3VGC: Building and Learning a Tri-Modal Data Set for Video Genre Classification

Battu Surender Harsha
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands
s4120310

Miculiță Andrei
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands
s4161947

Trantas Thanasis
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands
s4130162

Tziafas Giorgos
Department of Artificial Intelligence
University of Groningen
Groningen, The Netherlands
s3913171

***Abstract*—This paper serves as a project report for the course *Machine Learning* by *University of Groningen*. This project's aim is to tackle the task of *video genre classification*; that is to automatically assign any input video clip into one semantic category label (genre). We present our own tri-modal (video, audio and text) data set *3VGC*, collected from *YouTube* videos and comprised of the following genres: vlog, documentary, sports, music, e-sports, TV-show and cartoon. We also present an array of methods for learning this data set through fusing extracted features from each modality, along with an evaluation pipeline and some preliminary experiments for the implemented methods.**

***Index Terms*—Video Genre Classification, Multi-Modal Feature Extraction, Multi-Modal Deep Learning, Machine Learning**

I. Introduction

Video data are an essential part of all multimedia technologies, with a huge volume and a broad variation of content being uploaded, viewed and exchanged by people online every day. This enormous size, variation of content, as well as the big demand from consumers to have smart tools (search engines) to access them, poses a challenge for video database maintainers and multimedia retrieval engineers. In popular domains such as *YouTube*, a manually uploaded tagging method with keywords is used, resulting in community-based feature tags that describe contents of videos, often with a high amount of noise. As a result, an automated scheme for classifying video clips into some generic content categories (genres) seems like a way to address this challenge.

Approaches of video genre classification as well as general classification of videos (e.g. action recognition), such as the ones presented in II, are always based on data coming from one modality (video, audio and subtitle captions) or combination of audio - video. In real open-source challenges online, scientists are trying to classify using only text data (title, video description etc.). Troubled by this fact but also inspired by the demonstrated capacity of multi-modal deep learning models for the task of sentiment analysis over human dialogue videos, we attempt to propagate the video genre classification task into a multi-modal solution.

For this purpose we build a tri-modal video genre classification data set, called *3VGC*, consisting of *YouTube* video, audio and caption data footage of over 400 hours and from eight different genre categories (vlog, documentary, sports, music, e-sports, drama, tv-show and cartoon). Description of the data, statistics and a brief description of the downloading and pre-processing steps are included in section III. A thorough overview of a collection of deep learning methods adapted from the sentiment analysis task for learning the constructed data set follows in section IV. A detailed evaluation pipeline of the implemented methods, as well as some preliminary experiment results on a small-scale binary version of the task is presented in section V. Finally, we reflect on our project and suggest future directions in section VI.

II. Related Work

Previous work in video classification has been done in the fields of action recognition [4], as well as face and text recognition [7] [21], which provide example algorithms for video learning. Iyengar and Lippman [10] used optical flow and frame differencing as methods for detecting movement within a video, as well as another method based on the variation of the *RGB* color space histogram of each frame. Truong et al. [1] use hand-crafted low-level descriptors for video classification. A literature survey by Brezeale and Cook contains a more complete picture of the current state of the art [3].

For classifying video genres based on their content, Dharti M. Bhoraniya ; Tushar V. Ratanpara [2] conducted a survey on different video techniques presented over the past few years and discuss their potentials to extract audio and video features; and the types of fusions performed to combine the extracted data. In [12], Jin et al. attempt to use bi-modal features

of the input video data through encoding them to *MPEG-7* audiovisual descriptors. Other approaches, such as [8] attempt to classify using low-level descriptors extracted from signal processing operations over the video-audio data (color, texture etc. for video, pitch, voice intensity etc. for audio). Karthick et. al. [13] and Chen et. al. [5] train support vector machines on such audiovisual descriptors for long duration *YouTube* videos.

The inspiration for the suggested method is drawn from the context-aware multi-modal models employed in the *sentiment analysis* task [18]. In this work, uni-modal features are extracted from each modality and the resulting features are fused through tensor [22] or hierarchical [15] methods into a common representation. The video data are treated as sequences of smaller video durations (for each utterance in a human dialogue video) and the extracted features for each utterance are contextualized to integrate semantic information of the entire video.

III. Data Set Creation

In this section we present the decisions behind building the data, all pre-processing steps, some statistics of the collected samples as well as a mathematical framework for adaption to the deep learning scheme.

A. *Video Domain & Genres*

a) YouTube Domain: The data set is constructed based on videos hosted on `youtube.com`. For each category, appropriate videos were searched, using the filtered search function to ensure subtitle coverage. Then the videos were added to playlists, which were downloaded using the `youtube-dl` utility and placed into appropriate folders for each class label. They were all converted to the `mp4` format.

b) Genres: The genres are hand-crafted and chosen so that in 8 categories we can sum the most popular generic content categories based on our personal estimates. Also, all the genres can be further sub-categorized into interesting categories (sub-genres) for future inferences. The choices of videos for each genre category include:

1) ***Vlog****:* A video blog is a record of someone's thoughts, opinions or experiences that he/she films and publishes online. This list was created with vlogs of famous people, with millions of views, where subtitles were expected to be found more easily. The general themes can be categorized as daily life, travelling, shopping, building, gaming and beauty.

2) ***Documentary****:* The documentary list comprises of digitally edited narrations of historical events (e.g. war), space exploration, technology, nature and wildlife, manufacturing, modern life and more. Multiple different sources of documentaries are utilized to provide a robust collection of data.

3) ***Sports****:* The sports genre contains recordings of sports events. Examples include soccer matches, American football matches, tennis matches, boxing matches, as well as Olympic sports such as gymnastics, wrestling, weightlifting and judo. All videos were filtered, then hand-checked to ensure 100% subtitle coverage. Audio for his category often includes a single human voice speech narration, crowd cheering and rarely dialogue.

Category	total(hrs)	num.sam.	(μ, σ)(mins)	subs(%)
Vlog	55:24:57	210	15:50, 09:38	77.3
Documentary	72:04:41	96	45:02, 25:08	92.7
Sports	49:19:08	45	65:45, 47:49	100
Music	55:15:00	1345	04:02, 01:22	48
E-Sports	50:19:00	48,	61:59, 12:11	89.6
Drama	56:49:10	34	100:21, 23:36	94.11
TV-Show	50:36:23	113	26:52, 12:21	100
Cartoon	58:40:00	190	16.526, 04:09	70.49

TABLE I: Statistics describing the video samples collected for each genre category. From left to right: a) total duration of all samples in hours, b) total number of individual video samples, c) mean and standard deviation duration of each video sample and d) percentage of samples that do not include noisy subtitles.

4) ***Music****:* A music video is a short video that is combined along with a song for artistic purposes. It usually includes dance, arbitrary scenes and depicting the face of the artist. The videos can also include a lot of graphical effects and camera movements that sync with the audio. Its language is also poetic and rhyming in many cases, which makes it easier to differentiate from many other categories. The captions for this category comprise of the songs lyrics.

5) ***E-Sports****:* E-sports category is a very growing community-based category of videos that is all about competitive video games and its players. All videos are selected recordings of big e-sport tournaments for diffent game titles (*DoTa 2*, *Fortnite* etc.). E-sport videos often contain footage of digitally animated imagery (what the player watches in their monitor) and focuse on gameplay while having an auditory sports-like real-time commentary.

6) ***Drama****:* Drama are usually high quality production videos, as they all come from full feature films with a lot of different contents. The scenes may include dialogue, action sequences, fighting, landscapes etc. This genre is very abstract in nature and it's content is very often intertwined with other genres.

7) ***TV-Show****:* The videos in the TV show category are from various formats of TV shows, such as talk shows, reality shows and game shows. They usually include a presenter and a person/group of people taking turns in talking or performing some action. All of the videos were filtered and hand checked to ensure 100% subtitle coverage.

8) ***Cartoon****:* This category includes animated cartoon series or films, both from popular (Disney, Pixar etc.) but also from more eccentric cartoon animations, so that different art styles of animation are included. Captions for these category often are auto-generated for English and therefore require manual de-noising for alignment with the correct video's timeframe.

B. *Pre-Processing*

After downloading and saving the data, a pre-processing stage is followed so that each video can account to multiple video samples that our learning algorithm can train on and the three modalities are separated and saved individually. Each step in this pre-processing pipeline is described below and

is implemented with a combination of *FFmpeg* and *OpenCV* software packages:

- For each category, all raw video data are split into 5 second duration videos (e.g. a 2 minute video results in a total of $60/5 = 12$ samples) and saved after named appropriately so that their title can be interpreted to access succesive 5 second duration video samples of the same video source. All smaller duration leftover samples are disregarded.
- In order to compress our data volume we introduce a sub-sampling routine that keeps only 4 frames per second of each video sample (1st, 5th, 10th and 15th). The resulting videos have a total of $4{\cdot}5 = 20$ frames and can be thought as a 1 second compressed version of the real 5 second clip. These videos are *MPEG* encoded.
- For all the original 5 second duration videos, their corresponding mono-channel audio waveform is sampled at 192.500 Hz and $MPEG$ encoded.
- By using the *youtube_dl* package we are able to download time-stamped versions of the uploaded captions for each video. A parsing algorithm was used to detect 5 second windows in the time-stamps of the text file and include all captions (separated by space) within each window as a newline in a new file for each video source.

The resulting data set has a total of 322.872 samples for each modality, the video samples organized as 1 second *mp4* videos, the audio samples as the corresponding sampled audio content in *mp4* format and the text samples as newline separated sentences in the parsed captions file for the entire video.

By employing a suitable naming and memory saving convention while downloading and pre-processing the data, we are also able to access contiguous samples from the same video source. We make sure to maintain this utility even after randomly shuffling our data set for the feature extraction learning processes that follows.

C. Formalizing the Data

The resulting data set can be formalized as a set D of size $|D| = 322.872$, containing $N_s = 5$ second duration video data organized as multi-modal samples $s^{(i)},\ i = 1, ..., |D|$. Each sample is a unordered collection of four different attributes $s^{(i)} = \{\mathbf{V^{(i)}}, \mathbf{A^{(i)}}, \mathbf{T^{(i)}}, y^{(i)}\}$, encoding the three pre-processed data modalities (video, audio and text), along with the samples genre class. More specifically:

- The video input can be represented as a 4-dimensional tensor object $\mathbf{V^{(i)}} \in \mathbb{R}^{F\times 3\times h\times w}$, with F being the sub-sampling frequency that we described above ($F = 4$ frames per second), 3 the number of input channels for color image data and $h \times w$ the resolution of the video data. For our implementation, we utilized low resolution images of $h = w = 120$, as we downloaded low-quality versions of the video in order to manage computational and memory costs.
- The audio input can be represented as a 2-dimensional tensor $\mathbf{A^{(i)}} \in \mathbb{R}^{SR\times N_s}$, with $SR = 192.500$ denoting the sampling rate used while pre-processing and $N_s = 5$ the sample duration we have chosen. This results into the audio input being represented also as a continuous vector $\vec{a} \in \mathbb{R}^{577500}$.
- Each textual sample is a concatenation of all English sentences that are captioned for the video in the desired duration of 5 seconds, separated by space. This can be represented as a symbolic sequence of arbitrary length (the total number of words in the concatenated sentences) containing lexical items (words) w from a pre-defined English vocabulary V_{eng}:

$$\mathbf{T^{(i)}} = [w_0, w_1, ..., w_l], \tag{1}$$

  with $w_j \in V_{eng},\ j = 1, ..., l$. Details about the choice of vocabulary are given in the next section IV-C.
- The fourth attribute is the target attribute we want to classify, containing an integer mapping to the genre label of each video sample $y^{(i)} \in \{0, 1, ..., 7\}$.

.

Given that the data are organized contiguously as described in III-B, another way to formalize them that proves useful is to treat training data points as sequences of length $L^{(i)}$:

$$S^{(i)} = [s_0^{(i)}, s_1^{(i)}, ..., s_{L^{(i)}-1}^{(i)}], \tag{2}$$

comprising of consecutive $N_s = 5$ second duration multi-modal samples $s_k,\ k = 1, ..., L^{(i)}$, for the entire duration $L^{(i)}$ of each original video source. By modeling the data in this manner, we are able to treat our training data as discrete time sequences, exploiting the temporal correlation between consecutive video samples in the classification learning process.

IV. Feature Extraction & Learning

In this section we present our implemented methods utilized for developing a tri-modal classifier that learns how to combine extracted features from each modality in order to map an input video to the correct category label. A general schematic of the end-to-end model is demonstrated in Figure 1. We see that the overall process can be divided into two successive steps:

- First, extract uni-modal features from each modality. This step aims at "compressing" our data set so that each modality sample is represented as a fixed-length continuous feature vector that encodes the input data distribution in a low-dimensional vector space. This step is addressed by either employing signal processing based methods for extracting generic features from video - audio data or training deep learning architectures that learn how to extract features from raw data that are useful for classifying their genres.
- After uni-modal features are extracted for each data sample, find a way to fuse them appropriately so that the inter-modality dynamics can be learned and exploited to correctly classify each sample.

In the following subsections we describe the feature extraction methods that were used independently for each modality, a contextualization scheme adapted from literature aiming to

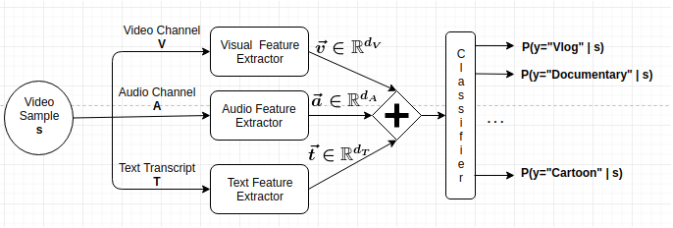

Fig. 1: A schematic of the proposed video genre classification system. Each modality channel is encoded into a continuous space feature vector. Features from all three modalities are fused into a single representation of the video sample, from which a classifier predicts the most likely genre label.

combine extracted features to create context-aware encoding vectors and a collection of methods to fuse these vectors to enhance classification learning.

A. *Visual Features*

*1) **Signal-Based Visual Features**:* Following the literature survey by Brezeale and Cook [3], 5 main methods for extracting visual features have been identified:

- Color-based features - color histograms (of entire video or multiple histograms for multiple regions),
- MPEG - extracting features from the undecoded video, such as discrete cosine transform coefficients and motion vectors
- Shot-based features - detecting shot changes such as hard cuts, fade-outs, fade-ins, and dissolves
- Object-based features - uncommon because of the difficulty of identifying objects, as well as computational expense. From these detected objects, features such as color, texture, size and trajectory are extracted.
- Motion-based features - extracting features based on the detected movement of objects within the video; can be done in computationally cheap ways, such as MPEG motion vectors or more demanding methods such as optical flow.

A vector containing all or a combination of the above features resolve the raw visual feature vector $\vec{v}'$.

*2) **3D Convolutional Network**:* Alternatively, we employ a 3D Convolutional Neural Network (CNN) [11] architecture to learn visual features from the video input. Augmenting the video tensor data with the $F = 20$ dimension that stacks up the sub-sampled frames of the input video will allow for a 3D convolution operation to "compress" the input data into feature vectors that model not only spatial regularities (as 2D convolutions in standard image classification), but also temporal across the successive frames.

Each layer in the 3D CNN consists of the application of a 3D kernel $\boldsymbol{K} \in \mathbb{R}^{f_K \times f_F \times c \times f_h \times f_w}$ in the previous tensor input, followed by a max pooling operation of size $d \times d \times d$ applied only to three dimensions of the convolution output relevant for features (number of frames and size of each frame f_F, f_h, f_w). The output of each layer is f_K feature maps $\boldsymbol{F_i} \in \mathbb{R}^{c \times (f_F - F) \times (f - f_h) \times (f - f_w)}$, $i = 1, ..., f_K$.

For our default implementation, we utilize a single 3D convolution - pooling layer with $f_K = 32$ kernels of size $4 \times 4 \times 4$ with $c = 3$ channels. The pooling window dimension is $d = 3$. The output features maps are flattened to obtain the feature vector $\vec{v'} \in \mathbb{R}^{d'_V}$, which can be given by

$$\vec{v'} = flat(max(d)_{F_i, h_i, w_i}\{\boldsymbol{K_i}(f_F, f_h, f_w) * \boldsymbol{V}(F, h, w)\}) \tag{3}$$

In both methods, the obtained feature vector is fed to a single linear layer with $d_V = 500$ output size, in order to reduce it's dimensionality. A softmax operation on this linear output is considered the video modality's feature vector $\vec{v} \in \mathbb{R}^{d_V}$.

B. *Audio Features*

*1) **Signal-Based Audio Features**:* Following [18] we extract low-level audio features offline by using audio engineering software (we used *librosa*) for all audio samples in our data set. The features are extracted at $30Hz$ with a sliding window of 200 ms. The extracted features include:

- **Generic Spectral Features**. The most commonly used features in audio classification tasks, these descriptors provide information about the Fourier transform spectrum of the input audio data. We extract: 1) 12-th order *Mel-Frequency Cepstral Coefficients* vector, 2) the fundamendal frequency, 3) signal energy and 4) zero-crossing rate. In the resulting feature vectors we include statistics (mean, variance) of each $30Hz$ sampling window accross the 5 second duration.
- **Speech-Specific Features**. For this we include pitch and voice intensity performed after Z-standardization.
- **Music-Specific Features**. Features include chroma and tonnetz, both as concatenations of a flattened version of the extracted 2D data.

Stacking up all the extracted feature vectors for the entire sample duration of 5 seconds amounts to the raw audio modality feature vector $\vec{a'} \in \mathbb{R}^{d_A\prime}$, with a total of $d'_A = 6413$ features. A trainable two-layer feed-forward network with hidden size 2048 and output size $d_A = 500$ followed by a softmax operation is applied on the raw feature vector. The softmax output is considered the audio modality's feature vector $\vec{a} \in \mathbb{R}^{d_A}$. The *Gaussian Error Linear Unit* (GELU) activation function [9] is utilized for the hidden layer.

C. *Textual Features*

*1) **Word Embeddings**:* Word embeddings have been established as the state-of-the-art method for representing text in continuous form since the *word2vec* algorithms [16]. These algorithms learn how to represent each word in a language's vocabulary as a vector so that words with similar semantics are mapped to points with a small distance in a Euclidean vector space. As a first part of our textual feature extractor, we utilize a pre-trained *GloVe* embedding layer [17] for the English language, with vector size of 300. Following the formalism of III-C, this mapping can be represented as a

function $g : V_{eng} \rightarrow \mathbb{R}^{300}$ upon the sequence $\mathbf{T} = [w_0, ..., w_l]$, so that:

$$g(\boldsymbol{T}) = \left(g(\vec{w}_0) \quad g(\vec{w}_1) \quad ... \quad g(\vec{w}_l)\right)^T \in \mathbb{R}^{l \times 300} \quad (4)$$

*2) **1D Convolutional Network**:* Features can be obtained from the textual modality by applying a 1D CNN architecture in the sequence of word embeddings. All word embeddings are concatenated with a padding window of maximum 20 words into a $l \times 300$ sized vector that is the input to the CNN. Two convolutional layers upon this input will create dense low-dimensional abstract representations of the words that comprise the input video caption, further equipped with implicit semantics due to the *GloVe* representation.

The CNN consists of two convolutional layers followed by max-pooling operations of window size d. The first layer comprises of two parallel kernels $\mathbf{K_1}, \mathbf{K_1}'$ of size f_1, f_1' with f_{K_1} feature maps and the second of one kernel $\mathbf{K_2}$ of size f_2 with f_{K_2} feature maps. The intuition behind this architecture is to slide over the input phrase using two different resolution windows f_1, f_1', and further abstract the concatenation of the two different initial representations, attempting to model longer dependencies between words in the video caption. The output of this feature extractor is given by the following equations for all $i = 1, ..., f_{K_2},\ j = 1, ..., f_{K_1}$:

$$CNN(g(\boldsymbol{T})) = max(d)_j\{(\boldsymbol{K_1} * g(\boldsymbol{T}))\ ;\ (\boldsymbol{K_1}' * g(\boldsymbol{T}))\}, \quad (5)$$

$$\vec{t}' = flat(max(d)_i\{\boldsymbol{K_2} * CNN(g(\boldsymbol{T}))\}), \quad (6)$$

For our default implementation we utilize $f_{K_1} = 50, f_{K_2} = 100$ feature maps, with kernel sizes $f_1 = 3, f_1' = 6, f_2 = 3$ and a max-pooling window of $d = 2$. The *GELU* activation function is used.

*3) **Self-Attention Encoder Network**:* The most recent state-of-the-art techniques regarding sequence processing suggest the use of the *Multi-Head Attention* mechanism, resulting in the *Transformer* encoder-decoder architectures [20] that operate in sequences in parallel (versus the iterative recurrent methods), utilizing attention operations only. For any input vectors $\vec{k}, \vec{q}, \vec{v}$ (that can be thought of as queries $\vec{q}$ over a dictionary of keys $\vec{k}$ with values $\vec{v}$), the multi-head attention operation gives:

$$MHA(\vec{k}, \vec{q}, \vec{v}) = softmax(\vec{q} \cdot \vec{k}) \cdot \vec{v} \quad (7)$$

We employ a single layer of a Self-Attention encoder, as suggested in [20], operating on the input *GloVe* embedding so that $\vec{q} = \vec{k} = \vec{v} = g(\vec{\boldsymbol{T}})$ and feeding its output to the above CNN for feature extraction. The intuition behind the self-attention mechanism is that for each $\mathbb{R}^{300}$ word vector, a new vector is built that represents in continuous manner a probability distribution over all the other $l - 1$ vectors of the $l \times 300$ matrix g of how much they must be "attended" for the encoding of this specific vector. This adds sentence-level semantic information encoded in these attention vectors, extending the word-level semantics of the word embeddings. The overall output of the encoder $\vec{e} \in R^{d_m}$ is then given by:

$$\vec{e}(g(\boldsymbol{T})) = \boldsymbol{W_0} \cdot [MHA(\boldsymbol{W_1} \cdot g(\boldsymbol{T}), \boldsymbol{W_2} \cdot g(\boldsymbol{T}), \boldsymbol{W_3} \cdot g(\boldsymbol{T}) + g(\boldsymbol{T})] \quad (8)$$

with $\boldsymbol{W}_1, ..., \boldsymbol{W_3}$ denoting linear trainable weights in $\mathbb{R}^{300 \times d_m}$ and $\boldsymbol{W}_0 \in \mathbb{R}^{300 \times d_{ff}}$.

For our default implementation we utilize a self-attention layer with dimensionalities $d_m = 300, d_{ff} = 1024$ and four heads in the encoder (actual model size $4 * d_m = 1200$). In both methods, the output of the CNN $\vec{t}'$ is fed to a single layer with output size $d_T = 300$ and a softmax activation, resulting in the text modality's feature vector $\vec{t} \in \mathbb{R}^{d_T}$.

D. Contextualization

Following the application of recurrent neural encoder-decoder architectures to text processing tasks [19], where an encoder architecture is used to create a single context-aware vector representation of an entire phrase by sequentially iterating over all vector rerpesentations, we want to further model the temporal correlation of our data by using larger video durations as "phrases", containing single 5 second duration samples as "words".

Our choice of $N_s = 5$ seconds for a video duration and therefore the great number of samples that exist in our data set allows us to insert a different hierarchy in out training data (what are considered individual training data points in our set) and treat videos as sequences of multi-modal samples, as described in the second formalism of III-C.

We employ a sequence sample window of $L = 6$ and account 6 successive video samples from the same original source to form the single training input $S = [s_1, ..., s_L]$ for our model. After the uni-modal feature extraction stage, each individual sample of the sequence will be encoded into three uni-modal feature vectors $\vec{v} \in \mathbb{R}^{d_V}, \vec{a} \in \mathbb{R}^{d_A}, \vec{t} \in \mathbb{R}^{d_T}$, or $\vec{m} \in \mathbb{R}^{d_m},\ m = \{v, a, t\}$. Each modality feature vector $\vec{m}$ is fed to a single bi-directional *Gated Recurrent Unit* (GRU) [6] layer with hidden size h and the concatenation of the last step built representation vectors (for the last sample s_L when iterating from the left and the first s_1 when from the right) is used as the context-aware encoding of the input video sequence S. This single step output $GRU_{L=6}(\vec{m}) \in \mathbb{R}^{2h}$ is fed into a linear layer $\mathbf{W}$ with output size d_0 and the softmax operation is performed. The resulting vector $\vec{c} \in \mathbb{R}^{d_0}$ is considered the *context vector* of the input sequence modalities:

$$\vec{c_m} = softmax(\boldsymbol{W} \cdot GRU_{L=6}(\vec{m})),\ \ m = v, a, t \quad (9)$$

For our default implementation we use a GRU hidden size of $h = 300$ and a softmax output of $d_0 = 200$.

E. Multi-Modal Fusion

The second module of our classifier is a fusion function $f : \mathbb{R}^{d_0 \times d_0 \times d_0} \rightarrow \mathbb{R}^{d_f}$ that attempts to combine the context vectors $\vec{c_m}$ of each modality to a single representation $\vec{f} = f(\vec{v}, \vec{a}, \vec{t}) \in \mathbb{R}^{d_f}$. Our implemented methods include:

- *Early Fusion*, which is the simple concatenation $\vec{f} = \vec{c_v}; \vec{c_a}; \vec{c_t}$ of all three modality context vectors ($d_f = 3d_0$),
- *Max Fusion*, which is the element-wise max operation

$$f_i = argmax_i\{c_{vi}, c_{ai}, c_{ti}\},\ i = 1, ..., d_0 \quad (10)$$

upon the three modality context vectors ($d_f = d_0$),

- *Mean Fusion*, which is the mean operation

$$\vec{f} = 1/3 \cdot (\vec{c_v} + \vec{c_a} + \vec{c_t}) \tag{11}$$

upon the three modality context vectors ($d_f = d_0$),

- *Tensor Fusion*, following [22], where the fusion vector is constructed as the three-fold cartesian product of the three modality context vectors:

$$\vec{f} = \begin{bmatrix} \vec{c_v} \\ 1 \end{bmatrix} \bigotimes \begin{bmatrix} \vec{c_a} \\ 1 \end{bmatrix} \bigotimes \begin{bmatrix} \vec{c_t} \\ 1 \end{bmatrix} \tag{12}$$

with $\bigotimes$ notating the outer product between vectors. The final product is the 3D cube of all possible combinations of uni-modal context features $\vec{c_m}$ with seven different semantic sub-regions (for six bi-modal and one tri-modal combination).

F. *Classifier*

Our classifier is a simple linear layer mapping input fused vectors $\vec{f}$ to eight outputs, one for each classified genre category. The softmax operation then provides us with the estimated probability distribution of genres upon the input video sequence. For the tensor fusion method, different handling of the 3D cube fusion output must be done to exploit the semantic spatial information within. In our default implementation, we simply flatten this cube ($d_f = d_0^3$) and add depth to our classifier with linear layers and *GELU* activations.

V. Experimental Design & Results

As our learning model consists of several independant modules (uni-modal feature extractors, contextualizer, fusion etc.), a specialized experimental pipeline for evaluating each module has been designed and is described analytically in this section. Unfortunately, due to computational resources constraints a thorough experimental evaluation of the full scale feature extraction and fusion models has not been performed, even though software for all has been implemented (can be found here). Instead, we perform a series of preliminary experiments for our feature extraction methods in a split of our data set that comprises of only two categories (e-sports, cartoon), using our default implementations of our models that attempt to solve a binary genre classification version of the task.

A. Evaluation Pipeline

For training our models we utilize the *categorical cross-entropy* loss function, as standardly applied in multi-class classification and the *Adam* optimization algorithm [14] with a learning rate of .001, beta parameters $\beta_1 = 0.5, \beta_2 = 0.99$ and a weight decay of 0.001. Dropout layers in all post-feature extraction networks have been added, with the exception of the classifier, with a dropout rate of 0.1. All models are trained until their training loss saturates and saved at their best validation performance. The evaluation metric utilized is the standard top-1 predicted label accuracy over a smaller split of our data. We follow a standard split of $80\% - 20\%$ into training and validation data sets. All reported results are the percentage of correctly classified video samples in the validation split.

For evaluating our feature extraction modules, we separate the three data modalities and train our models on each modality separately. The output of each uni-modal feature extractor is connected to the classifier. All models are trained and evaluated as described above and the best performing models are saved. This experiment sheds light as to which of the implemented methods of section IV are the best feature extractors, by demonstrating how far can a single classifier go by looking at them without any contextualization or fusion method included. After passing a single forward pass of each data sample into the best models and saving the activations of the pre-classifier layer (probability vectors $\vec{v}, \vec{a}, \vec{t}$), we compress our data set to a single vector representation per modality, as a way to further research the effect of contextualization and different fusion methods in a more computationally efficient way, using these vectors as pre-trained extracted features.

For evaluating contextualization, we train two versions of our classifier in the pre-trained features, one training on single 5 second duration video samples s and one training on $L = 6$ successive samples $S = [s_1, ..., s_L]$, with contextualization applied on the second model for re-encoding the extracted features for 6 samples into a single context-aware representation. We compare the performance of the two models by using the same fusion method (Early Fusion) and evaluate them on their corresponding validation splits, which is the same data but organized differently (samples vs. sequences of 6 samples).

After witnessing the positive effect of contextualization (or dissmising it), we train different models for all different fusion methods to compare their effect on classification, using the best model of the previous step and the pre-trained extracted features. Finally, an end-to-end version of the model for the best fusion method is trained, during which model parameters for the feature extraction, fusion and classification steps are all trained at once.

B. Preliminary Results

For our experiments, we follow the above evaluation pipeline for training data from the two categories for a total of 78.468 samples, or 13.078 sequences of $L = 6$ samples. For all models the default parameters are used across all experiments, with the exception of signal-based features where it was proven useful to use deeper feed-forward architectures before the softmax operation (with arbitrary chosen hidden sizes). Due to constraints in computational resources and time, results are reported after a single training session and no hyperparameter tuning trial-and-error trials were taken. As a result, the reported numbers are given as not to be representative of the full potential learning capacity of our models but rather provide an intuition as to which feature extraction algorithm performs relatively better on our data, if contextualization helps etc.

In Table II we demonstrate the top-1 accuracy of each uni-modal feature extraction method on a 20% split of our two-

Modality	Method	Accuracy
V	Signal-Based	*NYI*
V	3D CNN	74.76%
A	Signal-Based	80.30%
T	1D CNN	86.81%
T	Self-Attn + 1D CNN	88.89%

TABLE II: Preliminary results on uni-modal feature extraction methods for binary genre classification (e-sports vs. cartoon). *NYI* denotes Not Yet Implemented experiment.

Modality	Method	Accuracy
V	context.	83.45%
A	context.	88.61%
T	context.	88.63%
V+A	EF	74.13%
V+A	context. + EF	83.15%
A+T	EF	86.90%
A+T	**context. + EF**	**89.89%**
T+V	EF	86.12%
T+V	context. + EF	88.08%
V+A+T	EF	84.78%
V+A+T	**context. + EF**	**88.69%**

TABLE III: Preliminary results on contextualization effect using the Early Fusion (EF) method for uni-modal, bi-modal combinations and tri-modal features for binary genre classification (e-sports vs. cartoon).

label data set. We observe that the textual features are the most useful for uni-modal classification (as it is expected, cartoon dialogue is much more distinctive from video game narration captions in comparison e.g. with the video modality which includes animated pictures for both categories). As these are preliminary numbers, the only point that we can highlight is that self-attention seems to indeed enhance textual feature learning but not significantly.

In Table III we demonstrate the top-1 accuracy of two models trained on the pre-trained extracted features once with and once without contextualization. In all cases we observe the positive effect of contextualization (with exception to the video - audio bi-modal combination, where the fused video and audio vectors seem to "confuse" our model more than raw audio uni-modal features). This is also demonstrated by the fact that the contextualized audio - text bi-modal combination is the most accurate, but only with a small margin from the contextualized tri-modal model.

In Table IV we provide a comparative report of the different implemented fusion methods trained on the pre-trained extracted features. The high dimensionallity of the flattened $\mathbb{R}^{200^3}$ tensor fusion output accounts to the reported underfitting result, which is shown in [22] that performs better than classic early or element-wise fusion methods.

The results included in this section are preliminary and therefore cannot in any insightful way provide some evaluation conclusion for our constructed data and our methods. After a complete evaluation pipeline of the end-to-end model with tuned hyperparameters on the complete data set, one can has a say as to the performance of our suggested model. However, most of the results seem to suggest the positive effect of contextualization as well as multi-modal fusion to the reported accuracy.

Fusion Method	Accuracy (pre-trained)	Accuracy (end-to-end)
Early	89.69%	*NYI*
Max	88.61%	-
Mean	85.03%	-
Tensor	80.19%	-
Hierarchical	*NYI*	*NYI*

TABLE IV: Preliminary results on fusion methods and end-to-end vs. pre-trained model comparison for binary genre classification (e-sports vs. cartoon). *NYI* denotes Not Yet Implemented experiment.

VI. Discussion

In this section we reflect on our experiences and suggest some future steps.

A. Reflection

Some comments about our personal experiences with working in this project include:

- Building your own data set for a machine learning task is hard... After collecting and de-noising the data, the preprocessing step in reality sets a huge software infrastructure load, as data from different sources must interface with different software modules and be structured in way compatible with the machine learning algorithms.
- Even though building and training your own deep learning models bottom-up feels very straightforward and fast using modern toolkits (we use *PyTorch*), validation feels a bit underwhelming, leaving you with a number of sometimes ambiguous interpretation as a result of a long process of data preparation, formalization, software implementation etc. The whole project's evaluation scheme feels a bit "self-referential", using learning algorithms to evaluate the quality of data we have gathered and evaluating the quality of such algorithms based on the same data...
- Working with huge data sets and neural network models of many parameters (as our tri-modal end-to-end model) can only be sufficiently done on computers with big *NVidia* GPU capabilities. For augmenting our baby two-genre model (which was trained on a *GeFORCE 940 MX* of a laptop) to the entire data set we have to work on the *Peregrine* cluster of *University of Groningen*, to which we already have moved our data collection and prepare the experiments.

B. Future Directions

The obvious step ahead is to complete the described evaluation pipeline on the entire data set, select the best performing models though hyperparameter search, discriminate the best combination of methods and train an end-to-end version of it. Attempting to use the same model for transfer learning other video data sets or training other state-of-the art video classification models on the *3VGC* data set will also provide an insightful measure of the generalization capacity of the model and the classification-quality of our collected data respectively.

Finally, the implementation of an front-end interface for inference use on videos from other domains can be prove a useful tool for future use.

REFERENCES

[1] Ba Tu Truong and C. Dorai. Automatic genre identification for content-based video categorization. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 4, pages 230–233 vol.4, Sep. 2000.

[2] D. M. Bhoraniya and T. V. Ratanpara. A survey on video genre classification techniques. In *2017 International Conference on Intelligent Computing and Control (I2C2)*, pages 1–5, June 2017.

[3] D. Brezeale and D. Cook. Automatic video classification: A survey of the literature. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38:416 – 430, 06 2008.

[4] J. Carreira and A. Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[5] J.-F. Chen, H. Lu, R. Wei, C. Jin, and X. Xue. An effective method for video genre classification. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '10, page 97–104, New York, NY, USA, 2010. Association for Computing Machinery.

[6] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[7] N. Dimitrova, L. Agnihotri, and G. Wei. Video classification based on hmm using text and faces. In *2000 10th European Signal Processing Conference*, pages 1–4, Sep. 2000.

[8] H. Ekenel, T. Semela, and R. Stiefelhagen. Content-based video genre classification using multiple cues. 01 2010.

[9] D. Hendrycks and K. Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415, 2016.

[10] G. Iyengar. Models for automatic classification of video sequences giridharan iyengar and andrew lippman. 12 1997.

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, Jan 2013.

[12] S. Jin, T. M. Bae, J. Choo, and Y. Ro. Video genre classification using multimodal features. pages 307–318, 01 2004.

[13] S. Karthick, s. Abirami, S. Murugappan, S. Mariappan, and B. Ramachandran. Automatic genre classification from videos. 325:389–401, 11 2015.

[14] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[15] N. Majumder, D. Hazarika, A. F. Gelbukh, E. Cambria, and S. Poria. Multimodal sentiment analysis using hierarchical fusion with context modeling. *CoRR*, abs/1806.06228, 2018.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013. Curran Associates Inc.

[17] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *In EMNLP*, 2014.

[18] S. Poria, E. Cambria, D. Hazarika, N. Majumder, A. Zadeh, and L.-P. Morency. Context-dependent sentiment analysis in user-generated videos. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 873–883, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[19] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

[21] G. Wei, L. Agnihotri, and N. Dimitrova. Tv program classification based on face and text processing. In *2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532)*, volume 3, pages 1345–1348 vol.3, July 2000.

[22] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L.-P. Morency. Tensor fusion network for multimodal sentiment analysis. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1103–1114, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics.