Multimodal Joint Emotion and Game Context Recognition in League of Legends Livestreams

Charles Ringer*†, James Alfred Walker†, *Senior Member, IEEE*, Mihalis A. Nicolaou*‡

*Department of Computing, Goldsmiths, University of London, London, UK SE14 6NW
†Department of Computer Science, University of York, UK, York, UK YO10 5DD
‡Computation-based Science and Technology Research Center, The Cyprus Institute, Cyprus
Email: c.ringer@gold.ac.uk, james.walker@york.ac.uk, m.nicolaou@cyi.ac.cy

***Abstract*—Video game streaming provides the viewer with a rich set of audio-visual data, conveying information both with regards to the game itself, through game footage and audio, as well as the streamer's emotional state and behaviour via webcam footage and audio. Analysing player behaviour and discovering correlations with game context is crucial for modelling and understanding important aspects of livestreams, but comes with a significant set of challenges - such as fusing multimodal data captured by different sensors in uncontrolled ('in-the-wild') conditions. Firstly, we present, to our knowledge, the first data set of *League of Legends* livestreams, annotated for both streamer affect and game context. Secondly, we propose a method that exploits tensor decompositions for high-order fusion of multimodal representations. The proposed method is evaluated on the problem of jointly predicting game context and player affect, compared with a set of baseline fusion approaches such as late and early fusion. Data and code are available at https://github.com/charlieringer/LoLEmoGameRecognition**

***Index Terms*—Livestreaming, multimodal fusion, multi-view fusion, affective computing.**

I. Introduction

Livestreaming is an exciting and emerging area of video games entertainment. People find watching other people play games compelling [1], as evidenced by the popularity of streaming services like `TWITCH.TV`[1]. Typically, a live stream entails the broadcast of a set of both visual and auditory data. This includes game footage along with a webcam overlay showing the streamer (Fig. 1), as well as auditory data which includes in-game audio as well as speech and non-verbal cues. As a result, livestreaming presents a rare opportunity for studying streamer emotion and affect at the same time as the stimulus for this emotion, i.e. their game-play experience.

Modelling livestreams is an inherently 'in-the-wild' paradigm, using organically generated real-world data [2], [3], and therefore subject to many complicating factors such as visual and audio occlusions. Such occlusions can either be temporary, e.g. the streamer looking away from the camera, or permanent, e.g. overlays placed over the game scene or the music that the streamer is listening. Other difficulties include streamers having their webcams at a range of angles, using varying levels of lighting, and choosing different volume levels between their voice, music and the game audio. Additionally, modelling streamer affect and game context is a multimodal, or multi-view, problem where a key challenge is joining information from multiple views, e.g. webcam, game footage and audio, to form a single model. It is possible that these complicating factors contribute to the lack of past study into audio-visual stream data, but we feel that despite this there are compelling reasons to study livestreaming. In fact, one goal of this work is to invigorate the research community and spark interest in this topic.

This paper presents two contributions. Firstly, a data set of streamers playing the popular Multiplayer Online Battle Arena (MOBA) game *League of Legends*[2] is presented, along with annotations for both streamer affect and in-game context. Secondly, a novel method for fusing multiple views by modelling high-order interactions using a 'Tensor Train layer' [4] is presented and evaluated on this dataset. Additionally, we present a comparison of this method with several existing fusion approaches, thus providing baseline results for the dataset.

In this work, experiments are presented for all fusion methods in terms of modelling affect and game context both jointly as well as separately. In essence, this paper acts as a platform inviting further research into this problem, and a starting point for the study of supervised learning in terms of modelling the multiple facets of livestreams and their interactions.

II. Related Work

A. Analysis of Game Streams

Livestreaming is a young technology and so currently lacks a wealth of former work dedicated to it. Existing studies into modelling livestreams have focused on detecting 'highlights', i.e exciting or important moments, in streams. Chu *et al.* [5], [6] looked into various facets of *League of Legends* esports broadcasts, building models of highlight detection, focusing on modelling hand-crafted features such as the number of

This work was supported by the EPSRC Centre for Doctoral Training in Intelligent Games & Games Intelligence (IGGI) [EP/L015846/1] and the Digital Creativity Labs (digitalcreativity.ac.uk), jointly funded by EPSRC/AHRC/Innovate UK under grant no. EP/M023265/1.

[1]www.twitch.tv

[2]Riot Games, 2009

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

players on screen, and event detection, utilising text recognition on in-game messages. Additionally, our previous work [7] focused on using unsupervised learning to detect highlights in livestreams of *Player Unknown's Battlegrounds* using a technique similar to 'feature fusion', as presented in this paper, to fuse measures of novelty across views over time.

Often, instead of studying streams themselves, past research has focused on the social and community aspects of livestreaming. Examples include Smith *et al.* [8], Recktenwald [9] and, Robinson *et al.* [10] who studied streamer - viewer interactions. Others have examined the behaviour of streams in general, taking a higher level view by looking at features such as the number of viewers and the length of streams, such as Kaytoue *et al.* [1] and Nascimento *et. al* [11].

B. Studies of Player Experience Through Visual and Audio-Visual Data

Most prior work into audio-visual models of player experience do not use livestreaming platforms as their source of data. For example, the Platform Experience Dataset [12] is a data set of players playing the platformer game *Infinite Mario Bros*. This data set has been utilized in several studies, for example, Shaker *et al.* [13] developed player experience modelling techniques, while Asteriadis *et al.* [14] used this data set to cluster player types. Additionally, off-the-shelf affect models have been used to study player experience, for example Blom *et al.* [15] explored how these models could be used to personalise game content and Tan *et al.* [16] performed a feasibility study exploring player experience modelling via visual cues, concluding that facial analysis is a rich data source which warrants more exploration.

C. Multimodal Machine Learning

The problem of analysing multiple views is an open challenge in the affective computing and computer vision communities and describe situations where multiple data sources, e.g. audio and images, depict different views of the same event. Generally speaking, attempts to tackle this problem have focused on the question of fusion - how can multiple views be joined, or fused, into a single model. There is no general consensus on what the most appropriate fusion technique is with many studies attempting to perform fusion in different ways within the network architecture [17], [18]. Some of the most popular techniques for fusion in 'end-to-end' systems, where the inputs are raw data extracted from the video and the outputs are the classifications, include early (feature) fusion and late (decision-level) fusion. While early fusion refers to concatenating raw features or representations (e.g. [19], [20]), late fusion is usually performed on bottleneck features (e.g. [21]). Note that other fusion approaches exist, e.g. model fusion, where separate models are utilized for each view and subsequently aggregated (e.g. EmoNets [22]). However, such approaches are not naturally suited to an end-to-end system as two stages are often required during training.

A significant advantage of utilizing multimodal data, as Ngaim et al. [19] note, is that exploiting information from multiple views can significantly aid learning from noisy and imperfect data (e.g. when audio is noisy). This suggests that multimodal fusion may be well suited to the problem at hand, where noise is introduced due to uncontrolled conditions (discussed in Section I). This is also crucial for modelling behaviour and affect, as it is well known that audio information is more suited to predicting emotional arousal, while visual data is more suited for modelling valence [23]. Therefore, fusing these views allows for holistic modelling of affect. We refer the interested reader to [17] and [18] for more details on the literature in multimodal learning.

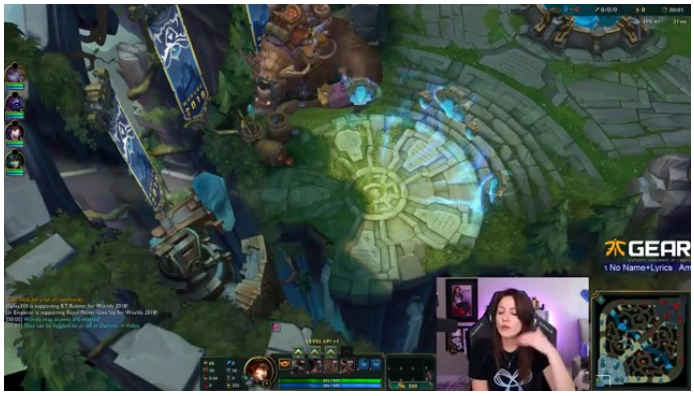

Fig. 1. Example screenshot from a *League of Legends* livestream.

D. Tensor Decompositions

This paper presents a method for utilizing tensor decompositions as a fusion mechanism in order to model high order relationships between multiple views. For the purposes of this work, a tensor refers to the general term for an array of values where the rank refers to the dimensionality of the array e.g. a vector is a rank-1 tensor and a matrix is a rank-2 tensor. The proposed approach is similar to Zadeh *et al.* [24], where Tensor Fusion was employed to fuse multiple views in a sentiment analysis task. Likewise, this work utilises a Tensor Train [25] [26] layer that decomposes a tensor into a set of simpler and smaller ones in a weight-efficient manner. Other researchers have used similar approaches to aid different tasks, e.g. Yang *et al.* [4] utilizes a (recurrent) Tensor Train layer, directly on the pixels of a video replacing the convolutional layers. Kossaifi *et al.* [27] used similar tensor methods on the unflattened output of a Convolutional Neural Network and showed that these types of decomposition can be trained in an end-to-end model. Anandkumar *et al.* [28] provide a comprehensive overview of tensor decompositions for learning latent variable models.

III. Data Set

League of Legends is a MOBA game where two teams of five players compete with the goal of reaching and destroying the opposing team's base. It is one of the most popular esports, with at least 5 professional leagues and numerous global competitions[3]. In addition, it is an incredibly popular game

[3]https://eu.lolesports.com/

for streamers and is regularly in the top 3 most popular games being streamed on TWITCH.TV[4]. Video data[5] was gathered from streamers playing *League of Legends* on TWITCH.TV. The data set consists of 10 streamers, five male and five female, streaming in English, and using TWITCH.TV. 20 minutes of footage was gathered from 3 games for each streamer for a total of 10 hours of footage. This data was then segmented into five second long non-overlapping segments, for a total of 7200 video clips. Each clip was manually annotated across three labels, two of which related to the streamers affect and one of which related to what the streamer was doing in the game. Note that while the number of streamers is limited by the effort required to manually annotated data the length of the data set is in keeping with other works e.g. [5], [6].

A. Emotional Affect Annotation

Each clip was annotated for affect, using the streamer's facial, bodily and vocal cues to judge their emotional state. Affect was annotated across two dimensions, valence and arousal. 'Valence' relates to the positive/negative axis of emotion whereas 'arousal' relates to how strongly someone is feeling/displaying emotion. For valence each clip was rated on a three-point scale, positive, neutral or negative. For arousal, a two-point scale, neutral or positive, was used because video games, especially *League of Legends*, are not generally designed to elicit negative arousal and so this did not appear often in the data set. Therefore each clip receives a valance and arousal classification according to observable displays of the following:

Negative Valence Negative feeling e.g sadness.
Neutral Valence A lack of discernible valence.
Positive Valence Positive feeling e.g. happiness.
Neutral Arousal A lack of discernible arousal.
Positive Arousal Strong emotional response e.g. anger or excitement.

As can be seen from Table I there is a huge imbalance between classes with a skew towards neutral affect in both dimensions. This is to be expected; often gamers are engrossed in gameplay, and as a result, do not show outward emotion. This adds another complicating factor to the difficulty of learning affect in a livestream setting.

B. Game Annotation

Each clip was also annotated for game context, relating to what the streamer was doing during the clip. This behaviour is not always represented on screen for the full duration of the clip, in most cases due to players quickly switching their camera in-game to observe what others are doing. Eight categories were chosen which represent the majority of gameplay and for the occasions where the player is doing something outside of the scope of the categories a ninth 'miscellaneous' category was used:

In Lane Farming 'creeps' (game controlled enemies) in a lane. Often the default action.
Shopping Spending gold earned in-game on items.
Returning to Lane Walking back to lane after re-spawning, shopping or returning to base for health.
Roaming Roaming the 'jungle' area, the space between lanes.
Fighting Engaging in player vs player combat.
Pushing Pushing into and attacking the enemy base.
Defending Defending their own base.
Dead Killed and is awaiting re-spawn.
Miscellaneous Something not covered above.

Similarly to the affect annotations, there is an imbalance between game event classes, although less pronounced. 'In Lane' and 'Roaming' are the most popular activities, representing 33.58% and 19.75% of the data respectively, shown in Table I.

C. Data Pre-Processing and Over-Sampling

Once annotated, all clips where the game annotation was 'miscellaneous' were removed because this label does not accurately represent the content of the clip. Next, the data set was split randomly into 20% testing data, 1375 clips, and 80% training data, 5517 clips. The training data underwent a further pre-possessing step, oversampling, to help address the class imbalance in the data. Traditionally, oversampling algorithms, e.g. SMOTE [29] and ADASYN [30], generate synthetic data for minority classes by finding points between two existing minority examples. However, these techniques are not applicable to videos, which features both incredibly high dimensionality and important inter-view relationships. Therefore, synthetic data is generated as clones of existing minority class data.

It is important to ensure that oversampling a minority class in one output does not increase the majority class if it belongs to a different output. Therefore, the least and most represented classes across all annotations are calculated, with a weighting applied to account for the varying number of classes between outputs, see Equation 1 where w_c is the representation weight for a class c, T_c is the total for this class, T_d is the total data points in the data set, and N_c is the number of classes for this output. Next, a data point is selected at random which is in the least represented class and not in the most represented class. The selected clip is then cloned in the data set. This process is repeated until either a predefined threshold is reached (the chosen threshold for this work was the size of the initial data-set) or no data satisfies the selection requirement. The result of this oversampling is shown in Table II.

$$w_c = \frac{T_c}{T_d/(1/N_c)} \tag{1}$$

The data is also processed by taking each five-second clip and extracting visual and audio frames at a rate of four frames per second. Game images are $128 \times 128 \times 3$ down-sampled images taken from the frame. They represent the game context on the screen and have a black patch placed over the streamer's webcam. The streamer's webcam images are $64 \times 64 \times 3$ down-sampled images taken from the frame and represent what is present in the streamer's webcam and contain no

[4]https://twitchstats.net/

[5]Code for all models along with the data set can be downloaded from https://github.com/charlieringer/LoLEmoGameRecognition

TABLE I
DISTRIBUTION OF ANNOTATIONS IN THE RAW DATA SET.

Valence			Arousal		Game Context								
Neg	Neut	Pos	Neut	Pos	In Lane	Shopping	Ret. to Lane	Roaming	Fighting	Pushing	Defending	Dead	Misc.
246	6,227	727	6,755	445	2,418	294	591	1,422	892	213	233	831	308

TABLE II
IMPACT OF THE OVERSAMPLING TECHNIQUE ON THE TRAINING DATA SET.

	Valence			Arousal		Game Context							
	Neg	Neut	Pos	Neut	Pos	In Lane	Shopping	Ret. to Lane	Roaming	Fighting	Pushing	Defending	Dead
Before	0.033	0.862	0.104	0.937	0.063	0.35	0.043	0.084	0.206	0.13	0.03	0.033	0.123
After	0.259	0.483	0.257	0.725	0.275	0.181	0.097	0.097	0.146	0.103	0.181	0.097	0.181

gameplay data. The audio represents a joint stream, containing the streamer's voice, the game audio, along with occlusions such as any music the streamer is listening to and represents the raw audio waveform as a single vector of length 5512. Therefore, the input data for each clip consists of 20 frames (4 fps $\times$ 5 seconds) represented as two image tensors and one audio vector. The temporal and spacial down-sampling was chosen empirically to provide a reasonable middle ground between representing the original clip and reducing the data passed into the network for performance reasons.

IV. METHODOLOGY

Three models with a similar underlying structure were developed to compare fusion technique. First, a set of latent features are extracted from each frame using a set of convolutional neural networks (CNNs). When considering views containing image data a 2D CNN is used, whereas on the audio stream a 1D CNN is applied, due to the one-dimensional shape of the audio data. These features are then modelled temporally using several Long Short-Term Memory (LSTM) [31] recurrent layers. Finally, several fully connected layers are used to extract a set of classifications, dependant on the task presented to the network. The difference between models, discussed in IV-B, lies in how the multimodal latent representations after the feature extraction stage are fused to build a shared representation of the input. The high-level architecture for each model is demonstrated in Fig. 2.

A. Feature Extraction

All models use the same set of CNN architectures for extracting a vector of latent features from each view. The two image networks, for streamer and game data, use a 2D CNN with a series of residual blocks [32] to aid in back-propagating gradient. The difference being that the streamer network expects a smaller input image and as a result requires fewer layers and weights to satisfactorily model the features required. For modelling audio, a feature extractor with 3 1D convolutional layers followed by a dense layer was used. After each convolutional layer across all feature extractors, ReLU activation and Batch Normalization are applied. A visual layout of these architectures can be found in Fig. 3. Convolutional and residual block structures are illustrated in Fig. 4. For more details on residual CNNs, see [32].

B. Multimodal Fusion and Temporal Modelling

Once latent features have been extracted frame by frame from the various views, the next step is to fuse the views together to form a single view in addition to modelling per-frame features temporally. The purpose of this process is to extract a single feature vector corresponding to a joint representation across both views and time. Traditionally, concatenating input or representation space vectors has been one of the most popular approaches [17]. Nevertheless, this method can fail in terms of modelling interactions between views. The fusion approaches detailed in this work include both early and late fusion, as well as a novel method based on tensor decompositions to facilitate modelling high-order interactions between views in an efficient manner. The methods are detailed in the following sections.

1) Early Fusion: In the early fusion model, fusion is performed by taking the 512 features extracted for each view per frame and fusing them into a single vector of $l = 1536$. A stack of 20 feature vectors, one for each frame, is then fed into 2 LSTM layers each with 384 neurons. Before and after the LSTM layers, batch normalization and 20% drop out are applied to guard against over-fitting and aid generalisation. The resulting vector represents the fused representation of this clip across views and time.

2) Late Fusion: Late fusion is implemented with separate LSTM layers, with 128 neurons per layer for each view. Batch Normalization and Dropout layers are applied before and after the layers. As such, a single feature vector for each view represents the latent representation of the entire clip across one view. Concatenation of the various views then occurs after the LSTM step, and right before classification.

3) TensorTrain Fusion: Both early and late fusion models concatenate feature vectors from each view into a single vector. However, this fusion approach has no explicit

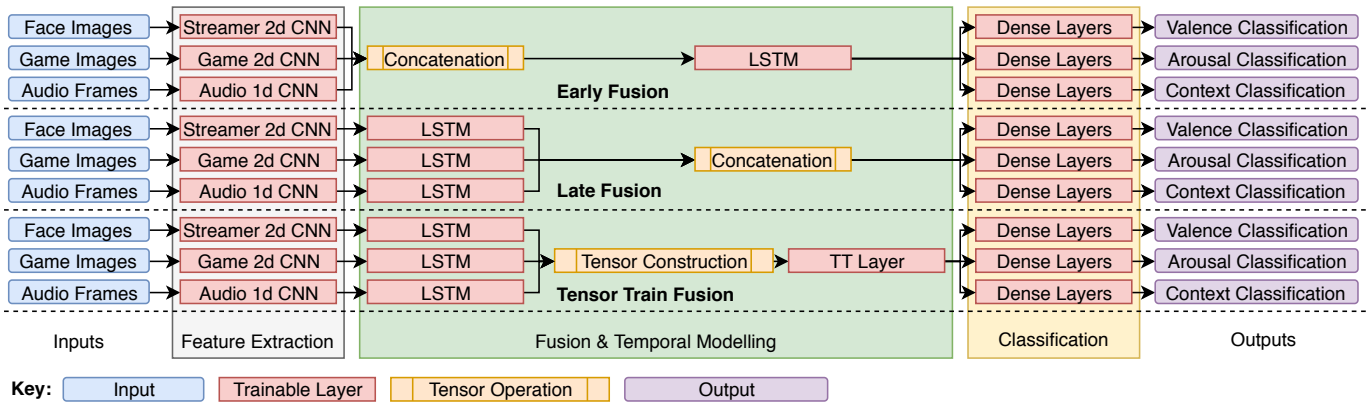

Fig. 2. Comparison between the high level architectures of the three fusion techniques presented in this paper. All modes use the same feature extraction and classification layer shapes but differ in how the input views are fused. Best viewed on a computer.

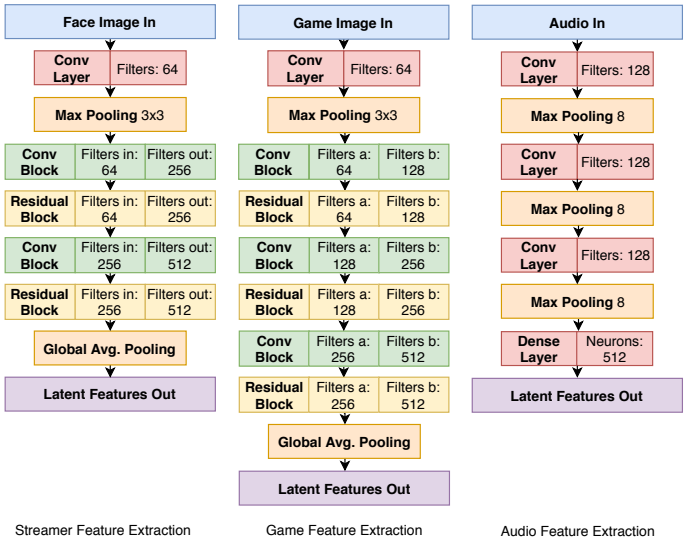

Fig. 3. Architectures of feature extraction modules. Note that each 'Conv Block' and 'Residual Block' are in turn multiple layers, resulting in a deeper network than shown. The makeup of these blocks is show in Fig. 4.

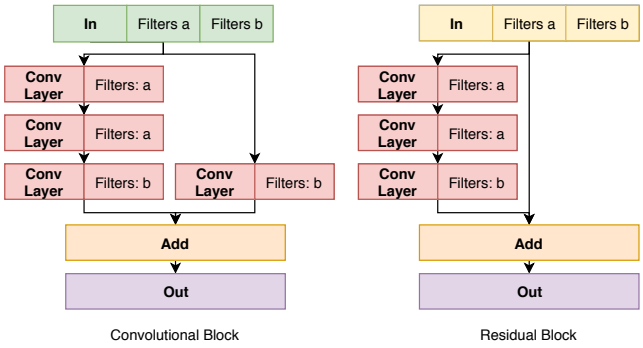

Fig. 4. Structure of the convolutional and residual blocks.

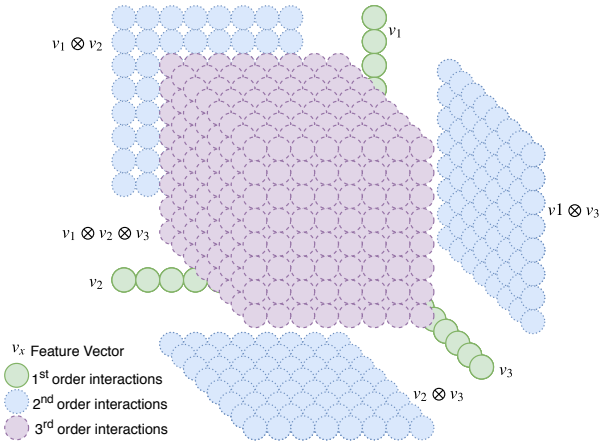

Fig. 5. Exploded structure of the 1st, 2nd and 3rd order interaction tensor constructed before the Tensor Train layer. Note: Each feature here represents 16 features in the model. Best viewed on a computer.

representations which capture the relationship between variables in different views. To better capture interactions between different views, we construct a tensor that models up to third order interactions between views. That is, given feature vectors v_x, v_y, v_z each corresponding to separate views, we take the cross product as

$$z = \begin{bmatrix} 1 \\ v_x \end{bmatrix} \otimes \begin{bmatrix} 1 \\ v_y \end{bmatrix} \otimes \begin{bmatrix} 1 \\ v_z \end{bmatrix} \tag{2}$$

were

$$\begin{bmatrix} 1 \\ v_x \end{bmatrix} \otimes \begin{bmatrix} 1 \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & v_x \\ v_y & v_x \otimes v_y \end{bmatrix}$$

As such the fused feature tensor $z \in \mathbb{R}^{129,129,129}$ is created which contains v_x, v_y, v_z, $v_x \otimes v_y$, $v_x \otimes v_z$, $v_y \otimes v_z$, and $v_x \otimes v_y \otimes v_z$. The resulting tensor is shown in Fig. 5.

At this point, it would be possible to flatten this feature tensor and then pass it through a series of dense classification layers. However, due to the size of the tensor $(2,146,689$ elements), this is computationally infeasible. Connecting this tensor to a fully connected layer with 128 neurons would result in $128 \times 2,146,689 = 274,776,192$ weights for this layer alone. Therefore, a Tensor Train layer [4] is used to connect the latent tensor with the classification layers. This layer replaces

a dense layer and represents its weight matrix as a series of smaller tensors, the tensor cores. In this case, a tensor train with ranks (1, 2, 4, 4, 2, 1) is used. Each element in the weight tensor is then approximately represented as a product of these tensors thus allowing the model to learn the weighted mapping between the input tensor z and the output vector with far fewer parameters, around 11,000, resulting in a space saving of 4×10^{-5}. The Tensor Train layer extracts 384 features, the same number features of concatenating the original feature vectors, which are then passed forward to the classification layers. This approach thus incurs only a small increase in weights whilst in theory modelling these important higher order interactions.

C. Classification

The fused feature vector is then passed into several dense layers to perform classification. These layers act as separate task specific 'heads'. For each task, first, the feature vector is passed through a 128 neuron dense layer with ReLU activation before the final classification layer, which is calculated by taking the softmax across n neurons where n = the number of classes (e.g. for Valence n = 3).

V. Experiment

Each model presented in Section IV was trained on three tasks. Firstly, to learn a joint representation of both game context and streamer affect, and thus classify valence, arousal and game context simultaneously. Secondly, the task of only learning the game context. Finally, the task was learning just the streamer's affect.

We used Keras [33] with the Tensorflow backend for implementing each model. Models were optimized with ADAM [34] where $\alpha = 0.0005$. For each learning task, each network was trained for 100 epochs using an NVIDIA GTX 1080 GPU. The number of weights per model can be found in Table III. For each output, a set of class weights were implemented as an additional measure to tackle the bias in the data set. To calculate class weights for each class x, an initial weight i_x is calculated then a scaled weight w_x is calculated so that all weights for an output sum to one:

$$i_x = \frac{T_d}{T_x \times N_c} \tag{3}$$

$$w_x, w_y, ... = \frac{i_x}{sum(i_x, i_y, ...)}, \frac{i_y}{sum(i_x, i_y, ...)}, ... \tag{4}$$

Where T_d is the total data points in the data set, T_x is the total data points for class x and N_c is the number of classes for this output, e.g. for a valence class $N_c = 3$ as there are 3 possible valence classifications.

VI. Results

A. Affect and Game Context Classification

For each task, model, and label values for Precision, Recall, and F1 Score are calculated by:

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

TABLE III
Comparison of Trainable Weights Across Models.

Task	Early Fusion	Late Fusion	Tensor Train Fusion
Affect + Game	9,382,029	6,629,517	6,640,939
Affect	9,331,717	6,579,205	6,590,627
Game	9,282,824	6,530,312	6,541,734

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{7}$$

Where TP, 'true positive', is the sum of correctly classified data-points for a label, FP, 'false positive', is the sum of all data points annotated with a different label but classified as this label and FN, 'false negative', being the sum of all data points with this label that were classified with a different label. These metrics were used because accuracy, e.g. the average correct classifications across a whole data set, is not necessarily the best measure of success when testing on very unbalanced data. In these cases, high accuracy can be achieved by simply always classifying the majority class e.g. for Arousal outputting only Neutral Arousal classifications would yield an accuracy of 0.94. As such, discussion of the results will focus on the individual class F1 Score values, as it is the harmonic average of Precision and Recall so is representative of both. F1 scores for all models across all classes in both single and joint task learning are shown in Table IV.

B. Joint vs Single Task Learning

One of the aims of this study is to explore if learning both game context and affect classifications at the same time would improve results. While learning these tasks simultaneously has the drawback that model has fewer variables to dedicate to each task, jointly learning the task may also lead to learning more generalisable, and thus robust, representations. Additionally, joint task learning has the benefit of requiring only a single model to perform recognition across all tasks thus resulting in faster training and inference as only one model needs to be trained/queried.

Fig. 6 shows the delta in performance between single and joint task learning, showing that whilst there is a large degree of between-class variance, the models which perform fusion after the LSTM step see an improvement when learning jointly, whereas the early fusion model performs worse.

VII. Discussion

A. Affect and Game Context Classification

Table IV shows that affect classification is a much harder task that game context classification, further reinforcing the discussion regarding the difficulty of in-the-wild affect detection. Importantly, while results such as a high of 0.362 for F1 Score for positive valence and a high of 0.286 for negative valence may seem poor, they are actually significantly higher than a random baseline, which has expected F1 scores

TABLE IV
F1 SCORES FOR EACH LABEL ACROSS ALL MODELS. FOR EACH MODEL F1 SCORES FOR EACH TASK AND EACH LABEL ARE REPORTED. BEST RESULT FOR EACH CLASS IN **BOLD**.

Model	Task	Neg V	Neut V	Pos V	Neut A	Pos A	In Lane	Shopping	Returning	Roaming	Fighting	Pushing	Defending	Dead
Early	Joint	0.194	0.911	**0.362**	0.969	0.509	0.778	0.797	0.496	0.667	0.515	0.568	0.544	0.899
Fusion	Single	0.206	0.905	0.345	0.966	**0.540**	0.842	0.724	0.591	0.794	0.565	0.610	**0.667**	0.924
Late	Joint	**0.286**	0.925	0.297	0.964	0.465	0.840	**0.828**	**0.615**	0.805	**0.635**	**0.652**	0.557	0.906
Fusion	Single	0.088	0.918	0.340	0.968	0.491	0.791	0.776	0.513	0.774	0.581	0.582	0.452	0.937
TT	Joint	0.102	0.926	0.325	**0.971**	0.476	**0.848**	0.765	0.580	0.791	0.630	0.635	0.574	0.930
Fusion	Single	0.135	**0.928**	0.315	0.969	0.537	0.837	0.777	0.518	**0.819**	0.557	0.405	0.473	**0.948**

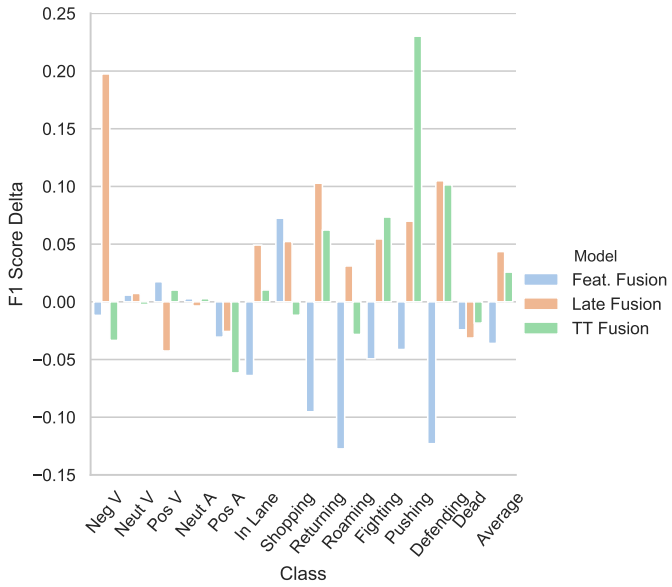

Fig. 6. Delta change in F1 Score performance. Positive values occur when joint task learning outperforms single task learning. Negative values occur when single task learning outperforms joint task learning.

of 0.044 (negative) and 0.106 (positive), because of the hugely imbalanced data set. There is clearly still a lot of interesting work to be done to accurately model streamer emotions, however, these results do provide a baseline. Additionally, Tensor Train models have a large scope for improvement gains via hyper-parameter tuning [4] so it is possible that future gains are possible using similar architectures.

Regarding game context we see that in general all models perform better when classifying examples of 'In Lane', 'Shopping', 'Roaming' and 'Dead', where F1 scores range from 0.667 (Early Fusion, 'Roaming', joint task learning) to 0.948 (TT Fusion, 'Dead', single task learning), compared to other context classes. It is difficult to know exactly why the models perform better on these categories but it is possible confusion arises for classes such as 'Pushing', and 'Defending' which can be visually similar. Consider the case where there are two players of opposite teams very close to each other and at the edge of a base. Only the positioning of the two players and prior knowledge about which side they are on provide clues as to if the streamer is pushing or defending. Contrast this with the 'Dead' class where the screen is mostly greyscale with a fixed message on the screen, resulting in easier classification.

B. Joint vs Single Task Learning

As Fig. 6 shows, for Late Fusion and TT Fusion we see a general if erratic improvement when learning a joint representation with average F1 Score improvements of 0.044 and 0.026. However, Early Fusion sees a degradation of F1 Score performance with an average change of −0.036. Furthermore by performing a Wilcoxon Signed-Rank Test we see that both the change in Early Fusion and Late Fusion are statistically significant at $P < 0.05$ ($P = 0.047$ and $P = 0.03$ respectively). Perhaps a key take away from these results is that seemingly fusion techniques occurring after temporal modelling see improved results from learning both tasks jointly, but Early Fusion which occurs before the LSTM see a performance reduction.

C. Determining the 'Best' Model

It is difficult to ascertain from these results which model performs the best as there is no clear 'winner' across all categories. Each model appears to perform better on some tasks and worse on others. Early fusion outperforms TT fusion and late fusion for emotion prediction tasks, although TT fusion appears best at classifying the neutral, majority, classes. This is possibly due to the LSTM layers applied to the fused representation being able to facilitate a temporal, multimodal representation that can account for cross-view cues manifesting at different points in time, e.g. anticipatory co-articulation. Furthermore, TT fusion slightly outperforms late fusion in emotion recognition due to TT fusion better modelling the interactions between views. In general, using post-LSTM fusion (late, TT-fusion) seems to provide better performance at game context recognition, possibly due to the game footage and game audio being the most important cues for the task, and there appears to be a benefit in modelling their individual dynamics separately.

The early fusion network is significantly larger in terms of weights than the other two models, in fact, they are roughly only 2/3$^{\text{rds}}$ of the size. Additionally, it is the only model that sees performance degradation when joint task learning. Therefore whilst its performance is comparable to other models it does so using approximately 3 million more weights and therefore has a much higher computational cost.

VIII. Conclusions and Future Work

In this work, we pose the problem of modelling streamer affect jointly with game context, within the framework of a deep learning architecture that fuses audio-visual stream data exploiting the power of tensor decompositions such as Tensor Train. Furthermore, the first annotated data set of emotion and game context for video game livestreams is presented, along with baseline results comparing early and late fusion approaches to the proposed model. Results show that the proposed approach generally outperforms Early Fusion and is comparable to Late Fusion across tasks. Additionally, the difficulty of affect classification in this environment is shown. Clearly, jointly modelling game context and streamer affect in a multimodal setting constitutes a challenging problem of an interdisciplinary nature, with challenges arising in the context of areas such as computer vision, machine learning, and affective computing. Ultimately, this paper acts as a call for action, inviting more researchers to work in this area, as improved results and models are crucial for facilitating audio-visual player experience modelling on livestreams with implications across a range of disciplines.

References

[1] M. Kaytoue, A. Silva, and L. Cerf, "Watch me playing, i am a professional: a first study on video game live streaming," *Proceedings of the 21st international conference companion on World Wide Web*, pp. 1181–1188, 2012.

[2] T. Stadelmann, M. Amirian, I. Arabaci, M. Arnold, G. F. Duivesteijn, I. Elezi, M. Geiger, S. Lörwald, B. B. Meier, K. Rombach, and L. Tuggener, "Deep learning in the wild," in *Artificial Neural Networks in Pattern Recognition*, L. Pancioni, F. Schwenker, and E. Trentin, Eds. Cham: Springer International Publishing, 2018, pp. 17–38.

[3] M. A. Nicolaou and C. Ringer, "Streaming behaviour: Live streaming as a paradigm for multi-view analysis of emotional and social signals," in *Twitch Workshop, 13th International Conference on the Foundations of Digital Games*, ser. FDG '18, 2018.

[4] Y. Yang, D. Krompass, and V. Tresp, "Tensor-Train Recurrent Neural Networks for Video Classification," 2017.

[5] "Event detection and highlight detection of broadcasted game videos," *HCMC 2015 - Proceedings of the 2nd Workshop on Computational Models of Social Interactions: Human-Computer-Media Communication, co-located with ACM MM 2015*, pp. 1–8, 2015.

[6] W.-T. Chu and Y.-C. Chou, "On broadcasted game video analysis: event detection, highlight detection, and highlight forecast," *Multimedia Tools and Applications*, vol. 76, no. 7, pp. 9735–9758, Apr 2017.

[7] C. Ringer and M. A. Nicolaou, "Deep unsupervised multi-view detection of video game stream highlights," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*, ser. FDG '18. New York, NY, USA: ACM, 2018, pp. 15:1–15:6.

[8] T. Smith, M. Obrist, and P. Wright, "Live-streaming changes the (video) game," *Proceedings of the 11th european conference on Interactive TV and video - EuroITV '13*, p. 131, 2013.

[9] D. Recktenwald, "Toward a transcription and analysis of live streaming on twitch," *Journal of Pragmatics*, vol. 115, pp. 68 – 81, 2017.

[10] R. Robinson, Z. Rubin, E. M. Segura, and K. Isbister, "All the feels: Designing a tool that reveals streamers' biometrics to spectators," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*, ser. FDG '17. New York, NY, USA: ACM, 2017, pp. 36:1–36:6.

[11] G. Nascimento, M. Ribeiro, L. Cerf, N. Cesrio, M. Kaytoue, C. Rassi, T. Vasconcelos, and W. Meira, "Modeling and analyzing the video game live-streaming community," in *2014 9th Latin American Web Congress*, Oct 2014, pp. 1–9.

[12] K. Karpouzis, G. N. Yannakakis, N. Shaker, and S. Asteriadis, "The platformer experience dataset," in *2015 International Conference on Affective Computing and Intelligent Interaction (ACII)*, Sept 2015, pp. 712–718.

[13] N. Shaker, S. Asteriadis, G. N. Yannakakis, and K. Karpouzis, "Fusing visual and behavioral cues for modeling user experience in games," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1519–1531, 2013.

[14] S. Asteriadis, K. Karpouzis, N. Shaker, and G. N. Yannakakis, "Towards detecting clusters of players using visual and gameplay behavioral cues," *Procedia Computer Science*, vol. 15, pp. 140–147, 2012.

[15] P. M. Blom, S. Bakkes, C. T. Tan, S. Whiteson, D. Roijers, R. Valenti, and T. Gevers, "Towards personalised gaming via facial expression recognition," in *Proceedings of the Tenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'14. AAAI Press, 2014, pp. 30–36.

[16] C. T. Tan, D. Rosser, S. Bakkes, and Y. Pisan, "A feasibility study in using facial expressions analysis to evaluate player experiences," *Proceedings of The 8th Australasian Conference on Interactive Entertainment Playing the System - IE '12*, pp. 1–10, 2012.

[17] A. K. Katsaggelos, S. Bahaadini, and R. Molina, "Audiovisual Fusion: Challenges and New Approaches," *Proceedings of the IEEE*, vol. 103, no. 9, pp. 1635–1653, 2015.

[18] S. Poria, E. Cambria, R. Bajpai, and A. Hussain, "A review of affective computing: From unimodal analysis to multimodal fusion," *Information Fusion*, vol. 37, pp. 98–125, 2017.

[19] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ser. ICML'11. USA: Omnipress, 2011, pp. 689–696.

[20] M. Wllmer, M. Kaiser, F. Eyben, B. Schuller, and G. Rigoll, "Lstm-modeling of continuous emotions in an audiovisual affect recognition framework," *Image and Vision Computing*, vol. 31, no. 2, pp. 153 – 163, 2013, affect Analysis In Continuous Input.

[21] M. Shah, C. Chakrabarti, and A. Spanias, "A multi-modal approach to emotion recognition using undirected topic models," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2014, pp. 754–757.

[22] S. E. Kahou, X. Bouthillier, P. Lamblin, C. Gulcehre, V. Michalski, K. Konda, S. Jean, P. Froumenty, Y. Dauphin, N. Boulanger-Lewandowski, R. Chandias Ferrari, M. Mirza, D. Warde-Farley, A. Courville, P. Vincent, R. Memisevic, C. Pal, and Y. Bengio, "Emonets: Multimodal deep learning approaches for emotion recognition in video," *Journal on Multimodal User Interfaces*, vol. 10, no. 2, pp. 99–111, Jun 2016.

[23] P. Tzirakis, G. Trigeorgis, M. Nicolaou, B. Schuller, and S. Zafeiriou, "End-to-end multimodal emotion recognition using deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. PP, 04 2017.

[24] A. Zadeh, M. Chen, and E. Cambria, "Tensor Fusion Network for Multimodal Sentiment Analysis," 2017.

[25] I. Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[26] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 442–450.

[27] J. Kossaifi, Z. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," 2017.

[28] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky, "Tensor decompositions for learning latent variable models," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 2773–2832, Jan. 2014.

[29] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: Synthetic minority over-sampling technique," *J. Artif. Int. Res.*, vol. 16, no. 1, pp. 321–357, Jun. 2002.

[30] and and E. A. G. and, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, June 2008, pp. 1322–1328.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," vol. 9, pp. 1735–80, 12 1997.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 770–778.

[33] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[34] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.