دانشگاه صنعتی شریف

دانشکده مهندسی برق

پایان نامه کارشناسی ارشد

گرایش سیستم های الکترونیک دیجیتال

پیاده سازی یک پلتفرم MLOps به صورت ابری روی GPU

نگارنده

ابوالفضل یاریان

استاد راهنما

دکتر متین هاشمی

خردادماه ۱۴۰۳

توجه

این پروژه بر اساس قرارداد شماره (.....) از حمایت مالی مرکز تحقیقات مخابرات ایران برخوردار شده است.

بسمه تعالی

دانشگاه صنعتی شریف
دانشکده مهندسی برق

پایان‌نامه کارشناسی ارشد

عنوان: **مدلسازی نهان نگاری تصویر بر اساس تئوری اطلاعات**

نگارش: «**نام و نام‌خانوادگی دانشجو**»

اعضا هیات داوران:

دکتر ... امضاء:.....

دکتر ... امضاء:.....

دکتر ... امضاء:.....

دکتر ... امضاء:.....

دکتر ... امضاء:.....

تاریخ: ۶ شهریور ۱۳۸۴.

تقدیم و قدردانی

در این صفحه از کسانی که مایلید تشکر می‌کنید.

چکیده:

در دنیای دیجیتال امروزه، نهان نگاری مقاومِ تصویر که در آن یک سیگنال حامل داده به صورت نامرئی و مقاوم در برابر حملات در تصویر تعبیه می‌شود، به عنوان یک راهکار برای حل مساله حفاظت از حق تالیف محصولات تصویری معرفی شده است. برای این منظور تاکنون جهت نهان نگاری روشهای متعددی به کار گرفته شده است که از آن جمله می‌توان به استفاده از مدلهای بینایی جهت یافتن میزان بیشینهٔ انرژی نهان نگاره برای تعبیه در تصویر و استفاده از حوزه های مقاوم در برابر حملات، اشاره نمود. در همین راستا در این پایان نامه به استفاده از مفاهیم حوزه تئوری اطلاعات به عنوان یک راهنما در توسعهٔ الگوریتمهای موجود، جهت قرار دادن بهینه نهان نگاره پرداخته شده است. همچنین در ساختار پیشنهادی که برای افزایش مقاومت در حوزه تبدیل تصویر پیاده می‌شود، از تبدیلات چنددقتی مانند تبدیل موجک گسسته و تبدیل MR-SVD که به سیستم بینائی انسان نزدیکترند، استفاده می‌شود. به طوریکه در حوزه تبدیل موجک، با استفاده از آنتروپی و تاثیر پدیدهٔ پوشش آنتروپی به اصلاح مدلهای بینایی مرتبط با این حوزه پرداخته و بدین ترتیب نهان نگاره با قدرت و مقاومت بالاتر در تصویر تعبیه نموده و همچنین کیفیت بهتر برای تصویر نهان نگاری شده بدست آمد. همچنین در حوزهٔ تبدیل MR-SVD ابتدا این تبدیل که تاکنون برای نهان نگاری استفاده نشده بود، جهت نهان نگاری بکار گرفته شد و سپس مشابه ساختار پیشنهادیِ مبتنی بر آنتروپی در حوزه تبدیل موجک، در حوزه این تبدیل نیز بکار رفت و نتایج شبیه‌سازیها مقاوم‌تر بودن ساختار پیشنهادی و کیفیت بالاتر تصویر نهان نگاری شده در این حوزه را نتیجه داد.

کلمات کلیدی:

۱- نهان نگاری تصویر Image Watermarking.
۲- تبدیل چنددقتی Multi-Resolution Transform.
۳- سیستم بینایی انسان Human Visual System (HVS).
۴- تبدیل موجک Wavelet Transform.
۵- تجزیه مقادیر تکین Singular Value Decomposition (SVD).
۶- آنتروپی Entropy.
۷- پوشش آنتروپی Entropy Masking.

فهرست مطالب

فهرست جداول

فهرست تصاویر

فهرست کلمات اختصاری

2D-DWT	2-Dimensional Discrete Wavelet Transform
CPD	Cycle Per Degree
CSF	Contrast Sensitivity Function
⋮	⋮

فصل ۱

مقدمه

گسترش روز افزون شبکه جهانی اینترنت و توسعه فناوری اطلاعات، نیاز فزاینده‌ای را به استفاده از سرویسهای چندرسانه‌ای دیجیتال، در پی داشته به طوریکه کاربردهای دیجیتال شاهد رشد شگرفی در طول دهه گذشته بوده است که نتیجه آن ایجاد سیستمهای کارآمد در ذخیره، انتقال و بازیابی اطلاعات است. مزایای فراوان فناوری دیجیتال، باعث محبوبیت و کاربرد هر چه بیشتر آن توسط اشخاص شده تا جاییکه حتی وسایل ضبط و پخش صدا و تصویر آنالوگ خانگی هم به سرعت با نمونه‌های دیجیتال جایگزین شده‌اند. اما این موضوع مسائل حاشیه‌ای دیگری برای بشر ایجاد نموده است. به طوریکه امکان تهیه کپی‌های متعدد از روی نسخه اصلی بدون کاهش کیفیت آن و یا سادگی جعل و تغییر محتوای اطلاعاتی نسخه اصلی، باعث شده که مالکیت معنوی[۱] صاحبان اثر به خطر افتاده و در نتیجه بسیاری از ارائه دهندگان سرویسهای چندرسانه‌ای (از جمله شرکتهای فیلم‌سازی) از ارائه نمونه دیجیتال محصولاتشان خودداری نمایند. لذا برطرف نمودن این مشکلات، یکی از زمینه‌های پژوهشی مهم در عرصه مخابرات و بخصوص پردازش سیگنال است.

[۱] Intellectual Property

فصل ۲

مرور مفاهیم پایه

۲-۱ DevOps

۲-۱-۱ تعریف

دِواپس که از اتحاد واژگان Development و Operation به وجود آمده است؛ ترکیبی از ابزارها، کنش‌ها و فرهنگ کاری است که تیم های توسعه[۱] و عملیات[۲] را به همکاری موثرتر نزدیک می کند و کسب و کارها با استفاده از می‌توانند اپلیکیشن‌ها و سرویس‌هایشان را با سرعت بالاتری نسبت به روش‌های سنتی تحویل دهند. همین سریع‌تر شدن سرعت توسعه و انتشار نرم‌افزار، سازمان‌ها را قادر می‌سازد تا در مقایسه با کسب‌وکارهایی که هنوز از روش‌های سنتی توسعه نرم‌افزار استفاده می‌کنند خدمات بهتری به مشتریانشان ارائه دهند. در واقع دواپس سعی دارد تا مشکل جدایی تیم‌های مختلف را رفع کرده و یک فرهنگ سازمانی یکپارچه را میان تیم‌های مختلفی که در حال توسعه یک نرم‌افزار هستند ایجاد کند. از این جهت بسیاری از کارها می‌تواند به صورت خودکار پیش رفته و در نهایت همه چیز با سرعت بیشتری صورت بگیرد [۲۴، ۱۲]. این خودکار سازی با استفاده از خط لوله CI/CD از منبع کد شروع می شود و تا مانیتورینگ محصول ادامه میابد [۳۵].

تا قبل از تشکیل دواپس، تیم‌های توسعه نرم‌افزار یا تیم عملیاتی در محیط‌های جداگانه کار می‌کردند. هدف تیم توسعه تولید محصول جدید و یا افزودن ویژگی‌های جدیدی روی محصولات قبلی بود. هدف تیم عملیاتی نیز ثابت نگه داشتن وضعیت موجود سرویس‌ها برای پایداری بیشتر بود. به مرور زمان در فرآیند توسعه نرم‌افزار، روش‌های چابک[۳] ایجاد شد تا با مشتری تعامل بهتری برقرار شود و نیازهایی که دارد به محصول اضافه شود [۲۵]. جدایی دو تیم توسعه و عملیات از هم باعث

[۱] Development
[۲] Operation
[۳] Agile

می‌شد که در فرآیند تولید محصول و استقرار[۴] آن، اتلاف وقت ایجاد شود و محصول دیرتر به دست مشتری برسد [۲۲].

۲-۱-۲ چرخه کاری DevOps

همانطور که در شکل ۲-۱ مشاهده می کنید، DevOps قصد دارد از ابزار و جریان های کاری[۵] برای خودکارسازی یک یا چند مورد از موارد زیر استفاده کند:

۱. کدنویسی: شامل توسعه، بازبینی کد و ابزارهای کنترل نسخه است. مثلا، یک تیم تصمیم می گیرد از گیت[۶] به عنوان ابزار کنترل نسخه و از گیت هاب[۷] نیز به عنوان یک مخزن راه دور استفاده کند. این تیم مجموعه‌ای از دستورالعمل‌های سبک کدنویسی را با استفاده از ابزاری نظیر Linter به همراه حداقل درصد پوشش تست تعریف کرده و با تعیین استراتژی انشعاب مبتنی بر تنه[۸] تغییرات خود را به منظور بازبینی برای ادغام با انشعاب اصلی[۹] برای توسعه دهنده ارشد ارسال می کند [۲۹].

۲. ساخت: شامل ایجاد و ذخیره خودکار مولفه[۱۰] ها می باشد. به طور مثال یک تیم تصمیم می گیرد یک Container image قابل اجرا از محصول خود ایجاد کند.

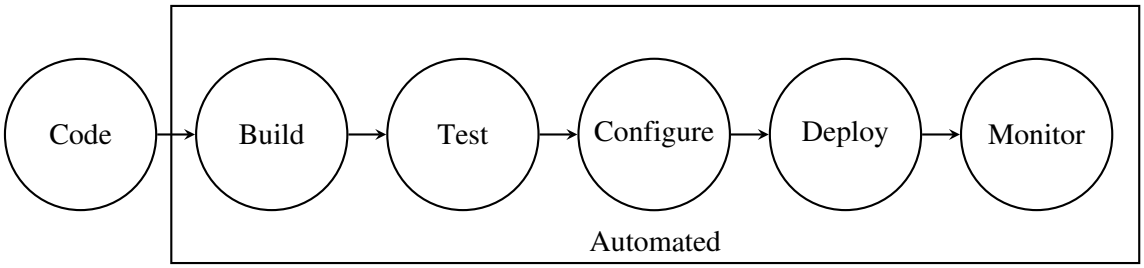
۳. تست: شامل ابزارهایی برای تست محصول می باشد. تیم محیطی را به منظور تست هر تغییر جدید راه اندازی می کند که در آن مجموعه‌ای از آزمایش‌ها مانند آزمون واحد[۱۱]، آزمون یکپارچگی[۱۲] و ... به‌طور خودکار در برابر هر ویرایش کد اجرا می‌شود.ادغام و تست کد به طور مکرر، به تیم‌های توسعه کمک می‌کند تا از کیفیت کدشان اطمینان حاصل کرده و جلوی خطاهای احتمالی را بگیرند.

۴. پیکربندی: شامل پیکربندی و مدیریت خودکار زیرساخت می باشد. این مورد شامل مجموعه ای از اسکریپت هایی برای بازتولید محیط در حال اجرا و زیرساخت نرم افزاری شامل سیستم عامل تا پایگاه داده و سرویس های خاص و پیکربندی شبکه آنها می باشد [۲۳، ۱۵].

۵. استقرار: این مرحله شامل استراتژی استقرار است. به طور مثال تیم می تواند تصمیم بگیرد که یک محصول به طور

---

[۴] Deploy
[۵] Workflow
[۶] Git
[۷] Github
[۸] Trunk-Based
[۹] Merge request
[۱۰] Artifact
[۱۱] Unit test
[۱۲] Integration test



شکل ۲-۱: مراحل DevOps

مستقیم منتشر شود یا ابتدا در یک محیط آزمایشی مورد ارزیابی قرار گیرد. هم چنین در مواقعی که مشکلی در استقرار وجود دارد چه کاری انجام دهند و استراتژی بازگشت[۱۳] خود را پیاده سازی کنند.

۶. نظارت: از عملکرد محصول تا نظارت بر تجربه کاربر نهایی را شامل می شود. به عنوان مثال، می‌تواند مدت زمان درخواست‌های پایگاه داده یا بارگذاری وب‌سایت یا تعداد کاربرانی که از ویژگی‌های خاص محصول استفاده می‌کنند یا تعداد بازدیدکنندگان از یک وب‌سایت که به ثبت نام ختم می‌شود یا تعداد کاربران جدید در یک مجموعه زمانی خاص را پوشش دهد. مرحله نظارت هم چنین شامل هشدار خودکار خرابی ها نیز می باشد (به عنوان مثال، آستانه استفاده از CPU) [۱۷]. درنهایت نظارت بر محیط تولید به منظور اطمینان از صحت کارکرد صحیح محصول ضروری است.

۲-۱-۳ خط لوله CI/CD

در دنیای توسعه نرم‌افزار دستیابی به بهره‌وری بالا، کیفیت مطلوب محصول و رضایت مشتری از اهداف اصلی هر سازمانی است. در متدلوزی DevOps و رویکردهای مرتبط با آن مانند ادغام مداوم[۱۴] ، تحویل مداوم[۱۵] و استقرار مداوم[۱۶] به‌طور فزاینده‌ای محبوب شده‌اند زیرا به سازمان‌ها کمک می‌کنند تا با سرعت و کارآمدی بیشتری به این اهداف دست یابند [۲۲].

ادغام مداوم به فرآیندی اطلاق می‌شود که در آن توسعه‌دهندگان برنامه‌های خود را به‌طور مداوم (معمولاً چندین بار در روز) در یک مخزن مشترک ادغام می‌کنند. به محض ادغام کد، یک سری از تست‌های خودکار اجرا می‌شود تا اطمینان حاصل شود که این تغییرات جدید باعث بروز مشکل در نرم‌افزار نشده‌اند [۱۴]. این تست‌ها شامل تست‌های واحد، تست‌های یکپارچگی و تست‌های کارکردی می‌باشند.از آنجایی که برنامه های کاربردی پیشرفته کنونی در چندین پلتفرم و ابزار های مختلف اقدام به توسعه می کنند، لذا نیاز به مکانیزمی برای ادغام و تایید تغییرات مختلف، اهمیت بالاتری پیدا می کند.

[۱۳] Rollback
[۱۴] Continuous Integration (CI)
[۱۵] Continuos Delivery (CD)
[۱۶] Continuous Deployment (CD)

جدول ۲-۱: نمونه هایی از ابزار برای مراحل خاص اتوماسیون خط لوله CI/CD

Phase	Tools
Build	Gradle, Bazel, Docker
Test	Selenium, pytest
Configure	Ansible, Terraform
Deploy	ArgoCD, Jenkins
Monitor	Prometheus, Sentry

تحویل مداوم ادامه ای بر ادغام مداوم است و به تیم‌ها این امکان را می‌دهد تا نرم‌افزار را پس هر تغییر مهم در کد به مرحله تولید برسانند. در این مدل، هر خروجی که از فرایند CI عبور کرده و تست‌های لازم را با موفقیت پشت سر گذاشته باشد، به صورت خودکار آماده انتشار می‌شود [۱۴]. به عبارتی دیگر، هدف از تحویل مداوم، داشتن پایگاه کدی است که همیشه آماده استقرار در محیط تولید باشد. این فرایند ممکن است شامل تست‌های اضافی برای ارزیابی عملکرد، امنیت و سازگاری با محیط‌های تولید نیز باشد.

استقرار مداوم که گاهی با تحویل مداوم اشتباه گرفته می‌شود، به فرآیندی اطلاق می‌شود که در آن هر تغییر در کد که تمام مراحل تست و تأیید را با موفقیت پشت سر می‌گذارد، به صورت خودکار در محیط تولید قرار می‌گیرد [۱۴]. این به معنای آن است که نسخه‌های جدید نرم‌افزار می‌توانند به طور مداوم و بدون دخالت دستی به کاربران نهایی تحویل داده شوند. این رویکرد به تیم‌ها کمک می‌کند تا سریعتر به بازخوردها پاسخ دهند و بهبودهای مستمری را در محصول خود اعمال کنند، اما نیازمند یک فرآیند آزمایشی بسیار قوی و اطمینان از کیفیت کد است.

فرآیند کامل CI/CD که در شکل ۲-۱ هم به عنوان بخشی از چرخه کاری توضیح داده شد [۲۲]، با یک فرآیند ساخت شروع می‌شود. در این مرحله کد توسط ابزارهای مرتبط که در جدول ۲-۱ ذکر شده است، تبدیل به نرم‌افزار قابل اجرا می‌شوند. پس از این مرحله، تست‌های خودکار که شامل تست‌های واحد، تست‌های یکپارچه‌سازی و تست‌های رابط کاربری هستند، اجرا می‌شوند تا اطمینان حاصل شود که تغییرات جدید باعث بروز خطا در نرم‌افزار نمی‌شوند. در صورت موفقیت‌آمیز بودن تست‌ها، یک نسخه قابل اجرا از کد نسخه گذاری شده و در مخازنی همانند Nexus نگه داری می شوند. در مرحله پیکربندی تنظیم محیط لازم برای نصب و استفاده از نرم‌افزار انجام می‌شود. دو رویکرد اصلی برای این کار وجود دارد: مرحله به مرحله[۱۷] و اعلامی[۱۸]. در رویکرد اول، پیش‌نیازها به ترتیب آماده‌سازی می‌شوند و شکست در هر مرحله می‌تواند به عدم انجام دادن مراحل بعدی منجر شود. این رویکرد، که اغلب با استفاده از ابزارهایی مانند Ansible پیاده‌سازی می‌شود، زمانی مفید است که نیاز به اعمال تغییرات جزئی بر محیط باشد. در مقابل، رویکرد اعلامی به طور همزمان کل محیط را بر اساس

[۱۷] Procedural
[۱۸] Declarative

یک حالت نهایی تعریف‌شده آماده می‌کند. این رویکرد باعث می‌شود که در صورت بروز خطا در یک بخش، سایر بخش‌ها تحت تأثیر قرار نگیرند. برای اجرای این رویکرد، می‌توان از ابزارهایی مثل Terraform استفاده کرد[۲۰]. پس از این، مرحله‌ی استقرار آغاز می‌شود که در آن نرم‌افزار به محیط‌های تست، توسعه یا تولید منتقل می‌شود. این فرایند اغلب شامل مکانیزم‌هایی برای پشتیبانی و بازگرداندن نسخه‌های قبلی در صورت بروز مشکل است. یکی از قسمت های فرآیند کامل این خط لوله نیز با استفاده از ابزاری مانند Gitlab CI، CircleCI و Jenkins می تواند انجام گردد. در آخر نیز مرحله نظارت انجام می شود. ابزارهای نظارتی نظیر Prometheus و Grafana معمولاً شامل نمودارها، گزارش‌ها، و آمارهایی هستند که به شما اطلاعاتی در مورد وضعیت فعلی خط لوله و عملکرد برنامه‌های آزمایشی و انتشارات را ارائه می‌دهند. هم چنین اطلاعاتی مانند زمان طول کشیده برای هر مرحله، تعداد خطاها و متوسط زمان بین خرابی‌ها[۱۹] از جمله آمارهایی هستند که ممکن است در این مرحله نمایش داده شوند. لازم به ذکر است که می توان به منظور بررسی سبک کدنویسی و اعمال استاندارد های تیم توسعه در ابتدا خط لوله بخشی را قرار داد تا از استاندارد بودن کد اطمینان یابد. در این بخش می توان از Linter ها یا pre-commit hooks استفاده کرد. معروف ترین ابزار برای هر مرحله را در جدول ۲-۱ نیز مشاهده می کنید.

به هنگام طراحی و پیاده سازی یک خط لوله CI/CD باید به نکات زیر توجه کرد [۲۲، ۲۰، ۱۴]:

- قابلیت بازگشت به حالت قبل[۲۰]
- قابلیت مشاهده[۲۱] و هشداردادن[۲۲]
- امنیت
- مدت زمان اجرای خط لوله

برای هر نسخه از کد باید یک استراتژی بازگشت وجود داشته باشد تا اگر مشکلی پیش آمد، به نسخه قبلی بازگردانده شود. یک راه حل آسان برای بازگشت می تواند اجرای نسخه قدیمی تر از طریق همان خط لوله CI/CD باشد. بازگشت به ورژن قبلی همیشه ساده نیست، چراکه اگر سرویس در حال اجرا قابل بازگشت باشد، باید به بازگرداندن داده‌ها قبلی و زمان توقف هنگام استقرار نسخه جدید نیز توجه کرد.

هر انتشار باید شفاف باشد که چه تغییراتی اعمال شده و چه کسی تغییرات را تأیید کرده است. هم چنین تیم توسعه باید بداند که استقرار موفقیت‌آمیز بوده و چه زمانی انجام شده است. درنهایت باید هشدارهای واضحی از کد خراب در خط لوله و

[۱۹] Mean time between failures (MBTF)
[۲۰] Rollback
[۲۱] Observability
[۲۲] Alerting

خطای احتمالی در انتشار وجود داشته باشد. اگر مشکلی در استقرار پیش آید و هیچ سابقه واضحی از تغییرات وجود نداشته باشد، بازگشت به حالت قبل دشوار خواهد بود. علاوه بر این دلیل بروز این مشکل در استقرار نیز برای تیم نامعلوم است. تصور کنید که یک توسعه‌دهنده به صورت دستی به یک ماشین محیط تولید دسترسی پیدا کرده و به طور تصادفی یک فایل کلیدی را حذف می‌کند که پس از چند روز باعث خرابی‌های سیستم می‌شود. هیچ ردی از این تغییرات وجود ندارد، هیچکس نمی‌داند کجا را باید به حالت قبلی برگرداند و حتی بازگشت به کد قبلی ممکن است کمکی نکند زیرا فایل گمشده ممکن است در ورژن قبلی بازتولید نشود.

توجه به امنیت در فرآیندهای CI/CD بسیار حیاتی است تا سلامت و امنیت فرآیندهای توسعه و ارسال نرم‌افزار حفظ شود. اجرای کنترل دسترسی بر اساس نقش[۲۳] ضروری است تا فقط افراد مجاز بتوانند تغییراتی در فرآیند CI/CD اعمال کرده و کد را ارسال کنند. این شامل کنترل دسترسی به ابزارهای CI/CD نظیر Jenkins و همچنین به هر سیستم متمرکز مانند مخازن کد منبع است. هم چنین مدیریت اسرار[۲۴] جنبه اساسی امنیت خط لوله است. اسراری مانند کلیدهای API، رمزعبورها و گواهی‌نامه‌ها باید به طور امن ذخیره و دسترسی‌پذیر باشند. استفاده از ابزارهایی مانند HashiCorp Vault می‌تواند به مدیریت امنیت اسرار کمک کند.

مدت زمان اجرای کامل یک خط لوله از ساخت تا استقرار برای حفظ چابکی بسیار مهم است. تست‌ها و ساخت‌های طولانی مدت می‌توانند منجر به تداخل با خط لوله های دیگر باشد. بهبود و بهینه‌سازی این فرآیند از طریق اجرای موازی تست‌ها، بهبود قابلیت مقیاس‌پذیری زیرساخت و بهینه‌سازی کد می‌تواند به کاهش زمان مورد نیاز برای تست و ساخت و بهبود جریان کار توسعه کمک کند.

در محصولاتی که از یادگیری ماشین استفاده می کنند نیز مراحل خط لوله برای رسیدگی به چرخه عمر مدل و داده ها افزایش می یابد، اما عناصر، مزایا و اهداف یکسان هستند.

۲-۱-۴ مزایای متدلوژی DevOps

این متدلوژی یک رویکرد نوآورانه در توسعه نرم‌افزار و عملیات است که مزایای بسیاری برای بهبود عملکرد سازمانی[۲۵] ارائه می دهد [۲۱]. ادغام این روش ها می‌تواند نحوه مواجهه تیم‌ها با چالش‌های پروژه و تعامل با فناوری را تغییر داده و منجر به افزایش کارایی، قابلیت اطمینان و رضایت شود [۱۲].

۱. افزایش سرعت و کارایی: با خودکارسازی فرایند انتشار نرم‌افزار از طریق CI/CD، تیم‌ها می‌توانند فرکانس و سرعت

[۲۳] Role-Based Access Control (RBAC)
[۲۴] Secrets
[۲۵] Organization performance

انتشارها را افزایش داده که منجر افزایش سرعت پاسخ دهی به مشتری شده و مزیت رقابتی ایجاد می کند [۳۵].

۲. ایجاد محیط‌های عملیاتی پایدارتر: تضمین قابلیت اطمینان به‌روزرسانی‌های برنامه و تغییرات زیرساخت یکی از مزایای مهم این متدلوژی می باشد. از طریق خط لوله CI/CD، هر تغییری برای اطمینان از کارایی و ایمنی ادغام با محیط تولید آزمایش می‌شود تا از انتشار نسخه‌های معیوب جلوگیری کند. یکی از شاخص‌های اصلی پایداری، انتشارهای متناوب و مکرر است. با استفاده از این متدلوژی توسعه‌دهندگان می‌توانند خطاها را سریع‌تر شناسایی و رفع کنند. این موضوع باعث کاهش شاخص MTTR[26] می‌شود. این شاخص مدت زمان برگشت به وضعیت پایدار بعد از وقوع خطا یا اشکال را نشان می‌دهد و هرچه مقدار آن کمتر باشد، پایداری سیستم بیشتر است [۲۲]. علاوه بر انتشار پیوسته و مستمر، نرم‌افزارهای مانیتورینگ هم با پایش مداوم نرم‌افزار و سرورها و ایجاد دسترسی به اطلاعات حیاتی نرم‌افزار و محیط عملیاتی برای مهندسان، نقش مهمی در شناسایی و رفع خطاها و در نتیجه حفظ پایداری دارند.

۳. مقیاس پذیری: تسهیل‌کننده مدیریت مقیاس‌پذیر زیرساخت‌ها و فرآیندهای توسعه است. تکنیک‌هایی مانند زیرساخت به عنوان کد[۲۷] مدیریت محیط‌های توسعه، آزمایش و تولید را به شکلی تکرارپذیر و کارآمد ساده‌سازی می‌کنند [۱۲].

۴. صرفه‌جویی در هزینه‌ها و منابع: علاوه بر مدیریت بهتر عملکرد و ارتباطات، هزینه‌ها و منابع را هم به نسبت روش‌های قدیمی کاهش می‌دهد. با استفاده از این متدلوژی و خط لوله CI/CD طول چرخه‌ها کوتاه‌تر و نتایج کمی و کیفی بهتر می‌شوند و در نتیجه هزینه‌ها نیز کاهش پیدا می‌کنند. این فرآیند حتی نیاز به منابع سخت‌افزاری و منابع انسانی را هم کاهش می‌دهد. با استفاده از معماری ماژولار، اجزا و منابع به خوبی دسته‌بندی شده و سازمان‌ها می‌توانند به راحتی از فضا و رایانش ابری برای انجام کارها استفاده کنند. چابکی در این متدلوژی اهمیت زیادی دارد لذا فناوری ابری نیز این چابکی را به تیم‌ها ارائه و سرعت و هماهنگی بین تیم‌ها را افزایش می‌دهد. با کمک این فناوری، حتی اگر در فرایند توسعه و عملیات نیاز به منابع جدید و بیشتر بود، با ثبت یک درخواست ساده در عرض چند دقیقه منابع جدید در اختیار سازمان قرار می‌گیرد. از مزایای دیگر استفاده از رایانش ابری می توان به حداقل شدن هزینه‌های شروع و عملیاتی پروژه، بهبود امنیت، افزایش مشارکت و بهبود دسترسی و کاربری داده‌ها اشاره کرد.

۵. تجزیه ایزوله گرایی: در بسیاری از سازمان‌ها، به دلایل امنیتی و مدیریتی، اطلاعات در تیم‌ها به طور جداگانه نگهداری می‌شوند و این باعث ایجاد سیلوهای سازمانی شده که مانع از گردش منظم داده و اطلاعات در سازمان می‌شود. با این حال، با بهره‌گیری از این متدلوژی و وجود همکاری فعال در تیم‌ها، ارتباطات بهبود می‌یابد. این امر باعث می‌شود که

[26] Mean Time To Recover
[۲۷] infrastructure as a code

اطلاعات به طور موثرتر جریان یابد، کارایی تیم‌ها افزایش یابد و در نتیجه، کارایی کلی سازمان بهبود پیدا کند.

۲-۲ مجازی سازی و کانتینرها

۲-۲-۱ مجازی سازی

تکنولوژی مجازی‌سازی[۲۸] به روشی اشاره دارد که در آن منابع سخت‌افزاری یک سیستم فیزیکی به چندین محیط مجازی تقسیم می‌شوند. این تکنولوژی به سازمان‌ها این امکان را می‌دهد تا منابع خود را به شیوه‌ای کارآمدتر استفاده کنند، زیرا می‌توانند چندین سیستم عامل و برنامه را روی یک سرور فیزیکی اجرا کنند. مجازی‌سازی انواع مختلفی دارد، از جمله مجازی‌سازی سرور، دسکتاپ، نرم‌افزار و شبکه، که هر کدام کاربردهای خاص خود را دارند [۱۹، ۱۳].

مجازی‌سازی سرور یکی از تکنولوژی‌های کلیدی در مدیریت و بهره‌برداری از داده‌ها و منابع سخت‌افزاری در مراکز داده است. این فناوری امکان تقسیم یک سرور فیزیکی[۲۹] به چندین سرور مجازی را می‌دهد، به طوری که هر سرور مجازی می‌تواند به صورت مستقل عمل کرده و سیستم عامل و برنامه‌های کاربردی خود را اجرا کند. مجازی‌سازی سرور معمولاً شامل سه جزء اصلی است [۱۹]:

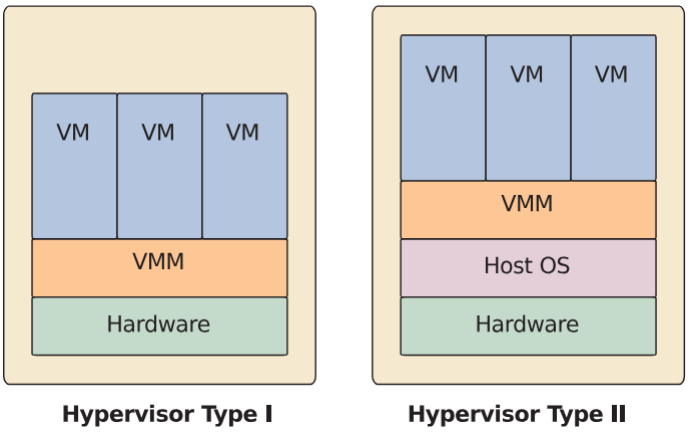
- هایپروایزر[۳۰]
- ماشین مجازی
- سیستم مدیریت مرکزی

هایپروایزر، که گاهی اوقات به عنوان مدیر ماشین مجازی[۳۱] شناخته می‌شود، نقش محوری در مجازی‌سازی سرور دارد. این نرم‌افزار بر روی سخت‌افزار سرور نصب می‌شود و وظیفه آن تقسیم منابع سرور فیزیکی، مانند CPU، حافظه، فضای دیسک و شبکه به چندین ماشین مجازی است. هایپروایزرها به دو دسته تقسیم می شوند.

هایپروایزر نوع ۱[۳۲] مستقیماً بر روی سخت‌افزار نصب می‌شود و به طور مستقل از سیستم عامل فیزیکی عمل می‌کند. می توان از هایپروایزرهای نوع ۱ معروف به VMware ESXi، Microsoft Hyper-V و KVM اشاره کرد که برای بهینه‌سازی عملکرد و امنیت طراحی شده‌اند.

---

[۲۸] Virtualization
[۲۹] Bare-metal
[۳۰] Hypervisor
[۳۱] Virtual Machine Manager (VMM)
[۳۲] Bare-metal



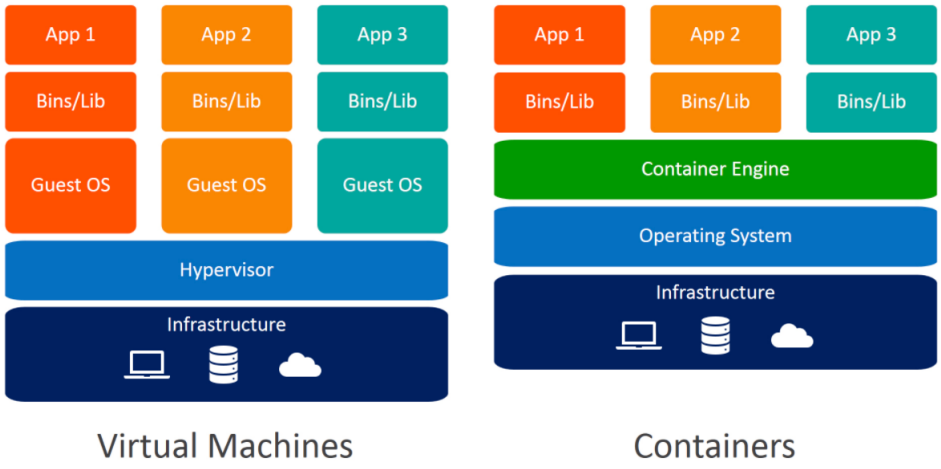
شکل ۲-۲: انواع هایپروایزر [۱۹]

هایپروایزر نوع ۲[۳۳] روی یک سیستم عامل میزبان نصب می‌شود و به عنوان یک برنامه درون سیستم عامل عمل می‌کند. از هایپروایزر نوع ۲ نیز می توان به VMware Workstation و Oracle VirtualBox اشاره کرد. این هایپروایزرها اغلب برای تست و توسعه مورد استفاده قرار می‌گیرند. در شکل ۲-۲ ساختار آن را مشاهده می کنید.

هایپروایزرها به لحاظ انعطاف پذیری و امکان پیکربندی متنوع، قابلیت‌های قدرتمندی را برای مدیریت سرورهای مجازی فراهم می‌کنند. آنها می‌توانند به طور خودکار منابع را بین ماشین‌های مجازی تخصیص دهند و امکاناتی مانند تکثیر[۳۴] و بازیابی فاجعه[۳۵] را ارائه دهند.

ماشین مجازی[۳۶] واحدی از منابع مجازی است که شبیه‌سازی یک سرور فیزیکی را انجام می‌دهد. هر VM می‌تواند سیستم عامل خود را داشته باشد و مستقل از دیگر VMها عمل کند. این امر به کاربران اجازه می‌دهد که برنامه‌های متعدد را بدون تداخل با یکدیگر اجرا کنند. VMها از منابع سخت‌افزاری تخصیص داده شده توسط هایپروایزر استفاده می‌کنند و می‌توانند به راحتی از یک سرور فیزیکی به دیگری با استفاده از تکنیک هایی نظیر Snapshot منتقل شوند. استفاده از تکنولوژی مجازی‌سازی نقش بسیار مهمی در فرآیندهای DevOps دارد. با امکان ایجاد و حذف سریع ماشین‌های مجازی، مجازی‌سازی به تیم‌های توسعه این امکان را می‌دهد که به سرعت محیط‌های نرم‌افزاری مورد نیاز خود را راه‌اندازی و پس از اتمام کار، آن‌ها را به راحتی حذف کنند، که این امر منجر به صرفه‌جویی در هزینه‌ها و منابع می‌شود. علاوه بر این، مجازی‌سازی ریسک‌های مرتبط با استقرار نهایی در محیط تولید را کاهش داده و با ایجاد محیط‌های شبیه‌سازی شده برای آزمایش‌های پیش از استقرار، اطمینان حاصل می‌کند که نرم‌افزار قبل از راه‌اندازی به درستی کار می‌کند.

[۳۳] Hosted
[۳۴] Replication
[۳۵] Disaster Recovery
[۳۶] Virtual Machine (VM)



شکل ۲-۳: تفاوت ماشین مجازی و کانتینر

۲-۲-۲ کانتینرها

کانتینرها محیط‌هایی هستند که به برنامه‌های نرم‌افزاری امکان می‌دهند تا با تمام وابستگی‌های خود در یک بسته واحد جمع‌آوری شوند. آن ها همانند برنامه‌های نرم افزاری سنتی که به شما اجازه می‌دهند مستقل از نرم افزارهای دیگر و خود سیستم عامل کار کنید، نصب نمی‌شوند. مهمترین دغدغه کانتینرها این است که چگونه محیطی فراهم کنند تا نرم افزارهایی که در یک محیط پردازشی اجرا می‌شوند با انتقال به محیط دیگر، بدون ایراد و مشکل اجرا شوند. این تکنولوژی از معماری میزبان بهره می‌برد تا از منابع سخت‌افزاری مشترک استفاده کند، اما اجرای برنامه‌ها را در یک محیط ایزوله و مستقل فراهم می‌کند. تمام اجزای ضروری مورد نیاز یک برنامه به صورت یک Image بسته‌بندی می‌شود. Image مربوطه در یک محیط ایزوله اجرا شده و فضای حافظه، CPU و فضای ذخیره سازی خود را با سیستم عامل به اشتراک نخواهد گذاشت. این عمل موجب می‌شود که فرآیندهای موجود در کانتینر، قادر به مشاهده سایر فرآیندها در خارج از آن نباشند.

کانتینرها و ماشین‌های مجازی هر دو ابزارهایی برای ایزوله‌سازی منابع نرم‌افزاری هستند، اما تفاوت‌های اساسی در معماری و کاربرد آن‌ها وجود دارد که در شکل ۲-۳ نشان داده شده است. ماشین‌های مجازی با ایجاد یک لایه انتزاعی کامل بر روی سخت‌افزار فیزیکی کار می‌کنند که به آن‌ها اجازه می‌دهد سیستم‌عامل‌های مستقل را بر روی هر VM اجرا کنند. این امر به هر ماشین مجازی امکان می‌دهد منابع سخت‌افزاری را به صورت مجزا استفاده کند، اما باعث می‌شود VMها نسبت به کانتینرها سنگین‌تر و کم استفاده تر باشند. در مقابل، کانتینرها به جای سیستم‌عامل‌های کامل، تنها برنامه‌ها و وابستگی‌های خود را ایزوله می‌کنند و همگی بر روی هسته سیستم‌عامل میزبان اشتراکی اجرا می‌شوند، که این امر باعث سبک‌تر، سریع‌تر و مقیاس‌پذیرتر شدن کانتینر ها نسبت ها به ماشین های مجازی باشند. از این رو، کانتینرها برای محیط‌هایی که نیازمند راه‌اندازی

سریع و مدیریت منابع مانند میکروسرویس‌ها و برنامه‌های کاربردی مبتنی بر Cloud هستند ایده‌آل می‌باشند [۳۱]. در کنار مزایای فراوان کانتینر ها، برخلاف ماشین های مجازی در امنیت و ایزولاسیون داده‌ها محدودیت‌هایی دارند و ممکن است نیازمند ابزارهای پیچیده‌تر برای مدیریت لاگ‌ها و نظارت باشند، که می‌تواند پیاده‌سازی و نگهداری آنها را چالش برانگیز سازد.

تکنولوژی کانتینر ریشه در مفهوم چارچوب‌های Unix مانند chroot دارد که در دهه ۱۹۷۰ معرفی شد. اما، پیشرفت‌های اصلی در این زمینه با ظهور Docker در سال ۲۰۱۳ آغاز شد. داکر یک پلتفرم متن باز است که استانداردسازی ایجاد، اجرا و مدیریت کانتینرها را فراهم کرد و به سرعت به یکی از مهم‌ترین ابزارها در این حوزه تبدیل شد.

اجزای کلیدی مورد استفاده در پیاده‌سازی کانتینرها شامل موارد زیر است [۳]:

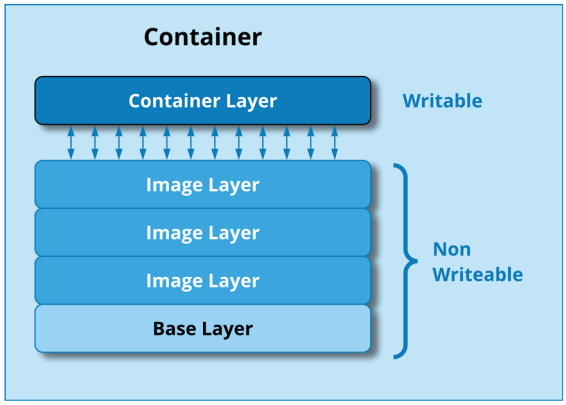
- موتورهای کانتینر[۳۷]
- هماهنگ سازی کانتینر[۳۸]

موتورهای کانتینری مانند Docker Engine و Containerd ابزارهایی هستند که کانتینرها را ایجاد، اجرا و مدیریت می‌کنند. این موتورها از فناوری‌های موجود در هسته لینوکس مانند Namespaces و Control groups (cgroups) برای ایزوله‌سازی کانتینرها استفاده می‌کنند و به آن‌ها امکان می‌دهند که فرایندها و منابع سیستمی را به صورت مستقل از یکدیگر مدیریت کنند. Namespaces بخشی از هسته لینوکس که امکان جداسازی عناصری مثل شبکه، فرایندها و فضای فایل سیستم را فراهم می‌کند. هر کانتینر در یک namespace جداگانه اجرا می‌شود که استقلال آن را نسبت به دیگر برنامه ها تضمین می‌کند. cgroups نیز به مدیریت استفاده از منابع سخت‌افزاری مانند CPU و حافظه توسط فرایندها کمک می‌کند. این فناوری امکان اختصاص دقیق منابع به کانتینرها را می‌دهد و از مصرف بیش از حد منابع توسط یک کانتینر جلوگیری می‌کند.

برای مدیریت و مقیاس‌بندی کانتینرها در محیط‌های تولید، ابزارهای هماهنگ سازی مانند Kubernetes و Docker Swarm کاربرد دارند. این ابزارها به توسعه‌دهندگان این امکان را می‌دهند که خوشه[۳۹] های بزرگ کانتینری را مدیریت کنند و برنامه‌ها را با انعطاف‌پذیری و دقت بالا مقیاس بندی نمایند. راجع به این موضوع در قسمت بعدی بیشتر صحبت خواهد شد.

Docker Image به عنوان اساسی‌ترین بخش در اکوسیستم داکر نقش کلیدی در پیاده‌سازی و توزیع برنامه‌های نرم‌افزاری دارد. تصاویر داکر از یک معماری لایه‌ای بهره می‌برند. معماری لایه این امکان را فراهم می‌کند که تغییرات نسبت به یک تصویر پایه به صورت دیفرانسیلی اعمال شود. هر لایه در تصویر داکر، تغییراتی را نسبت به لایه قبلی اضافه می کند. این رویکرد باعث می‌شود که بازسازی و به‌روزرسانی تصاویر کانتینری فقط بر روی لایه‌هایی که تغییر کرده‌اند انجام شود، که به

[۳۷] Container Engine
[۳۸] Container Orchestration
[۳۹] Cluster



شکل ۲-۴: معماری لایه ای تصویر داکر

نوبه خود باعث کاهش حجم داده‌های مورد نیاز برای ذخیره‌سازی و انتقال می‌شود. زمانی که Dockerfile نوشته می‌شود، هر دستور (مانند RUN، COPY و FROM) یک لایه جدید در تصویر داکر ایجاد می‌کند. این لایه‌ها به ترتیبی که در داکرفایل آمده‌اند، روی هم اضافه می‌شوند. داکر از یک فایل‌سیستم Union استفاده می‌کند که به آن این اجازه را می‌دهد تا لایه‌های مختلف را به گونه‌ای ترکیب کند که به نظر یک فایل‌سیستم یکپارچه است [۴]. ساختار لایه ای تصویر داکر را در شکل ۲-۴ مشاهده می کنید.

۲-۲-۳ هماهنگ سازی کانتینرها (کوبرنتیز)

در دنیای توسعه نرم‌افزار، استفاده از معماری‌های مبتنی بر میکروسرویس‌ها و کانتینرها افزایش یافته است که هر دو نیازمند مدیریت دقیق و خودکار سرویس‌ها در محیط‌های تولید هستند. در محیط‌های پویا و با مقیاس بزرگ که دستگاه‌ها و خدمات به طور مکرر تغییر می‌کنند، تقریباً غیرممکن است که با نیروی کار دستی، سرویسی با دسترسی بالا ارائه داد. در چنین شرایطی، ابزارهای هماهنگ سازی نقش حیاتی ایفا می‌کنند. آنها به خودکارسازی مدیریت کانتینرها، مدیریت شبکه و نظارت بر سلامت سیستم کمک می‌کنند. بدون هماهنگ سازی تیم‌های توسعه و عملیات با چالش‌های عدیده‌ای از جمله کنترل ناموفق بر پیکربندی‌ها، مشکلات مربوط به برقراری ارتباط بین سرویس‌ها، دشواری‌های مربوط به مقیاس‌پذیری و برقراری تعادل بار مواجه می‌شوند. در میان ابزارهای هماهنگ سازی Kubernetes به عنوان یکی از پیشروان بازار شناخته می‌شود که امکان مدیریت خودکار مجموعه‌های بزرگی از کانتینرها را فراهم می‌آورد. کوبرنتیز یک پلتفرم هماهنگ‌سازی کانتینر است که فرایند زمان‌بندی[۴۰]، خودکارسازی استقرار، مدیریت و مقیاس‌گذاری اپلیکیشن‌های کانتینری را تسهیل می‌کند.

[۴۰] Scheduling

معماری کوبرنتیز

یک سیستم کوبرنتیز با تمام اجزای آن را یک خوشه[۴۱] می‌گویند. هر خوشه شامل یک یا چند گره[۴۲] است که می توانند فیزیکی[۴۳] یا مجازی باشند. این گره ها به دو دسته اصلی یا همان سطح کنترل[۴۴] و کارگر[۴۵] تقسیم می شوند. گره اصلی به عنوان مغز متفکر کوبرنتیز عمل می‌کند و وظایف مدیریتی خوشه را بر عهده دارد. این گره شامل مولفه‌های اصلی زیر است:

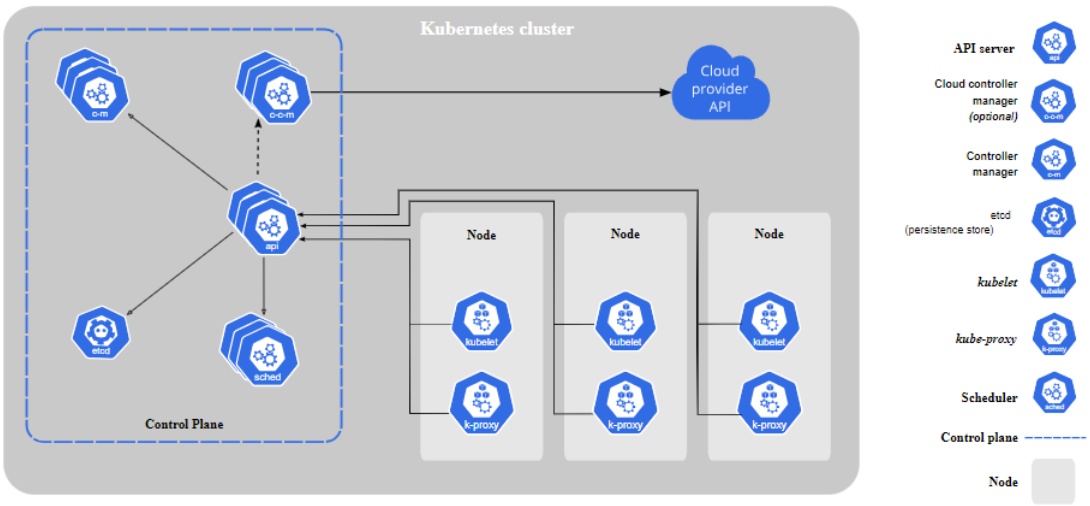
- API Server: نقطه اصلی دریافت فرمان‌های کوبرنتیز به صورت REST[۴۶] است و آن را پردازش می‌کند. این سرور مسئول اعتبارسنجی درخواست‌ها و اجرای آن‌ها بر روی خوشه است. همچنین، این مولفه به عنوان بخشی از مولفه‌های دیگر گره اصلی عمل می‌کند تا اطمینان حاصل شود که دستورات به درستی اجرا می‌شوند.
- Scheduler: مولفه‌ای است که تصمیم می‌گیرد کدام پادها بر روی کدام گره های کاری قرار گیرند. این فرایند بر اساس منابع موجود و الزامات مشخص شده برای پادها صورت می‌گیرد. علاوه براین، به طور مداوم وضعیت خوشه را رصد می‌کند تا بهترین تصمیم‌ها را برای مکان‌یابی پادها بگیرد.
- Controller Managers: مجموعه‌ای از فرآیندهایی است که حلقه‌های نظارتی را اجرا می‌کنند. این کنترل کننده ها وضعیت خوشه را با حالت مطلوب مطابقت می‌دهند. به عنوان مثال، اگر یک پاد از کار افتاده باشد، یک کنترل کننده وظیفه دارد تا یک پاد جدید را برای جایگزینی ایجاد کند.
- etcd: یک پایگاه داده توزیع شده است که تمام داده‌های مهم از جمله وضعیت خوشه در هر لحظه، پیکربندی خوشه، اطلاعات مربوط به هر گره و کانتینرهای درون آن را در خود ذخیره می‌کند.

گره های کاری نیز پادهای اپلیکیشن‌های کاربر را بر عهده دارند. این نودها شامل مولفه‌های زیر هستند:

- Kubelet: این مولفه وظفه مدیریت سلامت پادها را بر عهده دارد. علاوه بر این اطمینان حاصل می‌کند که کانتینرها در پادها بر اساس تنظیمات مشخص شده اجرا شوند و با API Server ارتباط برقرار کند تا وضعیت را به روز رسانی کند.
- Kube-proxy: وظیفه مدیریت ترافیک شبکه درون خوشه را برعهده دارند. این مولفه ارتباطات شبکه بین کانتینرها را تسهیل می کند و از قوانین IPTables برای مسیریابی ترافیک استفاده می‌کند.

---

[۴۱] Cluster
[۴۲] Node
[۴۳] Bare-metal
[۴۴] Control plane
[۴۵] Worker
[۴۶] Representational State Transfer



شکل ۲-۵: مولفه های یک خوشه کوبرنتیز

فصل ۳

MLOps

۳-۱ مقدمه

در حالی که مدل‌های یادگیری ماشین به طور گسترده توسعه یافته‌اند، انتقال آن‌ها از مفهوم آزمایشی به محیط تولید اغلب با شکست مواجه می شود. این فاصله بیشتر به خاطر این است که تاکنون توجه اصلی روی ساخت مدل‌ها بوده است، نه روی تولید محصولات یادگیری ماشین که قابلیت استفاده در محیط تولید را دارند. علاوه بر آن، مدیریت بخش‌ها و زیرساخت‌های پیچیده‌ای که برای یک استقرار موثر ضروری هستند نیز در این امر مغفول مانده اند. برای رفع این مسئله، مفهوم عملیات یادگیری ماشین یا MLOps معرفی شده است. MLOps بر روی خودکارسازی و عملیاتی کردن فرآیندهای یادگیری ماشین تمرکز دارد تا انتقال پروژه‌های یادگیری ماشین از مفهوم به تولید را تسهیل کند. این رویکرد شامل دیدگاه جامعی از طراحی سیستم، هماهنگی اجزا، تعریف نقش‌ها و مسئولیت‌ها می باشد. هدف کاهش خطا به منظور افزایش قابلیت اطمینان و کارایی سیستم‌های یادگیری ماشین در کاربردهای واقعی می باشد. این فصل به بررسی تعریف، اصول، ابزار و معماری جامعی از یک پلتفرم MLOps پرداخته و در نهایت، محصولات و رقبا این حوزه را بررسی می کنیم.

۳-۲ تعریف مفاهیم اولیه

MLOps یا عملیات یادگیری ماشین به مجموعه‌ای از فرایندها، ابزارها و شیوه‌ها جهت مدیریت چرخه توسعه مدل‌های یادگیری ماشین در یک محیط عملیاتی اشاره دارد. همچنین این چرخه شامل همکاری بین دانشمندان داده و مهندسان DevOps است به گونه ای که این اطمینان حاصل شود که مدل‌ها به طور مؤثر توسعه، استقرار، پایش و به‌روزرسانی می‌شوند. هدف MLOps افزایش سرعت، قابلیت اطمینان و مقیاس پذیری مدل‌های یادگیری ماشین و فرایند توسعه این مدل ها در تولید است؛

درحالی‌که خطرات ناشی از ریسک عدم موفقیت را نیز کاهش می‌دهد. همچنین به‌کارگیری MLOps فرایند مدیریت را ساده‌تر کرده، کیفیت را افزایش می‌دهد و استقرار مدل‌های یادگیری عمیق و یادگیری ماشین در محیط‌های تولید با مقیاس بزرگ را خودکار می‌کند. لذا می توان گفت یکی از اهداف MLOps، بهبود خودکارسازی و ارتقای کیفیت مدل‌های تولید و درعین‌حال توجه به الزامات تجاری و نظارتی است.

استقرار مدل‌های یادگیری ماشین روی محیط عملیاتی در MLOps اهمیت زیادی دارد، زیرا به سازمان‌ها کمک می‌کند تا مطمئن شوند که مدل‌هایشان در طول زمان دقیق، قابل‌اعتماد و کارآمد هستند. به‌طورکلی، MLOps با خودکار کردن بسیاری از مراحل مربوط به استقرار و مدیریت مدل‌های یادگیری ماشین، به دانشمندان و مهندسان داده اجازه می‌دهد تا با همکاری یکدیگر به ارائه سریع‌تر و کارآمدتر مدل‌های یادگیری ماشین دست یابند.

۳-۲-۱ اصول

برای تسهیل در رسیدن به اهداف فوق، تیم‌های MLOps از اصول زیر استفاده می کنند:

۱. خط لوله خودکار CI/CD و هماهنگ سازی جریان کار[۱]: خودکارسازی CI/CD شامل مراحل ساخت، آزمایش، تحویل و استقرار است که به توسعه‌دهندگان نسبت به موفقیت یا شکست مراحل مختلف بازخورد سریعی را ارائه داده و بهره‌وری کلی را افزایش می‌دهد [۳۶]. در همین حال، هماهنگ سازی جریان کاری وظایف یک خط لوله یادگیری ماشین را با استفاده از گراف‌های بدون‌حلقه‌ی جهت‌دار[۲] هماهنگ می‌کند، که ترتیب اجرای وظایف را با توجه به روابط و وابستگی‌ها تعیین می‌کند. ترکیب این دو رویکرد می تواند به بهبود عملکرد و کارایی تیم‌های توسعه و داده‌کاوی کمک کند [۳۴، ۲۷].

۲. کنترل نسخه مدل‌های یادگیری ماشین، مجموعه‌داده‌ها و کد منبع: با استفاده از نسخه‌بندی مدل، داده و کدمنبع، می‌توان هر تغییر و اصلاحی را در طول زمان دنبال کرد، که این امر به توسعه‌دهندگان و محققان اجازه می‌دهد تا به راحتی به نسخه‌های قبلی بازگردند و نتایج را بازبینی کنند. این قابلیت برای حفظ یکپارچگی و شفافیت در پروژه‌های نرم‌افزاری و علمی بسیار حیاتی است [۳۶].

۳. نظارت و آموزش مدوام مدل یادگیری ماشین: آموزش مداوم[۳] در یادگیری ماشین به معنای آموزش دوره‌ای مدل‌های یادگیری ماشین بر اساس داده‌های جدید است. این فرآیند همیشه شامل یک مرحله ارزیابی برای سنجش تغییرات

[۱] Workflow
[۲] Directed Acyclic Graph (DAG)
[۳] Continuous Training (CT)

کیفیت مدل است [۲۶]. نظارت مداوم به معنای ارزیابی دوره‌ای داده‌ها، مدل‌ها (مانند دقت پیش‌بینی)، کد منبع و منابع زیرساختی است تا خطاها یا تغییرات احتمالی که بر کیفیت محصول تاثیر می‌گذارند، شناسایی شوند. این فرآیند به توسعه‌دهندگان امکان می‌دهد تا به سرعت مشکلات را شناسایی و برطرف کنند و از افت عملکرد مدل جلوگیری کنند. یکی از دلایل لزوم آموزش مداوم، رانش داده یا مدل[۴] است، که به تغییرات تدریجی در داده‌ها یا عملکرد مدل در طول زمان اشاره دارد و می‌تواند باعث کاهش دقت پیش بینی‌ها شود [۱۶]. این اصل در MLOps برای اطمینان از عملکرد بهینه مدل‌ها و واکنش سریع به تغییرات محیطی و داده‌ها ضروری است. این فرآیند بهره‌وری را افزایش می‌دهد و کیفیت کلی سیستم‌های یادگیری ماشین را بهبود می‌بخشد. در نهایت، ترکیب آموزش و نظارت مداوم به توسعه‌دهندگان کمک می‌کند تا مدل‌ها را به‌روز نگه داشته و از تاثیرات منفی رانش داده یا مدل جلوگیری کنند [۱۸].

۴. ثبت فراداده[۵] یادگیری ماشین: ثبت فراداده برای هر مرحله در جریان کار یادگیری ماشین شامل ثبت جزئیات هر دوره آموزش مدل، مانند تاریخ و زمان آموزش، مدت زمان، پارامترهای استفاده شده و معیارهای عملکرد مدل می باشد [۲۷]. علاوه بر این، جزئیات مدل که شامل داده‌ها و کدهای استفاده شده است، باید ثبت شود تا قابلیت پیگیری کامل آزمایشات فراهم گردد. این امر به توسعه‌دهندگان کمک می‌کند تا تغییرات و نتایج را به دقت مستند کرده و در صورت نیاز به نسخه‌های قبلی بازگردند [۳۰].

۵. حلقه های بازخورد[۶]: حلقه‌های بازخورد به توسعه‌دهندگان اجازه می‌دهند تا به‌طور مداوم مدل‌ها را بهبود بخشند، مشکلات را شناسایی و رفع کنند و از افت کیفیت جلوگیری کنند. این رویکرد به تضمین کیفیت و کارایی مدل‌های یادگیری ماشین کمک می‌کند و فرآیند توسعه را به یک چرخه تکراری و قابل بهبود تبدیل می‌کند که به سرعت به تغییرات و نیازهای جدید پاسخ می‌دهد [۳۰]. به عنوان مثال، یک حلقه بازخورد از مرحله مهندسی مدل آزمایشی به مرحله قبلی مهندسی ویژگی می‌تواند بسیار مفید باشد.

می توان اضافه کرد که یکی از اصول مهم که کمتر جنبه فنی دارد و در روح فرهنگی DevOps نیز جایگاه ویژه‌ای دارد، اصل همکاری[۷] است. این اصل بر امکان همکاری مشترک افراد بر روی داده‌ها، مدل‌ها و کدها تاکید دارد. علاوه بر جنبه‌های فنی، اصل همکاری به ایجاد فرهنگ کاری مشارکتی توجه دارد که هدف آن کاهش ایزوله سازی های حوزه‌ای بین نقش‌های مختلف است. چنین رویکردی باعث می‌شود تا افراد با تخصص‌های گوناگون به طور هم‌افزا با یکدیگر کار کنند، دانش خود را به اشتراک بگذارند و از هم بیاموزند.

[۴] Data or Model Drift
[۵] Metadata
[۶] feedback loops
[۷] Collaboration

۳-۲-۲ اجزاء

پس از شناسایی اصولی که در قسمت قبل صحبت کردیم، اکنون اجزای دقیق یک معماری MLOps را توضیح داده و ارتباط هرکدام با اصول گفته شده را بیان می کنیم. ارجاعات داخل پرانتز به اصولی اشاره دارد که اجزای فنی در حال پیاده سازی هستند. ؟؟؟

سازآرایی جریان کاری

سازآرایی جریان‌کاری[۸] به عنوان یکی از اجزای حیاتی در مدیریت و خودکارسازی جریان‌های کاری پیچیده در حوزه‌های مختلف از جمله یادگیری ماشین و مهندسی داده، نقش مهمی ایفا می‌کنند. این سیستم‌ها مانند شکل ۳-۱ از گراف‌های بدون حلقه جهت دار برای نمایش ترتیب اجرای وظایف استفاده می کنند. هر مرحله از این جریان‌کاری ممکن است شامل استخراج داده، آموزش مدل یا استنتاج باشد. این سیستم‌ها نه تنها ترتیب اجرای وظایف را مدیریت می‌کنند، بلکه وابستگی‌های متقابل بین وظایف را نیز مورد توجه قرار می‌دهند. هم چنین این ابزارها به کاربران امکان می‌دهند تا جریان های کاری را به صورت خودکار و مقیاس پذیر اجرا کنند. این امر به ویژه در محیط‌های بزرگ با داده های کلان اهمیت دارد [۲۷].

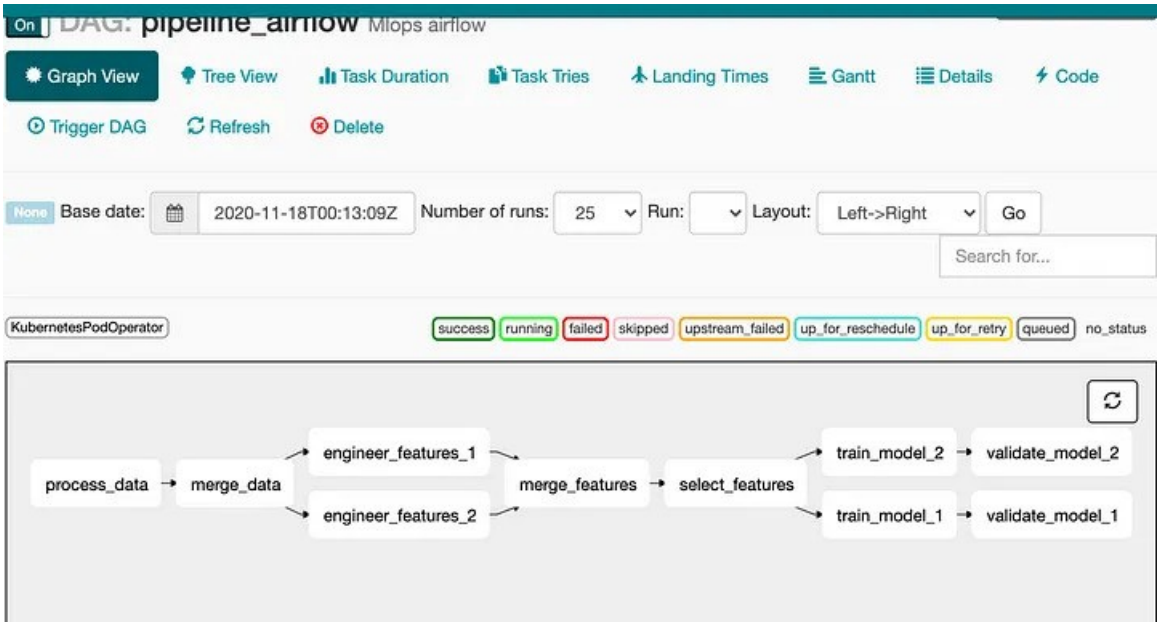
ابزارهای متن باز معروف در زمینه یادگیری ماشین Apache Airflow[۱] و Kubeflow pipeline می باشند. از Apache Airlfow بیشتر برای استخراج، تبدیل و بارگذاری[۹]داده های بزرگ استفاده می کنند. Kubeflow Pipelines نیز بخشی از پلتفرم Kubeflow[۱۰] است که برای اجرای جریان‌های کاری یادگیری ماشین بر روی کوبرنتیز طراحی شده است [۳۳]. از این ابزار به طور خاص برای توسعه و استقرار مدل‌های یادگیری ماشین در محیط‌های ابری مناسب است که در فصل های بعدی با آن بیشتر آشنا خواهیم شد.

انبار ویژگی

انباره ویژگی[۱۰] یک سیستم مدیریت داده است که به منظور ذخیره‌سازی، مدیریت و اشتراک‌گذاری ویژگی‌های مورد استفاده در مدل‌های یادگیری ماشین طراحی شده است. این سیستم (شکل ۳-۲) دارای دو بخش اصلی است: پایگاه داده آنلاین و پایگاه داده آفلاین. هر یک از این پایگاه‌های داده نقش خاصی در فرآیند مدیریت و استفاده از ویژگی‌ها ایفا می‌کنند.

پایگاه داده آفلاین برای ذخیره سازی و مدیریت ویژگی‌هایی استفاده می‌شود که در فرآیندهای آزمایش و تحلیل به کار می‌روند. این پایگاه داده معمولاً با تاخیر نسبتا بیشتری نسبت به پایگاه داده آنلاین استفاده می شود و برای مواردی مناسب است

[۸] Workflow Orchestration
[۹] Extract, Transform, Load (ETL)
[۱۰] Feature Store

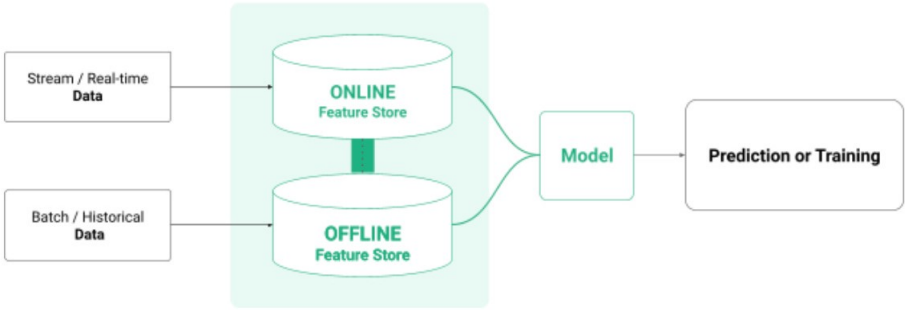


شکل ۳-۱: خط لوله در Apache Airlfow

که نیاز به پردازش حجم زیادی از داده‌ها در مدت زمان طولانی تر دارند. ویژگی هایی که در این پایگاه داده ذخیره می‌شوند، اغلب در فرآیندهای آموزش مدل‌های یادگیری ماشین مورد استفاده قرار می‌گیرند.

پایگاه داده آنلاین برای ارائه ویژگی‌ها به صورت بلادرنگ استفاده می‌شود و تأخیر کمی دارد. این پایگاه داده‌ها برای سیستم‌هایی مناسب هستند که نیاز به پاسخگویی سریع دارند. زمانی که یک مدل یادگیری ماشین نیاز به استفاده از ویژگی‌ها برای انجام پیش‌بینی‌های فوری دارد، داده‌ها از این پایگاه داده آنلاین بازیابی می‌شوند. این نوع پایگاه داده‌ها باید توانایی پشتیبانی از حجم بالای درخواست‌ها را داشته باشند تا بتوانند عملکرد مطلوبی را در شرایط عملیاتی فراهم کنند. ویژگی‌هایی که در این پایگاه داده ذخیره می‌شوند، اغلب در فرآیندهای استنتاج مدل‌های یادگیری ماشین مورد استفاده قرار می‌گیرند.

با استفاده از انباره ویژگی توسعه‌دهندگان می‌توانند ویژگی‌های از پیش پردازش شده را به صورت متمرکز ذخیره کرده و به راحتی در پروژه‌های مختلف به اشتراک بگذارند، که این امر به تسریع فرآیند توسعه مدل‌ها و بهبود دقت پیش بینی ها کمک می‌کند. این سیستم‌ها معمولاً بر روی زیرساخت‌های ابری اجرا می‌شوند تا مقیاس‌پذیری بالا و کارایی مورد نیاز برای پردازش داده های کلان را فراهم کنند [۱۸]. از ابزار معروف متن باز برای می توان به Feast[۵] اشاره نمود.



شکل ۳-۲: انباره ویژگی

بانک مدل

بانک مدل[۱۱] یکی از ابزارهای بسیار مهم در مدیریت مدل‌های یادگیری ماشین است که به تیم‌ها کمک می‌کنند تا مدل‌های خود را به صورت سازماندهی شده ذخیره، مدیریت و ردیابی کنند. هم چنین اطلاعات مربوط به هر مدل را از جمله نسخه، تاریخ آخرین آموزش، معیارهای ارزیابی و مستندات مربوطه را نگهداری می‌کند. این امر به تیم‌ها کمک می‌کند تا با استفاده از نسخه‌های مختلف مدل‌ها، آزمایش‌های مختلفی انجام دهند و بهترین مدل را انتخاب کنند. هم چنین به هنگام بروز مشکل در مدل های جدید، می توان از مدل های قابل قبول قبلی برای محیط عملیاتی استفاده کرد [۲۸]. از ابزار معریف متن باز می توان به MLflow[۱۱] اشاره کرد.

انبار فراداده

انبار فراداده یادگیری ماشین[۱۲] برای پیگیری و ذخیره‌سازی اطلاعات مربوط به هر مرحله از جریان کاری یادگیری ماشین استفاده می شوند. فراداده‌ها می‌توانند شامل جزئیاتی نظیر تاریخ و زمان آموزش مدل، مدت زمان هر مرحله از آموزش، پارامترهای استفاده شده، معیارهای عملکرد مدل، و سلسله‌مراتب مدل (مثل داده‌ها و کدهای استفاده شده) باشند. یکی از کاربردهای اصلی انبار فراداده‌ها، مدیریت کارآمد پروژه‌های پیچیده یادگیری ماشین است [۳۰]. به عنوان مثال، در پروژه‌های بزرگ که شامل آزمایش‌ها و مدل‌های متعددی هستند، پیگیری دقیق و منظم فراداده‌ها می‌تواند به تیم‌ها کمک کند تا نتایج قبلی را به راحتی بازبینی کنند، مشکلات را شناسایی کنند و بهینه‌سازی‌های لازم را انجام دهند. MLflow یک ابزار معروف برای یک سیستم پیشرفته مدیریت فراداده است که همراه با بانک مدل امکان مدیریت یکپارچه مدل‌ها و فراداده‌ها را فراهم می‌کند.

[۱۱] Model Registry
[۱۲] ML Metadata Store

استقرار مدل

استقرارکردن مدل[۱۳] به فرآیندی اشاره دارد که در آن مدل‌های یادگیری ماشین آماده برای استفاده، به کار گرفته می‌شوند تا به صورت عملیاتی به پیش‌بینی‌ها و استنتاج‌ها بپردازند. این فرآیند برای تبدیل مدل‌های آموزشی به ابزارهای قابل استفاده در محیط‌های تولیدی ضروری است و می‌تواند به صورت آنلاین برای پیش‌بینی‌های بلادرنگ یا به صورت دسته‌ای[۱۴] برای پردازش حجم بالای داده‌ها پیاده‌سازی شود. در محیط‌های عملیاتی، فرآیند استقرار مدل به سه شکل اصلی بلادرنگ، دسته ای و بدون سرور پیاده‌سازی می‌شود [۲۸].

در استنتاج بلادرنگ[۱۵]، مدل‌های یادگیری ماشین به گونه‌ای پیاده‌سازی می‌شوند که بتوانند به سرعت و با کمترین تأخیر ممکن پیش‌بینی‌ها را انجام دهند. این نوع استنتاج برای کاربردهایی نظیر سیستم‌های توصیه‌گر، تحلیل داده‌های حسگرها و برنامه‌های کاربردی که نیاز به پاسخ‌های سریع دارند، مناسب است. به عنوان مثال، در سیستم‌های پیشنهاددهی محتوا مانند نتفلیکس یا آمازون، مدل‌ها باید به صورت بلادرنگ تحلیل کنند و پیشنهادهای شخصی‌سازی شده را ارائه دهند. تکنولوژی‌های مانند RESTful APIs و gRPC معمولاً برای پیاده‌سازی این نوع سرویس‌دهی استفاده می‌شوند.

استنتاج دسته‌ای[۱۶] برای پردازش حجم وسیعی از داده‌ها به کار می‌رود که معمولاً به صورت زمان‌بندی شده انجام می‌شود. این روش برای تحلیل داده‌های کلان و پردازش‌های بزرگ مناسب است. به عنوان مثال، در تجزیه و تحلیل رفتار مشتریان یک فروشگاه آنلاین، داده‌های خریدهای گذشته می‌تواند به صورت دسته‌ای پردازش شود تا الگوهای مختلف شناسایی شود. ابزارهایی مانند Apache Spark[۲] و HadoopHadoop MapReduce معمولاً برای پیاده‌سازی استنتاج دسته‌ای استفاده می‌شوند.

در استنتاج بدون سرور[۱۷]، مدل‌ها به صورت پویا و بر اساس تقاضا اجرا می‌شوند که هزینه و مقیاس‌پذیری را بهینه می‌کند. این نوع استنتاج زمانی مورد استفاده قرار می‌گیرد که نیاز به سرویس‌دهی مقیاس‌پذیر و مقرون‌به‌صرفه باشد. در استنتاج بدون سرور، مدل‌ها فقط زمانی که لازم است اجرا می‌شوند و بنابراین منابع بهینه‌سازی می‌شوند. سرویس‌های ابری مانند AWS Lambda و Google Cloud Functions معمولاً برای پیاده‌سازی این نوع استنتاج استفاده می‌شوند. از ابزارهای معروف متن باز برای استقرار مدل می توان به Knative[۹] اشاره کرد.

[۱۳] Model Serving
[۱۴] Batch
[۱۵] Real-time Inference
[۱۶] Batch Inference
[۱۷] Serverless Inference

نظارت

نظارت[۱۸] در یادگیری ماشین یکی از مولفه‌های حیاتی برای تضمین عملکرد بهینه مدل‌ها و زیرساخت‌های مرتبط است. نظارت مداوم بر مدل‌های یادگیری ماشین به دلایلی از جمله اطمینان از دقت پیش‌بینی‌ها، شناسایی ناهنجاری‌ها و بهبود مداوم عملکرد مدل‌ها ضروری است [۳۲]. ابزارهایی مانند TensorBoard، Kubeflow و MLflow نیز نقش مهمی در نظارت بر مدل‌های یادگیری ماشین ایفا می‌کنند. TensorBoard به ویژه برای مصورسازی و تحلیل مراحل مختلف آموزش مدل‌ها مفید است.

نظارت در یادگیری ماشین تنها به مدل‌ها محدود نمی‌شود؛ بلکه زیرساخت‌های مرتبط با یادگیری ماشین نیز نیاز به نظارت دارند. این نظارت شامل نظارت بر فرآیندهای CI/CD، هماهنگی سرویس‌ها، خوشه های عملیاتی کوبرنتیز و گره های محاسباتی می‌شود [۳۰]. یکی از ابزارهای رایج برای نظارت، Prometheus است که به همراه Grafana برای مصورسازی داده‌ها استفاده می‌شود. علاوه بر این پشته ELK (Elasticsearch، Logstash، Kibana) نیز یک مجموعه قدرتمند برای جستجو، تحلیل و مصورسازی لاگ‌های سیستم است که می‌تواند به شناسایی و رفع سریع مشکلات کمک کند. ابزارهای نظارتی به مهندسان اجازه می‌دهند تا هر گونه ناهنجاری در زیرساخت‌ها را به سرعت شناسایی و رفع کنند، که این امر موجب کاهش زمان از کار افتادگی سیستم و افزایش بهره‌وری می‌شود.

زیرساخت آموزش و استقرار مدل

این زیرساخت شامل منابع محاسباتی اصلی مانند واحد پردازش مرکزی، حافظه واحد پردازش گرافیکی[۱۹] است که برای پردازش داده‌ها و اجرای الگوریتم‌های پیچیده می باشد. زیرساخت ها می‌توانند به دو شکل توزیع شده[۲۰] و غیرتوزیع شده پیاده‌سازی شوند. زیرساخت‌های غیرتوزیع‌شده معمولاً شامل ماشین‌های محلی هستند که با وجود سادگی در پیاده‌سازی، محدودیت‌هایی در مقیاس‌پذیری دارند. از سوی دیگر، زیرساخت‌های توزیع‌شده که معمولاً در بستر محاسبات ابری اجرا می‌شوند، امکان توزیع بار کاری بین چندین گره محاسباتی را فراهم می‌کنند و از این طریق مقیاس‌پذیری و کارایی بالاتری ارائه می‌دهند. یکی از ابزار محبوب برای مدیریت و سازآرایی محاسبات توزیع‌شده، کوبرنتیز است که امکان مدیریت کانتینرها و توزیع بار کاری بین گره‌ها را فراهم می‌کند. هم چنین، Red Hat OpenShift نیز به عنوان یک پلتفرم دیگر شناخته می‌شود که قابلیت‌های مشابهی ارائه می‌دهد [۲۸].

برای بهینه‌سازی عملکرد مدل‌های یادگیری عمیق، استفاده از واحد پردازش گرافیکی که برای ضرب ماتریسی

[۱۸] Monitoring
[۱۹] GPU
[۲۰] Distributed

بهینه‌سازی شده‌اند، استفاده می‌شوند. در دستگاه‌های لبه[۲۱] به دلیل محدودیت‌های فضا، توان محاسباتی و مصرف انرژی، اجرای مدل‌های یادگیری عمیق پیچیده به چالش‌های خاصی مواجه است. برای غلبه بر این محدودیت‌ها و بهینه‌سازی عملکرد مدل‌ها در این دستگاه‌ها، تکنیک‌های مختلفی مورد استفاده قرار می‌گیرد. یکی از این تکنیک‌ها، استفاده از شبکه‌های عصبی کوانتیزه شده است. در کوانتیزاسیون، وزن‌ها و محاسبات شبکه عصبی از دقت کامل (به عنوان مثال، اعداد با دقت ۳۲ بیت) به اعداد با دقت پایین‌تر (مانند ۸ بیت یا حتی کمتر) کاهش می‌یابند. این کاهش دقت باعث کاهش حجم مدل و کاهش نیاز به منابع محاسباتی می‌شود. علاوه بر این، با استفاده از عملیات نقطه شناور کم‌دقت، می‌توان محاسبات را سریع‌تر و با مصرف انرژی کمتری انجام داد. تکنیک دیگر، هرس کردن[۲۲] است که شامل حذف اتصالات غیرضروری و وزن‌های کوچک در شبکه عصبی می‌شود. این فرآیند باعث کاهش تعداد پارامترهای مدل می‌شود، بدون آنکه تاثیر قابل توجهی بر دقت مدل بگذارد. هرس کردن مدل را سبک‌تر و اجرای آن را سریع‌تر می‌کند، که این امر برای دستگاه‌های لبه با منابع محدود بسیار مفید است.

مخزن کد منبع

مخزن کد منبع به عنوان یک نقطه مشترک برای نگهداری و مدیریت کدهای مربوط به مدل‌های یادگیری ماشین یک سازمان عمل می‌کند. با استفاده از سیستم‌های مدیریت نسخه مانند گیت، تیم‌ها می‌توانند به راحتی تغییرات کد را پیگیری کرده و در صورت لزوم به نسخه‌های قبلی کد بازگردند. این مخزن همچنین به خودکارسازی فرآیند CI/CD کمک می‌کند، به طوری که هرگونه تغییر در کد به طور خودکار خط لوله را فعال کرده و تغییرات تست، ارزیابی و در محیط‌های مختلف مستقر می‌شوند. می توان از ابزار متن باز برای پیاده سازی آن به GitLab[۷] و Gerrit[۶] اشاره نمود.

خط لوله CI/CD

همان طور که در گذشته نیز راجع به آن صحبت کردیم، خط لوله CI/CD به تیم‌ها اجازه می‌دهند تا کدهای مدل و داده‌ها را به‌صورت مداوم تست، تأیید و استقرار دهند. در این فرآیند، مدل‌ها به طور خودکار بازآموزی و بهبود می‌یابند و در محیط‌های مختلف (توسعه، تست، تولید) به صورت پیوسته به‌روزرسانی می‌شوند. این کار نه تنها باعث افزایش کیفیت و دقت مدل‌ها می‌شود بلکه زمان توسعه و عرضه را نیز به طرز قابل‌توجهی کاهش می‌دهد. در MLOps این خط لوله ها در مراحل مختلف از جمله آموزش مدل، ارزیابی، استقرار و نظارت بر عملکرد مدل‌ها و هم چنین داده ها استفاده می‌شوند [۳۰]. از ابزارهای مناسب برای این کار می توان به Jenkins[۸] و GitLab CI[۷] نام برد.

---

[۲۱] Edge Devices

[۲۲] Pruning

فصل ۴

آنتروپی و استفاده از آن در نهان نگاری

۴-۱ مقدمه

در فصل گذشته سیستم بینایی انسان و ویژگی‌های آن را مورد بررسی قرار دادیم. در حوزهٔ تبدیل DCT و تبدیل موجک، دو مدل معروف و موجود برای سیستم بینایی معرفی کردیم. همچنین چارچوب کلی استفاده از مدلهای بینایی را برای کاربرد نهان نگاری، مطرح نمودیم و دو طرح مرجع P&Z و K&R را نیز مورد بررسی قرار دادیم. در این فصل به بیان هدف اصلی این پایان نامه که بررسی اثر آنتروپی در نهان نگاری است، می‌پردازیم. لذا ابتدا در بخش ۴-۲ به بیان مفهوم آنتروپی پرداخته، سپس در بخش ...

۴-۲ آنتروپی

در این قسمت ابتدا توضیح مختصری در بارهٔ مفهوم آنتروپی داده خواهد شد.

فصل ۵

نتیجه‌گیری و پیشنهادات

با پیشرفت فن‌آوری دیجیتال و گسترش هرچه بیشتر کاربردهای سرویسهای چندرسانه‌ایِ دیجیتال، نیازهای امنیتی جدیدی در سطح جهان مطرح گردیده است و لذا با نفوذ دنیای دیجیتال به زندگی مردم، طراحی سیستمهای امنیتی مرتبط به آن اهمیت فراوانی در سالهای اخیر پیدا کرده‌اند. به دنبال این نیاز، نهان نگاری به عنوان روشی مؤثر جهت تأمین برخی از این نیازها مورد توجه قرار گرفته و پیشرفت سریعی داشته است.

در این پایان نامه جهت آشنایی و نیل به یک دیدگاه کلی از سیستمهای نهان نگاری ابتدا به بیان کاربردهای نهان نگاری پرداختیم. ...

مراجع

[1] "Apache airflow," URL: https://airflow.apache.org/ [Accessed: 2023-11-02].

[2] "Apache spark," URL: https://spark.apache.org/ [Accessed: 2023-11-29].

[3] "Containerization," URL: https://www.ibm.com/topics/containerization [Accessed: 2023-05-21].

[4] "Docker," URL: https://docs.docker.com/ [Accessed: 2023-05-18].

[5] "Feast," URL: https://feast.dev/ [Accessed: 2023-11-05].

[6] "Gerritcodereview," URL: https://www.gerritcodereview.com/ [Accessed: 2024-05-16].

[7] "Gitlab," URL: https://about.gitlab.com/ [Accessed: 2024-05-16].

[8] "Jenkins," URL: https://www.jenkins.io/ [Accessed: 2023-11-01].

[9] "Knative," URL: https://knative.dev/docs/ [Accessed: 2023-11-29].

[10] "Kubeflow," URL: https://www.kubeflow.org/ [Accessed: 2023-11-02].

[11] "Mlflow," .

[12] "What is devops?," URL: https://aws.amazon.com/devops/what-is-devops/ [Accessed: 2024-05-07].

[13] "What is virtualization?," URL: https://aws.amazon.com/what-is/virtualization/ [Accessed: 2024-05-08].

[14] SAIBS Arachchi and Indika Perera, "Continuous integration and continuous delivery pipeline automation for agile software project management," in *Moratuwa Engineering Research Conference (MERCon)*, May 2018, pp. 156–161.

[15] Matej Artac, Tadej Borovssak, Elisabetta Di Nitto, Michele Guerriero, and Damian Andrew Tamburri, "Devops: introducing infrastructure-as-code," in *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 497–498.

[16] Lucas Cardoso Silva, Fernando Rezende Zagatti, Bruno Silva Sette, Lucas Nildaimon dos Santos Silva, Daniel Lucrédio, Diego Furtado Silva, and Helena de Medeiros Caseli, "Benchmarking machine learning solutions in production," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2020, pp. 626–633.

[17] Nelly Delgado, Ann Q Gates, and Steve Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," *IEEE Transactions on software Engineering*, vol. 30, no. 12, pp. 859–872, 2004.

[18] Behrouz Derakhshan, Alireza Rezaei Mahdiraji, Tilmann Rabl, and Volker Markl, "Continuous deployment of machine learning pipelines.," in *EDBT*, March 2019, pp. 397–408.

[19] L Navarro E Hernanchez-sanchez F Rodriguez-Haro, F Freitag, "A summary of virtualization techniques," .

[20] Anja Kammer Florian Beetz and Dr. Simon Harrer, *GitOps Cloud-native Continuous Deployment*, 2021.

[21] N. Forsgren and J. Humble, "The role of continuous delivery in it and organizational performance," March 2016.

[22] Jez Humble and David Farley, *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation*, Addison-Wesley Professional, 2010.

[23] M. Huttermann, *DevOps for Developers*, chapter Infrastructure as Code, 2012.

[24] Ramtin Jabbari, Nauman bin Ali, Kai Petersen, and Binish Tanveer, "What is devops? a systematic mapping study on definitions and practices," in *Proceedings of the Scientific Workshop Proceedings of XP2016*. 2016, Association for Computing Machinery.

[25] A. Van Bennekum A. Cockburn-W. Cunningham M. Fowler J. Grenning J. Highsmith A. Hunt R. Jeffries K. Beck, M. Beedle, "Manifesto for agile software development," 2001, URL: `https://agilemanifesto.org/` [Accessed: 2024-02-17].

[26] Ioannis Karamitsos, Saeed Albarhami, and Charalampos Apostolopoulos, "Applying devops practices of continuous automation for machine learning," *Information*, vol. 11, no. 7, 2020.

[27] Lucy Ellen Lwakatare, Ivica Crnkovic, Ellinor Rånge, and Jan Bosch, "From a data science driven process to a continuous delivery process for machine learning systems," in *Product-Focused Software Process Improvement*, Maurizio Morisio, Marco Torchiano, and Andreas Jedlitschka, Eds., Cham, 2020, pp. 185–201, Springer International Publishing.

[28] Álvaro López García, Jesús Marco De Lucas, Marica Antonacci, Wolfgang Zu Castell, Mario David, Marcus Hardt, Lara Lloret Iglesias, Germán Moltó, Marcin Plociennik, Viet Tran, Andy S. Alic, Miguel Caballer, Isabel Campos Plasencia, Alessandro Costantini, Stefan Dlugolinsky, Doina Cristina Duma, Giacinto Donvito, Jorge Gomes, Ignacio Heredia Cacha, Keiichi Ito, Valentin Y. Kozlov, Giang Nguyen, Pablo Orviz Fernández, Zděnek Šustr, and Pawel Wolniewicz, "A cloud-based framework for machine learning workloads and applications," *IEEE Access*, vol. 8, pp. 18681–18692, 2020.

[29] paul hammant, "Trunk based development," URL: `https://trunkbaseddevelopment.com/` [Accessed: 2023-11-01].

[30] Alexandra Posoldova, "Machine learning pipelines: From research to production," *IEEE Potentials*, vol. 39, no. 6, pp. 38–42, 2020.

[31] Amit M Potdar, DG Narayan, Shivaraj Kengond, and Mohammed Moin Mulla, "Performance evaluation of docker container and virtual machine," *Procedia Computer Science*, vol. 171, pp. 1419–1428, 2020.

[32] Cédric Renggli, Luka Rimanic, Nezihe Merve Gürel, Bojan Karlas, Wentao Wu, and Ce Zhang, "A data quality-driven view of mlops," *CoRR*, vol. abs/2102.07750, 2021, URL: `https://arxiv.org/abs/2102.07750` [Accessed: 2023-11-30].

[33] M. Schmitt, "Airflow vs. luigi vs. argo vs. mlflow vs. kubeflow," URL: `https://www.datarevenue.com/en-blog/airflow-vs-luigi-vs-argo-vs-mlflow-vs-kubeflow` [Accessed: 2023-11-02].

[34] Damian A. Tamburri, "Sustainable mlops: Trends and challenges," in *2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2020, pp. 17–23.

[35] Manish Virmani, "Understanding devops and bridging the gap from continuous integration to continuous delivery," in *Fifth International Conference on the Innovative Computing Technology (INTECH 2015)*, 2015, pp. 78–82.

[36] Y. Yu Y. Zhou and B. Ding, "Towards mlops: A case study of ml pipeline platform," October 2020.

ABSTRACT

In the digital world today, invisible and robust image watermarking which embeds invisible signals in to the digital images has been proposed as a major solution to the problem of copyright protection of digital images. Several approaches such as exploiting Human Visual System (HVS) and invariant domain watermarking have been proposed to achieve this goal. In this thesis we use the information-theoretic concepts as tools to develop methods for embedding watermark in an optimized way. Also multi-resolution transforms such as wavelet transform and MR-SVD (Multi-Resolution form of the Singular Value Decomposition) are used in the proposed structure, because theses transforms resemble the HVS characteristics for an optimized watermarking structure. Entropy concept and entropy masking effects were proposed to use to develop a model in DWT domain to increase the strength and robustness of the watermark, while perceived quality of the electronic image is not altered. Then, the structure similar to the entropy-based proposed structure in DWT domain, is used for watermarking in the MR-SVD transform domain, which is found a new approach to robust image watermarking. Simulation results show that the proposed methods outperform conventional methods in terms of both invisibility and robustness.

KEYWORDS

1. Image Watermarking.
2. Multi-Resolution Transform.
3. Human Visual System (HVS).
4. Wavelet Transform.
5. Singular Value Decomposition (SVD).
6. Entropy.
7. Entropy Masking.

SHARIF UNIVERSITY OF TECHNOLOGY
ELECTRICAL ENGINEERING DEPARTMENT

M.Sc. THESIS

Title:

An Information-Theoretic Model for Image Watermarking

by:
AAAAA BBBBBB

Supervisor:

Dr. ...

August 2005