Video Verticalization

Hackaton Project

boring-wozniak crew

Project Scope

Problem Statement

- IVI is the platform for video streaming and its users consumes content both on desktop and mobile devices
- Most of films are recorder in landscape format (e.g. 16:9)
- A lot of content consumption is being done on mobile devices
- In case of adding a padding to horizontal video for showing it on vertical oriented device, a lot of screen space is left unused

Goals & Objectives

Goals

- To increase the engagement of users on mobile devices

Objectives

- To develop a system that converts horizontal videos (landscape format) to vertical (portrait) keeping the maximum of saliency in the frame

Constraints

Performance

- Model evaluation: versus labeled dataset + subjective expert opinion
- Baselines: ??? есть ли решения, на которые мы можем ориентироваться?
- Performance tradeoffs: ???
- Interpretability: Not required
- Confidence measurement: Not required

Data

- Existing data: 10 labeled videos, X unlabelled (limited features)
- New data: no limitations - manual collection&annotation, crowdsourcing, etc.
- Heterogeneity: different genres implies different saliency

Constraints

Budget

- All costs is covered by the development team
- Budget compensation by the client (Ivi) ???

Stakeholders

- Ivi CV team: client, product owner, mentoring party
- MIPT & SF: observer, facilitator
- Other parties: are allowed upon the decision of team?client

Constraints

Privacy

- Model, code, system architecture and other discoveries are not the subjects of intellectual property
- Videos provided by the client are copyrighted
- Datasets, including labels and annotations provided by the client are ?not? the subjects of copyright
- Datasets, including labels and annotations collected by the team are the property of the team

Legacy

- The developed system should be compatible with client data pipelines

Other

- The system development is a part of hackathon challenge and

ML vs non-ML

Do we really need ML to solve the problem?

- No strict patterns to define (e.g. stationary actor or moving object)
- We have data for training (frames with known crops)
- A lot of unseen frames with common patterns
- The problem has big scale

Based on indirect signs provided, we have a task for ML

System Design

System classification

Choosing a type of ML system

- ✖ ~~Online prediction~~ vs. ✅ Batch prediction
- ✖ ~~Edge computing~~ vs. . ✅ Cloud computing
- ✖ ~~Online learning~~ vs. . ✅ Offline learning vs. ✖ ~~Continuous learning~~

Data

Data

Collection

- Data (video and labels) is provided by Ivi
- Data is collected from video hosting (e.g. Youtube)
- Data type to collect - clips up to 5 minutes, film previews
- Data splitting (by year, country, culture, genre, etc.) - not necessary

Data

Storage

- Format to store: ????
- Location: ???? S3 / Object storage
- Data partitioning: Not performed

Data

Transformation

- Clipping videos: not performed
- Data cleaning: not performed
- Data encoding/decoding: ????
- Anomalies detection/correction: not performed

Data

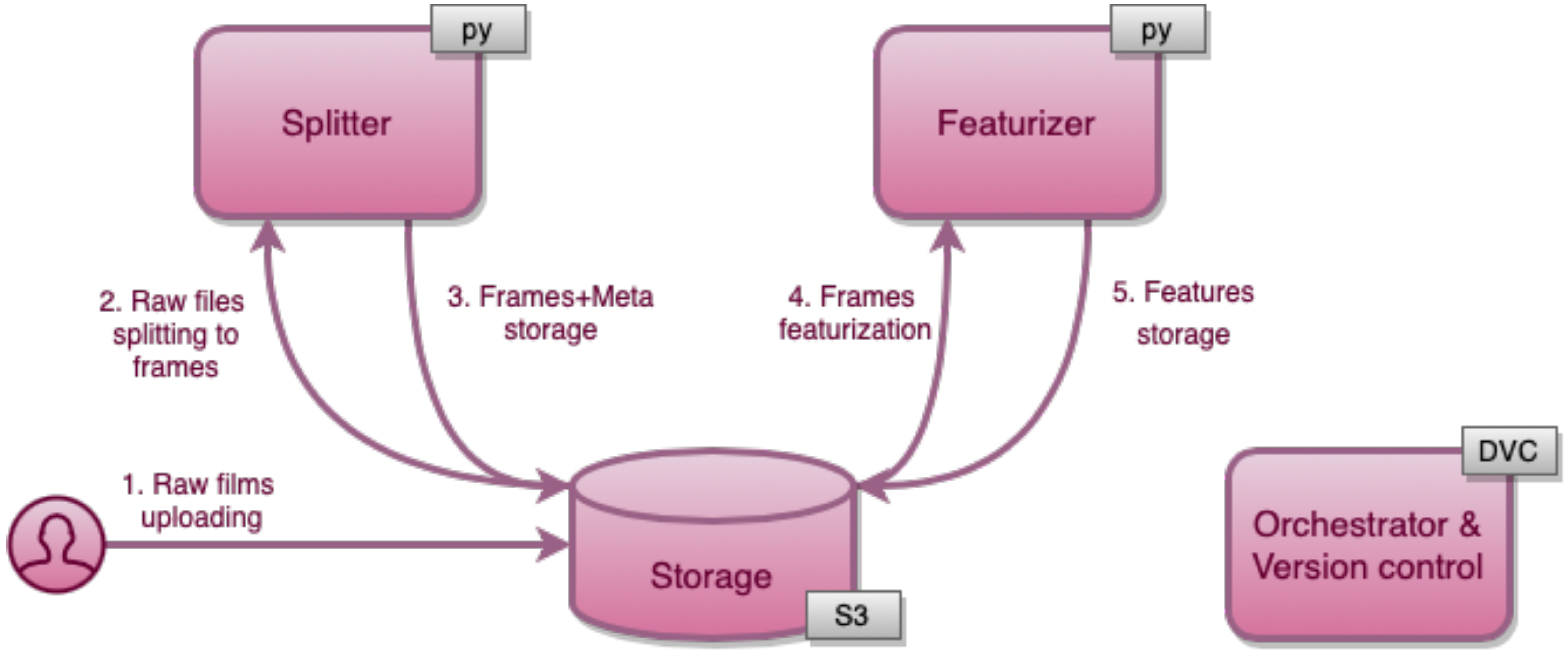
Encoding and Labeling

- Video clip is splitting into frames
- Each frame is encoded with a range of features
- Downstreams modules work with frames data
- Frame’s data should contain clip’s metadata (e.g. clip id, size, length, year, country, genre etc.) as well as frame’s extracted features (actors in the frame, objects, activities, etc.)
- Feature extraction is performed using pre-trained models
- Labeling (crop positioning) is done in manual mode or with a help of crowdsource (supervised learning)

High-Level Design

Data preparation

1. User uploads raw (uncropped) film to the storage
2. Splitter gets the video file and explode it to frames
3. Splitter generates metadata for each frame and saves to the storage
4. Featurizer gets frames and detect objects, actions etc.
5. Featurizer adds extracted features to the original frame meta and stores them
6. Data/pipeline version control as well as pipeline orchestration is handled by Orchestrator



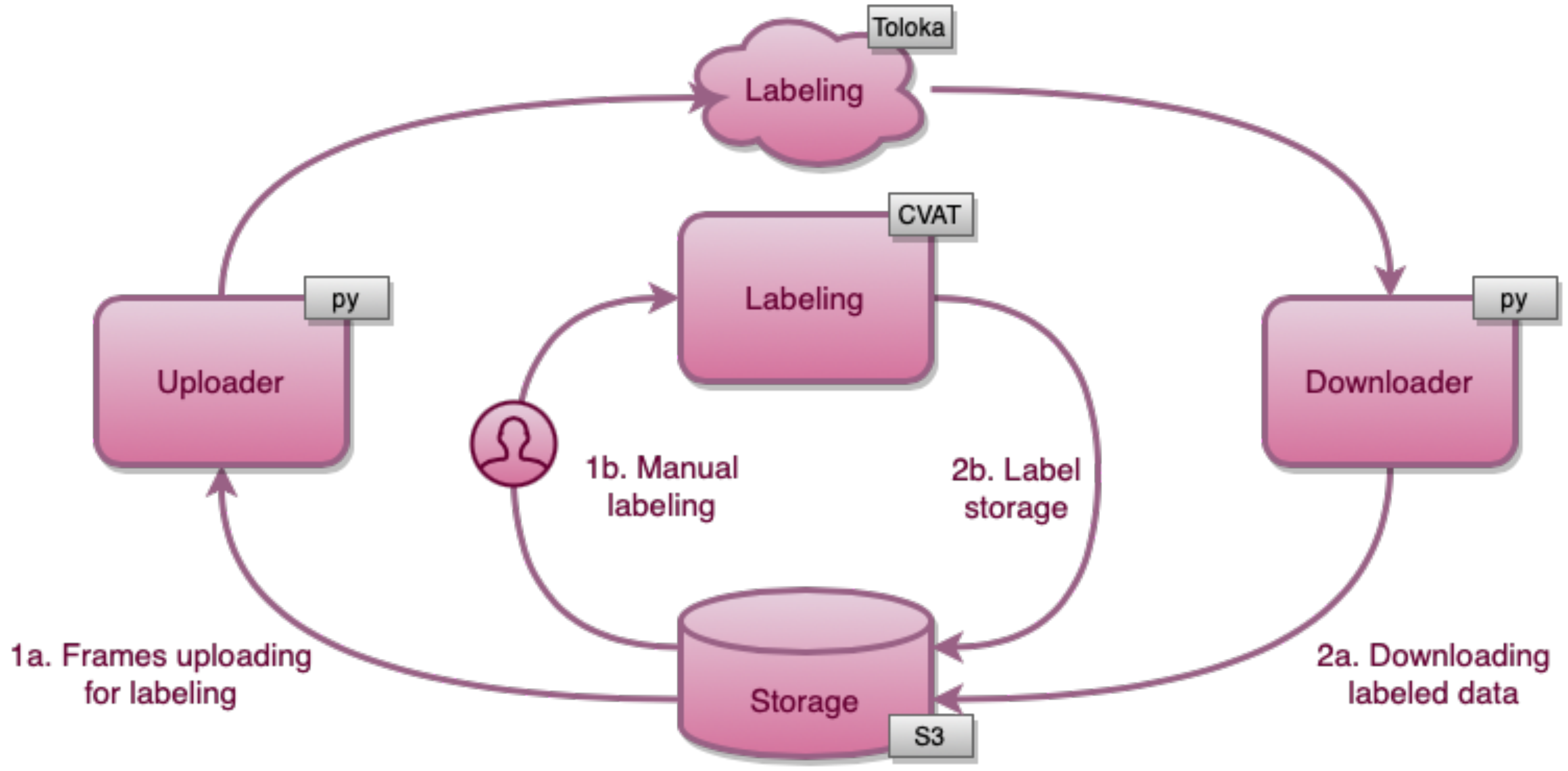
Data annotation

a. Automated labelling

1. Uploader gets frames from storage and uploads them to crowdsourcing tool
2. Downloader monitors annotation process and downloads labels for frames to the storage

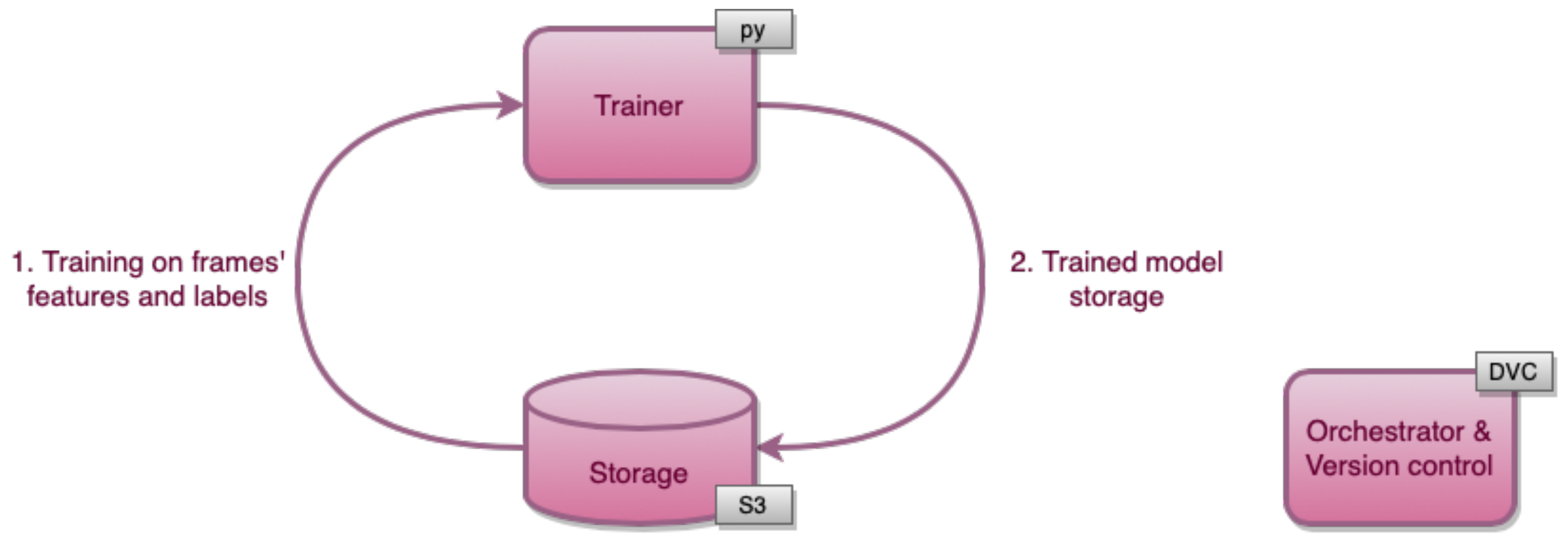
b. Manual labelling

1. User labels the frames using local tool
2. Labeled data is stored



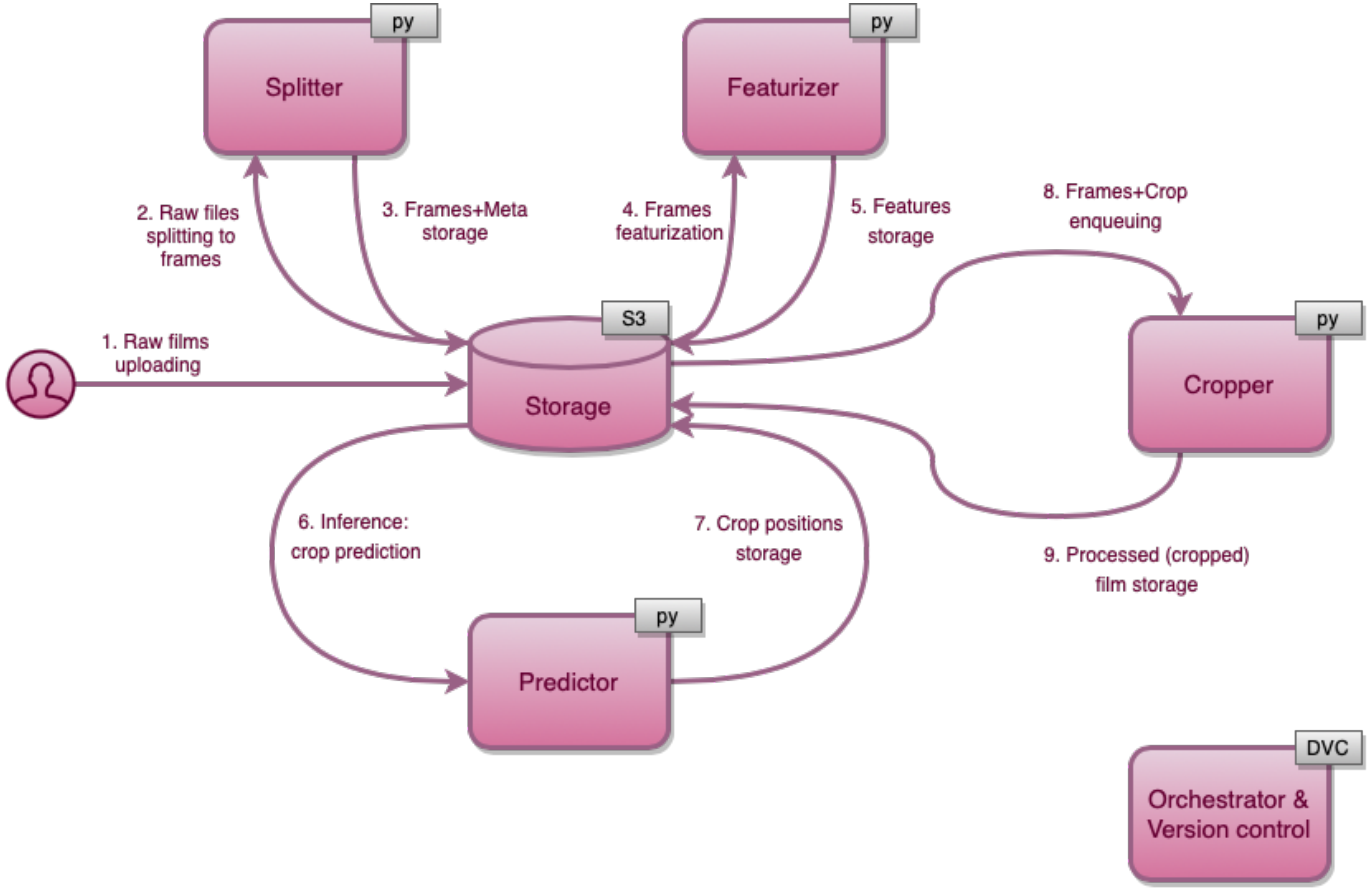
Model training

1. Trainer gets data (features and labels) from the storage and trains the model
2. Trained model is serialised and stored
3. Orchestration, experiment tracking, pipeline moderation, version control is done by Orchestrator


py
Trainer
1. Training on frames' features and labels
2. Trained model storage
Storage
S3
DVC
Orchestrator & Version control

Inference

1. User uploads raw (uncropped) film to the storage
2. Splitter gets the video file and explode it to frames
3. Splitter generates metadata for each frame and saves to the storage
4. Featurizer gets frames and detect objects, actions etc.
5. Featurizer adds extracted features to the original frame meta and stores them
6. Predictor gets features and makes a forecast for a crop position.
7. Predictor stores crop's coordinate.
8. Cropper use crop coordinates and original raw video to crop video file
9. Cropped film is stored for further usage.
10. Data/pipeline version control as well as pipeline orchestration is handled by Orchestrator



Components and tech stack

System components

Tech.stack

Splitter - python script, cv2

Featurizer - python script, yolo/detectron/mediapipe/xxx

Trainer - python file, pytorch/xgboost/catboost/xxx

Predictor - python script, pytorch/xgboost/catboost/xxx

Cropper - python script, cv2

Uploader - python script, toloka-kit

Downloader - python script, toloka-kit

Storage - S3 object storage

Orchestrator - Git + DVC, MLFlow

Market research

Existing solutions, libraries, tools etc.

Existing solutions

Google AutoFlip

Yolo

Detectron

OpenCV

Xxx

Project plan

Team

Crew and Roles

- Alex - writing code of xxx, ….
- Anton - solution architecture, setting up pipelines…
- Danil - project management …
- Dima - to be xxx
- Katya - to be xxx
- Kirill - to be xxx

Roadmap

- Architecture design - xx.xx
- Formats and specifications standartization - xx.xx
- Infrastructure and pipeline creation - xx.xx
- Components development - xx.xx
- Testing and debug - xx.xx
- Documenting and presentation - xx.xx