University of Idaho

Department of Computer Science

CS 404/504 Special Topics: Python Programming For Data Science

Dr. Alex Vakanski

Lecture 30

Continuous Deployment

Lecture Overview

- Continuous deployment
- Stateless retraining vs stateful training
- Frequency of model updates
- Challenges
- Model updating
  - Shadow deployment
  - A/B testing
  - Canary release
  - Interleaved experiments

From Monitoring to Continuous Deployment

Continuous Deployment

- In the previous lecture we discussed the importance of monitoring Data Science projects in production
  - An important aspect of project monitoring is detecting data distribution shifts
- In this lecture, we will focus on the problem of continuous deployment
  - That is, this step in the project life cycle is about regularly updating the deployed models to preserve the performance, e.g., by continually adapting to changing data distributions
- Continuous deployment requires to develop automated CI/CD pipelines and related infrastructure so that deployed models in production can uninterruptedly learn from new data
- As we explained in the DSOps lecture, this approach is the most suitable for companies that need to:
  - Manage a large number of models
  - Retrain the models frequently (e.g., daily, or hourly)
  - Redeploy the models on many servers simultaneously

CI/CD

Continuous Integration and Delivery

- ***Continuous Integration (CI)*** is the practice of automatically developing, testing, and merging code changes in Data Science projects in a structured manner
  - CI allows Data Science teams to confidently and quickly apply changes to the projects, since the CI pipeline provides for proper integration of made changes, where any errors in the code can be easily detected and resolved
- ***Continuous Delivery (CD)*** is the practice of regularly delivering the code and models from CI to production environment
  - CD allows to serve the end-users with the latest updated version of the model
- Deployment pipelines with CI/CD enable to quickly develop and deploy effective systems that adapt to changes

Continuous Deployment

Continuous Deployment

- Continuous deployment **use cases**
  - It is especially important for model performance in rare events
    - E.g., Black Friday, Prime Day shopping, Christmas
    - Such events can significantly degrade performance of models trained on regular historical data
    - Solution: apply continuous deployment using fresh data throughout the day/the season
  - Continuous cold start
    - It means that the model has to make predictions for a new user without any historical data
    - Other examples include cases when existing users haven't logged in for a very long period of time, or existing users who rarely log in (a user books a hotel a few times a year)
    - Continuous deployment is used to adapt to the user's preference during the current session
      - For example, Tik Tok can make recommendations for new users within several minutes
- **Disadvantages** of continuous deployment
  - Risk of catastrophic forgetting – refers to the tendency of a neural network to completely forget previously learned information when training with new data
  - Higher cost – requires infrastructure, developed pipelines, compute resources for training and evaluating multiple models, storage capacity

Triggering Model Updates

Continuous Deployment

- Deciding when to update the model requires to define a trigger mechanism
  - Time based trigger
    - E.g., every 5 minutes
  - Performance-based trigger
    - E.g., the accuracy drops below 90%
  - Data volume-based trigger
    - E.g., the total amount of available new data is increased by 5%
  - Drift-based trigger
    - E.g., a major data distribution shift is detected

Stateless Retraining vs Stateful Training

Stateless Retraining vs Stateful Training

- Updating the model can be achieved via stateless retraining or stateful training
- ***Stateless retraining***, refers to training the model from scratch with new volume of data
  - I.e., the current state of the model is not preserved when updating the model
- ***Stateful training*** refers to fine-tuning an existing model on new data
  - This training mode is also called incremental learning
  - It allows to update the model with less data
    - It can be performed without storing the incoming data (important for data with strict privacy requirements)
  - The model converges faster, requires less computer power than stateless retraining
  - Stateful training is the preferred approach in continuous deployment
    - However, there are cases when stateless retraining is required
      - E.g., when new input features are added to an existing model
      - E.g., when a new model architecture is applied

Stateless Retraining vs Stateful Training

Stateless Retraining vs Stateful Training

- Model updating with stateless retraining creates new versions of the model v1, v2, v3, by training from scratch when new data become available
- Model updating with stateful training uses an initial model v1 and creates fine-tuned versions of the model v1.1., v1.2, v1.3 as new data become available

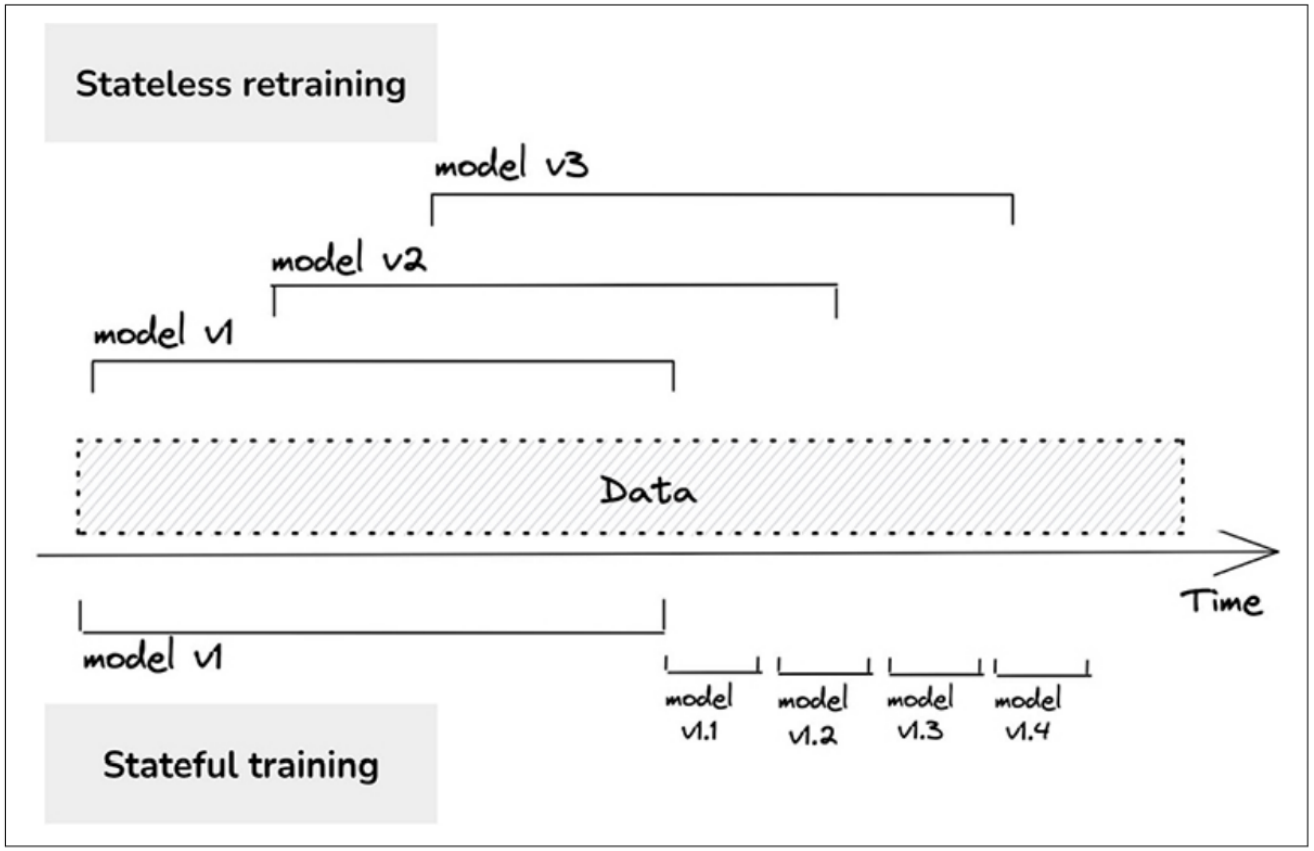

Figure from: Chip Huyen, Designing Machine Learning Systems

Frequency of Model Updates

Frequency of Model Updates

- One of the most important decisions for continuous deployment is the ***frequency of model updates***
- Important considerations for this decision are based on:
  - Quantifying the value of data freshness
    - E.g., what is the gain in model's performance when switching from retraining monthly to weekly to daily?
    - To calculate the gain, train models on data from different time windows in the past and evaluate them on most recent data to see how the performance changes
      - Adopting a smaller window for updating usually can increase business performance (such as higher click rate, lower prices for rides)
  - Deciding on model iteration vs. data iteration
    - Model iteration: change the model architecture, employ new algorithm
    - Data iteration: train the same model using new data
    - It is recommended to periodically perform model iteration and data iteration
      - E.g., introducing a new model architecture can significantly improve the performance

Continuous Deployment Challenges

Challenges

- Fresh data access
  - Continuous deployment requires access to fresh data
    - E.g., to update the model hourly, requires to obtain fresh data on an hourly basis
    - If the data is collected from several sources, it may require long time to be collected
    - If the data is unbalanced, the time for fresh data access can be long
      - E.g., an ML model is used by a bank to detect fraudulent transactions: most transactions are regular, it takes time to collect fraudulent transactions
- Model evaluation challenge
  - Evaluating model performance to ensure that the updated model is good enough to be deployed is challenging
    1. Continuous deployment increases the risk of catastrophic forgetting and model failure
    2. Continuous deployment makes models more susceptible to adversarial attacks
       - It makes it easier for end-users to input malicious data to trick the model into making wrong predictions

Model Updating

Model Updating

- Model updating can be based on:
  - Offline evaluation – the model is evaluated using historical data
  - Online evaluation – the model is evaluated using fresh production data
    - Pros: the online data used for model evaluation has the same distribution as the incoming production data
    - Cons: online evaluation in production is risky
- Several approaches for safe ***online model evaluation*** are employed
  - Shadow deployment
  - A/B testing
  - Canary release
  - Interleaved experiments

Shadow Deployment

Model Updating

- ***Shadow deployment***
  1. Deploy the new model in parallel with the existing model
  2. Route each incoming request to both models to make predictions, but only serve the predictions by the existing model to the users
  3. When the results by the new model are satisfactory, switch to the new model

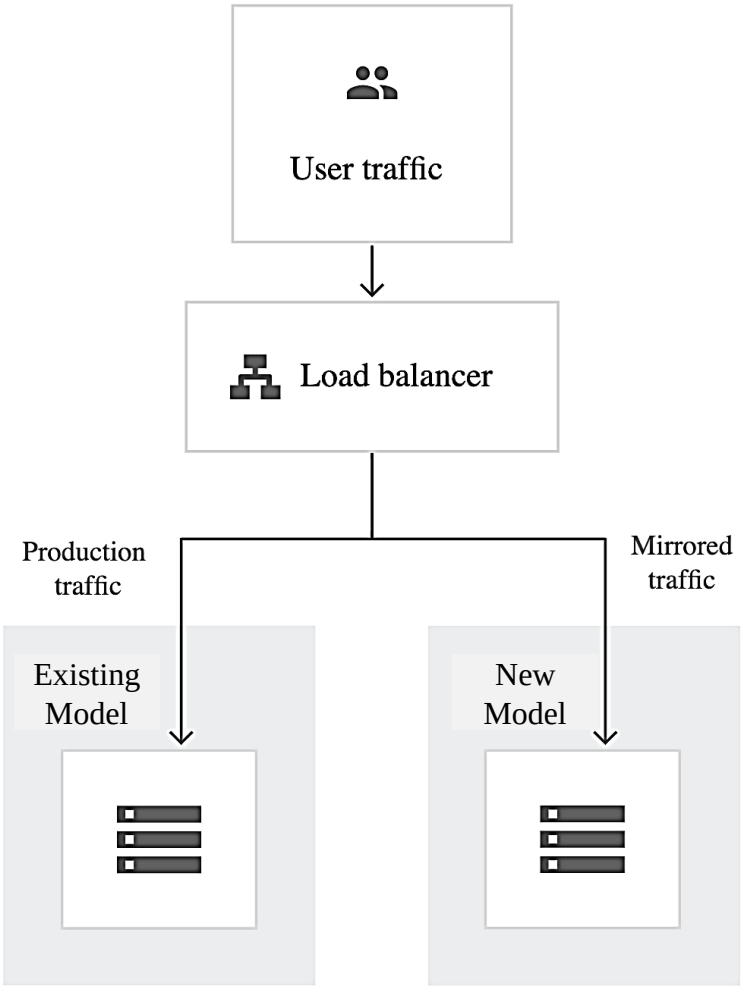

Figure from: Chip Huyen, Designing Machine Learning Systems

A/B Testing

Model Updating

- ***A/B testing***
  1. Deploy the new model alongside the existing model (e.g., model A and model B)
  2. A percentage of traffic is routed to the new model for predictions and serving, based on routing rules
     - The rest of the traffic is routed to the existing model for prediction and serving
  3. Monitor the predictions by both models and collect feedback from end-users, to determine if there is a statistically significant difference in the performance based on two-sample tests
  4. Select the better-performing model to keep
- Notes:
  - The traffic routed to each model needs to be random
  - A/B test should be run on sufficient number of samples, to ensure confidence in the performance
  - It is possible to evaluate not only one, but multiple candidate models (e.g., A/B/C testing)

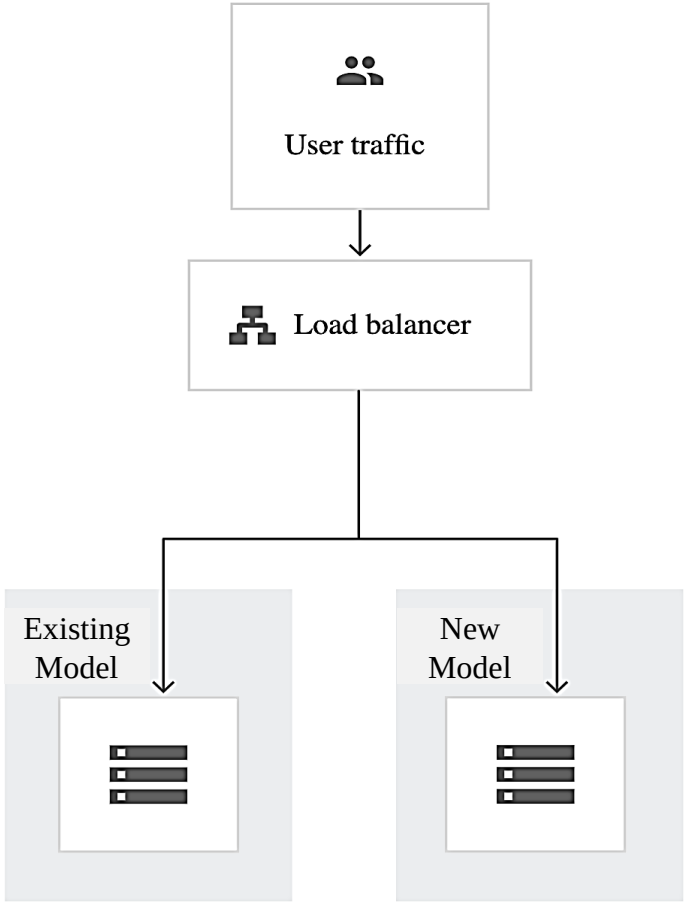

Figure from: Chip Huyen, Designing Machine Learning Systems

Canary Release

Model Updating

- ***Canary release***
  1. Deploy the new model (called canary) alongside the existing model
  2. Portion of the traffic is routed to the new model
  3. If the performance is satisfactory, slowly increase the traffic to new model until it fully replaces the existing model
     - E.g., roll out to USA first, then Europe, Asia, and the rest of the world
     - If the performance is poor, keep the existing model

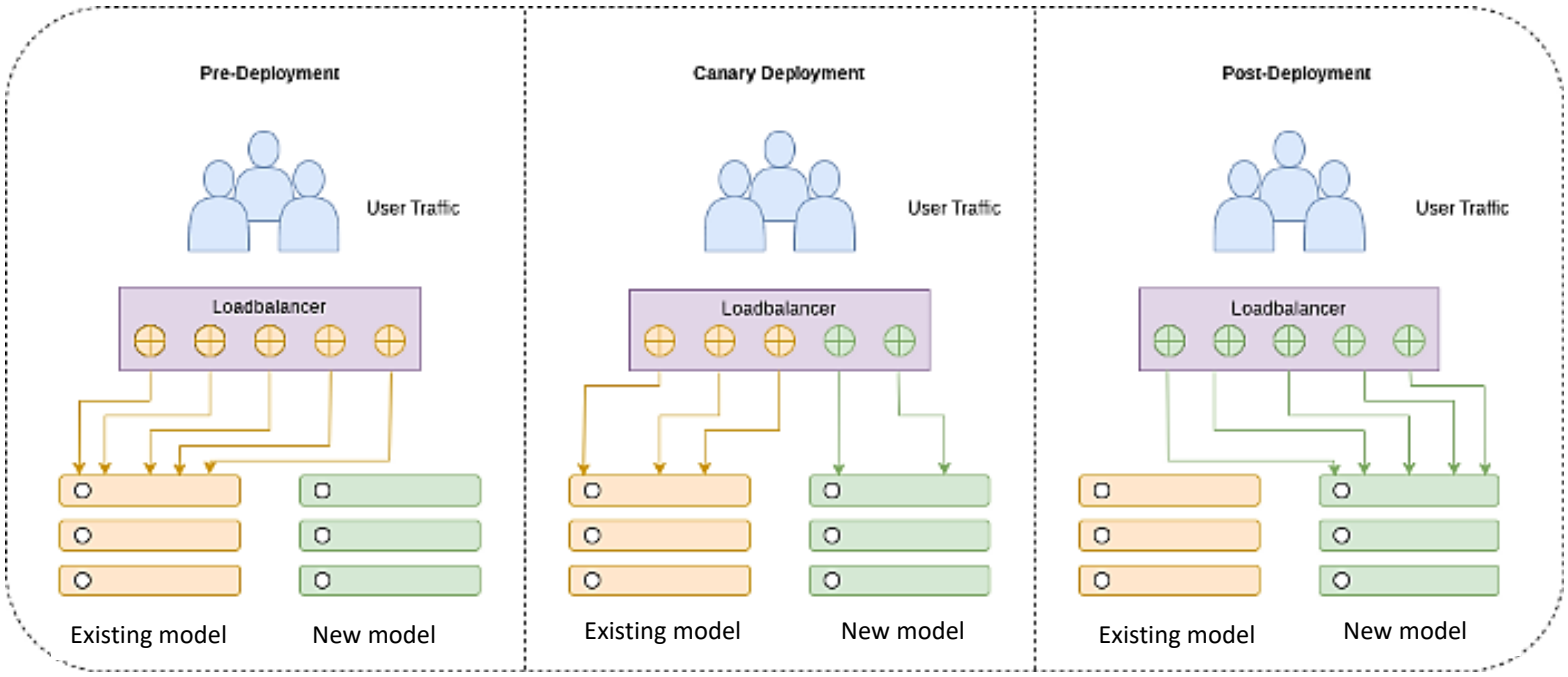

Figure from: https://blog.getambassador.io/kubernetes-canary-testing-and-release-with-istio-4cbdedcc9914

Interleaved Experiments

Model Updating

- ***Interleaved experiments***
  1. Deploy the new model alongside the existing model (e.g., model A and model B)
  2. Mix (interleave) the predictions by both models A and B together, and show them to the end-users
  3. Analyze end-users' preferences (e.g., click rates) to decide which model to keep
- This approach is especially useful for recommender systems and search ranking
- Comparison to A/B testing
  - In A/B testing, one group of users is served the predictions by model A, and the rest of the users are served the predictions by model B
  - In interleaved experiments, all users are served predictions by both models A and B

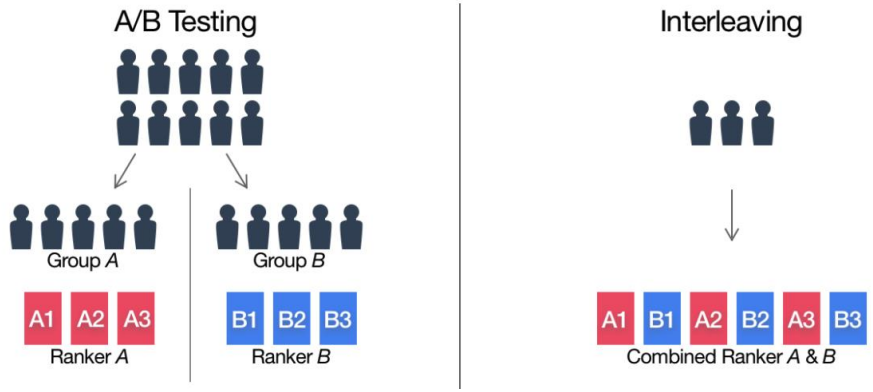

Figure from: Chip Huyen, Designing Machine Learning Systems

References

1. Chip Huyen, Designing Machine Learning Systems, O'Reilly Media, 2022.