### Q4. Deep Learning

### Data Pipeline:

A data labeling pipeline is an end-to-end workflow for labeling a dataset. It includes the process for selecting and training the annotators, managing data labeling, and aggregating and verifying the results. As an output, it provides high-quality labeled data.

A high level data pipeline is depicted as below.

Remote Devices

Business Application Systems

DB

KAFKA

**Data Collection**

-MySQL

-AWS s3

**Data Ingestion**

ETL

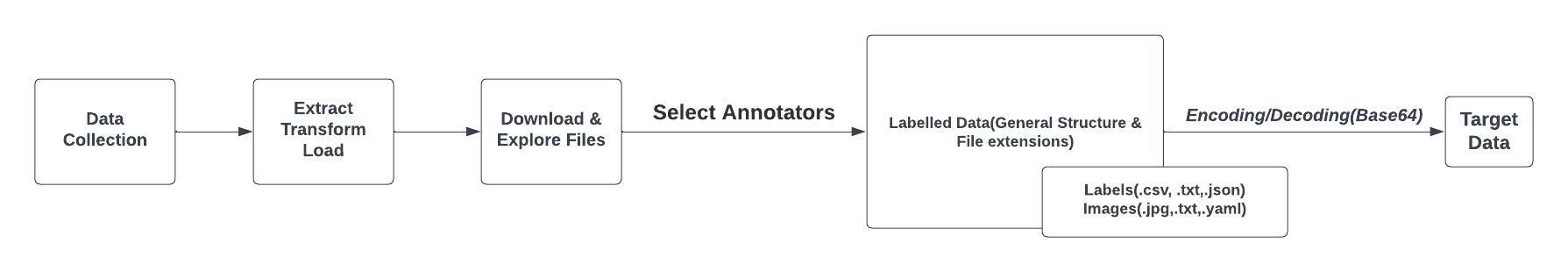
**Data preparation**

Consumer Apps

**Consumption**

**Data pipeline development stages are shown as below.**

### Data Pipeline



**Technologies Used**: PyTorch, Fastai, LabelImg etc.

* PyTorch features a Pythonic API, a more opinionated programming pattern and a good set of built-in utility functions such as a built-in DataLoader to feed data when training a model.

**Limitations**:

* If you only have a few classes, you can run labeling for each class separately. If you have too many classes to label in parallel then it is cumbersome.
* you can encode image to Base64 and embed it using the data URI. Base64 encoder accepts any images types with a size of up to 50 MB.

### MLOps Pipeline

* ***YOLO***
* ***Detectron2***
* ***TFOD2***

**Get data from gitHub**

**Validation status=True**

**ModelTrainer(Get best model)**

**Data Validation**

**Data Ingestion**

**Data zip file path**

**Feature store path**

**Bucket name, s3 model path**

**Model Pusher**

Save Model (best.pt)

### Deployment Pipeline:

AWS Cloud

**EC2**

**trigger**

Local Development Environment

GitHub

VM

**S3 bucket**

***Continuous integration/Continuous Deployment (Docker)***

### Inference Pipeline:

An inference pipeline is a program that takes input data, optionally transforms that data, then makes predictions on that input data using a model. Inference pipelines can be either batch programs or online services.

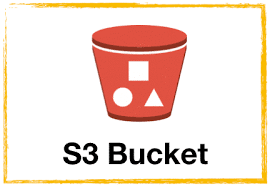
The inference step involves setting up the input data pipeline, making predictions on the input data using the model, and optionally post-processing the predictions.

An inference pipeline typically involves downloading the trained model from the saved s3 path, loads any required dependencies (such as importing libraries), and establishing a connection to a feature store. Since we are using a feature store, the inference pipeline can read precomputed features directly from the feature store, perform any necessary on-demand feature transformations and any feature encodings, and then pass the transformed features to the model for prediction.

**Download the best trained model** (***best.pt)***

Connect to **FeatureStore**

**Prediction**



***-transformations (SparkML, Scikit-learn) -Feature encoding***

***-Import CV2, torch,PIL,()***

model.cpu() # CPU

model.cuda() # GPU

model.to(device) # torch device

**Advantage with the AWS Sagemaker:**

* When you configure the pipeline, you can choose to use the built-in feature transformers already available in Amazon SageMaker. Or, you can implement your own transformation logic using just a few lines of scikit-learn or Spark code.
* A pipeline model is immutable, but you can update an inference pipeline by deploying a new one using the UpdateEndpoint operation. This modularity supports greater flexibility during experimentation.
* There are no additional costs for using this feature. You pay only for the instances running on an endpoint.

**For the use of GPUs, the following points can be taken into consideration**.

* Consider factors such as GPU memory, compute capability, and power efficiency when making hardware selection
* Use deep learning frameworks like TensorFlow or PyTorch that provide GPU acceleration capabilities
* Process your data in batches as batch processing helps maximize GPU utilization and improves training time.

**Inference settings can be made as below for example.**

* model confidence threshold,
* model IoU threshold
* pretrained YOLOv5s model
* number of classes
* force reload (force a fresh download of the latest YOLOv5 version from PyTorch Hub) etc.