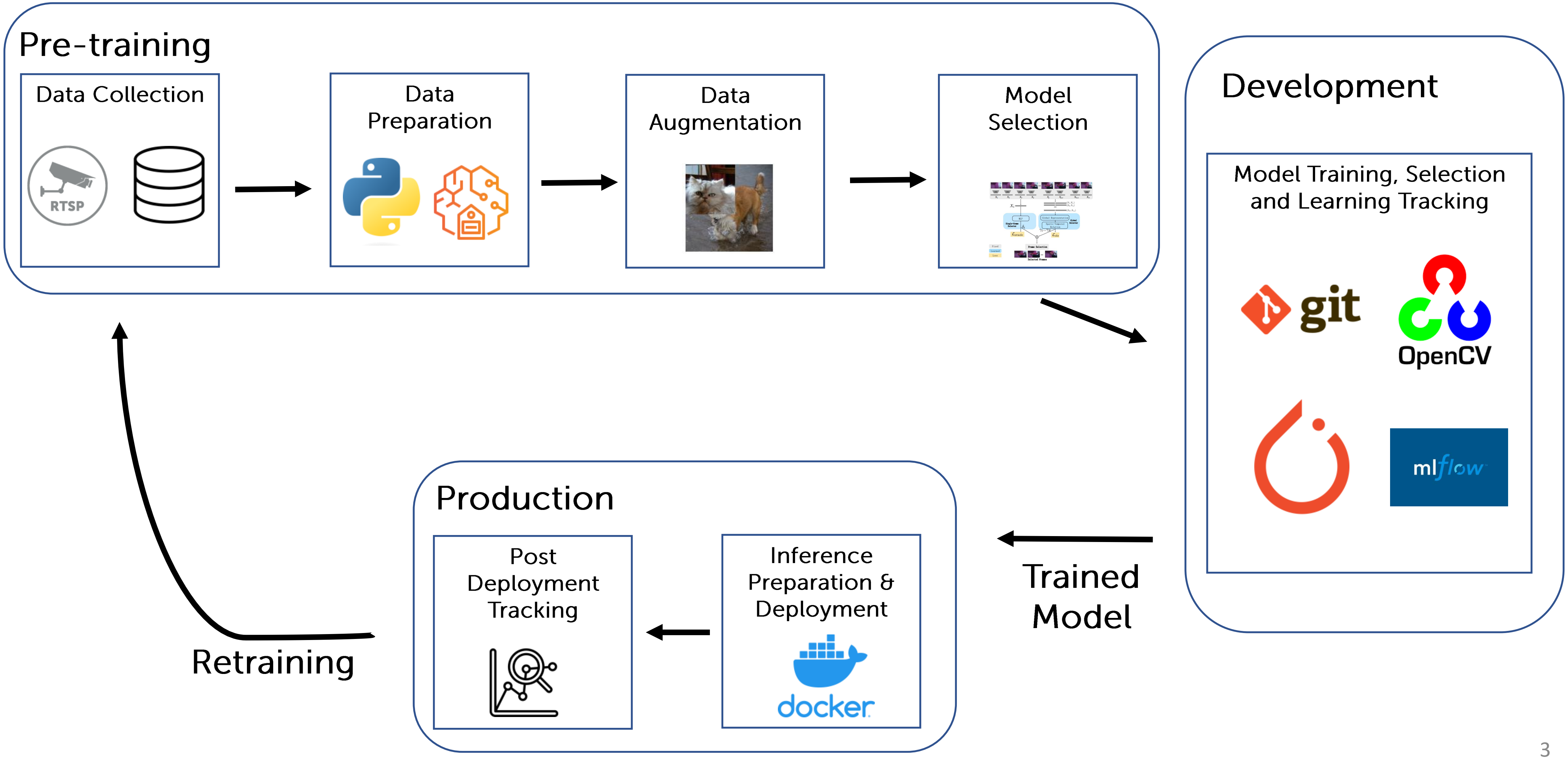
Action Recognition

Table of Contents:

Overview

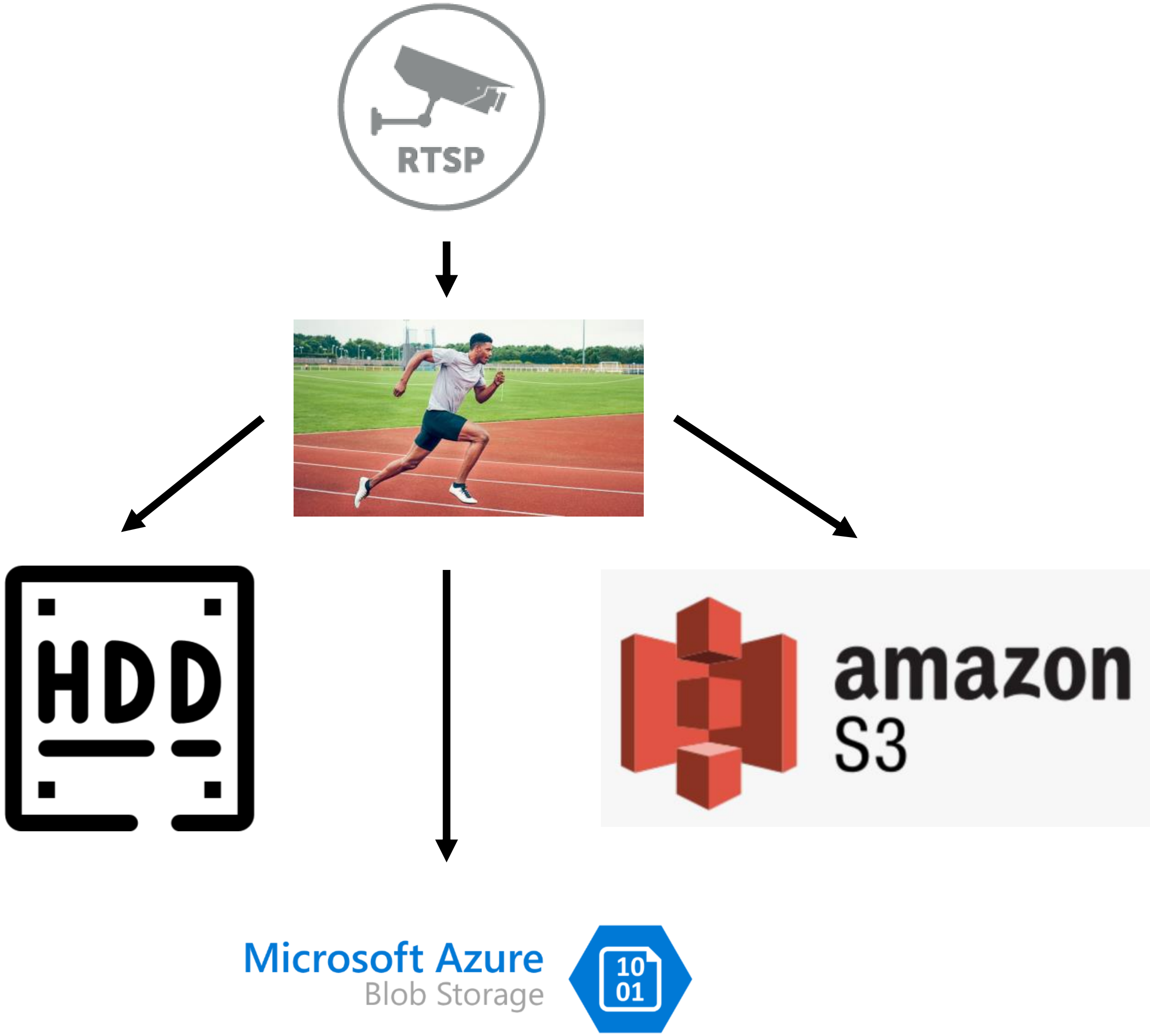
Overview


Pre-training
Data Collection
RTSP
Data Preparation
Data Augmentation
Model Selection
Development
Model Training, Selection and Learning Tracking
git
OpenCV
mlflow
Trained Model
Production
Post Deployment Tracking
Inference Preparation & Deployment
docker
Retraining

Data Collection

The train and test data should come from similar domain

- **Scene** is important to computer vision task
- Although a lot of [public datasets](#) are freely available, perform **dataset collection** at the targeted deployed setting is still crucial for best model performance
- Camera Setup:
  - IP camera renders live feed video using **RTSP** protocol
  - Minimum **30** FPS
  - Resolution: 1920x1080 (**2**MP)
- Collected data can be **stored** either **locally** or on **cloud**



Data Preparation

Curate and Label Images

- Quality of collected images should be inspected for **high quality training** process in subsequent stage. E.g. Blurred images should be removed
- Images with disturbances such as different lighting condition or noise can be treated using image processing tools.
- Image labelling can be done using **open-source tools** or **cloud services**

Open-sourced Image Labelling Tools
(LabelMe, LabelImg)

AWS Ground Truth

Data Augmentation

Data augmentation is crucial for optimal performance

Plenty **augmentation** techniques are available to achieve the following

- To ensure a **variety** of images are available for training
- To prevent **overfitting**

Popular data augmentation techniques offered by [Pytorch] include

- Random resize
- Random cropping
- Random affine transformation
- Random color jitter
- [CutMix]

Etc

Augmentation can be performed **offline** or **on-the-fly** during training

Raw Image

Mixup

Cutout

CutMix

Model Selection (Pre-training)

SMART is selected as solution

- UCF101 is the one of the most **popular** public dataset in action recognition domain with **4592 citations** as recorded by Google Scholar
- Sampling through Multi-frame Attention and Relations in Time (SMART) produces **most astounding** results (**98.64% accuracy**) based on the leaderboard in paperswithcode
- Thus, it serves as good **baseline** to kickstart the development and more work can be done to further improvise its performance

Sample Images from UCF101

Snapshot of SMART Technical Paper

Research Problem

- Video has huge information **redundancy**
- Processing full video is computationally **expensive**

Research Objective

- To perform frame selection SMART-ly for action recognition by **considering** only **important** frames

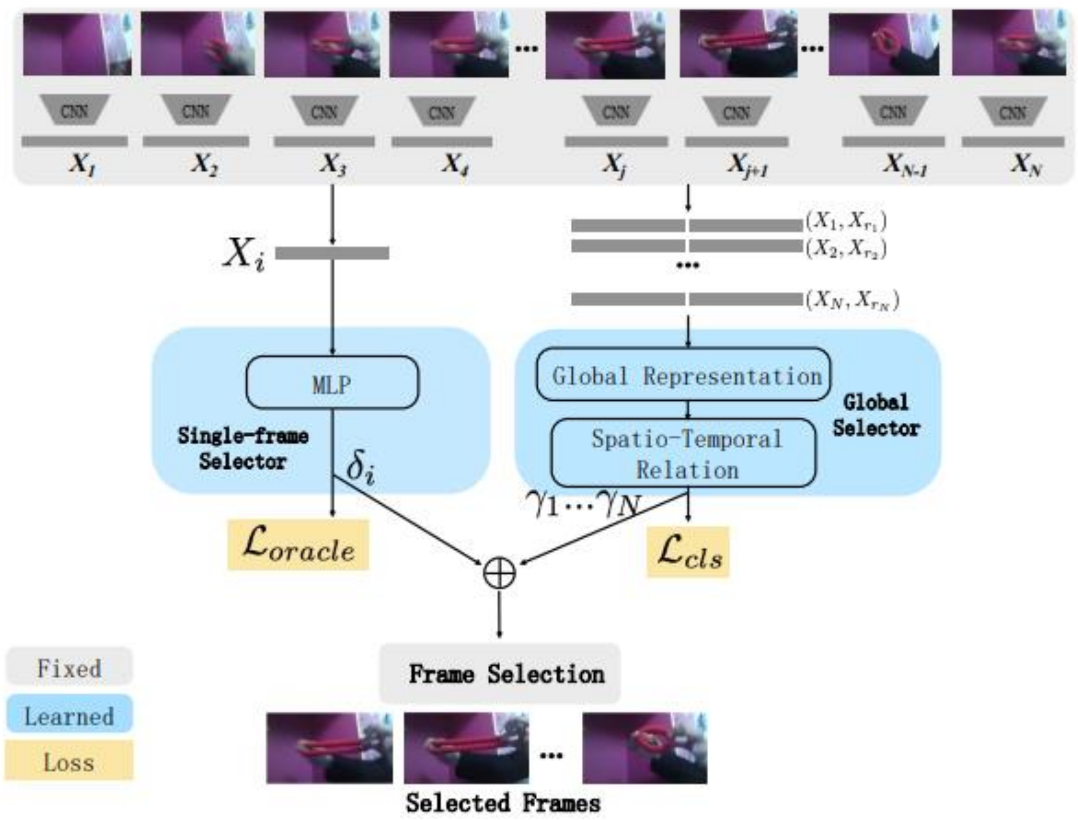


Illustration of SMART

Snapshot of SMART Technical Paper (Cont'd)

Methodology

- Select top-N frames using SMART selection strategy by employing single-frame and global selector
- Frame is represented using MobileNet and GloVE features jointly
- Single-frame selector: MLP is used to express frame importance score
- Global Selector:
  - Information from individual frame is aggregated into global features through attention module.
  - Relation model and Long-Short Term Memory (LSTM) are used to compute temporal relationships among frames.
- Action of the selected frames is recognized using deep CNN

Model Training, Selection & Learning Tracking

Model Training, Selection & Learning Tracking

A **commendable** deep learning-based **training script** should cover the following:

- Data Ingestion Pipeline (Datasets & Dataloaders)
- Data Augmentation
- Loss Function (Cross entropy loss/Focal loss/Label smoothing loss/Contrastive loss/Center Loss etc.)
- Metrics (Accuracy/Recall/Precision/F1-score etc.)
- Optimizer
- Learning rate scheduler

Multiple trials can be run using different combinations of parameters. The **best model** should be selected by considering following:

- Model that gives highest metrics based on evaluation dataset
- The chosen metrics is selected based on class distribution
  - Accuracy can be used for dataset with almost evenly distributed classes
  - Recall/Precision/F1-score is opted for imbalanced class

Model Training, Selection & Learning Tracking (Cont'd)

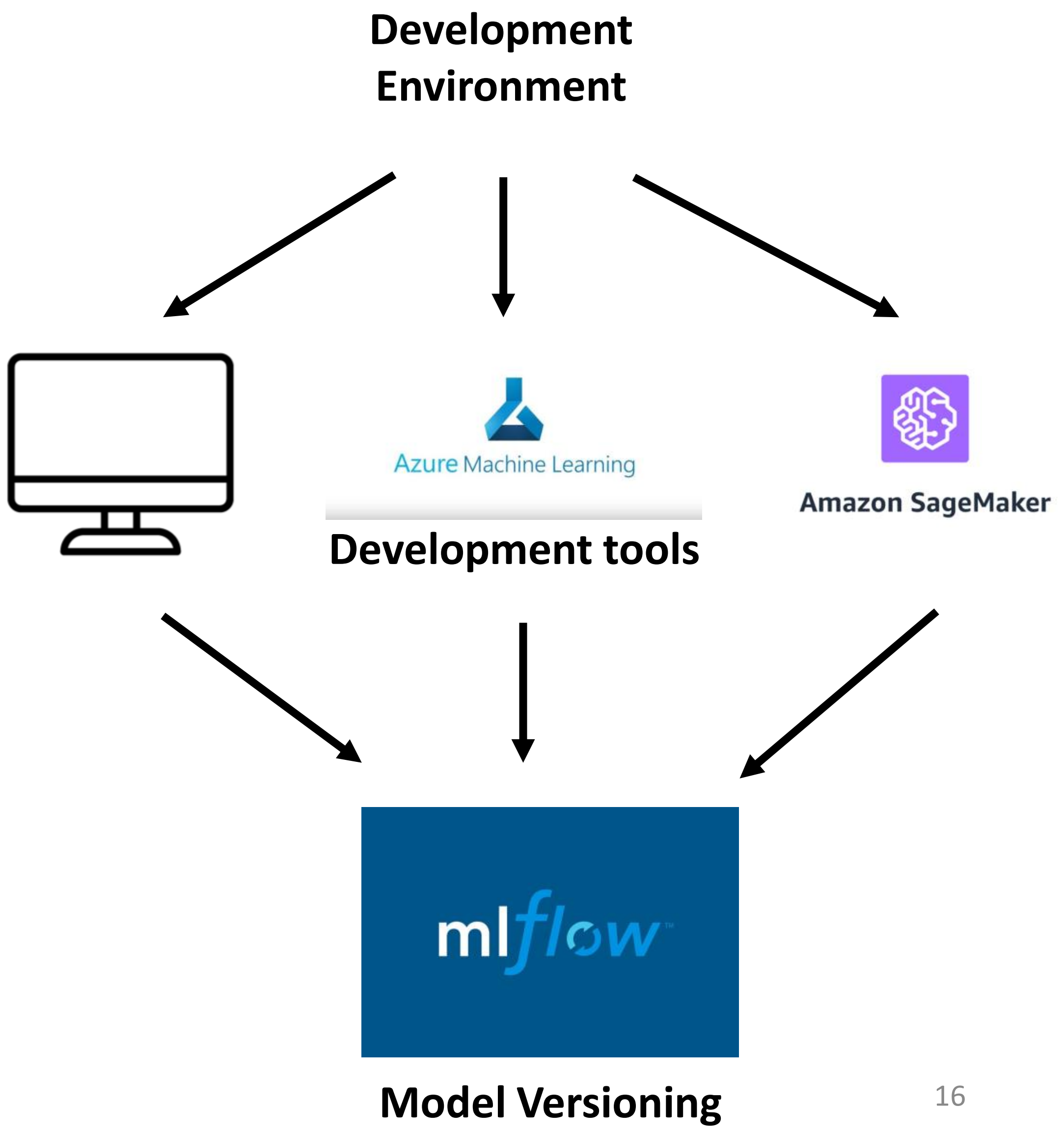
Training process can be **monitored** by:

- Observing training and evaluation loss
- Observing training and evaluation accuracy

The above should not have large discrepancy else it can be either **underfitting** or **overfitting**

Development environment can be either local or on cloud

MLFLow can be used for model versioning to ease final model selection



Inference Preparation & Deployment

Inference Preparation & Deployment

Set up deployment environment with following consideration

- computational power (CPU, RAM, GPU specification)
- On-premise server (sensitive data) or cloud server

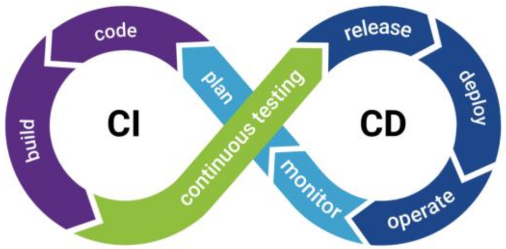
Optimizing inference speed

- Converting native pytorch model to **TensorRT**
- Quantization: Inference on **fp16** and even **int8** but with slight drop in accuracy
- **Model pruning** by removing less important layers using NVIDIA TAO Toolkit
- Use NVIDIA Trition Server or NVIDIA Deepstream as inference pipeline for speed optimization

Inference Preparation & Deployment (Cont'd)

CICD Pipeline using YAML file

- Automated **unit testing i.e.** pytest to avoid code breaking using CI pipeline
- **Containerization** and upload docker image to DockerHub/Azure Container Registry/Amazon Elastic Container Registry
- One-click away **deployment** to production environment using CD pipeline



Azure Container Registry

Amazon ECR

Post Deployment Tracking

Post Deployment Tracking

Data drift occurs when images used during development is different from deployment. For instance

- Change in background
- Lighting condition
- People behavior (pre-covid vs covid vs Post-covid time)
- Etc

Continuous monitoring model accuracy is important.

Defining bad performance is important as well. E.g. impact of 5% accuracy drop in animal classification is different from that of lung disease classification based on X-ray

Post Deployment Tracking (Cont’d)

Model retraining process can be **triggered** when model performance is below predefined level.

For real-time application, tracking **inference speed** is crucial. Higher number of objects in frame will require more computational power to render result in real-time. E.g. Covid lockdown results in less number of crowd as compared to normal circumstances.