Similarity Based Video Clip Retrieval

1st Petros Kafkas (mtn2204)
University of Piraeus
NCSR Democritus

***Abstract*—In this report we present the theory and technical aspects behind a *video retrieval* project that we developed as part of the course *Multimodal Machine Learning* for the msc "Artificial Intelligence". We present our basic methodology which consists of extracting image and audio features for a number of video clips serving as our *database* of information and then we compare a random video in order to find the most similar within our database.**

Contents

I. Introduction

Content Retrieval is a task of great importance for both researching purposes as well as industrial applications. In respect to that, extensive research has been carried out on this particular problem and many methods have been proposed and tested (see an overview of the methods here: [2] [1]). In the most basic setting, a content retrieval problem can be summarized as following: a user wants to retrieve some information of interest which is part of database, and he formulates his request for information as a specific query. This query then must be somehow compared to the information stored to the database and the most relevant (ie. *similar*) information must be returned to the user. In general, this process can be time consuming and difficult to be performed correctly. Luckily, machine learning and deep learning can provide solutions for these type of information retrieval problems.
In this project we have created a video clip retrieval application, where the user can give a video-clip of his choice as input to the application and the program compares this video to a number of videos already stored internally (the database). The algorithm returns the five most similar videos to the one that was given as input.

II. Technical Aspects

A. The Process Pipeline

This implementation consists of three (3) discrete parts that work in tandem:

- A Database: this includes ninety (90) of the most popular videos of the recent years, as was published in Spotify.
- A feature Extraction Segment which includes:
  - Audio-based Feature Extractor (powered by *pyAudioAnalysis*)
  - Image-based Feature Extractor (powered by *Clip*)
- A Content Retrieval Segment:
  - Variational Auto Encoder based similarity calculator

The work process can be described as this: following a user defined query (a music video clip) two separate feature extractors are used to produce video-features and audio-features which are then combined into a single feature vector (*early-fusion model*). This combined feature vector is then sent through a VAE similarity calculator which compares the input to a set of prior knowledge (our database) and outputs the results as seen in figure 1.

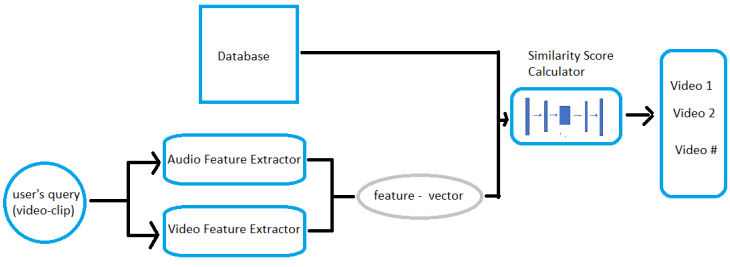

Fig. 1. Implementation Representation

B. Data Preprocessing

In order to prepare the data for the latter feature extraction we utilize the *ffmpeg library* for the handling of the input video files. In more detail, from each video file is extracted a raw audio file (.wav) in mono format at a standard sampling rate of 44100hz.

signal energy:	represents the magnitude of the audio signal (corresponds to "loudness")
mel-frequency cepstrum (MFC)	represents the short-term power spectrum of a sound
Chroma	represents harmonic and melodic characteristics of music

TABLE I
AUDIO FEATURES

C. Audio Feature Extraction

For the audio feature extraction we have used the *pyAudioAnalysis* [3] which is a Python library for audio signal analysis. From each audio we compute the average and the standard deviation of a large number of features (signal energy, zero-crossing rate, pitch, spectral features, mfcc etc. *an analytical list can be found in the project's github repo). In particular, a total number of 138 of individually averaged features is produced for each audio file and this information is stored as an *audio feature vector*. See table I for a short description of the most important audio features.

D. Video Feature Extraction

For the video feature extraction we have used the *CLIP (Contrastive Language-Image Pretraining)* [5] deep network. This network is constituted by two parts, a language encoder and an image encoder. The image encoder is based on a CNN architecture (vision transformer, see fig.2) which is used in order to analyze and obtain visual features at various levels of abstraction. In our project we use this image-encoder in order to process the video files in the following way: we extract image embeddings for one (1) *frame per second* of video. For each video file we obtain a feature matrix with size of [*number of frames of the video * 512 (size of encoded representation features used by the model)*]. For each video we have taken the average of the values for each of the 512 features and so we extract a feature vector of dimension 1x512. This last vector is our video *feature vector*. By taking the average of the features for all frames we inevitably loose information, a better solution would be to concatenate the features per frame. We have not done that due to computational power restrictions.

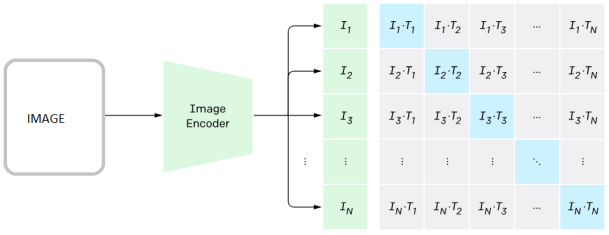

Fig. 2. Clip Image Encoder

E. Similarity Extraction Tool

For measuring the similarity between different video clips we have used a Variational Auto Encoder (VAE). A Variational Autoencoder (VAE) is an unsupervised generative model that takes the basic autoencoder and adds to it a probabilistic model. A VAE model is comprised of two sub-networks: an encoder and a decoder (see fig. 5). The encoder projects the input data into a lower dimension (compression) which is represented by a mean vector and a variance vector which means that it resembles a normal distribution [4]. This compressed latent space can then be sampled in order to reconstruct data by the decoder as shown in figure 4. As we mentioned before, a key difference between a plain autoencoder and a variational is that the latter's latent space is *well organised*. For this reason, the regions within the VAE's latent space obey a probability distribution and distances within this space have a direct correlation to the data produced by the decoder. Based on this property we proposed a similarity measure as the euclidean distance of different points in the VAE's latent space. The closer the two points, the more statistically similar are the samples represented by them and vice versa.

1) VAE training: For the training of the variational auto encoder we used the data we generated from performing feature extraction on our database videos (90 videos). The duration of the training was approximately one-thousand (1000) epochs. The loss function we used in order to measure the efficiency of the auto encoder has two parts: a reconstruction loss and the KL-divergence term (see fig. 3). The KL divergence in essence measures the relative entropy between two distributions and since we pass it as a loss term it *guides* the neural network to learn how to approximate the latent distribution that it generates to a target distribution (in our case the normal distribution).

$$\text{Loss } L = \underbrace{E_{q_\phi(x|z)}[\log p_\theta(x|z)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}(q_\phi(z|x)||p(z))}_{\text{Approximate Normal(0,1)}}$$

Fig. 3. Variational auto Encoder Loss

In more detail, the comparison and similarity retrieval is performed in the following way:

- We have encoded all our prior video-clips using the VAE. The latent representation of the videos is our *database* (see fig. 6).
- The input piece is projected by the same VAE encoder to a point in the latent space.
- We calculate the k-nearest neighbors of the input video's latent representation's point. We take the nearest neighbors to be the most similar videos to the input, based on the justification given in paragraph II-E
- Finally, we perform same basic transformations to the distances between our input data and our output and pass the augmented-distance values to a softmax function, so we get a probability estimation of the similarity as depicted in figure 7

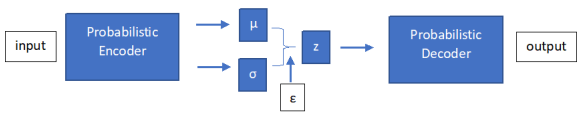

Fig. 4. Variational Auto Encoder

```
VAE(
  (fc1): Linear(in_features=650, out_features=64, bias=True)
  (fc2_mean): Linear(in_features=64, out_features=2, bias=True)
  (fc2_logvar): Linear(in_features=64, out_features=2, bias=True)
  (fc3): Linear(in_features=2, out_features=64, bias=True)
  (fc4): Linear(in_features=64, out_features=650, bias=True)
)
```

Fig. 5. VAE Pytorch Model

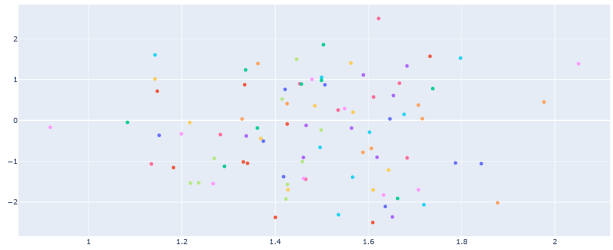

Fig. 6. Latent space - Database Representation

```
track1: Khalid-Young-Dumb-Broke, probability: [49.]
track2: Farruko-Pepas, probability: [24.4]
track3: Elton-John-Dua-Lipa-Cold-Heart, probability: [16.3]
track4: Alec-Benjamin-Let-Me-Down-Slowly, probability: [6.6]
track5: Imagine-Dragons-Bad-Liar, probability: [3.6]
```

Fig. 7. probability of similarity

III. Experimental Results

We have experimented using two different database representations: audio only and audio-video combined. The results of the classification task can vary depending on whether we are using audio-based only classification and combined audio-visual feature vector classification.

A. Experiment A

Remarks about experiment:

- We have deliberately used two video clips that aesthetically are very similar (ie. settings, colors etc.) but with very dissimilar audio tracks: this was chosen as to better showcase the variance in similarity match between the two methodologies. Results are depicted in figures 8, 9:

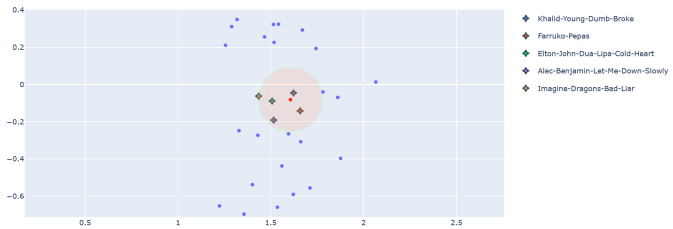

Fig. 8. Audio Only Similarity Matching

B. Experiment B

Remarks about experiment:

- In this experiment we used the same video file as input to both the *audio-only* and the *visual-audio* VAE based similarity calculators and, again, the results vary significantly (fig. 10, 11). Note that the latent distribution representations (the database) varies between audio only and combined audio-visual similarity matching. This is a limitation of our model which is based on an early fusion approach.

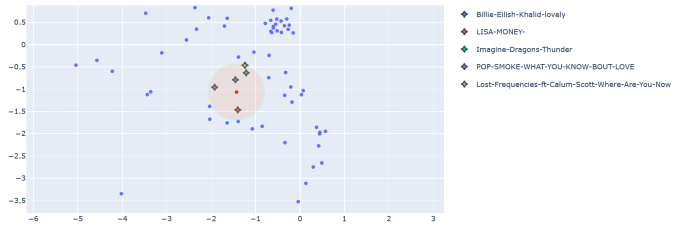

Fig. 9. Audion-visual Similarity Matching

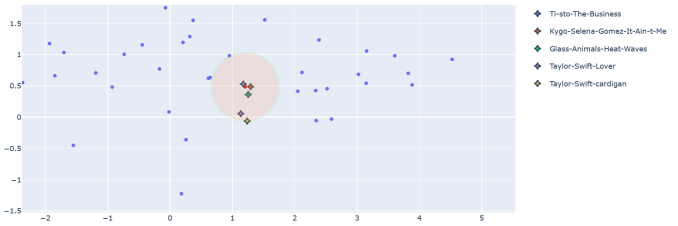

Fig. 10. audio only match

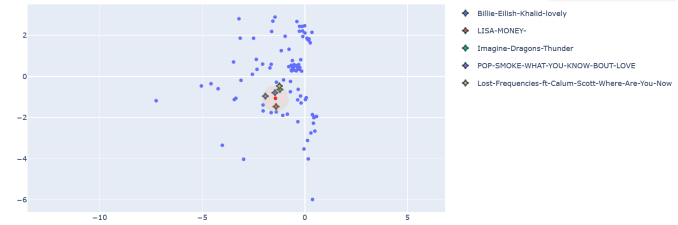

Fig. 11. audio-visual match

C. Equations

The similarity measure that we have used throughout this project (based on proximity on latent space) is given by the following equation. PS is the probability of similarity and SF is a softmax function applied twice, once on the euclidean distance of two pieces in the latent space, and a second time on the value one minus the distance:

$$PS = SF(1 - SF(\sqrt{(z_0^2 - z_1^2)})) \quad (1)$$

IV. Conclusion

In this project we have implemented a content retrieval algorithm based on the statistical properties of a variational auto-encoder. The information is classified based on audio and visual characteristics which we extract by using two well-established neural networks, namely CLIP for the image features and pyAudioAnalysis for the audio features. We have performed several experiments on content retrieval by querying the system with different input videos and by using a dual search process whereby we perform similarity matching based (1) only on audio and (1) audio combined

with video. We have noted that results gathered using the same query but different modalities for the feature vector yield different content-matches as a rule. A limitation of our model is that it requires a different database representation for each separate modality. This is because the latent space representation generated by the autoencoder varies relative to the size and values of the input data. However, since we extract similarity-probability scores for each matching, it is possible to implement a late fusion model. Finally, the algorithm proved to be robust, being able to handle large amounts of data withing small time frames.

V. Feature Work

Some features that we would like to add/explore in a feature revision of the program include the following:

- Implementing text base feature extraction and information retrieval
- Giving the option of *late fusion* instead of only *early fusion* in terms of feature vectors. This can be further refined with the addition of the option of the type of classifier to be used. In this scenario the user would be able to choose if they want an audio based search, a visual or a textual.
- Finally, modyfying the architecture of the auto encoder so that the input are not column vectors but rather image representations. This way we could generate mel-spectogram images of the audio tracks and use this computer vision technique in order to classify audio. A comparison of the efficiency between this and the implementation we did for the project could yield intresting results.

VI. References

References

[1] Mohammed Ansari and Muzammil Mohammed. Content based video retrieval systems -methods, techniques, trends and challenges. *International Journal of Computer Applications*, 112:975–8887, 03 2015.

[2] T. Dharani and I. L. Aroquiaraj. A survey on content based image retrieval. In *2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*. IEEE, February 2013.

[3] Theodoros Giannakopoulos. pyAudioAnalysis: An open-source python library for audio signal analysis. *PLOS ONE*, 10(12):e0144610, December 2015.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.