن بیمار) با استفاده از مدل‌های فشرده و کم‌مصرف امکان‌پذیر است.
۳. صنعت ۴.۰ و اینترنت اشیا صنعتی **TinyML**: به کنترل کیفیت محصولات و پیش‌بینی نقص در ماشین‌آلات کمک می‌کند. مدل‌های تعبیه‌شده با نظارت بر وضعیت ماشین‌آلات می‌توانند به تعمیرات پیش‌بینی‌شده و کاهش خرابی‌های ناگهانی کمک کنند.

امنیت در یادگیری ماشین تعبیه‌شده

یکی از چالش‌های کلیدی TinyML و یادگیری ماشین تعبیه‌شده، مسئله‌ی امنیت و پایداری مدل‌ها در برابر حملات سایبری است:

- حملات تزریق داده: در این نوع حملات، داده‌های مخرب به سیستم تزریق می‌شوند تا خروجی‌های نادرست تولید کنند. توسعه تکنیک‌های مقاوم‌سازی مدل در برابر این حملات ضروری است.
- حملات استنتاج: مهاجمان می‌توانند با بررسی خروجی مدل‌ها، اطلاعات حساس را استخراج کنند. برای جلوگیری از این موضوع، تکنیک‌های مبهم‌سازی مدل و حفاظت از داده‌ها توصیه می‌شود.

توسعه و آزمون مدل‌ها با استفاده از ابزارهای شبیه‌سازی

برای بهبود کارایی و پایداری مدل‌ها، ابزارهای شبیه‌سازی کمک می‌کنند که پیش از پیاده‌سازی، مدل‌ها را در محیط‌های مختلف تست کنیم:

- ابزارهایی مانند **Edge Impulse** و **TensorFlow Model Optimization Toolkit**، امکان شبیه‌سازی و ارزیابی مدل‌ها را در سناریوهای مختلف فراهم می‌کنند.
- شبیه‌سازی به تیم‌های تحقیقاتی این امکان را می‌دهد که رفتار مدل‌ها را در شرایط مختلف (مانند تغییرات محیطی و شبکه‌ای) ارزیابی کنند و به بهینه‌سازی و مقاوم‌سازی آن‌ها بپردازند.

سیستم‌های تعبیه‌شده به‌طور فزاینده‌ای از قابلیت‌های یادگیری ماشین (ML) برای افزودن هوشمندی در محیط‌های محدود از نظر منابع استفاده می‌کنند. ادغام ML در چنین سیستم‌هایی به دلیل محدودیت‌های حافظه، توان پردازشی و منابع انرژی، چالش‌هایی را ایجاد می‌کند. هدف این گزارش:

- بررسی کاربردهای مختلف الگوریتم‌های ML در سیستم‌های تعبیه‌شده.
- استخراج و دسته‌بندی بنچمارک‌های مشابه مورد استفاده در ادبیات موجود.
- تحلیل مقالات مرتبط برای درک وضعیت فعلی و جهت‌گیری‌های آینده.

---

۱. الگوریتم‌های یادگیری ماشین در سیستم‌های تعبیه‌شده و کاربردهای آن‌ها

سیستم‌های تعبیه‌شده از مجموعه‌ای از الگوریتم‌های ML استفاده می‌کنند، از روش‌های کلاسیک تا مدل‌های یادگیری عمیق سبک، که برای محیط‌های با منابع محدود بهینه‌سازی شده‌اند.

۱.۱ الگوریتم‌های کلاسیک یادگیری ماشین

این الگوریتم‌ها به دلیل سادگی و کارایی در دستگاه‌های با منابع محاسباتی محدود معروف هستند.

- **K**-نزدیک‌ترین همسایه‌ها (**KNN**):
  - کاربردها: تشخیص ناهنجاری، شناسایی ژست‌ها و وظایف ساده طبقه‌بندی.
  - مزایا: مصرف حافظه کم، پیاده‌سازی آسان و عملکرد خوب روی داده‌های کوچک.
- ماشین بردار پشتیبان (**SVM**):
  - کاربردها: طبقه‌بندی تصاویر، شناسایی اشیا و تشخیص خرابی.
  - مزایا: مؤثر در فضاهای با بعد بالا و مناسب برای مسائل طبقه‌بندی دودویی.
- درخت تصمیم:
  - کاربردها: نگهداری پیش‌بینی‌شده، نظارت محیطی و شناسایی فعالیت‌ها.

- مزایا: قابل تفسیر، سربار محاسباتی کم و قابلیت کار با داده‌های دسته‌ای.

۱.۲ مدل‌های یادگیری عمیق سبک

این مدل‌ها به‌طور خاص برای اجرای کارآمد روی سخت‌افزار تعبیه‌شده طراحی شده‌اند.

- **MobileNet** and **MobileNetV2**
  - کاربردها: طبقه‌بندی تصاویر، تشخیص اشیا و وظایف شناسایی بصری.
  - مزایا: استفاده از کانولوشن‌های تفکیکی عمقی که منجر به کاهش محاسبات و اندازه مدل می‌شود.
- **EfficientNet**
  - کاربردها: سناریوهایی که دقت بالاتر مورد نیاز است، مانند تصویربرداری پزشکی.
  - مزایا: تعادل در عمق، عرض و رزولوشن شبکه برای عملکرد بهینه.
- مدل‌های **TinyML**
  - کاربردها: شناسایی کلمات کلیدی، تشخیص ناهنجاری، وظایف استنتاج با تأخیر کم.
  - مزایا: اندازه بسیار کوچک، مناسب برای میکروکنترلرها و دستگاه‌های با توان فوق‌العاده کم.

۱.۳ تکنیک‌های بهینه‌سازی

برای مناسب‌سازی مدل‌های ML برای سیستم‌های تعبیه‌شده، چندین تکنیک بهینه‌سازی به کار گرفته می‌شوند:

- کوآنتیزاسیون (**Quantization**):
  - تبدیل اعداد شناور ۳۲ بیتی به اعداد صحیح ۸ بیتی.
  - مزایا: کاهش اندازه مدل، افزایش سرعت استنتاج و کاهش مصرف انرژی.
- برش (**Pruning**):
  - حذف اتصالات غیرضروری در شبکه‌های عصبی.
  - مزایا: کاهش پیچیدگی مدل و نیازهای حافظه.

- تقطیر دانش (**Knowledge Distillation**):
  - آموزش یک مدل کوچک‌تر با استفاده از خروجی‌های یک مدل بزرگ‌تر.
  - مزایا: حفظ عملکرد در حالی که اندازه مدل کاهش می‌یابد.

---

۲. بنچمارک‌ها در سیستم‌های یادگیری ماشین تعبیه‌شده

معیارگذاری برای ارزیابی عملکرد مدل‌های ML روی سخت‌افزار تعبیه‌شده تحت محدودیت‌های مختلف حیاتی است.

۲.۱ مرور بنچمارک‌ها

بنچمارک‌ها مدل‌ها را براساس معیارهای زیر ارزیابی می‌کنند:

- تأخیر: زمان مورد نیاز برای استنتاج.
- دقت: صحت پیش‌بینی‌های مدل.
- مصرف انرژی: میزان توان مصرفی در حین عملیات.
- استفاده از حافظه: نیازهای RAM و ذخیره‌سازی.

۲.۲ مجموعه‌های بنچمارک رایج

- **MLPerf Tiny:**
  - تمرکز بر معیارگذاری مدل‌های ML برای دستگاه‌های کوچک.
  - ارزیابی عملکرد در زمینه تأخیر، دقت و کارایی انرژی.
- بنچمارک‌های خاص سازنده:
  - ابزارهای معیارگذاری **STM32Cube.AI**: برای میکروکنترلرهای 32STM.
  - ابزارهای عملکردی **NVIDIA Jetson**: برای دستگاه‌های لبه‌ای Jetson.
  - ابزارهای معیارگذاری **Google Coral**: برای TPUهای Coral.

---

۳. دسته‌بندی بنچمارک‌ها

بنچمارک‌ها را می‌توان براساس پلتفرم‌های سخت‌افزاری، مدل‌های ML و حوزه‌های کاربردی دسته‌بندی کرد.

۳.۱ براساس پلتفرم‌های سخت‌افزاری

- میکروکنترلرها:
  - مثال‌ها: سری Ambiq Apollo، STM32، ARM Cortex-M3.
  - تمرکز بنچمارک: مصرف توان فوق‌العاده کم، استفاده حداقلی از حافظه.
- دستگاه‌های لبه‌ای:
  - مثال‌ها: Google Coral TPU، NVIDIA Jetson Nano
  - تمرکز بنچمارک: تعادل بین عملکرد و کارایی توان برای مدل‌های پیچیده‌تر.

۳.۲ براساس مدل‌های یادگیری ماشین

- الگوریتم‌های کلاسیک:
  - بنچمارک‌ها بر سرعت اجرا و ردپای حافظه تأکید دارند.
  - مناسب برای وظایف ساده روی میکروکنترلرها.
- مدل‌های یادگیری عمیق:
  - بنچمارک‌ها تأخیر استنتاج، مصرف انرژی و دقت را ارزیابی می‌کنند.
  - مناسب برای وظایف پیچیده روی دستگاه‌های لبه‌ای با شتاب‌دهنده‌ها.

۳.۳ براساس کاربردها

- شناسایی کلمات کلیدی:
  - دیتاست‌ها: Google Speech Commands
  - معیارهای بنچمارک: پردازش بلادرنگ، مصرف توان کم.

- طبقه‌بندی تصاویر و تشخیص اشیا:
  - دیتاست‌ها: CIFAR-10 و COCO
  - معیارهای بنچمارک: دقت، زمان استنتاج.
- تشخیص ناهنجاری:
  - کاربردها: نگهداری پیش‌بینی‌شده، نظارت امنیتی.
  - معیارهای بنچمارک: نرخ تشخیص، مثبت‌های کاذب.
- شناسایی ژست‌ها:
  - کاربردها: تعامل انسان و کامپیوتر، دستگاه‌های پوشیدنی.
  - معیارهای بنچمارک: زمان پاسخ، کارایی انرژی.

---

۴. تحلیل مقالات و دسته‌بندی کاربردها

۴.۱ مقاله: یادگیری ماشین در سیستم‌های تعبیه‌شده: محدودیت‌ها، راه‌حل‌ها و چالش‌های آینده

- یافته‌های کلیدی:
  - چالش‌ها: محدودیت منابع، کمبود داده، ایمنی و نیاز به پردازش بلادرنگ.
  - راه‌حل‌ها: بهینه‌سازی مدل، سخت‌افزارهای مخصوص، محاسبات درون حافظه.
  - اهمیت بنچمارک‌ها: برای انتخاب مدل‌های مناسب براساس مصرف انرژی، سرعت و دقت.
- دسته‌بندی کاربردها:
  - بر نیاز به مدل‌های ML مخصوص در کاربردهای حساس به ایمنی مانند دستگاه‌های پزشکی تأکید دارد.
  - نقش معیارگذاری در انتخاب مدل برای سیستم‌های تعبیه‌شده را برجسته می‌کند.

۴.۲ مقاله **TinyML**: مروری سیستماتیک و ترکیب تحقیقات موجود

- یافته‌های کلیدی:

- پلتفرم‌های سخت‌افزاری: استفاده از میکروکنترلرهای فوق‌العاده کم‌مصرف.
- فریم‌ورک‌ها: TinyOL ،TensorFlow Lite برای Arduino.
- کاربردها: شناسایی کلمات کلیدی، طبقه‌بندی تصاویر، تشخیص ناهنجاری.
- چالش‌ها: کمبود دیتاست‌های مناسب، معیارهای ارزیابی متمرکز بر کارایی انرژی.

- دسته‌بندی کاربردها:
  - بر کاربردهای لبه‌ای که نیاز به پاسخ فوری و مصرف انرژی کم دارند تمرکز می‌کند.
  - به گسترش TinyML به بخش‌هایی مانند کشاورزی هوشمند و سلامت الکترونیک اشاره دارد.

۴.۳ مقاله: مروری بر یادگیری ماشین در کاربردهای تعبیه‌شده و موبایل

- یافته‌های کلیدی:
  - الگوریتم‌های بررسی‌شده: KNN، SVM، درخت تصمیم، نسخه‌های MobileNet و EfficientNet.
  - پلتفرم‌های سخت‌افزاری: ARM Cortex-M, NVIDIA Jetson, Google Coral TPU
  - نتایج:
    - الگوریتم‌های کلاسیک روی میکروکنترلرها عملکرد خوبی دارند.
    - MobileNetV2 روی Coral TPU استنتاج بلادرنگ ارائه می‌دهد.
    - EfficientNet در شرایطی که دقت مهم‌تر از سرعت است، عملکرد بهتری دارد.
- دسته‌بندی کاربردها:
  - قابلیت اجرای هر دو مدل کلاسیک و یادگیری عمیق روی سیستم‌های تعبیه‌شده با بهینه‌سازی مناسب را نشان می‌دهد.
  - پیشنهاد می‌کند که انتخاب مدل براساس نیازهای کاربردی (سرعت در مقابل دقت) صورت گیرد.

---

۵. کاربردهای پیشرفته و جهت‌گیری‌های آینده

۵.۱ حوزه‌های کاربردی

- کشاورزی هوشمند:
  - استفاده‌ها: تشخیص بیماری‌های گیاهی، پیش‌بینی آفات، مانیتورینگ رشد محصولات.
  - الگوریتم‌های ML: CNNهای سبک، الگوریتم‌های تشخیص ناهنجاری.
- سلامت الکترونیک و نظارت بر بیماران:
  - استفاده‌ها: شناسایی فعالیت‌ها، تشخیص افتادن، مانیتورینگ علائم حیاتی.
  - الگوریتم‌های **ML**: الگوریتم SVM، درخت تصمیم، مدل‌های یادگیری عمیق فشرده.
- صنعت ۴.۰ و اینترنت اشیا صنعتی:
  - استفاده‌ها: نگهداری پیش‌بینی‌شده، کنترل کیفیت، تشخیص خرابی.
  - الگوریتم‌های **ML**: الگوریتم KNN، تشخیص ناهنجاری، مدل‌های تحلیل سری زمانی.

۵.۲ ملاحظات امنیتی در یادگیری ماشین تعبیه‌شده

- چالش‌ها:
  - حملات تزریق داده: داده‌های مخرب که منجر به خروجی‌های نادرست می‌شوند.
  - حملات استنتاج: استخراج اطلاعات حساس از خروجی‌های مدل.
- استراتژی‌های کاهش خطر:
  - مقاوم‌سازی مدل: تکنیک‌های آموزشی مقاوم در برابر ورودی‌های مخرب.
  - حفاظت از داده‌ها: رمزنگاری، روش‌های حفظ حریم خصوصی تفاضلی.

۵.۳ توسعه و آزمون مدل‌ها با استفاده از ابزارهای شبیه‌سازی

- ابزارها:
  - **Edge Impulse**، پلتفرمی جامع برای جمع‌آوری داده، آموزش مدل و پیاده‌سازی.
  - **TensorFlow Model Optimization Toolkit**، ارائه قابلیت‌های کوآنتیزاسیون و برش.
- مزایا:
  - شبیه‌سازی: تست مدل‌ها تحت شرایط مختلف قبل از پیاده‌سازی.
  - بهینه‌سازی: تنظیم مدل‌ها برای عملکرد و محدودیت‌های منابع.

نتیجه‌گیری

ادغام یادگیری ماشین در سیستم‌های تعبیه‌شده در حال تحول صنایع مختلف است و قابلیت‌های هوشمند را در محیط‌های با منابع محدود فراهم می‌کند. الگوریتم‌های کلاسیک ML و مدل‌های یادگیری عمیق بهینه‌شده، هنگامی که با فریم‌ورک‌ها و تکنیک‌های بهینه‌سازی سخت‌افزار-محور ترکیب می‌شوند، می‌توانند به‌طور مؤثر چالش‌های سیستم‌های تعبیه‌شده را برطرف کنند.

بنچمارکینگ نقش مهمی در ارزیابی و انتخاب مدل‌های مناسب براساس نیازهای کاربردی ایفا می‌کند. مقالات تحلیل‌شده بر اهمیت موارد زیر تأکید دارند:

- بهینه‌سازی مدل: تکنیک‌هایی مانند کوآنتیزاسیون، برش و تقطیر دانش ضروری هستند.
- ملاحظات سخت‌افزاری: استفاده از سخت‌افزارهای مخصوص و فریم‌ورک‌های بهینه‌شده برای پلتفرم‌های خاص.
- امنیت و حریم خصوصی: پرداختن به آسیب‌پذیری‌ها از طریق طراحی مقاوم و روش‌های حفظ حریم خصوصی.

تحقیقات و توسعه‌های آینده باید بر گسترش کاربردهای TinyML، تقویت اقدامات امنیتی و بهبود ابزارهای شبیه‌سازی برای تسهیل استقرار مدل‌های ML در سیستم‌های تعبیه‌شده تمرکز کنند.

مقاله Edge AI in Sustainable Farming

خلاصه: این مقاله به بررسی و توسعه یک چارچوب IoT مبتنی بر هوش مصنوعی لبه‌ای می‌پردازد که از مدل‌های یادگیری عمیق برای حفاظت از محصولات کشاورزی در برابر تهدیدات حیات وحش استفاده می‌کند. با استفاده از سنسورها و دوربین‌های نصب‌شده در محیط‌های کشاورزی، داده‌های بلادرنگ جمع‌آوری و تحلیل می‌شوند تا تهدیدات احتمالی مانند ورود حیوانات وحشی به مزارع شناسایی و واکنش نشان داده شود. این سیستم با هدف کاهش مصرف انرژی و بهبود سرعت پردازش در دستگاه‌های لبه‌ای بهینه‌سازی شده است، که برای کاربردهای کشاورزی پایدار بسیار حیاتی است.

نکات کلیدی:

- تکنولوژی **:IoT** استفاده از سنسورها و دوربین‌ها برای جمع‌آوری داده‌های محیطی.
- یادگیری عمیق: تحلیل بلادرنگ داده‌ها برای شناسایی تهدیدات.
- حفاظت از محصولات: پیشگیری از خسارات ناشی از حیوانات وحشی.
- بهینه‌سازی انرژی: کاهش مصرف انرژی و افزایش کارایی در دستگاه‌های لبه‌ای.

مقاله TinyissimoYOLO

خلاصه: این مقاله TinyissimoYOLO را معرفی می‌کند، که یک شبکه تشخیص اشیاء سبک و کم‌حافظه است که برای میکروکنترلرهای با توان پایین طراحی شده است. با استفاده از تکنیک‌های کوانتیزاسیون و بهینه‌سازی معماری شبکه، TinyissimoYOLO قادر است اشیاء را با دقت مناسبی شناسایی کند در حالی که نیاز به منابع محاسباتی و حافظه‌ی محدودی دارد. این شبکه به‌ویژه برای کاربردهای IoT و سیستم‌های تعبیه‌شده که محدودیت‌های سخت‌افزاری دارند، بسیار مناسب است.

نکات کلیدی:

- کوانتیزاسیون: کاهش دقت داده‌ها برای کاهش حجم مدل.
- کم‌حافظه بودن: طراحی شبکه با مصرف حافظه کم.
- کارایی بالا: حفظ دقت در تشخیص اشیاء با مصرف انرژی پایین.
- میکروکنترلرهای توان پایین: مناسب برای دستگاه‌های با منابع محدود.

مقاله Efficient Deep Learning Infrastructures for Embedded Computing Systems

خلاصه: این مقاله یک بررسی جامع از زیرساخت‌های یادگیری عمیق بهینه‌شده برای سیستم‌های کامپیوتری تعبیه‌شده ارائه می‌دهد. نویسندگان تکنیک‌های مختلفی مانند بهینه‌سازی سخت‌افزار، معماری‌های شبکه عصبی سبک و چارچوب‌های نرم‌افزاری را که برای اجرای مدل‌های یادگیری عمیق بر روی دستگاه‌های با منابع محدود مناسب هستند، مورد بررسی قرار داده‌اند. علاوه بر این، مقاله به چالش‌های فعلی و مسیرهای آینده در این حوزه اشاره می‌کند، از جمله نیاز به توسعه الگوریتم‌های بهینه‌تر و سخت‌افزارهای اختصاصی.

نکات کلیدی:

- بهینه‌سازی سخت‌افزار: استفاده از سخت‌افزارهای خاص برای اجرای مدل‌های یادگیری عمیق.
- معماری‌های سبک شبکه عصبی: طراحی شبکه‌های عصبی با مصرف منابع کم.
- چارچوب‌های نرم‌افزاری: ابزارها و فریم‌ورک‌هایی که اجرای مدل‌های یادگیری عمیق را در سیستم‌های تعبیه‌شده تسهیل می‌کنند.
- چالش‌ها و آینده: نیاز به توسعه بیشتر در الگوریتم‌ها و سخت‌افزارهای بهینه‌شده.

MLPerf Inference Tiny

MLPerf Inference Tiny یک مجموعه استاندارد بنچمارک است که کارایی سیستم‌ها در پردازش ورودی‌ها و تولید نتایج با استفاده از مدل‌های آموزش‌دیده را اندازه‌گیری می‌کند. این بنچمارک‌ها برای ارزیابی عملکرد سیستم‌ها در سناریوهای مختلف و با استفاده از معیارهای مشخص طراحی شده‌اند.

سناریوها و معیارها در MLPerf چهار سناریوی مختلف برای تست کردن تنوع پلتفرم‌های استنباط و کاربردهای آن‌ها تعریف شده است:

۱. جریان تکی | **Single stream**: بارگیری یک درخواست بعد از تکمیل درخواست قبلی.
۲. جریان چندگانه | **Multiple stream**: برای نسخه‌های قبلی 1.1، درخواست جدید بعد از هر محدودیت زمانی ارسال می‌شود و اگر درخواست قبلی تکمیل شده باشد. در نسخه‌های 2.0 و بعد، بلافاصله بعد از تکمیل درخواست قبلی، درخواست بعدی ارسال می‌شود.
۳. سرور | **Server**: درخواست‌های جدید براساس توزیع پواسون ارسال می‌شوند.
۴. آفلاین | **Offline**: همه درخواست‌ها در ابتدا ارسال می‌شوند.

بنچمارک‌ها هر بنچمارک براساس داده‌ها و هدف کیفیت تعریف شده است. به عنوان مثال، برای تشخیص کلمات کلیدی، داده‌های Google Speech Commands و مدل DS-CNN استفاده می‌شود و کیفیت براساس درصد دقت ارزیابی می‌شود.

MLPerf دو دیویژن را معرفی می‌کند که امکان بازنویسی پیاده‌سازی‌های مرجع را فراهم می‌کنند:

- بخش بسته | **Closed Division**: برای مقایسه سخت‌افزارها یا فریم‌ورک‌های نرم‌افزاری با شرایط یکسان.
- بخش باز | **Open Division**: برای تشویق نوآوری، استفاده از مدل‌های متفاوت یا بازآموزی مجاز است.

نتایج نتایج هر بنچمارک براساس معیارهای تعیین شده ارائه می‌شوند و شامل جزئیات مربوط به سخت‌افزار، نرم‌افزار، و نتایج اختصاصی هر بنچمارک می‌شوند.

MLPerf Tiny به ویژه برای سیستم‌های بسیار کم‌قدرت مانند میکروکنترلرها طراحی شده است و هدف آن ارائه مجموعه‌ای از شبکه‌های عصبی عمیق و کد بنچمارک برای مقایسه عملکرد بین دستگاه‌های تعبیه شده است.