SCENE UNDERSTANDING ON STREAMING VIDEOS

Vishnu Beji
NetID: vb2409

Dhanush Raya
NetID: dr3631

Yashwanth Alapati
NetID: ya2351

I. Abstract:

This project delves into the nuanced understanding of scenes within frame-level video datasets, employing advanced deep learning models to extract meaningful insights from video features. Leveraging the capabilities of multiple big data technologies, our methodology combines the power of Spark for Exploratory Data Analysis (EDA), Kafka for real-time streaming scenarios, and MongoDB for the efficient storage of predictions and video ID mapping. The culmination of these technologies facilitates robust learning from video features, leading to a sophisticated setup for inference. Additionally, an interactive visualization of model predictions has been developed using IPython, enhancing the interpretability and accessibility of the deep learning model's outcomes.

II. Big Data problem

The YouTube 8M dataset [1] contains over 1.5 TB of data spanning millions of videos. Handling, storing and processing such vasts amount of data makes it a big data challenge. Processing such a large Dataset on a single machine is impractical and scaling the data processing can be achieved easily with big data technologies . Scaling can be carried out in Vertical and Horizontal manner, however Horizontal scaling is efficient because multiple machines can work together. This approach reduces the time taken to process the data. To carry out the Exploratory Data Analysis process we employ Spark. The project involves the implementation of a real-time video streaming pipeline with Kafka to facilitate the seamless integration of video data into the Deep learning model. This setup aims to enable the model to dynamically analyze and create labels for video segments as they are streamed in real time. The integration of Kafka adds a layer of complexity for efficient handling of data ingestion, processing, and communication between components. Overall, by horizontally scaling the data processing, we can reduce the time taken to process the data, while maintaining accuracy and cost efficiency.

III. Dataset

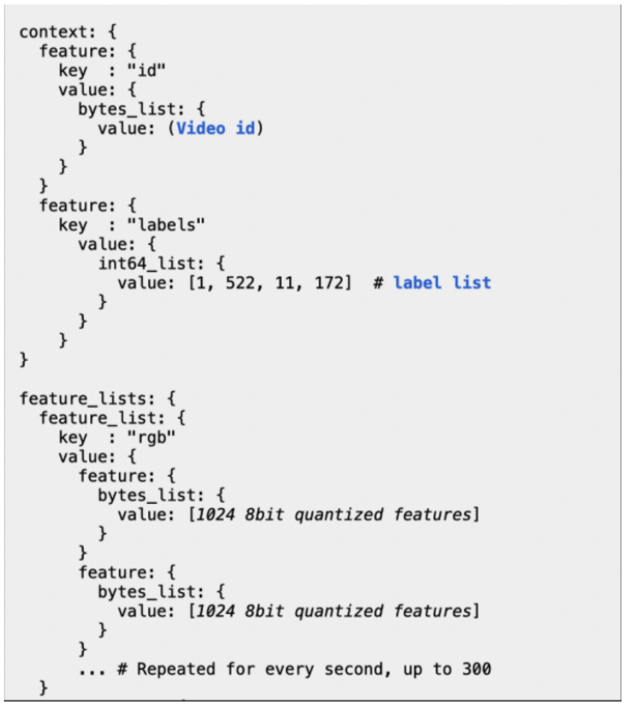
The YouTube 8M dataset consists of over 1.5 TB frame level features. It was released in June 2019 with segment-level annotations. Human-verified labels on about 237K segments on 1000 classes are collected from the validation set of the YouTube-8M dataset.

Below are some of the features of the dataset:

- Size of frame level data: 1.71 TB
- Data is broken into 3844 shards
- Schema: video-id, labels

Description:

- "video-id": unique id for the video
- "labels": list of labels of that video.
- Each frame has "rgb": float array of length 1024
- Each frame has "audio": float array of length 128



```
context: {
  feature: {
    key  : "id"
    value: {
      bytes_list: {
        value: (Video id)
      }
    }
  }
  feature: {
    key  : "labels"
      value: {
        int64_list: {
          value: [1, 522, 11, 172]  # label list
        }
      }
    }
}

feature_lists: {
  feature_list: {
    key  : "rgb"
    value: {
      feature: {
        bytes_list: {
          value: [1024 8bit quantized features]
        }
      }
      feature: {
        bytes_list: {
          value: [1024 8bit quantized features]
        }
      }
      ... # Repeated for every second, up to 300
  }
```

Fig. 1. Schema of the Dataset

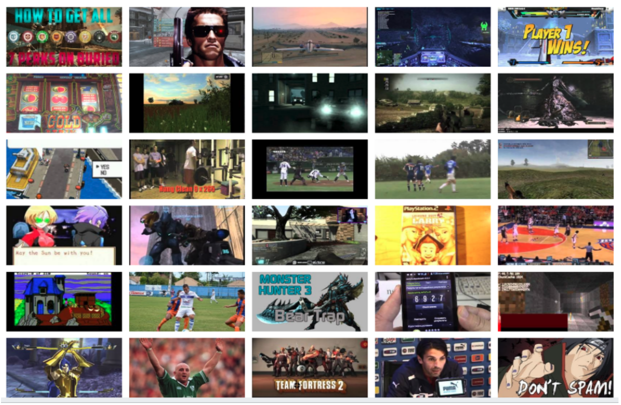


Fig. 2. Sample Dataset of Action games verticle

IV. Expolatory Data Analysis

Exploratory data analysis (EDA) is carried out using Spark's distributed computing capabilities. We conducted a thorough analysis providing visualizations of the dataset's structure and content. EDA has been performed on 1/100 th of the Dataset.

A Verticle is a representation of category under which many classes can exist. . Fig.3, Fig.4, Fig.5 display detailed information of the dataset in terms of verticles. Fig.6 on the other hand visualizes the relationship between multiple verticles.Each edge in the graph represents the classes that are common to two verticles. For example: There are 21 classes that are fall under both the categories of Arts-Entertainment and Sports.

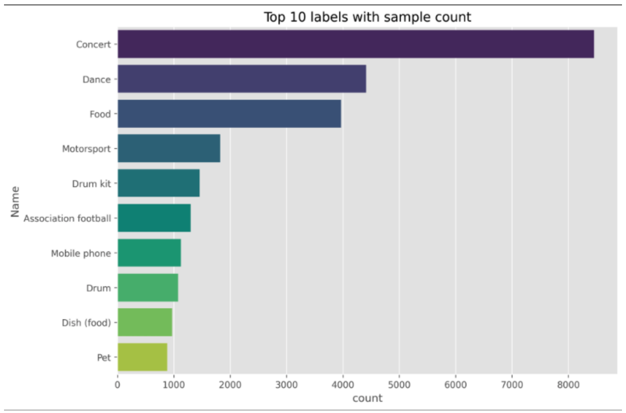


Fig. 3. Top 10 labels with sample count

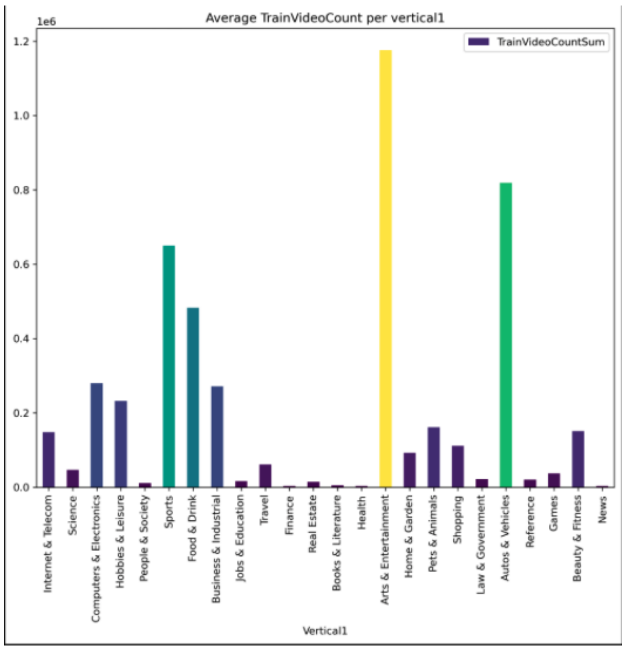


Fig. 4. Average TrainVideoCount per verticle1

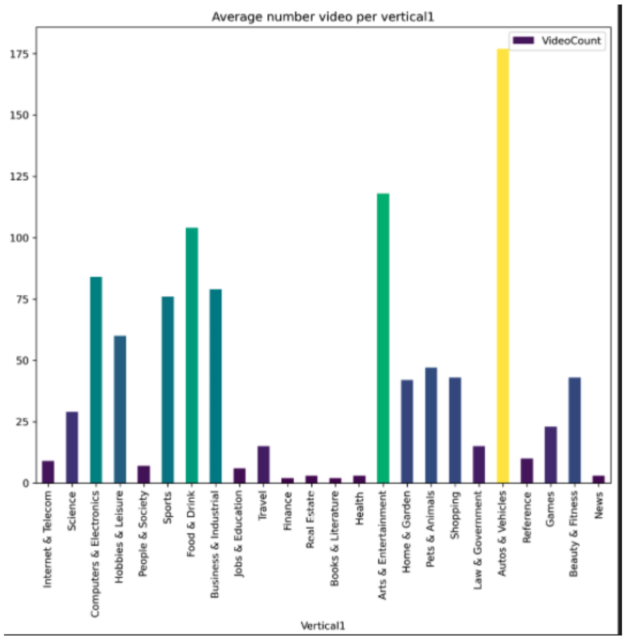


Fig. 5. Average NumberVideo per vericle 1

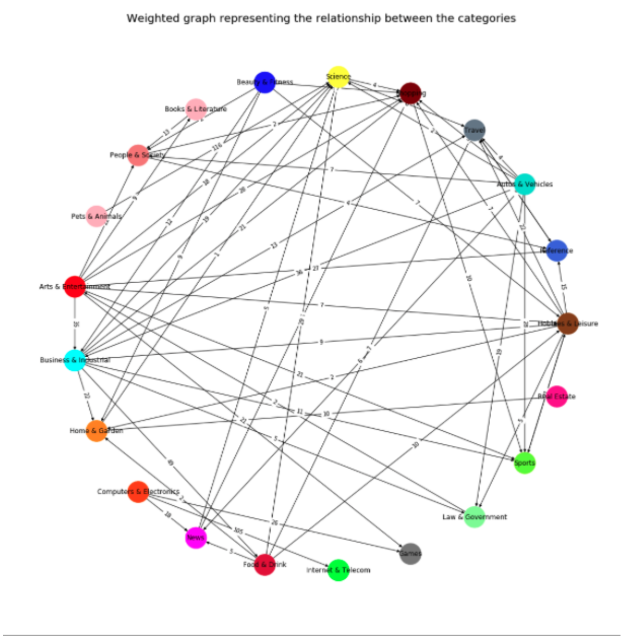


Fig. 6. Weighted graph representing the relationship between categories

YouTube-8M is the largest multi-label video classification dataset, with about 8 million videos totalling to 500K hours with human-verified segment annotations. The dataset has been preprocessed extensively performing Principal Component Analysis (PCA) to reduce dimensionality to 1024,followed by Quanitzation. The pre-processing preserves the important features while enabling faster processing of the data. Even though EDA has been performed on a subset, the results can be extrapolated to the entirety of the dataset.

V. Architecture

The project architecture is designed around Apache Kafka, a distributed streaming platform, and MongoDB, a NoSQL database. Kafka serves as the backbone for real-time data streaming, facilitating seamless communication between producers, which generate data, and consumers, responsible for processing and analyzing it. The architecture prioritizes scalability, fault tolerance, and interactive analysis to deliver a comprehensive solution for real-time data processing and Deep learning model evaluation.

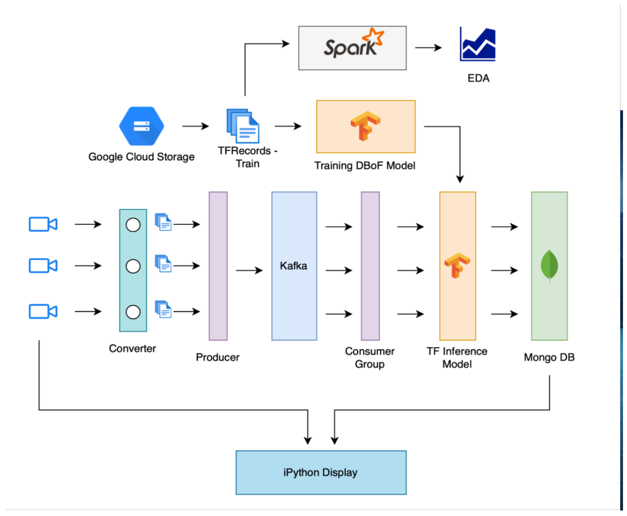


Fig. 7. Architecture

Producer: The Kafka producer captures frames from video files, serializes them, and publishes them to the Kafka topic.We utilize concurrent threads to capture and publish frames from multiple videos simultaneously and serialize frame-bytes to publish a message to Topic.
Topic: The central component in KAFKA setup is the Topic which plays key role in the fault tolerance and distributed mechanisms. Topics in Kafka are divided into partitions, and each partition is an ordered and immutable sequence of records. Partitions allow Kafka to parallelize the processing of messages across multiple brokers and consumers. Each message within a topic is assigned to a specific partition. A Kafka Cluster is composed of Multiple brokers, where each broker is a single server.Brokers are responsible for storage, retrieval and transmission of messages to and from producers and consumers. Brokers can exist on multiple machines and this distributed architecture is easily scalable horizontally according to the work load.
Replication in Kafka is the process of maintaining multiple copies of a partition across different brokers to ensure fault tolerance and data durability.Each partition can have one leader and multiple followers (replicas).The leader handles all reads and writes, while the followers replicate the data to stay in sync. If a broker or partition fails, one of the replicas can be promoted to be the new leader. If leader fails zookeeper elects new leader from the replicas to server data to consumer.
Key constraints with replication and partition are:

- Number of replicas should be less than the number of servers in the cluster.
- Number of consumers cannot be greater than the number of partitions. If number of consumers is more than the number of partitions then some consumers will be left inactive.

Zookeeper: Apache ZooKeeper is a distributed coordination service that plays a crucial role in managing and maintaining configuration information, providing distributed synchronization, and helping in group services within a distributed environment. It is responsible for Management of brokers, Coordination and Leader election.

Consumer: Consumers subscribe to topics and process messages, ensuring they stay up-to-date with real-time data. Their role is to analyze, store, or take actions based on the information received from Kafka producers. Kafka provides a key feature 'offset' which is a unique identifier assigned to each message within a partition of a topic. It represents the position of a consumer in a partition and indicates the last successfully processed message. This offset is used to keep track of the progress of the consumer within a partition of a topic. If consumer goes down in the case of failure or restart, Kafka can track back and start from where it was left off. Configuring producers and consumers in Apache Kafka involves striking a balance between latency and throughput based on specific use cases. For low-latency scenarios, producers can benefit from techniques like batching and compression, while consumers may adjust parameters like fetch.min.bytes and max.poll.records. On the other hand, optimizing for throughput includes increasing batch sizes and adjusting linger times for producers, and adjusting max.poll.records and fetch.max.bytes for consumers.
Deep Learning model: The DL model employed is context-gated DBoF(Deep Bag of Frames) [3], [2]. First Input frame features are fed into a up-projection layer (with shared parameters for all frames). Next, a pooling layer converts frame-level sparse codes into video-level representation.Hidden layers and a classification layer provide the final video-level predictions.

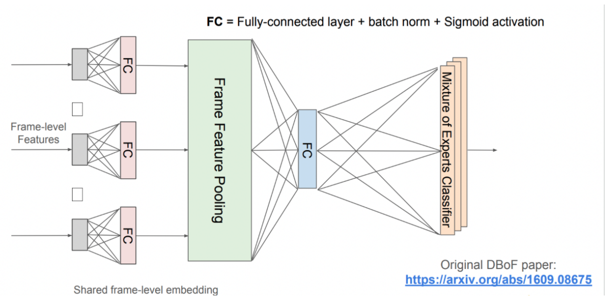
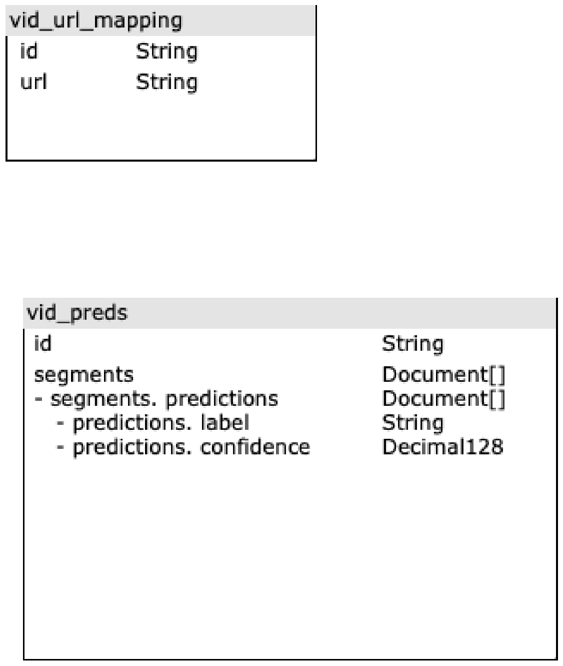

FC = Fully-connected layer + batch norm + Sigmoid activation
Frame-level Features
FC
FC
FC
Frame Feature Pooling
FC
Mixture of Experts Classifier
Shared frame-level embedding
Original DBoF paper:
https://arxiv.org/abs/1609.08675

Fig. 8. DBoF model

MongoDB We employed MongoDB as the database solution to store and manage the predictions generated by the Deep Learning model.



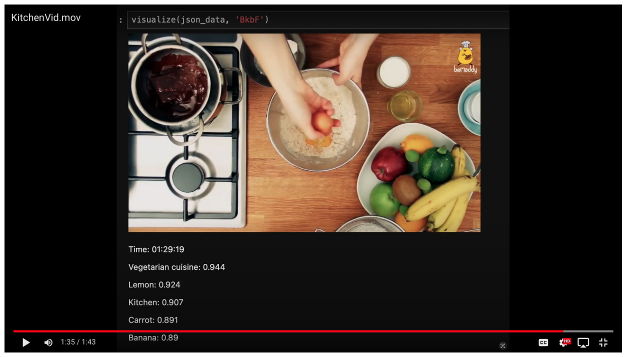
vid_url_mapping	
id	String
url	String

vid_preds	
id	String
segments	Document[]
- segments. predictions	Document[]
- predictions. label	String
- predictions. confidence	Decimal128

Fig. 9. MongoDB schema

VI. Results

The project utilizes interactive capabilities of Jupyter Notebooks along with IPython widgets to enhance the visualization of video dataset predictions. Using IPython widgets, we created an interactive environment within the Jupyter Notebook, allowing us to explore and analyze the results of the Deep learning model's predictions. The use of IPython widgets provided insights into the model's performance on specific segments of the dataset.This interactive approach facilitated a seamless and user-friendly experience for reviewing and validating the model's labeling.



```
visualize(json_data, 'BkbF')
```

```
Time: 01:29:19
Vegetarian cuisine: 0.944
Lemon: 0.924
Kitchen: 0.907
Carrot: 0.891
Banana: 0.89
```

Fig. 10. Visualizing the results

VII. Conclusion

The implemented KAFKA setup aims to enable the DL model to dynamically analyze and create labels for video segments as they are streamed in real time. Intersection of machine learning and streaming technologies, opens avenues for real-world applications such as: Live content labeling: Real-time labeling to filter and moderate content on social media platforms, ensuring compliance with community guidelines and policies. Adaptive video recommendation systems-use cases in Netflix and other streaming services such as personalized content delivery. The challenge here is to design a robust architecture that ensures low-latency, smooth interactive video streaming.

VIII. Limitations and Future Work

Since we do not have an open-source converter for raw video format to TFRecord with enhanced video features, we use a video ID mapping as a workaround:

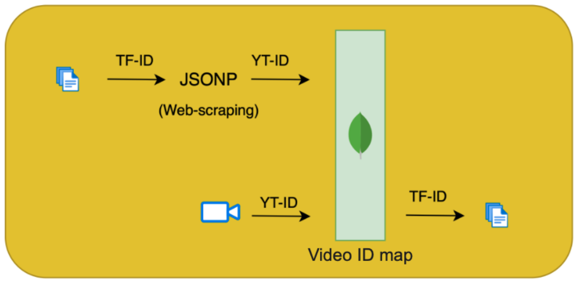


Fig. 11. Video URL scraper

The model's compatibility can be extended with various file formats, beyond the constraints of enhanced TFRecords, which is useful for broader applicability. Extension to open-vocabulary for general purpose usage. Continuously refining the Machine Learning model by incorporating state-of-the-art techniques and algorithms and regularly updating the model with new training data from the YouTube 8M dataset or other relevant sources can help improve its accuracy and ability to handle diverse video content. The model can be fine-tuned for specific use cases such as burglary detection, accident detection, and providing fire hazard warnings derived from CCTV footage, thereby addressing niche use cases with precision and efficacy.

IX. Acknowledgements

The team expresses gratitude to Professor Juan Rodriguez of NYU Tandon School of Engineering for his inputs and the opportunity to work on a challenging project.

References

[1] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Apostol (Paul) Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m: A large-scale video classification benchmark. In *arXiv:1609.08675*, 2016.
[2] Joonseok Lee, Walter Reade, Rahul Sukthankar, George Toderici, et al. The 2nd youtube-8m large-scale video understanding challenge. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.
[3] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. *arXiv preprint arXiv:1706.06905*, 2017.