

بسم الله الرحمن الرحيم



دانشکده مهندسی برق و کامپیوتر - گروه مهندسی کامپیوتر

پایان نامه پروژه کارشناسی

رشته : مهندسی کامپیوتر

عنوان :

استفاده از ابزار "داکر" برای پیاده سازی مدل های یادگیری عمیق

استاد راهنما :

دکتر راضیه راستگو

دانشجو :

محمدنوید افضلی

شماره دانشجویی:

۹۷۱۱۱۲۶۰۱۲

نیمسال اول سال تحصیلی ۰۲-۰۱

## فهرست مطالب

چکیده .....	۷
فصل اول .....	۸
مقدمه .....	۸
۱-۱ داکر چیست؟ .....	۹
۱-۱-۱ تاریخچه کوتاهی از داکر : .....	۹
۱-۱-۲ نگهدارنده چیست؟ .....	۱۰
۱-۱-۳ ماشین مجازی چیست؟ .....	۱۰
۱-۱-۴ تفاوت داکر و ماشین مجازی در چیست؟ .....	۱۱
۲-۱ دلیل نیاز به استفاده از ابزار داکر چیست؟ .....	۱۱
۳-۱ کاربرد های داکر .....	۱۳
۴-۱ مزایا و معایب داکر .....	۱۴
۱-۴-۱ مزایا: .....	۱۴
۲-۴-۱ معایب: .....	۱۴
۵-۱ دلایل محبوبیت داکر .....	۱۵
۶-۱ اجزای اصلی داکر .....	۱۵
۱-۶-۱ موتور داکر .....	۱۵
۲-۶-۱ کلاینت داکر .....	۱۶
۳-۶-۱ داکر دیمون .....	۱۶
۵-۶-۱ داکر ایمج .....	۱۷
۶-۶-۱ داکر کانٹینر .....	۱۸
۷-۶-۱ داکر هاب .....	۱۹

۱۹.....	۸-۶-۱ سی گروپ ها
۲۰.....	۹-۶-۱ نیم اسپیس ها
۲۰.....	۷-۱ مهم ترین دستورات داکر
۲۰.....	۱-۷-۱ دستور DOCKER INFO :
۲۲.....	۳-۷-۱ دستور DOCKER IMAGES :
۲۲.....	۴-۷-۱ دستور DOCKER PULL :
۲۳.....	۵-۷-۱ دستور DOCKER PUSH :
۲۳.....	۶-۷-۱ دستور DOCKER RUN :
۲۳.....	۷-۷-۱ دستور DOCKER BUILD :
۲۴.....	۸-۷-۱ دستور DOCKER COMMIT :
۲۵.....	فصل دوم
۲۵.....	داکر و یادگیری عمیق
۲۶.....	۱-۲ هوش مصنوعی چیست؟
۲۷.....	۲-۲ یادگیری ماشین
۲۸.....	۳-۲ یادگیری عمیق
۲۹.....	۴-۲ تفاوت های یادگیری ماشین و یادگیری عمیق
۲۹.....	۵-۲ شبکه های عصبی
۳۰.....	۱-۵-۲ شبکه های عصبی کانولوشن
۳۱.....	۷-۲ کانولوشن
۳۲.....	۸-۲ انواع شبکه های عصبی کانولوشن
۳۲.....	۹-۲ کاربرد های شبکه های عصبی کانولوشن
۳۳.....	۱۰-۲ چرا از کانتینر های داکر در یادگیری عمیق استفاده میکنیم؟

۳۴	..... ۲-۱۰-۱- ساخت نگهدارنده
۳۶	..... فصل سوم
۳۶	..... پیاده سازی یادگیری ماشین(عمیق) با استفاده از داکر
۳۷	..... ۳-۱ پروژه یادگیری ماشین
۳۷	..... ۳-۱-۱ مقدمات لازم برای ساخت تصویر داکر
۳۸	..... ۳-۱-۲ شروع ساخت تصویر
۴۱	..... ۳-۱-۳ اجرای تصویر روی نگهدارنده
۴۲	..... ۳-۲ پروژه یادگیری عمیق
۴۲	..... ۳-۲-۱ مقدمه
۴۲	..... ۳-۲-۲ شروع ساخت تصویر
۴۷	..... ۳-۳ جمع بندی و نتیجه گیری
۴۸	..... سخن پایانی
۴۹	..... منابع
۵۱	..... ABSTRACT

## فهرست اشکال

- شکل ۱-۱ تفاوت عملکرد داکر و ماشین مجازی ..... ۱۱
- شکل ۲-۱ معماری داخلی داکر ..... ۱۲
- شکل ۳-۱ عملکرد داکر ؛ از ساخت نگهدارنده تا ذخیره سازی و استفاده از آن... ۱۳
- شکل ۴-۱ تعریف داکر رجیستری ..... ۱۷
- شکل ۵-۱ تعریف و نحوه عملکرد داکر ایميج ..... ۱۸
- شکل ۶-۱ تعریف و نحوه عملکرد نگهدارنده داکر ..... ۱۹
- شکل ۷-۱ - دستور **docker info** ..... ۲۱
- شکل ۸-۱ - دستور **docker ps** ..... ۲۱
- شکل ۹-۱ - دستور **docker images** ..... ۲۲
- شکل ۱۰-۱ - دستور **docker pull** ..... ۲۲
- شکل ۱۱-۱ - دستور **docker run** ..... ۲۳
- شکل ۱-۲ نمودار ون هوش مصنوعی، یادگیری ماشین و یادگیری عمیق ..... ۲۷
- شکل ۲-۲ عمل کانولوشن ..... ۳۱
- شکل ۳-۲ پارامتر های اصلی کانولوشن ..... ۳۲
- شکل ۱-۳ داکر فایل ..... ۳۸
- شکل ۲-۳ مراحل ساخت تصویر ..... ۴۰
- شکل ۳-۳ تصویر های موجود در مخزن محلی (دریافت شده **Docker hub**) ..... ۴۰
- شکل ۴-۳ اجرای تصویر روی نگهدارنده ..... ۴۱
- شکل ۵-۳ محتویات داکر فایل ..... ۴۳
- شکل ۶-۳ اجرای تصویر ۲ ..... ۴۴
- شکل ۷-۳ اجرای برنامه با کانتینر ..... ۴۵
- شکل ۸-۳ تصویر نمونه ..... ۴۵
- شکل ۹-۳ خروجی کانتینر تحت وب ..... ۴۶

## چکیده

در طی سالیان اخیر و به ویژه امروزه، از داکر به عنوان یکی از ابزارهای مهم و کاربردی در حوزه برنامه‌نویسی یاد می‌شود. داکر به عنوان ابزاری برای ساخت و اجرای برنامه‌ها در محیط‌های نگهدارنده‌ای باعث افزایش سرعت و بهره‌وری قابل توجه برنامه‌ها گردیده‌است. با ابزار داکر دیگر نگرانی بابت تفاوت نسخه‌های متعلقات برنامه و کتابخانه‌های آن وجود نخواهد داشت. هم‌اکنون این ابزار در توسعه‌ی برنامه‌های مربوط به هوش مصنوعی و یادگیری ماشین/عمیق نیز قابل استفاده است و توسعه‌دهندگان این حوزه می‌توانند برنامه‌های خود را بدون دغدغه از طریق داکر با همکاران خود به اشتراک بگذارند.

همچنین داکر برای این پروژه‌ها بسیار مفید است؛ زیرا پروژه‌های یادگیری ماشین/عمیق دارای کتابخانه‌های زیادی با چندین نسخه هستند. از طرف دیگر، آنها در اکثر دستگاه‌ها یکسان نیستند، بنابراین در این موارد داکر به ما کمک شایانی می‌کند. در این پروژه قصد داریم با استفاده از ابزارهای موجود در داکر، یک نگهدارنده روی مدل یادگیری ماشین/عمیق خود ساخته و پیاده‌سازی کنیم. در واقع ایده‌ی این پروژه ساخت سریع و آسان نگهدارنده داکر با یک مدل یادگیری عمیق ساده و اجرای آن است. پیاده‌سازی این پروژه در <https://github.com/NavidAfzali/Docker-for-deep-learning/tree/master> موجود است.

# فصل اول

## مقدمه



## ۱-۱-۱ داکر چیست؟

داکر پروژه‌ای متن باز<sup>۲</sup> برای توسعه و اجرای برنامه‌ها بر مبنای نگهدارنده‌ها می‌باشد و عملیات ساخت<sup>۳</sup>، اجرا<sup>۴</sup> و مدیریت برنامه را تسهیل می‌بخشد. داکر می‌تواند یک برنامه و متعلقات آن را (کتابخانه‌ها و ...) در یک نگهدارنده مجازی اجرا کند. این ابزار توسط زبان برنامه‌نویسی GO توسعه یافته است. [۳]

### ۱-۱-۱-۱ تاریخچه کوتاهی از داکر

در سال ۲۰۰۶ پروسس نگهدارنده<sup>۵</sup> ارائه شد. در سال ۲۰۰۸ LXC<sup>۶</sup> ارائه شد که این موضوع مقدمه‌ای مناسب و خوب برای استفاده از نگهدارنده‌ها بود. در سال ۲۰۱۳ سرویس داکر ارائه شد. در همین سال شرکت بزرگ رد هت<sup>۷</sup> اعلام کرد که پروژه‌های لینوکس<sup>۸</sup>، اوپن شیفت<sup>۹</sup> و فدورا<sup>۱۰</sup> را به سمت این پروژه خواهد برد و از این پروژه حمایت کرد. در سال ۲۰۱۴ شرکت مایکروسافت اعلام کرد که نسخه‌ی آینده ویندوز سرور خودش را به سمتی خواهد برد که بتواند داکر کلاینت<sup>۱۱</sup> را پشتیبانی کند. شرکت IBM نیز در همین سال اعلام کرد که برنامه‌ریزی خواهد کرد تا در IBM CLOUD از این سرویس استفاده کند و در همین سال شراکت خودش را با سرویس داکر اعلام کرد. در سال ۲۰۱۶ شرکت داکر سرویس تجاری<sup>۱۲</sup> خود را ارائه کرد. در حال حاضر سرویس داکر دارای جامعه‌ی متن باز بسیار پویا و کاملی می‌باشد که بسیار فعال بوده و همواره در حال رشد و شکوفایی بیش‌تر می‌باشد [۴].

در ادامه به تعریف نگهدارنده و ماشین مجازی می‌پردازیم.

---

<sup>۱</sup> Docker

<sup>۲</sup> Open source

<sup>۳</sup> build

<sup>۴</sup> run

<sup>۵</sup> Process container

<sup>۶</sup> Linux container

<sup>۷</sup> Red hat

<sup>۸</sup> Linux

<sup>۹</sup> Open Shift

<sup>۱۰</sup> Fedora

<sup>۱۱</sup> Docker client

<sup>۱۲</sup> Commercial

### ۱-۱-۲ نگهدارنده چیست؟

نگهدارنده<sup>۱</sup> یک واحد نرم‌افزاری استاندارد است که کدها و تمام متعلقات آن را بسته‌بندی می‌کند. به این ترتیب، اپلیکیشن در محیط‌های محاسباتی مختلف، سریع‌تر و با اطمینان بیش‌تر اجرا می‌شود. هر نگهدارنده یک محیط ایزوله شده را مشابه یک ماشین مجازی<sup>۲</sup> فراهم می‌کند. برخلاف ماشین‌های مجازی، نگهدارنده‌های داکر یک سیستم‌عامل کامل را اجرا نمی‌کنند، بلکه هسته<sup>۳</sup> میزبان را به اشتراک می‌گذارند و مجازی‌سازی را در یک سطح نرم‌افزاری انجام می‌دهند. برای درک بهتر نگهدارنده‌ها، بهتر است از یک مثال استفاده کنیم. فرض کنید در شرکتی مشغول به کار هستید و ناهار خود را هر روز در خانه درست کرده و آن را داخل یک ظرف به شرکت می‌برید تا آنجا میل بفرمایید. دیگر لازم نیست داخل شرکت شروع به پختن غذا کنید چون احتمالاً زمان زیادی را از شما می‌گیرد. کار نگهدارنده هم تا حدودی شبیه به این است. شما پروژه خود را (غذا) داخل نگهدارنده (ظرف غذا) قرار داده و آن را هر کجا که دوست داشتید (مثلاً شرکت) می‌برید. به لطف این تکنیک، برنامه‌نویس خیالش راحت است که برنامه او می‌تواند در سیستم‌های دیگر بدون نیاز به تنظیمات خاص یا ابزارهای جانبی اجرا شود.

### ۱-۱-۳ ماشین مجازی چیست؟

ماشین مجازی برنامه‌ای است که به عنوان یک رایانه مجازی عمل می‌کند. این برنامه بر روی سیستم‌عامل فعلی شما اجرا می‌شود و سخت‌افزاری مجازی برای یک سیستم‌عامل میهمان ارائه می‌کند. از نگاه سیستم‌عامل میهمان، ماشین مجازی مشابه یک رایانه واقعی عمل می‌کند. ماشین‌های مجازی از سخت‌افزارهای سیستم همچون CPU، حافظه RAM و سایر سخت‌افزارها استفاده می‌کند. البته نحوه عملکرد داکر با ماشین مجازی متفاوت است که در ادامه شرح داده می‌شود.

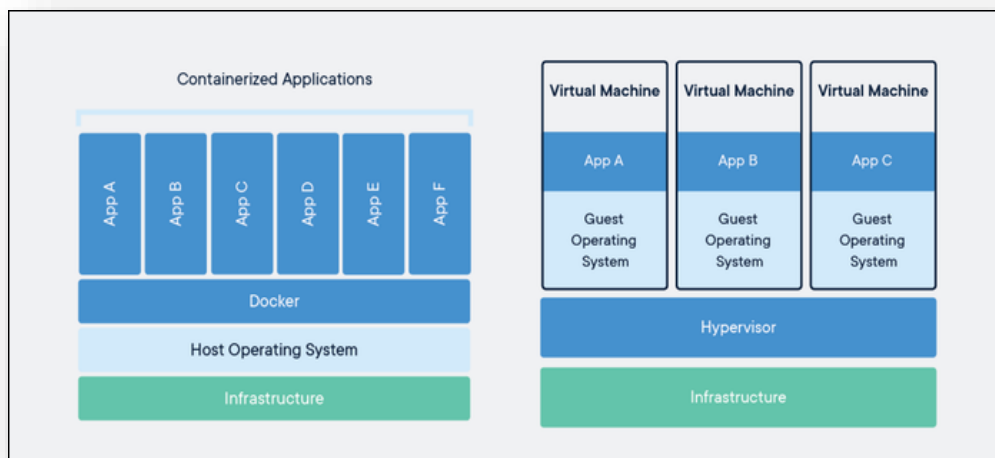
---

<sup>۱</sup> Container

<sup>۲</sup> Virtual machine

<sup>۳</sup> Kernel

## ۴-۱-۱ تفاوت داکر و ماشین مجازی در چیست؟



شکل ۲-۱ تفاوت عملکرد داکر و ماشین مجازی

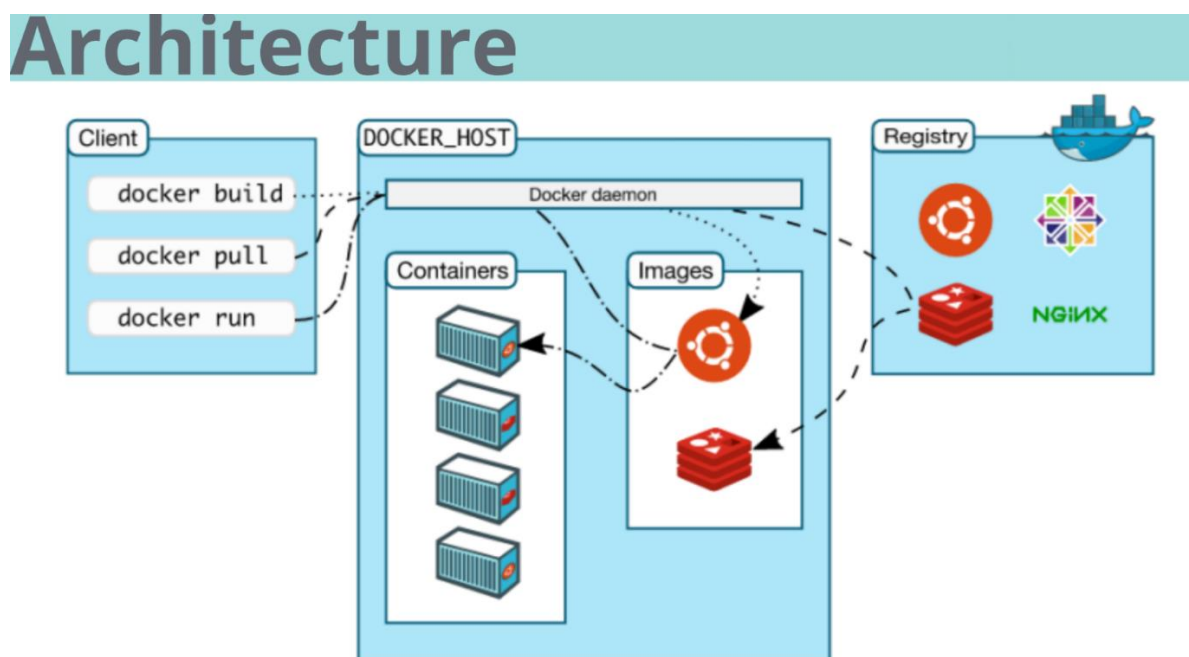
در ماشین مجازی، قسمتی از سخت‌افزار سیستم به ماشین مجازی اختصاص داده می‌شود و روی آن یک سیستم‌عامل کامل نصب و اجرا می‌شود. در واقع می‌توان گفت در ماشین مجازی امکانات سخت‌افزاری سیستم تقسیم می‌شود و بر روی هر قسمت، سیستم‌عامل بخصوصی به اجرا در می‌آید، اما در داکر این طور نیست. در داکر امکانات سخت‌افزاری به تناسب نیاز هر نگهدارنده به صورت موقت اختصاص داده می‌شود و داکر این امکان را فراهم می‌آورد که برنامه‌ها روی کرنل سیستم‌عامل اجرا شوند. این عمل، بازدهی و کارایی سیستم را تا حد زیادی بالا می‌برد. در این حالت دیگر نیازی به نصب پیش‌نیازها و نیازمندی‌هایی که برنامه‌ی ما می‌خواهد و به طور پیش‌فرض روی سیستم وجود ندارد، نیست. این سرویس به شما کمک می‌کند یک محیط را به چند بخش تقسیم کرده و و در هر بخش یک برنامه مجزا را اجرا کنید.

در ادامه به دلایل نیاز به استفاده از داکر می‌پردازیم.

## ۲-۱ دلیل نیاز به استفاده از ابزار داکر چیست؟

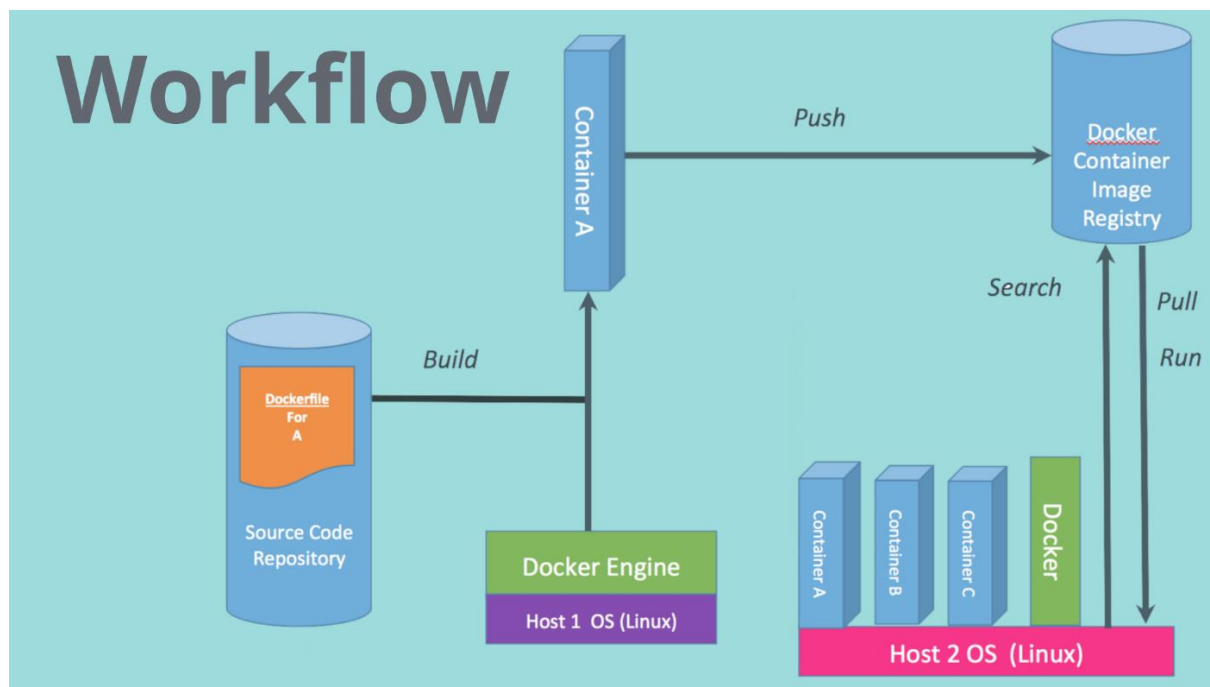
با ذکر یک مثال، دلیل نیاز به این ابزار را نشان می‌دهیم. برای مثال، سه اپلیکیشن مختلف مبتنی

بر پایتون<sup>۱</sup> وجود دارد که قرار است روی یک سرور میزبانی شوند (چه فیزیکی چه مجازی). همچنین، فرض می‌کنیم که در هر یک از این برنامه‌ها نسخه‌های متفاوتی از پایتون استفاده شده و کتابخانه‌ها و متعلقات به کار گرفته شده نیز در هر اپلیکیشن با دیگری فرق دارد. با توجه به اینکه نمی‌توان نسخه‌های مختلفی از پایتون را روی یک ماشین نصب کرد، امکان میزبانی از این سه اپلیکیشن روی یک کامپیوتر واحد وجود نخواهد داشت. اینجاست که فناوری داکر به کار می‌آید. دیگر نیاز نیست که چند ماشین مجازی مختلف را برای نسخه‌های مختلف پایتون اجرا کنیم و قسمت زیادی از سخت‌افزار خود را از دست دهیم. تنها یک بار داکر را نصب می‌کنیم؛ داکر، خود نگهدارنده‌های مجزا را ایجاد می‌کند. بدین ترتیب یک محیط را به چند بخش تقسیم کرده و در هر قسمت، یک برنامه مجزا را اجرا می‌کنیم.



شکل ۱-۱ معماری داخلی داکر

<sup>۱</sup> زبان برنامه نویسی Python



شکل ۱-۲ عملکرد داکر؛ از ساخت نگهدارنده تا ذخیره سازی و استفاده از آن

### ۳-۱ کاربردهای داکر [۶]

#### ✓ تولید برنامه‌های سریع و پایدار

داکر منجر به ساده‌سازی چرخه حیات توسعه می‌شود. این ساده‌سازی به وسیله فراهم کردن امکان کار در محیط‌های استانداردسازی شده برای توسعه‌دهندگان با استفاده از نگهدارنده‌های محلی فراهم شده است که اپلیکیشن‌ها و خدمات لازم را ارائه می‌دهند.

#### ✓ اجرای حجم کاری بیش‌تر بدون نیاز به ارتقای سخت‌افزار

داکر کم‌حجم و سریع است. داکر جایگزینی مقرون به صرفه برای ماشین‌های مجازی مبتنی بر لایه نرم‌افزاری<sup>۱</sup> به حساب می‌آید. با داکر می‌توان ظرفیت محاسباتی بیش‌تری را برای دستیابی به اهداف کسب و کار به کار گرفت. داکر

<sup>۱</sup> Hypervisor-Based

برای محیط‌هایی با تراکم بالا و همچنین برای استقرار نرم‌افزارهای کوچک و متوسط یعنی جایی مناسب است که نیاز به اجرای وظایف بیش‌تر با منابع کم‌تر وجود داشته باشد.

### ✓ مقیاس‌پذیری

نگهدارنده در داکر امکان ایجاد حجم کاری<sup>۱</sup> قابل حمل را فراهم می‌سازد. نگهدارنده‌های داکر می‌توانند روی لپ‌تاپ توسعه‌دهنده، ماشین‌های فیزیکی یا مجازی در یک مرکز داده، ارائه دهندگان فضای ابری یا روی محیط‌های تلفیقی اجرا شوند.

## ۴-۱ مزایا و معایب داکر

در ادامه به مزایا و معایب داکر می‌پردازیم: [۶]

### ۱-۴-۱ مزایا:

- ✓ ثبات و پایداری
- ✓ صرفه‌جویی در فضای ذخیره‌سازی
- ✓ وسعت و انعطاف جامعه توسعه‌دهندگان
- ✓ امکان استفاده در مک و ویندوز
- ✓ خودکارسازی

### ۲-۴-۱ معایب:

- ✓ کمبود در مستندات
- ✓ در حال حاضر مشکلات کارکردی برای Mac وجود دارد
- ✓ بروز مشکلات در محیط‌های غیر بومی

---

<sup>۱</sup> Workload

## ۱-۵ دلایل محبوبیت داکر

داکر به دلیل راهاندازی بر بستر نگهدارنده کیفیت برنامه‌ها را بالاتر برده و همچنین به دلیل قابل حمل بودن آن، شرایط استفاده در هر مکانی را داراست.

امکان محفظه‌سازی تنها برای یک بار و اجرا در همه محیط‌ها، منجر به کاهش فاصله میان محیط توسعه و سرورهای تولید محصول می‌شود. استفاده از نگهدارنده‌ها این اطمینان و اعتماد را به وجود می‌آورد که همه محیط‌ها برابر هستند. در صورتی که عضو جدیدی به تیم توسعه اضافه شود، این فرد تنها لازم است از دستور «docker run» برای راهاندازی نمونه توسعه خود استفاده کند (این دستورات در ادامه معرفی خواهند شد).

داکر نسبت به یک ماشین مجازی بسیار ساده‌تر است. ماشین‌های مجازی ابزارهایی همه‌منظوره هستند که برای پشتیبانی از هر میزان حجم کاری ممکن طراحی شده‌اند. در مقابل آن، نگهدارنده‌های کم حجم داکر، خودکفا و برای کاربردهای یک بار مصرف مناسب‌تر هستند. با توجه به اینکه داکر کرنل میزبان را به اشتراک می‌گذارد، نگهدارنده‌ها تاثیر قابل چشم‌پوشی در کاهش عملکرد و کارایی سیستم دارند. زمان اجرای نگهدارنده‌ها تقریباً لحظه‌ای و بلافاصله<sup>۱</sup> است.

## ۱-۶ اجزای اصلی داکر [۱]

### ۱-۶-۱ موتور داکر<sup>۲</sup>

موتور داکر هسته شروع تمام کارها در این ابزار است. تا زمانی که موتور داکر آغاز به کار نکند استفاده از داکر ممکن نیست. این موتور حکم اتصال اینترنت را برای این ابزار دارد. در سیستم-عامل‌های ویندوز این موتور به وسیله برنامه docker desktop آغاز به کار می‌کند.

---

<sup>۱</sup> real time

<sup>۲</sup> Docker engine

## ۱-۶-۲ کلاینت داکر<sup>۱</sup>

کلاینت داکر جزئی است که کاربر نهایی با آن در ارتباط خواهد بود. برای درک بهتر می‌توانید آن را یک رابط کاربری<sup>۲</sup> برای داکر بنامید. هر چند که فاصله آن از واژه کاربرپسند<sup>۳</sup> بسیار زیاد است. شما به عنوان کاربر نهایی با کلاینت داکر ارتباط خواهید داشت و کلاینت داکر دستورات شما را به داکر دیمون منتقل می‌نماید.

## ۱-۶-۳ داکر دیمون<sup>۴</sup>

داکر دیمون جزئی است که دستورات ارسال شده به کلاینت داکر را اجرا می‌نماید. دستوراتی مانند ساختن، راه‌اندازی و یا توزیع نگهدارنده‌ها. داکر دیمون، خود بر روی هاست اجرا می‌شود. اما به عنوان کاربر شما هیچ‌گاه به صورت مستقیم با آن در ارتباط نخواهید بود. همان طور که در قسمت قبل در توضیح کلاینت داکر گفته شد، کلاینت داکر انتقال‌دهنده دستورات شما به داکر دیمون خواهد بود.

## ۱-۶-۴ داکر رجیستری<sup>۵</sup>

داکر رجیستری فضای ذخیره‌سازی تصویرهای داکری می‌باشد. (تصویرهای داکر در ادامه توضیح داده خواهد شد). مخزن Docker Hub یک مخزن عمومی داکر است که تمام کاربران می‌توانند تصویرهایی که برای برنامه خود ایجاد می‌کنند را در آن ذخیره کنند تا دیگران هم بتوانند از آن استفاده کنند.

---

<sup>۱</sup> Docker client

<sup>۲</sup> User interface

<sup>۳</sup> User friendly

<sup>۴</sup> Docker daemon

<sup>۵</sup> Docker registry



# Docker Registry

A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries that anyone can use, and Docker is configured to look for images on Docker Hub by default.

<https://docker.dockerme.ir>



شکل ۱-۳ تعریف داکر رجیستری؛ [4]

## ۱-۶-۵ داکر ایمج<sup>۱</sup>

داکر ایمج یک فایل غیر قابل تغییر است که شامل کد منبع، کتابخانه‌ها<sup>۲</sup>، وابستگی‌ها، ابزارها و سایر فایل‌های مورد نیاز برای اجرای یک برنامه است.

از آنجا که تصویرها به نوعی فقط الگو یا قالب<sup>۳</sup> هستند، نمی‌توانید آن‌ها را شروع یا اجرا کنید. کاری که می‌توانید انجام دهید این است که از آن الگو به عنوان پایه برای ساخت نگهدارنده استفاده کنید.

---

<sup>۱</sup> Docker image

<sup>۲</sup> Libraries

<sup>۳</sup> Template

# Image

An image is a read-only template with instructions for creating a Docker container. Often, an image is based on another image, with some additional customization.



شکل ۴-۱ تعریف و نحوه عملکرد داکر/یمیج؛ [4]

## ۱-۶-۶ داکر کانتینر

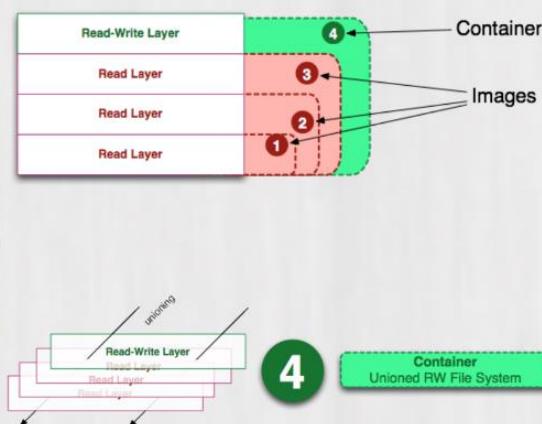
نگهدارنده داکر یک محیط مجازی **run-time** است که در آن کاربران می‌توانند برنامه‌ها را از سیستم اصلی جدا کنند. این نگهدارنده‌ها واحدهای قابل حمل و فشرده‌ای هستند که می‌توانید در آن‌ها به راحتی یک برنامه را راه‌اندازی کنید. یک ویژگی مهم دیگر، استانداردسازی محیط محاسبات و رایانش در حال اجرا در داخل نگهدارنده است. این نه تنها از این که برنامه شما در شرایط یکسانی در حال کار است اطمینان حاصل می‌کند، بلکه اشتراک‌گذاری با سایر هم‌تیمی‌ها را نیز ساده می‌سازد.

# Container

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI

A container is defined by its image as well as any configuration options you provide to it when you create or start it.

When a container is removed, any changes to its state that are not stored in persistent storage disappear.



شکل ۱-۵ تعریف و نحوه عملکرد نگهدارنده داکر؛ [4]

## ۱-۶-۷ داکر هاب<sup>۱</sup>

داکر هاب سرور ذخیره ساز<sup>۲</sup> مبتنی بر ابر است که مصرف کنندگان و کاربران داکر می توانند در آن ایمج های نگهدارنده را ایجاد، آزمون، ذخیره سازی و توزیع کنند. داکر هاب نسخه ای از داکر رجیستری می باشد که میزبانی آن مبتنی بر ابر<sup>۳</sup> است.

## ۱-۶-۸ سی گروپ ها<sup>۴</sup>

سی گروپ ها اجازه تخصیص منابع را به ما می دهند. منظور از منابع، به طور کلی سخت افزار که شامل سی پی یو<sup>۵</sup>، حافظه، شبکه، دیسک ورودی و خروجی<sup>۶</sup> و ... می باشد. منابع در داکر توسط سی گروپ ها مدیریت می شوند.

<sup>۱</sup> Docker hub

<sup>۲</sup> Repository

<sup>۳</sup> cloud -hosted

<sup>۴</sup> Cgroups

<sup>۵</sup> CPU

<sup>۶</sup> I/O disk

سی‌گروپ‌ها به موتور داکر<sup>۱</sup> اجازه می‌دهند تا منابع سخت‌افزاری موجود را به نگهدارنده‌ها اختصاص دهند و به صورت اختیاری می‌توانند محدودیت‌های لازم را به اجرا درآورند؛ برای مثال، شما می‌توانید حافظه‌ی موجود را برای نگهدارنده‌ی خاصی محدود کنید.

### ۱-۶-۹ نیم‌اسپیس‌ها<sup>۲</sup>

داکر از نیم‌اسپیس‌ها برای تهیه فضاهای مجزا به نام نگهدارنده استفاده می‌کند. از طرف دیگر نیم‌اسپیس مکانیزم دیگری در کرنل برای محدود کردن دید گرهی از فرآیندها، نسبت به بقیه سیستم‌ها است. برای مثال می‌تواند یک یا چند فرآیند را محدود کرد تا امکان دیدن و تعامل با بقیه فرآیندهای در حال اجرا روی سیستم یا مثلاً امکان دسترسی به فایل سیستم‌های مانت<sup>۳</sup> شده را نداشته‌باشد.

### ۱-۷-۷ مهم‌ترین دستورات داکر

در ادامه به برخی از مهم‌ترین دستورات داکر که از آن‌ها به صورت متداول در خط فرمان<sup>۴</sup> استفاده می‌شود، می‌پردازیم:

#### ۱-۷-۱ دستور **docker info** :

این دستور توضیح کامل و جامعی از سرویس داکری که بر روی سیستم ما قرار دارد، در اختیار ما قرار می‌دهد. اطلاعاتی مانند تعداد نگهدارنده‌ها و وضعیت هر کدام از آن‌ها، تعداد تصویرها، نسخه‌ی کرنل سیستم‌عامل، میزان منابعی که در اختیار دارید و ...

---

<sup>۱</sup> Docker engine

<sup>۲</sup> Namespaces

<sup>۳</sup> Mount

<sup>۴</sup> Command line

```
Command Prompt
Server:
Containers: 17
Running: 1
Paused: 0
Stopped: 16
Images: 4
Server Version: 20.10.13
Storage Driver: overlay2
Backing Filesystem: extfs
Supports d_type: true
Native Overlay Diff: true
userxattr: false
Logging Driver: json-file
Cgroup Driver: cgroupfs
Cgroup Version: 1
Plugins:
Volume: local
Network: bridge host ipvlan macvlan null overlay
Log: awslogs fluentd gcplogs gelf journald json-file local logentries splunk syslog
Swarm: inactive
Runtimes: io.containerd.runc.v2 io.containerd.runtime.v1.linux runc
Default Runtime: runc
Init Binary: docker-init
containerd version: 2a1d4dbdb2a1030dc5b01e96fb110a9d9f150ecc
runc version: v1.0.3-0-gf46b6ba
init version: de40ad0
Security Options:
seccomp
Profile: default
Kernel Version: 5.10.102.1-microsoft-standard-WSL2
```

شکل ۱-۶ - دستور `docker info`

## ۱-۷-۲ دستور `docker ps`:

این دستور لیست نگهدارنده‌ها روی سرویس‌دهنده داکر شما را ارائه می‌دهد. شامل تنظیمات<sup>۱</sup> بسیاری می‌باشد. در حالت کلی و بدون تنظیمات، فقط نگهدارنده‌های در حال کار<sup>۲</sup> را نمایش می‌دهد.

```
C:\Users\M.Navid afzali>docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
8a3b1b935424   busybox   "sh"      7 seconds ago    Up 4 seconds        afzali123
C:\Users\M.Navid afzali>
```

شکل ۱-۷ - دستور `docker ps`

---

<sup>۱</sup> Options  
<sup>۲</sup> Running

### ۱-۷-۳ دستور `docker images` :

این دستور لیست تصویرهای ما را نمایش می‌دهد. همانند دستور قبل شامل تنظیمات زیادی است. برای مثال، با تنظیم `-q` تنها شناسه تصویر<sup>۱</sup>های موجود را برمی‌گرداند.

```
C:\Users\M.Navid afzali>docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
redis         latest    2e50d70ba706   3 weeks ago    117MB
python        latest    0f95b1e38607   3 weeks ago    920MB
busybox       latest    62aedd01bd85   5 weeks ago    1.24MB
hello-world   latest    feb5d9fea6a5   9 months ago   13.3kB

C:\Users\M.Navid afzali>docker images -q
2e50d70ba706
0f95b1e38607
62aedd01bd85
feb5d9fea6a5
C:\Users\M.Navid afzali>
```

شکل ۱-۸ - دستور `docker images`

### ۱-۷-۴ دستور `docker pull` :

توسط این دستور، شما تصویرهای مورد نیاز خود را از مخازن مختلف (به طور پیش‌فرض `hub.docker.com`) دانلود و دریافت می‌کنید. تنظیمات خاصی ندارد و برای دریافت تصویر مورد نیاز خود، نام تصویر و برچسب<sup>۲</sup> مورد نظر را وارد می‌کنید. اگر برچسبی وارد نکنید، داکر به صورت پیش‌فرض برچسب `latest` را در نظر می‌گیرد.

```
C:\Users\M.Navid afzali>docker pull busybox
Using default tag: latest
latest: Pulling from library/busybox
Digest: sha256:3614ca5eacf0a3a1bcc361c939202a974b4902b9334ff36eb29ffe9011aaad83
Status: Image is up to date for busybox:latest
docker.io/library/busybox:latest

C:\Users\M.Navid afzali>
```

شکل ۱-۹ - دستور `docker pull`

---

<sup>۱</sup> Images ID

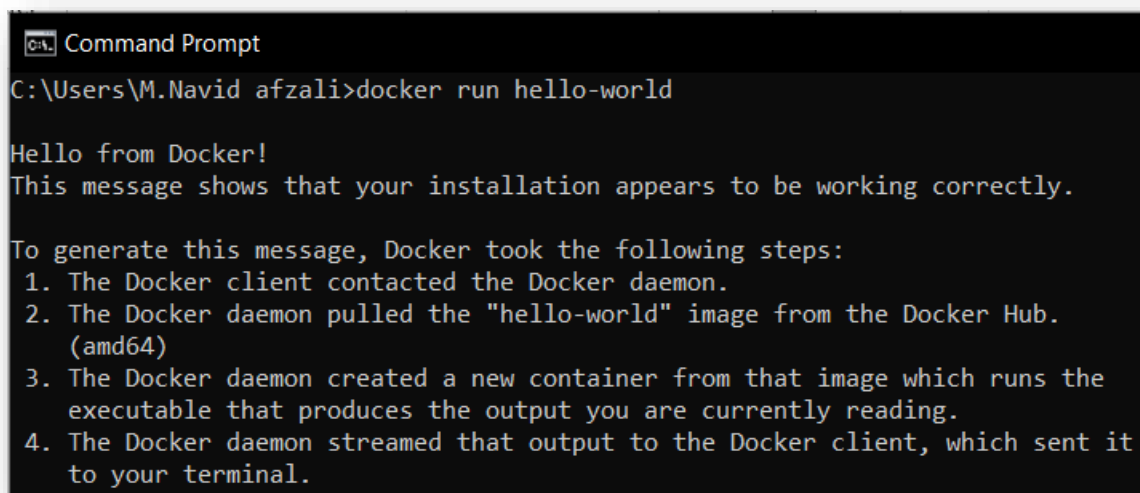
<sup>۲</sup> Tag

## ۵-۷-۱ دستور **docker push** :

برعکس دستور قبلی عمل می‌کند. به طوری که با این دستور می‌توانید تصویر خود را در داخل مخازن مختلف داکر قرار دهید. تنظیمات خاصی ندارد، فقط باید توجه داشت که برای قرار دادن تصویر درون مخزن، باید از قبل وارد<sup>۱</sup> یک مخزن شده باشید.

## ۶-۷-۱ دستور **docker run** :

به وسیله‌ی این دستور می‌توانید از روی تصویرهای موجود در سرویس داکر، یک نگهدارنده راه‌اندازی کرد. دارای تنظیمات بسیار زیادی می‌باشد که توضیح آنها از حوصله‌ی این پروژه خارج است. (دستورات مهم در ادامه پروژه شرح داده خواهند شد)



```
Command Prompt
C:\Users\M.Navid afzali>docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.
```

شکل ۱-۱۰ - دستور *docker run*

## ۷-۷-۱ دستور **docker build** :

به وسیله‌ی این دستور، قادر خواهیم بود از روی داکرفایل<sup>۲</sup>، تصویر داکر ایجاد کنیم. فرآیند ساخت<sup>۳</sup> به این صورت است که از روی یک تصویر پایه<sup>۴</sup> مابقی تغییرات در آن داده می‌شود تا تصویر مد نظر شما ایجاد شود. نکته این که در این مسیر ممکن است چندین تصویر ایجاد شود تا به تصویر نهایی

<sup>۱</sup> login

<sup>۲</sup> Dockerfile

<sup>۳</sup> Build

<sup>۴</sup> Base image

برسیم. این قابلیت لایه‌ای بودن بسیار به کم حجم شدن تصویر نهایی می‌تواند کمک کند. ممکن است در لایه‌ی آخر تصویر شما مشکلی رخ دهد که مجبور باشید بعد از اصلاح آن از ابتدا فرآیند ساخت را انجام دهید. قابلیت کش<sup>۱</sup> که از همین روش لایه‌ای بودن تصویرها فراهم شده است به شما کمک می‌کند که تمام لایه‌های درست و صحیح را مجدد دریافت نکنید و تنها از همان لایه‌ای که خطا داشته است ادامه دهید.

### ۱-۷-۸ دستور **docker commit** :

به کمک این دستور، می‌توانیم از نگهدارنده‌ی در حال اجرا، یک تصویر ایجاد کنیم؛ این دستور زمانی موثر است که تغییری که در نگهدارنده ایجاد کرده‌ایم، مهم است و در آینده بارها به آن نیاز می‌شود. پس مطلوب است که این تغییرات در تصویر جدیدی ثبت شود.

دستوراتی که تاکنون ذکر شدند دستورات اصلی داکر هستند. داکر شامل دستورات زیادی است که مابقی آن‌ها دستورات پیشرفته داکر محسوب می‌شوند و فراتر از این پروژه هستند. بنابراین به معرفی این دسته از دستورات نمی‌پردازیم. همچنین داکر به دلیل تحریم به صورت مستقیم قابل استفاده نمی‌باشد [۱۰].

فصل اول در این جا به پایان می‌رسد. در فصل دوم به معرفی هوش مصنوعی و توضیحاتی پیرامون یادگیری ماشین و یادگیری عمیق و کاربرد داکر در آنها خواهیم پرداخت.

---

<sup>۱</sup> Cache



## فصل دوم

# داکر و یادگیری عمیق

## ۱-۲ هوش مصنوعی چیست؟

هوش مصنوعی به هوشمندی گفته می شود که از ماشین ها حاصل می شود. به عبارتی، هر ماشینی که می تواند محیط را درک کند و فعالیت های وابسته به فکر و ذهن انسان را به خوبی انجام دهد. [۱۱] آلن تورینگ یکی از تاثیرگذارترین افراد در حوزه هوش مصنوعی است و بسیاری مقاله ای که او در سال ۱۹۵۰ منتشر کرد را تولد هوش مصنوعی می دانند. او همچنین تست تورینگ<sup>۱</sup> را پیشنهاد داد که معیاری برای تشخیص هوشمندی ماشین است.

یکی از اولین پروژه های عملی هوش مصنوعی، ایده گرفتن از مفهوم شبکه عصبی و نورون ها بود. دو نفر از دانشجویان دانشگاه هاروارد در سال ۱۹۵۰ اولین شبکه عصبی مصنوعی را که شامل ۴۰ نورون بود ساختند. نورون ها واحدهایی یک شکل در مغز هستند که پیام های عصبی را منتقل می کنند. جالب است بدانید شبکه های عصبی مصنوعی جدید که از آن ها برای طبقه بندی تصاویر استفاده می شود دارای هزاران نورون هستند!

باخت کاسپاروف، قهرمان شطرنج جهان، از سیستم *deep blue* باعث شد تا هوش مصنوعی در کانون توجه قرار گیرد. این سیستم توسط *IBM* طراحی شده بود. در اولین مسابقه در سال ۱۹۹۶ کاسپاروف پیروز شد اما در رقابت بعدی که در سال ۱۹۹۷ اتفاق افتاد، *deep blue* توانست کاسپاروف را شکست دهد.

این حوزه دارای شاخه هایی است که در زمینه های مختلف نیز مورد استفاده قرار می گیرند. در ادامه ما یادگیری ماشین را به عنوان زیرمجموعه ای از هوش مصنوعی بررسی می کنیم.

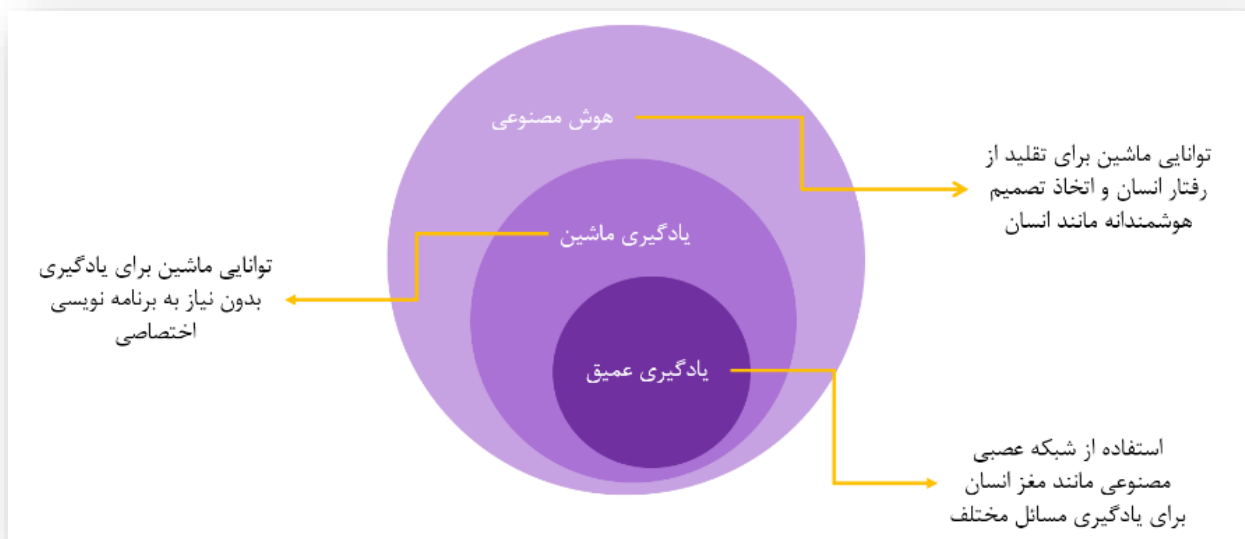
---

<sup>۱</sup> این آزمون در مقاله معروف تورینگ دستگاه محاسباتی و هوشمندی مطرح شد و هدف این تست بررسی هوشمندی ماشین بود. در این آزمون شخص *C* به شکلی غیر مستقیم با شخص *B* و یک ماشین (*A*) در ارتباط است و نمی داند کدامشان ماشین است. او از آنها سوالاتی می پرسد و اگر از روی جواب ها نتواند شخص *B* را از ماشین تشخیص دهد، می گوییم که ماشین تست تورینگ را قبول شده است.

## ۲-۲ یادگیری ماشین<sup>۱</sup>

یادگیری ماشین بعد از هوش مصنوعی موضوعیت پیدا کرد و آن را می‌توان زیرمجموعه هوش مصنوعی در نظر گرفت. یادگیری ماشین ویژگی‌هایی دارد که آن را از دیگر روش‌های هوش مصنوعی متمایز می‌کند.

یکی از مهم‌ترین ویژگی‌های یادگیری ماشین، یادگیری با استفاده از داده و بدون استفاده از دستورات مستقیم است. در انواع دیگر هوش مصنوعی ممکن است شاهد نوعی از هوشمندی باشیم که از داده‌ها استفاده نمی‌کند. ویژگی دیگر یادگیری ماشین، تغییر خود با دیدن داده‌های جدید است؛ یعنی همانند انسان با کسب تجربیات جدید رفتار خود را تغییر می‌دهد. [۷]



شکل ۲-۱ نمودار ون مربوط به هوش مصنوعی، یادگیری ماشین و یادگیری عمیق

در یادگیری ماشین به طراحی ماشین‌هایی پرداخته می‌شود که با توجه به دیتاهای داده شده به آن ماشین و تجربیات ماشین، بیاموزند. به عبارتی، یادگیری ماشین کامپیوترها را قادر می‌سازد بدون این که صریحاً برنامه‌ریزی شده باشند، به روشی خودآموز تبدیل شوند.

<sup>۱</sup> Machine learning

"یک برنامه یادگیرنده برنامه رایانه‌ای است که به آن گفته شده تا از تجربه E مطابق با برخی وظایف T، و کارایی عملکرد P برای وظیفه T که توسط P سنجیده می‌شود، یاد بگیرد که تجربه E را بهبود ببخشد." (تام میشل)

یافتن الگوها در داده‌های موجود در سیاره زمین، تنها برای مغز انسان ممکن است. اما هنگامی که حجم داده‌ها بسیار زیاد می‌شود و زمان لازم برای انجام محاسبات افزایش می‌یابد، نیاز به یادگیری ماشین به عنوان علمی مطرح می‌شود که به افراد در کار با داده‌های انبوه در حداقل زمان کمک می‌کند [۶]. امروزه یادگیری ماشینی در بسیاری زمینه‌ها از جمله مهندسی، کسب و کار، زبان‌شناسی و پزشکی کاربرد دارد.

استفاده از یادگیری ماشین به دلایل زیر خیلی آسان‌تر و فراگیرتر شده است:

- افزایش بی حد و حصر داده‌ها
- ذخیره سازی مقرون به صرفه داده‌ها
- افزایش قدرت پردازنده‌ها و کاهش هزینه‌های آن‌ها

در ادامه به توضیح مختصر یادگیری عمیق، یکی از مهم ترین زیرمجموعه‌های یادگیری ماشین می‌پردازیم:

## ۲-۳ یادگیری عمیق

شاید در مورد یادگیری عمیق هم چیزهایی شنیده باشید. یادگیری عمیق عمری بسیار کوتاه‌تر از یادگیری ماشین دارد. استفاده از شبکه‌های عصبی مصنوعی یکی از انواع روش‌های یادگیری ماشین است که از سال‌ها پیش در حال گسترش است. این شبکه‌ها از ساختار ذهن انسان الگوبرداری شده‌اند و ساختاری لایه لایه دارند. با پیشرفت تکنولوژی و افزایش توان محاسباتی کامپیوترها، مهندسان توانستند تعداد لایه‌های این شبکه‌ها را افزایش دهند و این شروعی برای یادگیری عمیق بود. یادگیری عمیق می‌تواند الگوهای پیچیده‌تری را در داده پیدا کند و امروزه به یکی از جذاب‌ترین بخش‌های یادگیری ماشین تبدیل شده است. [۲]

یادگیری عمیق اشاره به مجموعه ای از الگوریتم‌های یادگیری ماشین دارد؛ که معمولاً مبتنی بر شبکه‌های عصبی مصنوعی اند و تلاش دارند تا انتزاعات سطح بالای موجود در داده‌ها را مدل نمایند. یادگیری عمیق دقت بالاتری نسبت به سایر مدل‌های یادگیری ماشین دارا می‌باشد.

یادگیری عمیق کاربردهای زیادی دارد همچون : بازسازی تصاویر، حذف نویز در تصاویر، تشخیص گفتار، تبدیل متن به گفتار و ...

## ۲-۴ تفاوت های یادگیری ماشین و یادگیری عمیق [۲]

- استخراج ویژگی‌ها : استخراج ویژگی‌ها در یادگیری ماشین به صورت دستی انجام می‌شود؛ در حالی که در یادگیری عمیق به صورت خودکار انجام می‌پذیرد.
- دقت: یادگیری عمیق دقت بالاتری نسبت به یادگیری ماشین دارد.
- زمان آموزش : یادگیری ماشین زمان کوتاه‌تری برای آموزش نیاز دارد؛ در حالی که در یادگیری عمیق این زمان طولانی‌تر است.
- نیازمندی داده : یادگیری عمیق نیازمند داده‌های زیاد و کلان است؛ یادگیری ماشین با داده‌های کم نیز قابل آموزش است.
- شکل پیاده‌سازی : یادگیری عمیق از شبکه‌های عصبی استفاده می‌کند؛ در حالی که یادگیری ماشین از الگوریتم‌هایی مانند خوشه‌بندی و رگرسیون بهره می‌برد.

## ۲-۵ شبکه های عصبی<sup>۱</sup>

وقتی به یادگیری عمیق که حیطه‌ای از یادگیری ماشین است، اشاره می‌شود، احتمالاً پای شبکه‌های عصبی در میان است. شبکه‌های عصبی از مغز ما الگو گرفته‌اند. گره‌آهایی وجود دارند که لایه‌ها را در شبکه تشکیل می‌دهند و دقیقاً مانند نورون‌های مغز ما، نواحی مختلف را به هم متصل می‌کنند. به ورودی‌های گره‌ها در یک لایه، وزنی اختصاص می‌یابد که تأثیری را که پارامتر بر نتیجه پیش‌بینی

---

<sup>۱</sup> Neural Network

<sup>۲</sup> node

کلی دارد، تغییر می‌دهد. از آنجا که وزن‌ها به پیوندهای بین نودها اختصاص داده می‌شوند، ممکن است هر گره تحت تأثیر وزن‌های مختلف قرار گیرد [6]

## ۲-۵-۱ شبکه‌های عصبی کانولوشن<sup>۱</sup>

شبکه‌های عصبی کانولوشن، دسته‌ای از شبکه‌های عصبی هستند که به طول معمول برای تحلیل تصاویر و گفتار در یادگیری ماشین و به طور جزئی‌تر در یادگیری عمیق استفاده می‌شوند. این نوع شبکه دارای چندین لایه است که داده‌ها را پردازش کرده و ویژگی‌های مهم آنها را استخراج می‌کنند. فیلترهای این شبکه بر اساس روش‌های اکتشافی<sup>۲</sup> بدست می‌آیند. شبکه‌های عصبی کانولوشن می‌توانند مهم‌ترین ویژگی فیلترها را بیاموزند و چون به پارامترهای زیادی احتیاج نیست، صرفه‌جویی زیادی در وقت و عملیات آزمون و خطا صورت می‌گیرد.

تا زمانی که با تصاویر با ابعاد بالا که هزاران پیکسل دارند کار نکنید، این صرفه‌جویی چندان به چشم نمی‌آید. هدف اصلی این الگوریتم‌ها این است که با حفظ ویژگی‌هایی که برای فهم آنچه داده‌ها نشان می‌دهند مهم هستند، داده‌ها را به فرم‌هایی که پردازش آن‌ها آسان‌تر است، درآورد.

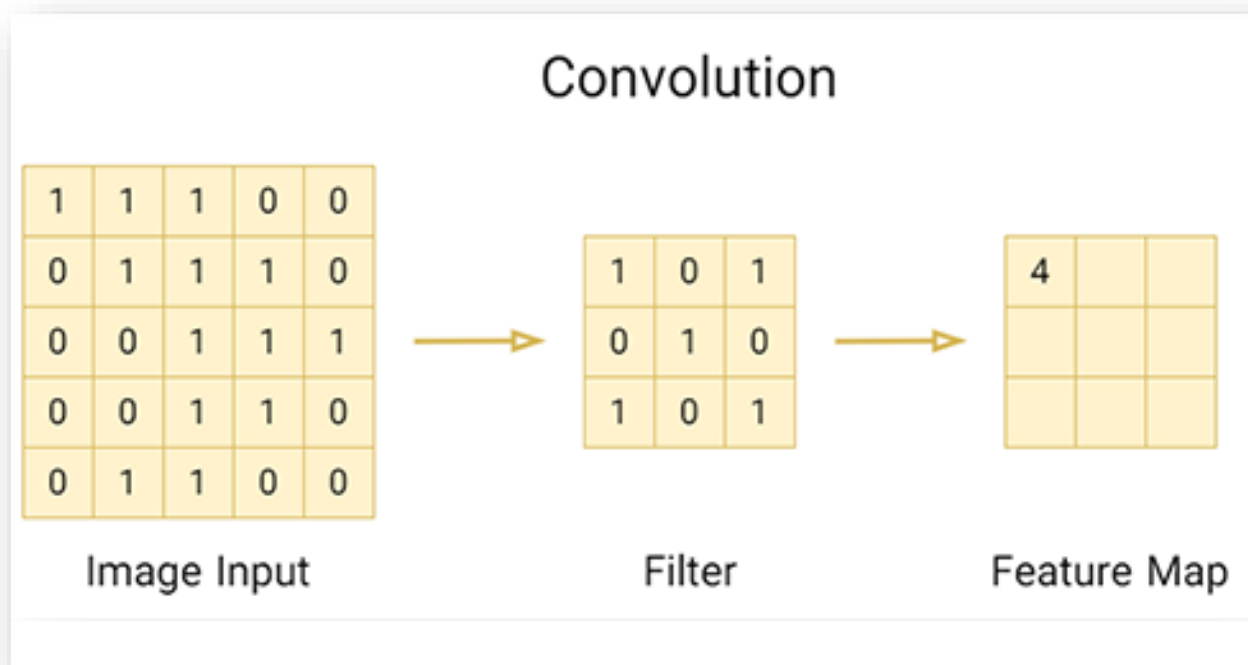
شبکه‌های عصبی کانولوشن بر خلاف شبکه‌های عصبی معمولی، به جای ضرب ماتریس از کانولوشن استفاده می‌کند. این شبکه‌ها با اعمال فیلتر روی داده‌های ورودی شما پردازش خود را انجام می‌دهند. چیزی که آن‌ها را بسیار خاص می‌کند، این است که شبکه‌های عصبی کانولوشن می‌توانند فیلترها را هم‌زمان با فرایند آموزش، تنظیم کنند. بنابراین، حتی وقتی مجموعه داده‌های عظیمی مانند تصاویر داشته باشید، نتایج به‌خوبی و در لحظه دقیق‌تر می‌شوند.

از آنجا که می‌توان فیلترها را برای آموزش بهتر شبکه‌های عصبی کانولوشن به‌روزرسانی کرد، نیاز به فیلترهای دستی از بین می‌رود و این انعطاف‌پذیری بیشتری در تعداد و ارتباط فیلترهایی که بر روی مجموعه داده‌ها اعمال می‌شوند، به ما می‌دهد. با استفاده از این الگوریتم، می‌توانیم روی مسائل پیچیده‌تری مانند تشخیص چهره کار کنیم.

---

<sup>۱</sup> Convolutional Neural Network (CNN)

<sup>۲</sup> Heuristic

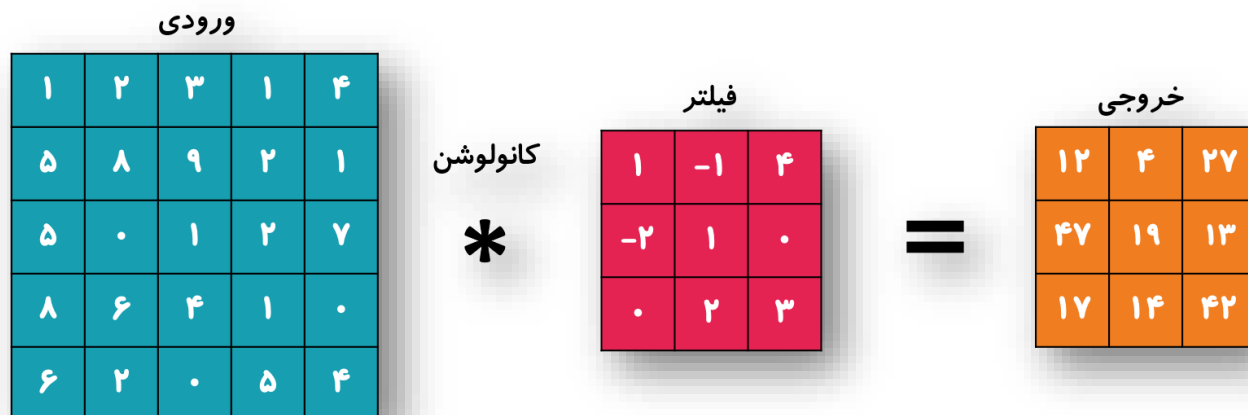


شکل ۲-۲ عمل کانولوشن

در عمل کانولوشن چند پارامتر اساسی وجود دارد که عبارتند از:

- ماتریس ورودی
- فیلتر کانولوشن
- عملگر کانولوشن
- ویژگی خروجی کانولوشن

کارکرد اصلی کانولوشن به این صورت است که عملگر کانولوشن، فیلتر کانولوشنی را برمی‌دارد و روی ماتریس ورودی می‌لغزاند. (فیلتر روی تصویر حرکت می‌کند). [۸]



شکل ۲-۳ پارامتر های اصلی کانولوشن

اعدادی که در ماتریس خروجی ذخیره می‌شوند، تابعی از ورودی و فیلتر هستند. فیلتر به دنبال پیدا کردن نواحی مشابه خود در تصویر است و هر جایی ناحیه مشابه خود را پیدا کرد بلند فریاد می‌زند (عدد بزرگ). پس کانولوشن منجر به یافتن الگوهای خاص در تصویر با توجه به فیلتر می‌شود. اعداد موجود در فیلتر بسیار مهم هستند.

## ۲-۸ انواع شبکه های عصبی کانولوشن

- شبکه عصبی کانولوشن یک‌بعدی: در این حالت، کرنل CNN در یک جهت حرکت می‌کند. CNN های یک‌بعدی معمولاً روی داده‌های سری زمانی استفاده می‌شوند.
- شبکه عصبی کانولوشن دوبعدی: در این نوع از CNN، کرنل‌ها در دو جهت حرکت می‌کنند. CNN های دوبعدی در برچسب‌گذاری و پردازش تصویر کاربرد دارند.
- شبکه عصبی کانولوشن سه‌بعدی: این نوع CNN دارای کرنلی است که در سه جهت حرکت می‌کند. محققان از این نوع CNN در تصاویر سه‌بعدی مانند سی‌تی‌اسکن و MRI استفاده می‌کنند.

## ۲-۹ کاربرد های شبکه های عصبی کانولوشن

- تشخیص دست‌خط‌های مختلف



- تشخیص تصاویر با پیش‌پردازش کم
- استفاده در سرویس‌های پستی برای خواندن کدپستی
- استفاده در بانکداری برای خواندن ارقام در چک
- کاربردهای بینایی کامپیوتر<sup>۱</sup>
- و ...

## ۲-۱۰ چرا از نگهدارنده‌های داکر در یادگیری عمیق استفاده می‌کنیم؟

اولین چیزی که قبل از صحبت در مورد ساخت نگهدارنده باید درک کرد، مفهوم میکروسرویس<sup>۲</sup> است. اگر یک برنامه بزرگ به سرویس‌های کوچک‌تر تقسیم شود، هر یک از آن سرویس‌ها یا فرآیندهای کوچک را می‌توان سرویس‌های میکرو نامید که از طریق یک شبکه با یکدیگر ارتباط برقرار می‌کنند. رویکرد میکروسرویس‌ها برعکس رویکرد یکپارچه است که مقیاس‌بندی آن دشوار است. اگر یک ویژگی خاص دارای برخی مشکلات یا خرابی باشد، همه ویژگی‌های دیگر مشابه آن مشکل را تجربه خواهند کرد. مثال دیگر این است که وقتی تقاضا برای یک ویژگی خاص به طور جدی در حال افزایش است، ما مجبور می‌شویم منابعی مانند سخت‌افزار را نه تنها برای این ویژگی خاص بلکه برای کل برنامه افزایش دهیم و هزینه‌های اضافی را ایجاد کنیم که ضروری نیستند. این هزینه می‌تواند به حداقل برسد اگر رویکرد خدمات خرد با تقسیم برنامه به گروهی از خدمات کوچک‌تر در نظر گرفته شود. هر سرویس یا ویژگی‌های برنامه به گونه‌ای ایزوله شده است که می‌توانیم بدون تأثیر بر سایر ویژگی‌های برنامه، آن را مقیاس‌بندی یا بروزرسانی کنیم. برای وارد کردن یادگیری ماشین به چرخه تولید، بیایید در نظر بگیریم که برنامه باید به سرویس‌های خرد کوچک‌تری مانند مصرف، آماده‌سازی، ترکیب، جداسازی، آموزش، ارزیابی، استنتاج، پس‌پردازش و نظارت تقسیم شود. [۹]

<sup>۱</sup> Computer vision

<sup>۲</sup> Micro-service

## ۲-۱۰-۱- ساخت نگهدارنده

معماری میکرو سرویس نیز دارای اشکالاتی است. هنگامی که برنامه یادگیری ماشین خود را در یک سرور توسعه می‌دهید، به همان تعداد ماشین مجازی نیاز خواهید داشت که میکروسرویس حاوی وابستگی است. هر ماشین مجازی به یک سیستم‌عامل، کتابخانه‌ها و باینری‌ها نیاز دارد و منابع سخت‌افزاری بیشتری مانند پردازنده، حافظه و فضای دیسک را مصرف می‌کند، حتی اگر میکرو سرویس واقعاً در حال اجرا نباشد. به همین دلیل است که داکر وارد می‌شود. اگر یک نگهدارنده در حال اجرا نباشد، منابع باقی مانده به منابع مشترک تبدیل می‌شوند و برای سایر نگهدارنده‌ها قابل دسترسی هستند. شما نیازی به اضافه کردن سیستم‌عامل در یک ظرف ندارید. بیا یک راه حل کامل متشکل از برنامه‌های کاربردی ۱ و ۲ (مثلاً APP<sup>۱</sup> و APP<sup>۲</sup>) را در نظر بگیریم. اگر می‌خواهید APP<sup>۱</sup> را کوچک کنید یا برنامه‌های کاربردی دیگری را اضافه کنید، می‌توانید با استفاده از منابع در دسترس از ماشین‌های مجازی به جای نگهدارنده‌ها محدود شوید. اگر تصمیم دارید فقط APP<sup>۱</sup> و نه APP<sup>۲</sup> را کوچک کنید (فقط یک مورد را نگه دارید)، APP<sup>۲</sup> تبدیل به سهمی از همه فرآیندهای نگهدارنده می‌شود.

ایجاد یک مدل یادگیری ماشینی که در رایانه ما کار می‌کند واقعاً پیچیده نیست. اما وقتی برای مثال با مشتری کار می‌کنید که می‌خواهد از مدل در مقیاس استفاده کند، مدلی که می‌تواند در همه انواع سرورها در سراسر جهان کار کند، چالش‌برانگیز است. پس از توسعه مدل شما، ممکن است در لپ‌تاپ یا سرور شما به خوبی اجرا شود، اما واقعاً در سیستم‌های دیگر مانند زمانی که مدل را به مرحله تولید یا سرور دیگری منتقل می‌کنید، کار نمی‌کند. بسیاری از چیزها ممکن است رخ دهد مانند مشکلات عملکردی، از کار افتادن برنامه یا بهینه نبودن برنامه. موقعیت چالش برانگیز دیگر این است که مدل یادگیری ماشین ما مطمئناً می‌تواند با یک زبان برنامه‌نویسی منفرد مانند پایتون نوشته شود، اما برنامه مطمئناً نیاز به تعامل با برنامه‌های کاربردی دیگر نوشته شده در زبان‌های برنامه‌نویسی دیگر برای جذب داده، آماده سازی داده، فرانت‌اند و غیره دارد. داکر اجازه می‌دهد تا همه این تعاملات را بهتر مدیریت کنید زیرا هر میکرو سرویس را می‌توان به زبانی متفاوت نوشت که امکان مقیاس‌پذیری و افزودن یا حذف آسان سرویس‌های مستقل را فراهم می‌کند. داکر تکرارپذیری، قابلیت حمل، استقرار آسان، بروزرسانی، سبکی و سادگی را به ارمغان می‌آورد. [۹]

فصل دوم به پایان رسید. فصل بعدی پیرامون پیاده‌سازی پروژه‌ها به وسیله ابزار داکر است. در فصل بعدی، چند پروژه یادگیری ماشین و یادگیری عمیق را با استفاده از ابزار داکر پیاده‌سازی می‌کنیم و برای پروژه‌های خود تصویر ساخته و از آن در پیاده‌سازی پروژه خود استفاده می‌کنیم.

## فصل سوم

### پیاده سازی یادگیری

### ماشین/عمیق با استفاده از داکر

## ۳-۱ پروژه یادگیری ماشین

### ۳-۱-۱ مقدمات لازم برای ساخت تصویر داکر

تصویر<sup>۱</sup> بر روی مجموعه‌ای از کدها و متعلقات ساخته می‌شود. بنابراین در ابتدا باید پروژه‌ای که قصد ساخت تصویر آن را داریم، انتخاب و فایل‌ها و اسناد مربوط به پروژه را جمع‌آوری کنیم.

نوار مغزی، الکتروانسفالوگرافی<sup>۲</sup> یا EEG ثبت فعالیت الکتریکی مغز از طریق نصب الکترودهای سطحی بر روی سر و به صورت غیر تهاجمی می‌باشد. به طور کلی، در یک سیستم EEG، اثر الکتریکی فعالیت نورون‌های مغز از طریق الکترودهای نصب شده بر روی سر به دستگاه انتقال داده شده و پس از تقویت و حذف نویز به صورت سیگنال زمانی ثبت و نمایش داده می‌شود. سیگنال ثبت شده می‌تواند مستقیماً و یا پس از پردازش کامپیوتری توسط پزشک و یا متخصص علوم اعصاب مورد تحلیل قرار بگیرد [۱۲].

پروژه فرضی ما حاوی اسناد زیر می‌باشد:

- فایل `train.py`: یک فایل به زبان برنامه‌نویسی پایتون است که داده‌های EEG<sup>۳</sup> را از یک فایل به نام `train.csv` دریافت و نرمال‌سازی<sup>۴</sup> می‌کند. همچنین دو مدل را برای طبقه‌بندی داده‌ها آموزش می‌دهد: تحلیل تشخیصی خطی و مدل چند لایه شبکه‌های عصبی.
- فایل `inference.py`: این فایل هم که به زبان برنامه‌نویسی پایتون نوشته شده، برای استنتاج دسته‌ای با بارگذاری دو مدلی که در فایل قبلی ایجاد و آموزش داده شدند، فراخوانی می‌شود. این برنامه داده‌های EEG جدید را که از یک دیتاست می‌آید، نرمال‌سازی می‌کند. همچنین استنتاج روی مجموعه داده انجام داده و دقت طبقه‌بندی و پیش‌بینی‌ها را چاپ می‌کند.
- فایل `train.csv`: دیتاستی که توسط فایل `train.py` دریافت و آموزش داده می‌شود.

---

<sup>۱</sup> Image

<sup>۲</sup> Electroencephalography

<sup>۳</sup> EEG نوسانات ولتاژ ناشی از جریان یونی در نورون‌های مغز را اندازه‌گیری می‌کند. از نظر بالینی، EEG به ثبت فعالیت الکتریکی خود به خودی مغز در طی یک دوره زمانی اشاره دارد، همانطور که از چندین الکتروود قرار داده شده بر روی پوست سر ثبت می‌شود. (ویکی پدیا)

<sup>۴</sup> Normalize

- فایل test.csv : دیتاستی که توسط inference.py نرمال سازی می شود.

### ۳-۱-۲ شروع ساخت تصویر

حال ساخت تصویر را آغاز می کنیم. ساخت تصویر بر روی داکر فایل صورت می گیرد. برای ساخت تصویر ابتدا یک فایل تحت نام داکر فایل در مسیر پروژه ایجاد کرده و آن را با یک ویرایش گر متن<sup>۱</sup> باز می کنیم. حال دستورات لازم در داکر فایل نوشته می شود تا تصویر طبق این دستورات ساخته شود.

قالب یک داکر فایل به شکل زیر است:

```
Dockerfile > ...
1 FROM jupyter/scipy-notebook
2
3 RUN mkdir my-model
4 ENV MODEL_DIR=/home/jovyan/my-model
5 ENV MODEL_FILE_LDA=clf_lda.joblib
6 ENV MODEL_FILE_NN=clf_nn.joblib
7
8 RUN pip install joblib
9
10 COPY train.csv ./train.csv
11 COPY test.csv ./test.csv
12
13 COPY train.py ./train.py
14 COPY inference.py ./inference.py
15
16 RUN python3 train.py
```

شکل ۳-۱ داکر فایل

- خط ۱ : شروع ساخت تصویر از یک تصویر پایه : ما از تصویر jupyter/scipy-notebook استفاده کرده ایم. این تصویر از مجموعه ای از تصویر های آماده ای اجراست که شامل برنامه های کاربردی jupyter و ابزارهای محاسباتی تعاملی می باشد.
- خط ۳ : ساخت یک دایرکتوری<sup>۲</sup> به نام my-model برای استفاده در ادامه کار.

<sup>۱</sup> Text editor

<sup>۲</sup> Directory-Repository

- خط ۴-۵-۶ : برای آینده توسعه ی پروژه خود، می توان متغیرهای محیطی را از ابتدا، فقط یک بار در زمان ساخت، برای تداوم مدل آموزش دیده تنظیم کرد و شاید داده ها یا ابرداده های<sup>۱</sup> اضافی را به یک مکان خاص اضافه کرد. مزیت تنظیم متغیرهای محیطی این است که کار خود را بهتر با دیگران در یک ساختار دایرکتوری توافق شده به اشتراک بگذارید.
  - خط ۸ : اجرای دستور به منظور نصب پکیج `joblib` . `Joblib` مجموعه ای از ابزارها برای ارائه خط لوله<sup>۲</sup> سبک در پایتون است. `Joblib` به گونه ای بهینه سازی شده است که به ویژه در داده های بزرگ سریع و قوی باشد و دارای بهینه سازی های خاصی برای آرایه های `numpy` است. ما باید `joblib` را نصب کنیم تا سریال سازی<sup>۳</sup> و سریال زدایی<sup>۴</sup> مدل آموزش دیده مان را امکان پذیر کنیم.
  - خطوط ۱۰ تا ۱۴ : کپی کردن اسناد و فایل های پروژه و انتقال آن ها به تصویر. این دستورالعمل برای کپی کردن فایل های جدید، از قسمت منابع<sup>۵</sup> به فایل سیستم هایی که در نگهدارنده قرار دارند است.
  - خط ۱۶ : اجرای دستور پایتون بر روی فایل `train.py` جهت نمایش خروجی هنگام اجرای تصویر. ما `train.py` را اجرا می کنیم که مدل های یادگیری ماشین را به عنوان بخشی از فرآیند ساخت تصویر ما متناسب و سریال می کند.
- در ادامه پس از نوشتن داکر فایل، وارد خط فرمان سیستم عامل خود شده و در مسیر فایل های پروژه، با دستور زیر شروع به ساخت تصویر از پروژه یادگیری ماشین خود می کنیم:

```
docker build -t docker-ml-model -f Dockerfile .
```

<sup>۱</sup> Meta data

<sup>۲</sup> Pipeline

<sup>۳</sup> Serialization

<sup>۴</sup> Deserializaiton

<sup>۵</sup> Source

با اجرای این دستور، داکر شروع به ساخت تصویر برای مدل یادگیری ماشین می‌کند و با توجه به دستورات داکر فایل، تصویر پایه و متعلقات آن را در صورت نیاز از مخزن داکر<sup>۱</sup> دریافت می‌کند. خروجی نهایی به شکل زیر است:

```
C:\Program Files\WindowsApps\Microsoft.PowerShell_7.2.6.0_x64_8wekyb3d8bbwe\pwsh.exe
PS C:\Users\M.Navid afzali\Desktop\final proj\test2> docker build -t afzali-cn2 -f Dockerfile .
[+] Building 6.2s (13/13) FINISHED
=> [internal] load build definition from Dockerfile 3.2s
=> => transferring dockerfile: 346B 0.3s
=> [internal] load .dockerignore 2.4s
=> => transferring context: 2B 0.1s
=> [internal] load metadata for docker.io/jupyter/scipy-notebook:latest 0.0s
=> [1/8] FROM docker.io/jupyter/scipy-notebook 0.0s
=> [internal] load build context 0.7s
=> => transferring context: 1.56kB 0.2s
=> CACHED [2/8] RUN mkdir my-model 0.0s
=> CACHED [3/8] RUN pip install joblib 0.0s
=> CACHED [4/8] COPY train.csv ./train.csv 0.0s
=> CACHED [5/8] COPY test.csv ./test.csv 0.0s
=> CACHED [6/8] COPY train.py ./train.py 0.0s
=> CACHED [7/8] COPY inference.py ./inference.py 0.0s
=> CACHED [8/8] RUN python3 train.py 0.0s
=> exporting to image 2.8s
=> => exporting layers 0.0s
=> => writing image sha256:7abed835c8f9918f3392b658e82d306b439c4162ba4c2959f71e71a65d346315 0.1s
=> => naming to docker.io/library/afzali-cn2 0.1s
PS C:\Users\M.Navid afzali\Desktop\final proj\test2>
```

شکل ۳-۲ مراحل ساخت تصویر

حال تصویر ما ساخته شده و در مخزن سیستم عامل میزبان موجود است:

```
C:\Program Files\WindowsApps\Microsoft.PowerShell_7.2.6.0_x64_8wekyb3d8bbwe\pwsh.exe
PS C:\Users\M.Navid afzali\Desktop\final proj\test2> docker images
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
afzali-cn2           latest          7abed835c8f9   2 days ago     2.96GB
afzali-container     latest          7abed835c8f9   2 days ago     2.96GB
redis                latest          2e50d70ba706   2 months ago   117MB
python              latest          0f95b1e38607   2 months ago   920MB
busybox             latest          62aedd01bd85   3 months ago   1.24MB
hello-world         latest          feb5d9fea6a5   11 months ago  13.3kB
PS C:\Users\M.Navid afzali\Desktop\final proj\test2>
```

شکل ۳-۳ تصویر های موجود در مخزن محلی (دریافت شده از مخزن Docker hub)

<sup>۱</sup> Docker Hub



### ۳-۱-۳ اجرای تصویر روی نگهدارنده

حال می‌توانیم استنتاج<sup>۱</sup> خود از مدل‌ها را روی داده‌های جدید یعنی داده‌های موجود در test.csv اجرا کنیم. به صورت زیر:

```
C:\Program Files\WindowsApps\Microsoft.PowerShell_7.2.6.0_x64_8wekyb3d8bbwe\pwsh.exe
PS C:\Users\M.Navid afzali\Desktop\final proj\test2> docker run afzali-cn2 python3 inference.py
Shape of the test data
(1300, 160)
(1300,)
/home/jovyan/my-model/clf_lda.joblib
LDA score and classification:
0.6915384615384615
[ 0  0  0 ... 25 25 25]
NN score and classification:
0.6615384615384615
[ 0  0  0 ... 25 25 24]
PS C:\Users\M.Navid afzali\Desktop\final proj\test2>
```

شکل ۳-۴ اجرای تصویر روی نگهدارنده

همانطور که مشاهده می‌شود تصویر ساخته‌شده روی یک نگهدارنده اجرا شده و خروجی کد inference.py که طبقه بندی<sup>۲</sup> و دقت مدل‌ها را نمایش می‌دهد، به نمایش در آمده است. می‌بینیم که مدل‌ها از دقت نسبتاً خوبی برخوردارند. همچنین مسیر اجرای تصویر را هم مشاهده می‌کنید.

در ادامه یک پروژه یادگیری عمیق را مورد بررسی قرار می‌دهیم و تصویر مربوط به برنامه را ایجاد می‌کنیم.

---

<sup>۱</sup> Inference  
<sup>۲</sup> Classification

## ۲-۳ پروژه یادگیری عمیق

### ۱-۲-۳ مقدمه

در این پروژه یک مدل CNN که تصاویر مربوط به غذاها را دسته‌بندی می‌کند، را در قالب یک سرویس API پیاده‌سازی می‌کنیم.

همانطور که در فصل دوم شرح داده‌شد، شبکه عصبی کانولوشن نوع خاصی از شبکه عصبی با چندین لایه است که داده‌هایی را که آرایش شبکه‌ای دارند، پردازش کرده و سپس ویژگی‌های مهم آن‌ها را استخراج می‌کند. یک مزیت بزرگ استفاده از CNN ها این است که نیازی به انجام پیش‌پردازش زیادی روی تصاویر نیست.

FastAPI یک چهارچوب تحت وب به زبان پایتون است و به وسیله آن برنامه‌های api را با سرعت بالا ایجاد می‌کنند.

پروژه حاوی اسناد زیر است:

- Main.py : فایل اصلی پروژه به زبان پایتون که یادگیری روی مدل را انجام می‌دهد؛ در نهایت روی یک پورت محلی سرویس fastAPI را اجرا می‌کند.
- Requirments.txt : یک فایل متنی است که در آن پکیج‌های مورد نیاز پروژه نوشته شده است. از آن در داکر فایل هنگام دریافت پکیج‌ها به وسیله دستور pip استفاده می‌شود.
- داکر فایل : یک فایل داکر که در آن نحوه ساخت تصویر نوشته شده‌است و تصویر طبق این فایل ساخته می‌شود.

### ۲-۲-۳ شروع ساخت تصویر

محتویات داکر فایل به شکل زیر است (تصویر در صفحه بعد) :

```
Dockerfile M X
Dockerfile > ...
1 FROM python:3.7.3-stretch
2
3 # Make working directories
4 RUN mkdir -p /food-vision-api
5 WORKDIR /food-vision-api
6
7 # Upgrade pip with no cache
8 RUN pip install --no-cache-dir -U pip
9
10 # Copy application requirements file to the created working directory
11 COPY requirements.txt .
12
13 # Install application dependencies from the requirements file
14 RUN pip install -r requirements.txt
15
16 # Copy every file in the source folder to the created working directory
17 COPY . .
18
19 # Run the python application
20 CMD ["python", "main.py"]
```

شکل ۳-۵ محتویات داکر فایل

- خط ۱: تصویر پایه را نام می‌بریم؛ در این پروژه ما از تصویر پایتون نسخه ۳.۷ استفاده کرده ایم. کل پروژه روی این تصویر بنا می‌شود.
- خط ۴ و ۵: مانند پروژه قبل ساخت دایرکتوری برای ادامه کار را انجام می‌دهد.
- خط ۸: یک دستور پایتون است که پکیج pip را بروزرسانی می‌کند.
- خط ۱۱: فایل متنی پروژه یعنی requirements.txt را کپی می‌کند و در دایرکتوری ای که در خطوط قبلی ساختیم قرار می‌دهد (دایرکتوری در نگهدارنده ساخته می‌شود).
- خط ۱۴: دستور پایتون است که با دستور pip محتویات فایل requirements.txt را دریافت می‌کند.
- خط ۱۷: تمام فایل‌های پروژه را کپی می‌کند و در دایرکتوری ساخته شده قرار می‌دهد.

- خط ۲۰: یک دستور اجرایی است که فایل main.py را اجرا می‌کند؛ معادل دستور زیر در خطفرمان است:

```
Python main.py
```

در ادامه پس از نوشتن داکر فایل، وارد خطفرمان یا ترمینال سیستم‌عامل خود شده و در مسیر فایل‌های پروژه، با دستور زیر شروع به ساخت تصویر از پروژه خود می‌کنیم:

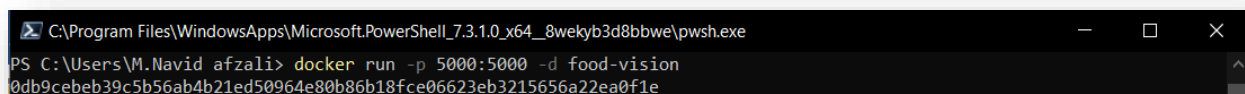
```
docker build -t <your-image-name> .
```

نحوه ساخت همانند شکل ۱۶ است.

پس از ساخت تصویر با دستور زیر آن را در یک نگهدارنده اجرا می‌کنیم:

```
docker run -p ۵۰۰۰:۵۰۰۰ -d <your-image-name>
```

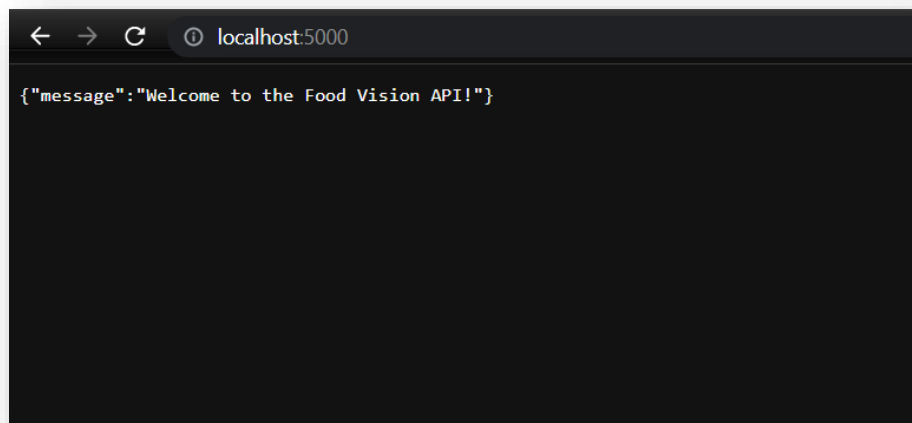
با تنظیمات -p پورت ۵۰۰۰ سیستم خود را به پورت ۵۰۰۰ نگهدارنده متصل می‌کنیم؛ همچنین با تنظیمات -d نگهدارنده را به صورت detached mode اجرا می‌کنیم؛ به این معنی که در پس‌زمینه اجرا می‌شود. در صورت موفقیت‌آمیز بودن اجرا، شناسه نگهدارنده در خطفرمان به عنوان خروجی برمیگردد.



```
C:\Program Files\WindowsApps\Microsoft.PowerShell_7.3.1.0_x64_8wekyb3d8bbwe\pwsh.exe
PS C:\Users\M.Navid afzali> docker run -p 5000:5000 -d food-vision
0db9cebeb39c5b56ab4b21ed50964e80b86b18fce06623eb3215656a22ea0f1e
```

شکل ۳-۶ اجرای تصویر ۲

اکنون با باز کردن صفحه <http://localhost:۵۰۰۰> در رایانه خود می‌توانیم مشاهده کنیم که آیا نگهدارنده با موفقیت اجرا شده است یا خیر:



شکل ۷-۳ اجرای برنامه با نگهدارنده

با وارد کردن دستور <http://localhost:5000/docs> به صفحه fastAPI هدایت می‌شویم؛ در آن جا می‌توانیم با متد POST یک تصویر ارسال کنیم و نتیجه طبقه‌بندی تصویر را در قالب JSON تماشا کنیم.

برای مثال ما از تصویر زیر استفاده می‌کنیم:



شکل ۸-۳ تصویر نمونه

خروجی fastAPI پس از تزریق یک ورودی در قالب تصویر، به صورت زیر می‌باشد:

Responses

Curl

```
curl -X 'POST' \
  'http://localhost:5000/net/image/prediction/?image_link=https%3A%2Ftallypress.com%2Fwp-content%2Fuploads%2F2017%2F09%2Flime-chili-chicken-salad-with-fork.jpg' \
  -H 'accept: application/json' \
  -d ''
```

Request URL

```
http://localhost:5000/net/image/prediction/?image_link=https%3A%2Ftallypress.com%2Fwp-content%2Fuploads%2F2017%2F09%2Flime-chili-chicken-salad-with-fork.jpg
```

Server response

Code	Details
200	<p>Response body</p> <pre>{   "model-prediction": "caesar salad",   "model-prediction-confidence-score": 2.64 }</pre> <p>Response headers</p> <pre>access-control-allow-credentials: true access-control-allow-origin: http://localhost:5000 content-length: 76 content-type: application/json date: Wed, 28 Dec 2022 12:31:34 GMT server: uvicorn vary: Origin</pre>

شکل ۳-۹ خروجی نگهدارنده تحت وب

بدین ترتیب موفق به ساخت یک تصویر از پروژه یادگیری عمیق شدیم. حال می‌توانیم پروژه خود را در ریپازیتوری عمومی خود در [hub.docker.com](https://hub.docker.com) با سایرین به اشتراک بگذاریم.

### ۳-۳ جمع‌بندی و نتیجه‌گیری

همان‌طور که در بخش‌های قبل مشاهده کردید، ساختن تصویر برای یک پروژه یادگیری ماشین و یادگیری عمیق با نوشتن داکر فایل برای آن و اجرای دستور `docker build` امکان‌پذیر است.

حال ما با پوش کردن تصویر خود در مخزن `Docker hub` می‌توانیم پروژه یادگیری ماشین/عمیق خود را در دسترس عموم قرار دهیم تا سایر کاربران با دریافت آن تصویر و اجرای آن روی یک نگهدارنده در سرور میزبان خود، بتوانند بدون آشنایی با کتابخانه‌ها و متعلقات آن پروژه، پروژه را اجرا و نتیجه آن را مشاهده نمایند. فرضاً پروژه با زبان پایتون ۳ نوشته شده و فرد دریافت‌کننده تصویر روی سیستم خود پایتون ۲ را دارد. با داکر دیگر مشکلی برای اجرا وجود نخواهد داشت و دریافت‌کننده نیازی به تغییر نسخه پایتون خود نیز ندارد؛ همچنین به تبع رفع این مشکل، فرد نیازی به استفاده از ماشین مجازی نیز ندارد و در نتیجه سرعت و فضای سیستم را از دست نخواهد داد.

در این پروژه دیدیم که علاوه بر سایر برنامه‌ها، برای برنامه‌های یادگیری ماشین نیز می‌توان ایمج داکر ساخت و از آن استفاده کرد. این تنها یک مثال از ساخت تصویر و اجرای آن برای پروژه یادگیری ماشین است. برای سایر برنامه‌های یادگیری ماشین نیز به همین ترتیب عمل می‌شود. استفاده از تصویر داکر برای پروژه‌های یادگیری ماشین، باعث راحتی، افزایش سرعت و اتلاف وقت بسیار کمتر توسعه‌دهندگان در استفاده از آن در سایر سرورها و سیستم‌های خارجی است.

## سخن پایانی

در پایان تشکر می‌کنم از استاد محترم دکتر راستگو که بنده را در نوشتن این پایان‌نامه کمک کردند و باعث شدند که بنده با این ابزار کاربردی و خوب و همچنین با مفاهیم جالب یادگیری عمیق آشنا شوم. موفقیت‌های روز افزون را برای ایشان از ایزد منان خواهانم.

و تشکر می‌کنم از همه شما خوانندگان گرامی بابت صبر و شکیبایی که در خواندن این پایان‌نامه از خود نشان دادید.



## منابع

- [۱]. Docker (n d) . Docker Documentation. <https://docs.docker.com>
- [۲]. کیانی، کوروش ؛ راستگو، راضیه ؛ "لذت آموزش گام به گام یادگیری عمیق با مثال عددی(کتاب دوم: شبکه عصبی کانولوشن)"؛ سمnan ؛ دانشگاه سمnan ؛ بهار ۱۴۰۰
- [۳]. Docker (n d) . Docker:Accelerated,Containerized Application Development.  
<https://docker.com/>  
[۴]. احمد رفیعی (بی تا). آموزش داکر و پلتفرم به زبان فارسی. <https://dockerme.ir/>
- [۵]. <https://youtube.com/>
- [۶]. فرادرس (آذر ۱۴۰۰) . آموزش داکر به زبان ساده- از صفر تا صد.
- [۷]. کوئرا (آبان ۱۴۰۰) . شبکه عصبی کانولوشن چیست؟. <https://quera.org/blog/what-is-a-convolutional-neural-network/>
- [۸]. هوسم (اردیبهشت ۱۴۰۰) . شبکه عصبی کانولوشن-<https://howsam.org/convolutional-neural-network/>
- [۹]. Xavier vazques (Apr ۲۰۲۱) . why use Docker containers for machine learning? . <https://towardsdatascience.com/why-using-docker-for-machine-learning-۷c9۲۷ceb6c۴>

[۱۰]. ویرگول (۱۳۹۷) . گریز از تحریم داکر با چند روش .

<https://virgool.io/DockerMe/%DA%AF%D%A%B%D%DB%AC%D%B%D%A%D%B%D%AA%D%AD%D%B%D%DB%AC%D%8D%A%D%B%D%AF%D%A%D%DA%A%D%B%D%A%D%A%D%A%D%8D%8D%D%AF-%D%B%D%8D%D%B-zczoxibqnyk>

[۱۱]. <https://wikipedia.org> . هوش مصنوعی . Wikipedia (n d) .

[۱۲]. شرکت طراحی نگاراندیشان (بی‌تا) . نوار مغزی چیست؟ . <https://negand.com/fa/۳۵-۲/> .

پیش‌نویس: منظور از (n d) همان بی‌تا یا "بدون تاریخ" می‌باشد.

# Abstract

Nowadays, “Docker” is mentioned as one of the important and practical tools in the field of programming. Docker as a tool for building and running programs in container environments has significantly increased the speed and productivity of these programs. With Docker, we will no longer worry about the difference between the versions of the program's components and its libraries. Currently, this tool can be used in the development of programs related to artificial intelligence and machine/deep learning, and developers in this field can share their programs with public or their colleagues without worry through Docker.

Also Docker is so helpful for these projects because machine/deep learning projects have many libraries with several versions. On the other hand, they're not same in most devices, so in these cases Docker help us! In this project, we are going to build and implement a container (explained in the introduction) on our machine/deep learning model using the tools available in Docker. In fact, the idea of this project is to build a quick and easy Docker container with a simple machine/deep learning model and run it. Implementation is available at <https://github.com/NavidAfzali/Docker-for-deep-learning/tree/master> .



**Faculty of Electrical and Computer Engineering**  
**Department of Computer Engineering**

**Bachelor project thesis**

Title:

**Using Docker to implement deep learning projects**

*Supervisor:*

**Dr.Rastgoo**

Student:

**Mohammad navid afzali**

Student number:

**۹۷۱۱۱۲۶۰۱۲**

first semester of ۰۱-۰۲