UNIT V – MACHINE LEARNING DEPLOYMENT AND APPLICATIONS

Model deployment lifecycle – MLOps pipeline – Model monitoring and drift detection – Cloud AI platforms – Edge AI deployment – Federated Learning – TinyML – Real-time inference systems – AI security and adversarial attacks – Applications of ML in healthcare, finance, agriculture, autonomous systems, robotics, cybersecurity, IoT, smart manufacturing, generative AI applications (chatbots, recommendation systems, digital twins)..

MODEL DEPLOYMENT

What is model deployment?

- Model deployment involves placing a machine learning (ML) model into a production environment.
- Moving a model from development into production makes it available to end users, software developers, other software applications and artificial intelligence (AI) systems.

Model deployment methods

- Enterprises can choose between different deployment approaches depending on the applications and use cases they envision for their new models. Here are some common model deployment methods:
  - Real-time
  - Batch
  - Streaming
  - Edge

Real-time

- Real-time deployment entails embedding a pretrained model into a production environment capable of immediate handling of data inputs and outputs. This method allows online ML models to be updated continuously and generate predictions rapidly as new data comes in. Having a model process input data on a continual basis is known as real-time inference.
- Real-time deployment can be implemented for AI applications such as recommendation engines swiftly serving suggestions or chatbots providing live support for customers.

Batch

- Batch deployment involves offline processing of data inputs. Datasets are grouped into batches, then periodically applied to machine learning algorithms. As such, batch deployment doesn't need as robust an infrastructure as real-time

deployment. Having a trained model process input data in batches is known as batch inference.

- This method is suitable for huge volumes of data that can be processed asynchronously, such as financial transactions, healthcare records or legal documents. Batch deployment use cases include document analysis, forecasting, product description generation, image classification and sentiment analysis.

Streaming

- Streaming deployment feeds regular streams of data to a machine learning system for continuous calculations and near-real-time predictions. It generally requires the same infrastructure as real-time deployment.
- This method can be employed for fraud detection and Internet of Things (IoT) applications like power plant monitoring and traffic management that rely on flows of sensor data.

Edge

- Edge deployment refers to deploying AI models on edge devices such as smartphones and wearables. The edge method of model serving can be used for edge AI applications, including health monitoring, personalized mobile experiences, predictive maintenance and predictive routing on autonomous vehicles

Model deployment and MLOps

- MLOps, short for machine learning operations, is a set of practices designed to create an assembly line for deploying, monitoring, managing and improving machine learning models within production environments. MLOps builds upon the principles of DevOps—which focuses on streamlining the development, testing and deployment of traditional software apps—and applies them to the machine learning lifecycle.
- Model deployment is just one component of the MLOps pipeline. However, some steps in the model deployment process overlap with those in MLOps

How model deployment works

- Model deployment can vary according to an organization's existing IT systems and any DevOps or MLOps procedures already in place. But the process typically encompasses these series of steps:
  - Planning
  - Setup
  - Packaging and deployment
  - Testing
  - Monitoring
  - Continuous integration and continuous deployment (CI/CD)

Planning

- Before deployment even starts, companies must prepare for the process. Here's how enterprises can fulfill technical readiness during the planning stage:

- Make sure the ML model is in a production-ready state.
- Create a model registry to store, track and manage model versions.
- Choose a deployment method.
- Select the type of deployment environment, whether it's on premises, through cloud computing services or on edge devices.
- Assess the availability and sufficiency of computational resources such as CPUs, GPUs, memory and storage.
- This is also the time to develop a timeline for deployment, define the roles and responsibilities of those involved and create clear guidelines and standardized workflows for the model deployment process.

Setup

- Like planning, setup is a multistep phase. Here's what usually happens during this stage:
- Any necessary dependencies like frameworks and libraries are installed.
- Production environment settings are configured to optimize model performance.
- Security measures, such as access control, authentication and encryption, are established to safeguard data and models.
- Current backup and disaster recovery strategies are modified to incorporate ML models and their accompanying data and infrastructure.
- Documenting all setup procedures and configuration settings is essential for troubleshooting and resolving issues in the future.

Packaging and deployment

- The model and its dependencies are packaged into a container (a technique called containerization) to maintain consistency regardless of the chosen deployment method and environment. The packaged model is then loaded into the production environment

Testing

- Thorough testing is crucial to validate that the deployed model functions as intended and is capable of handling edge cases and erroneous instances. Testing includes verifying the model's predictions against expected outputs using a sample dataset and making sure model performance aligns with key evaluation metrics and benchmarks.

Monitoring

- Keeping track of model performance, especially model drift, is a critical element of model monitoring. Insights gained from continuous monitoring feed into iterative model retraining, wherein models are updated with improved algorithms or new training data containing more recent and relevant samples to refine their performance.
- Vital metrics such as error rates, latency, resource utilization and throughput must also be logged using monitoring tools. Model monitoring occurs immediately

after deployment, but it usually falls under the purview of MLOps in the long term

Continuous integration and continuous deployment (CI/CD)

- The combined practices of continuous integration and continuous deployment (known as CI/CD) can automate and streamline the deployment and testing of ML models. Implementing CI/CD pipelines helps ensure model updates and enhancements can be easily and swiftly applied, resulting in more efficient deployment and accelerated delivery cycles for AI initiatives.

Model deployment platforms and tools

Version control

- Version control systems and model registries record model versions and their related data sources and metadata. Choices include Data Version Control (DVC), Git, GitLab and Weights & Biases.

Packaging

- Docker is a widely used open-source platform for containerization. It's compatible with cloud service providers like Amazon Web Services (AWS), Google Cloud, IBM Cloud® and Microsoft Azure. Alternatives include the Buildah command line interface (CLI), Podman and Rancher Desktop.

Orchestration

- Kubernetes is a well-known open-source container orchestration platform for scheduling and automating the deployment of containerized applications. Kubernetes and Docker are typically used in tandem. Similar orchestration tools include Red Hat® OpenShift®, Amazon Elastic Container Service (ECS) and managed Kubernetes solutions like Azure Kubernetes Service (AKS) and IBM Cloud Kubernetes Service.

Deployment

- Multiple platforms exist for deploying models. For instance, BentoML is a Python-based platform for serving ML models as application programming interface (API) endpoints and even large language models (LLMs) as API endpoints. Kubeflow facilitates model deployment on Kubernetes, while TensorFlow Serving is an open-source serving system for TensorFlow models.
- Meanwhile, other platforms not only assist with model deployment but also manage machine learning workflows. These include Amazon SageMaker, Azure Machine Learning, Google Vertex AI Platform, IBM Watson® Studio and MLflow.

CI/CD

- CI/CD tools automate model deployment and testing. Common tools include Continuous Machine Learning (CML), GitHub Actions, GitLab CI/CD, and Jenkins.

Challenges of model deployment

- Deploying deep learning models entails a lot of moving parts, which can make it a complicated endeavor. Here are some challenges associated with model deployment:

- Cost
- Complexity
- Integration
- Scalability

Cost

- Model deployment can be expensive, with infrastructure and maintenance costs eating up most of the budget. Companies must be prepared to invest in robust infrastructure and resources for efficient deployment.

Complexity

- Automating model deployment can help reduce complexity, but teams must still understand the basics of machine learning and be familiar with new technologies for deployment. Bridging this gap requires training and upskilling.

Integration

- Integrating AI models into current IT systems can be a challenge. Conducting a detailed assessment can help enterprises determine if any APIs, middleware or upgrades are needed for seamless connection and communication between models and other systems.

Scalability

- Scaling models according to demand without degrading performance can be tricky. Implementing auto scaling and load balancing mechanisms can help support multiple requests and varying workloads.