End-to-End Let's Play Commentary Generation using Multi-Modal Video Representations

Chengxi Li
University of Kentucky
Lexington, Kentucky
chengxili@uky.edu

Sagar Gandhi*
University of Kentucky
Lexington, Kentucky
sga267@uky.edu

Brent Harrison
University of Kentucky
Lexington, Kentucky
harrison@cs.uky.edu

ABSTRACT

In this paper, we explore how multi-modal video representations can be applied in an end-to-end fashion for automatically generating game commentary based on *Let's Play* videos using deep learning. We introduce a comprehensive pipeline that involves directly taking videos from YouTube and then using a sequence-to-sequence strategy to learn how to generate appropriate commentary. We evaluate our framework using *Let's Play* commentaries for the game *Getting Over It with Bennet Foddy*. To test the quality of the commentary generation, we apply perplexity to evaluate our language models using different input video representations to highlight different aspects of gameplay that might influence commentary.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; **Natural language generation**; **Video summarization**;

KEYWORDS

Commentary Generation, Sequence to Sequence, Multi-Modality

ACM Reference Format:
Chengxi Li, Sagar Gandhi, and Brent Harrison. 2019. End-to-End Let's Play Commentary Generation using Multi-Modal Video Representations. In *The Fourteenth International Conference on the Foundations of Digital Games (FDG '19), August 26–30, 2019, San Luis Obispo, CA, USA.* ACM, New York, NY, USA, Article 4, 7 pages. https://doi.org/10.1145/3337722.3341870

1 INTRODUCTION

One of the primary strengths of deep learning is its ability to automatically generate informative features from complex inputs. This has made it an ideal tool for interpreting visual data. As such, one of the most popular applications of deep learning is in extracting informative features to better generate some desired output. One such output is text. Deep learning has proven effective in image-to-text tasks [4, 22, 23]. Given this success, researchers have began exploring the related, but more difficult, task of video-to-text.

Converting video streams to text presents a significant step up in difficulty when compared to image-to-text tasks. Video-to-text tasks need to be able to model the temporal relationships that exist between potentially long sequences of image frames. Sequence to sequence networks [1, 11] have emerged as a common method for performing video-to-text tasks [20, 21]. These approaches have focused on the specific task of video captioning: providing descriptions of what is occurring in some input video. In this paper, we explore the use of sequence to sequence approaches for *commentary generation.*

Commentary generation from video task is a new research topic that focuses on generating general commentary about a given input video. The primary difference between caption generation and commentary generation is that commentary generation is not restricted to merely describing the actions that are occurring. Effective commentary can include many different types of textual information including descriptions of current behaviors, personal feelings about what is going on in the video, or stories about other things that have happened outside of the video.

To simplify this task, one can use different modality data representations as input to a deep learning architecture. Different representations can highlight different aspects of the input data that could be relevant to generating commentary. For example, some representations can highlight how objects move in the original input video, which could have a great deal of influence on the commentary generated. Similarly, audio data could contain information regarding the commentator's emotional state, which could also affect the commentary that's being produced. One of the primary goals of this paper is to explore how different types of input representations affect the type of commentary produced by deep learning methods.

We summarize our contribution as below: First, we examine how additional input information such as optical flow or audio data can impact automatic commentary generation. Second, we explore how deep learning approaches, specifically sequence to sequence models, can be used for automatic commentary generation. Third, we present a pipeline that can be used for automatic, end-to-end commentary generation.

2 RELATED WORK

The rapid development of deep learning approaches for automatically generating captions or commentary for video stems from the success of deep learning approaches for images captioning [4, 22, 23]. There has also been work on applying analogous attention models to further improve captioning efforts [26]. While a video is composed of multiple continuous frames, care must be taken to maintain any temporal relationships that could exist between frames when generating the desired text.

*Now at Google

FDG '19, August 26–30, 2019, San Luis Obispo, CA, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7217-6/19/08...$15.00
https://doi.org/10.1145/3337722.3341870

To account for this, some of the earliest work in video captioning took advantage of different deep learning approaches to learn relationshps between frames of an input video [14, 20, 25]. While there has been success in video captioning using deep learning, commentary generation represents a unique challenge in that subjective commentaries often contain subjective information that may be difficult to generate using video features alone.

While commentary generation may be a more daunting task than video captioning, there have been successes in this area in the past. In 2016, Yan [24] used video events and frames number as video features to train a LSTM-based recurrent neural network and structured SVM for commentary generation. While successful, their approach was trained on limited data and required extensive hand authoring to generate the requisite training commentaries. In 2018, Guzdial et al. [8] automatically generated Let's Play commentary by first clustering videos and then learning a mapping from videos clusters to commentary types using machine learning methods. In their work, videos were represented as a bag of sprites and the comments were represented as a bag of words. This proved effective at generating commentaries for simple arcade games, but this approach is likely to struggle to generalize for more complex games where sprites may be difficult to extract. In this paper, we explore how neural techniques paired with different input representations and data can address this limitation by automatically learning visual features useful for commentary generation rather than manually extracting sprite information.

3 METHOD

The primary contribution of this paper is an exploration of machine learning-based approaches to perform end-to-end commentary generation using different types of input data. Specifically, we examine the applicability of sequence-to-sequence networks using different video input representations to this task. In the following sections, we will provide additional details about the type of data used for this problem as well as the networks and input data used for this problem.

3.1 Dataset

In this paper, we focus on generating *Let's Play* style commentaries. A *Let's Play* video is typically a video of a person playing through a video game while providing real-time commentaries related to the events occurring in the game. For our experiments, we choose to focus on generating commentaries using *Let's Play* videos made for a specific game: *Getting Over It with Bennet Foddy*[1]. *Getting over it*, released in 2017, is a climbing game where the player's goal is to move his avatar, a man inside of a cauldron, higher and higher using a hammer controlled by the mouse. Using this hammer, the player can jump, swing, climb, and even fly in some cases. An example frame from this game can be seen in Figure 1 (left). We choose this game because it is a relatively popular game, ensuring that we have access to many playthroughs of the game, and because it is a relatively short game.

Along with these videos (and the associated audio), it is possible to obtain text transcripts of the commentary by using the closed captions associated with each video. The closed captions to those video recordings are usually created by different human annotators. As such, most closed captioning transcripts contain informal language with potential grammar errors or nonstandard spellings of common words. This ends up complicating the learning process as it results in very sparse word dictionaries.

Using this data, we are able to create input representations using video or audio data while using the associated text as the target output. With this information, we can train different sequence to sequence models for commentary generation.

3.2 Architectures and Representations

In this paper we explore how different video input data modalities impact automatic *Let's Play* commentary generation using various sequence to sequence architectures. In particular, we focus on three input variations using two deep neural networks: RGB frames representing a video clip using sequence to sequence architecture (baseline), RGB frames representation using sequence to sequence networks with attention, sequence to sequence networks using additional information about optical flow, and sequence to sequence networks using multi-modal input consisting of video features and audio features. Each of these variations will be discussed in greater details below.

3.2.1 ***Basic Architecture.*** The baseline uses RGB frames features to represent each video clip. We choose RGB frames features as baseline feature input because this allows the deep learning architecture to extract what it views as the most relevant features for this task. Continuous RGB frames features will also implicitly convey the motions, position and color changes that occur from frame to frame. We use sequence to sequence networks to perform commentary generation since they have proven effective on other video to text tasks [19].

Sequence to sequence networks comprising encoders and decoders made of long short-term memory (LSTM) nodes have achieved state-of-the-art performance in video-to-text work [20]. In this paper, we use a stacked LSTM encoder-decoder, which is able to exploit temporal information from video clips [13], to encode and decode video information. This involves using a video frame as one unit of the input sequence, represented using either pixel values or as a vector of visual features. As a video clip consists of several video frames in time order, we treat one video clip as a sequence. We use the closed caption transcript information associated with the video clip as the target sequence.

3.2.2 ***Sequence to Sequence Architecture with Attention .*** As the basic sequence to sequence model compresses a source sequence to a fixed length vector, partial vector information could get lost during the translation process. In the baseline architecture, the only access that the decoder network has to the input sequence is through this fixed length context vector. This can make it difficult for the decoder to identify complex relationships between various parts of the input, especially if the encoder struggles to create a context vector that accurately encapsulates the input sequence.

Attention mechanisms address this inherent limitation of baseline sequence to sequence networks by enabling the decoder to "peek" at the input sequence. This allows the decoder to form a better alignment between the output sequence and input sequence.

[1]https://en.wikipedia.org/wiki/Getting_Over_It_with_Bennett_Foddy

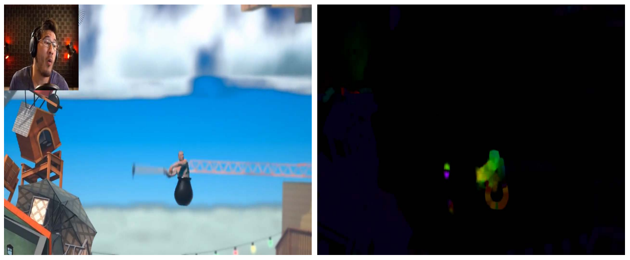

Figure 1: RGB frame (left) and optical flow frame (right) sample

As videos and comments are naturally aligned by time, adding attention mechanism would help align the source frame in the video clip with the current target token in the comment [1, 11]. Also, an attention model could enable the sequence to sequence networks to better generate words which occur very rarely in the corpus [11], which is ideal given the nature of *Let's Play* commentaries.

3.2.3 ***Architecture Guided by Motion Stimulation Using Optical Flow.*** With video data, especially videos concerning video games, it is possible that neural networks could get distracted by unimportant parts of the video (e.g. animated background environments). However, game commentaries are closely related to the activities and motions happening in the video. While these videos contain a lot of information, we hypothesize that most of what is being talked about will be confined to what's moving on the screen to keep the audience engaged. The commentator sometimes will add other comments which have no connection to the gameplay to increase the entertainment of the whole comments. As such, recognizing the motions on the screen can potentially help identify when the speaker is talking about other things rather than gameplay. For example, while nothing on the screen is moving, we could assume the speaker is making comments not related to the gameplay.

Dense optical flow has been used successfully to identify motions in video data [15] and has been shown to improve performance on video-to-text tasks when compared against networks that use only visual input [20]. Figure 1 shows a sample of what dense optical flow looks like in a frame we extracted from the videos clips.

The dense optical flow is calculated by first obtaining a 2-channel array with optical flow vectors through computing all the pixel displacements between current frame and previous frame using polynomial expansion coefficients [6]. And then according to the optical flow vectors, we get their magnitude and direction. Direction is assigned as the Hue value of the image and magnitude as Value plane. Finally, this is converted back into a RGB representation. Through above process, we calculate the dense optical flow between frames across videos and explore generating commentary based on motion stimulation. To incorporate motion stimulation into sequence to sequence and maintain the rich information from RGB features, we directly combine optical flow with RGB frame features through concatenation and use this concatenated vector as input.

3.2.4 ***Architecture Guided by Audio Features.*** It is possible that the commentator's mood could have a large effect on the type of commentary that they produce. Thus, additional information about the commentator's emotional state could prove to be invaluable in creating high quality commentaries.

To address this, we propose using a multi-modal input sequence composed of video frames paired with audio features to make the video commentary generation more "subjective". We choose to include audio as an additional input because audio information can be used to determine the speaker's feelings and speaking style [3] and these feelings have great impact about which words will be used in the commentary.

To represent the audio data, we convert the raw audio input into a Mel-spectrogram. Mel-spectrograms have been shown to be an effective audio representation for transferring audio features [9, 17], which is the primary reason that we use this representation in this work.

4 EXPERIMENT

To evaluate the impacts of different video representations using the deep learning architectures outlined above, we examine the performance of each when generating commentary for the game, *Getting Over It with Bennet Foddy*. Specifically, we compare the perplexity of each video input representation in the following two experimental conditions:

- Ground truth versus random words: In this condition, we compare the perplexity of a network when generating the actual ground truth sentences versus the perplexity of the network when generating sequences of random words drawn from our word dictionary.
- Ground truth versus random sentences: In this condition, we compare the perplexity of a network when generating the actual ground truth sentences versus the perplexity of the network when generating other naturally occurring sentences in our test set.

We will discuss our data cleaning pipeline in addition to these experiments in greater detail below.

4.1 Dataset Pipeline

We downloaded 8 *Let's Play* videos of the same commentator playing through the game *Getting Over It with Bennet Foddy* from Youtube. We then downloaded the closed captioning transcripts along with timestamped information about when they occur. In this game, Bennet Foddy, the game's creator, will provide commentary periodically. This commentary is also included in the closed captioning transcripts associated with each video. We use the timestamp information included with the closed caption transcripts to segment the video and audio so that it is tightly coupled with the text. At the end of this process, each video is divided into several clips, each lasting roughly 2-3 seconds, with the associated text extracted from the closed captions.

The reason to use closed captioning transcripts as our ground truth and parse our videos as above is because closed captioning data is readily available for these types of videos. Closed caption transcripts can be problematic, however, because they often contain spelling or grammatical errors as well as informal language. This can result in data sparsity with respect to the input vocabulary.

We use the Social Tokenizer [2] to extract tokens from the original sentences and make all uppercase tokens into lowercase. This is the only preprocessing of the text data that has been done. This ensures that our training word corpus maintains the word diversity present in the original corpus. This does, however, make the learning problem more difficult since some words that are clearly related (e.g. *nooo* and *no*) are considered separate tokens. In total, our vocabulary size is 2598.

Youtube game commentary videos often contain an introduction and sign off messages which may be necessary for guiding and building rapport with the audience, but has no relation with the gameplay. Thus, we remove these parts of videos and their associated transcripts. After this, we have a total of 2274 video segments with associated audio and natural language commentary transcripts.

To generate data sources, we first extract all the frames from each video clips with default video frame rate 30fps. The dimension of each frame is 720×1280. Then we read the raw pixels from each frame and concatenate the frame RGB pixels from one video clip into one vector as one raw data source. Before we feed data to in our system, we convert each raw data source into a sequence of 4096 dimension vectors through vgg19 fc7 layer [16]. Vgg19 is a VGG Net with 19 layers and it is constructed with very small convolution filters and very deep networks. And since it works well with different datasets and get state-of-art results [5, 7, 23], vgg19 is always used as a pretrained model to extract images features and ease future training. So, we embed each segmented video into a sequence of vgg19 style vectors by passing each frame into a 4096 dimension vector. And we feed these embeddings as our input vector into the model.

To make sure every video among the 8 entire videos participates in training, we randomly split the data into a training, test, and validation set. We set aside 10% of the data as testing data, 10% as validation data, and 80% as training data from every video in the 8 videos. In total, the training set contained 1828 video clips (each video clip contain about 90 frames, represented as a vector with shape around 90*4096), and the validation and test sets contained 223 video clips each.

4.2 Commentary Generation with RGB features

Here, we will describe details related to each specific architecture that we examine in our experiments.

4.2.1 Baseline. We apply RGB features as video input using a sequence to sequence model as our base model. There are two steps to this process. We first embed all the video clips by using the vgg19 network as described in the above section. And then we feed the embedded features into the encoder to get a final state vector as initial state for decoder. Specifically, we use 3-stacked LSTM cells for both the encoder and the decoder. Each LSTM cell has 128 hidden units. To help enable the network to generalize, we apply 0.5 drop out before stacking. We also use word embeddings of size 200. These word embeddings are initialized randomly and trained along with our sequence to sequence networks.

We train the base model for total 500 epochs using a batch size of 10. For this network, we have used flexible output sequence lengths, meaning that the expected length of the output vector for each batch is different. We assign this length for a given batch to be equal to the size of the longest expected output sequence out of all examples in the batch. Sequences shorter than this are padded up to this maximum length. We use softmax cross-entropy as our loss function and use Adam optimization [10] with a learning rate of 0.001 during training. To help reduce training variance and prevent exploding gradients, we also clip gradients between -1 and 1.

4.2.2 Sequence to Sequence Attention Mechanism. Based on baseline model we have implemented, we add an attention mechanism [11] to enhance the alignment between the source and target. To create the attention, in details, at each time decoding, the current target token is fed to the stacked LSTM layers and generate the current target hidden states. Then alignment between current target and original source is calculated using current target hidden states and all the source states. Finally, the context vector is derived from the inner product of the alignment and all source states. Instead using current target as next input, we combine the current target hidden states and the context vector to generate the next input.

Data sources and targets are the same as baseline. We train the attention model with 500 epochs and batch size is 10. All the other hyper-parameters is the same as our base model.

4.3 Commentary Generation using Optical Flow

To help capture the activities and motion features from videos, we calculate the dense optical flow [6] between frames in each clip and generate the optical flow frames which has the same dimension as the raw frames. Same as the baseline, we extract all the optical flow frame features by pre-trained vgg19 layers. To include both RGB visual features and optical flow features as input, we concatenate each RGB feature vector with the optical flow feature vector to create a new input embedding of size 8192.

We use the same splitting as baseline to split all the data into training, validation and testing. Data sources are the new video clips embeddings we generate and data targets are the same comment vectors as before. We feed the new video clip embeddings into the attention model as above. The model is trained 500 epochs and batch size is 10, all other hyper-parameters are kept the same as baseline model.

4.4 Commentary Generation with Audio Features

To incorporate audio information into the learning process, we first encode the audio stream for each clip as a mel-spectrogram [12, 18]. Each mel-spectragram is represented as a 2-D vector with size 128 x duration. Since the mel-spectrogram dimensions are vastly different from the frame embeddings used above, instead of combining RGB features and audio features directly, we combine RGB features and audio features by concatenating two encoded hidden states. (see Figure 2). As you can see in the figure, we encode both the audio and visual features into 128 length feature vectors first. By encoding both, we are able to capture the temporal information from video frame features and audio features. The two feature vectors are then concatenated into a 256 length input vector which is passed into a subsequent encoder-decoder structure with attention as described for the previous approaches.

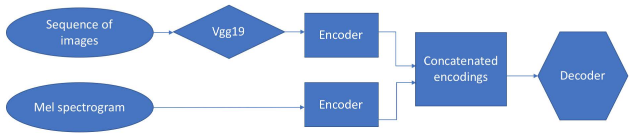

Figure 2: Encoder Decoder Network Architecture using Audio Features

We train the model for 500 epochs and all other hyper-parameters remain the same as baseline.

4.5 Evaluation

Since commentary generation is an inherently generative process, we use perplexity to evaluate the quality of each of the architectures described above.The perplexity of a model can be calculated as:

$$perplexity = 2^{\frac{-1}{N}\sum_{i=1}^{N} log_2 q(x_i)} \tag{1}$$

Among (1), N is the count of tokens in one comment, x_i is the $i-th$ token in the comment and $q(x_i)$ is probability of x_i in the language model. Informally, perplexity represents how "confused" a model is by a given output sequence. Thus, we would expect a model to have higher perplexity for sequences that it is unlikely to be able to generate and lower perplexity for sequences that it is likely to be able to generate. Note, when calculating perplexity in these experiments we do not factor in padding tokens. For these evaluations, we measure the perplexity of each architecture on the sentences contained in the test set and compare these values against the perplexity obtained on sequences of random words and on random sentences.

Recall that we consider the following two comparisons for our experiments: (1) comparing ground truth perplexity versus the perplexity of random words and (2) comparing ground truth perplexity versus the perplexity of random sentences. We choose these two conditions because they give a clear picture on the capabilities of each sequence to sequence architecture. By comparing against sequences of random words, we identify if the architecture has the ability to recognize syntactically correct (at least in terms of the training captions) sequences. By comparing against random complete sentences, we evaluate if each architecture can correctly identify the context that certain sentences should be used in.

When generating sequences of random words, we select random words from the word corpus until we generate a sentence of the same length as the ground truth sentence. When selecting random sentences we simply select from the set of sentences available in the test set.

Since this is a preliminary exploration, we are primarily concerned with whether the ground truth perplexity is lower than the perplexity for the sequence of random words or random sentences. Thus, we report the results of this evaluation by reporting the percentage of test examples that outperform either the sequence of random words or random sentences, depending on the test condition.

Since the comparison sequences we generate are all done so randomly, we evaluate the full test set 20 times, generating new random sentences each time. Across all of these runs, we calculate the maximum percentage, minimum percentage, mean percentage, and median percentage obtained. These values are used to compare each of our test architectures with each other.

5 RESULT AND DISCUSSION

The results of our experiments are contained in Table 1 for the random words experiments and in Table 2 for the random sentences experiments. In addition to these quantitative results, we also provide a discussion on the qualitative quality of the commentary generated by each model.

5.1 Quantitative analysis

From Table 1 we can see that all networks achieved above 90% accuracy across all relevant measures. This means that each model excels at identifying syntactically correct sentences compared to sequences of random words. It is interesting to note that the base model achieves 100% accuracy on average, meaning that it is very likely to be able to recognize syntax. This could mean, however, that it is unlikely to generalize. It is also interesting that using audio features performs poorly when compared to the other architectures. It is possible that this is due to the complexity of the input data. It is also possible, however, that the commentary generated by this architecture would be more likely to generalize since it has access to more information than the other networks.

From Table 2, the accuracy for each model is around 50%, peaking at 60% for the audio feature model. This indicates that each model struggles to identify the correct context based on the input for a given output sentence. One potential explanation for this is that the commentary for these videos is generic. This could potentially be improved by using semantic parsing to give our models a greater understanding of what is happening in each video. Also, there are many potential explanations for why this occurred. First, our video clips are very short (on the order of 2-3 seconds). It's possible that this is so short that the commentary that we captured is actually not very related to the current clip because of a natural delay that occurs when people provide commentary. Also, the scenes in *Getting Over It* are very similar to each other, which could also add to the difficulty of determining correct context from video information.

Even with small range difference, one positive trend that we notice, however, is that models with more information are, at least, less easily confused by other random sentences given an input video segment. Sequence to sequence with attention outperforms the baseline in terms of accuracy mean, max, and median. The attention model, however, tends to be weaker when looking at minimum accuracy and standard deviation. This indicates that there is more variance in the commentary generated from the attention model compared to the others, including the baseline sequence to sequence model. This may be because the attention network is more likely to identify potentially spurious correlations between output words and the input image. Although the optical flow network and the multi-modality network are also based on the attention network, they contain more data as input. This helps minimize the chance of the attention network finding spurious correlations. In terms of maximum accuracy, the audio features architecture achieves 60% accuracy, the highest among all models, within 20 runs.

Table 1: Random words experiment result of all the Models

Model	Minimum	Maximum	Mean	Median	Standard Deviation
Sequence to Sequence(baseline RGB)	1.0	1.0	1.0	1.0	0.0
Sequence to Sequence with Attention	0.991	1.0	0.999	1.0	0.003
Optial Flow Model	0.995	0.995	0.995	0.995	0.0
Audio Feature Model	0.936	0.973	0.952	0.95	0.009

Table 2: Random sentence experiment result of all the Models

Model	Minimum	Maximum	Mean	Median	Standard Deviation
Sequence to Sequence(baseline RGB)	0.454	0.555	0.504	0.507	0.023
Sequence to Sequence with Attention	0.432	0.564	0.513	0.518	0.033
Optical Flow Model	0.464	0.568	0.520	0.523	0.030
Audio Feature Model	0.482	0.6	0.524	0.518	0.030

Table 3: Examples of generations

Model	Target	Generated Text
Sequence to Sequence (Baseline RGB)	no god , hooooh no	oh noho come on ! ! ! (* fun time ! *)
Sequence to Sequence with Attention	no god , hooooh no	i don ' t want anything else for christmas . i just want to get back to where i was
Optical Flow Model	no god , hooooh no	that doesn ' t mean it ' s any less frustrating . .
Audio Features Model	no god , hooooh no	i never thought i would get back to this point -

This provides support for our claim that the manner of speaking influences the type of words used in commentary.

5.2 Qualitative analysis

While the quantitative analysis provides definitive metrics on the capabilities of each network, we feel it is important to have a qualitative perspective on the types of commentary that is being generated. Output samples from each network are outlined in Table 3. In this table, output for each of the architectures explored earlier is listed for the same input clip. We can see that the ground truth test target in Table 3 is a negative commentary. It is interesting to note that the input video used here involves the player falling down. The commentaries generated by the attention model, mixed RGB and optical flow model, and multi-modality model all apply to this situation. This implies that it's possible the agent is picking up more context than we realize, and poor performance in the quantitative analysis is due to the nature of the dataset we curated or the metrics we chose for evaluation.

One thing that we notice from each output is that each network generates human readable text that are syntactically correct. Although the output from the baseline model contains some unorthodox characters, it is similar to the training text provided to the network. One other thing to note is that these sentences are very general and could potentially be fit for many different types of situations. This potentially explains why these models struggled to achieve high accuracy in the quantitative study. By generating broad commentary, the models would struggle to reproduce any commentary that is tightly coupled with the test input.

It is not clear, however, if these broad comments are necessarily a limitation. It is possible that humans are more accepting of this broad commentary as long as it at least somewhat correlates with the context of the input video clip. So, even though these networks may struggle from a quantitative perspective, it is possible that they will produce effective commentary from a qualitative perspective.

6 CONCLUSIONS

Recently, there has been increased interest in exploring how deep learning approaches could be used to generate novel text based on video input. In this paper, we explore how deep learning approaches can be used to generate *Let's Play* commentaries based on different representations of video game playthrough data. Commentary generation is a difficult task due to the complex nature of commentary. Since commentary often involves knowledge that cannot always be directly inferred from the video alone, this is a difficult task for many machine learning algorithms. Specifically, formulate an end-to-end pipeline for generating *Let's Play* commentaries and explore how different types of input data can affect the quality of the generated commentaries.

Each sequence to sequence model examined in this paper is able to produce commentaries that are syntactically correct. While the networks examined struggle to produce contextually relevant commentaries according to our quantitative evaluation, our qualitative analysis provides some positive evidence that the commentaries generated using deep learning could still potentially be satisfying for human viewers. In the future, we will continue our research on how deep learning and multi-modal data can be used for commentary generation. We hope that this work helps to show the potential that deep learning has for this task, but also the considerations that need to be made for it to be effective.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[2] Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Association for Computational Linguistics, Vancouver, Canada, 747–754.

[3] Carlos Busso, Zhigang Deng, Serdar Yildirim, Murtaza Bulut, Chul Min Lee, Abe Kazemzadeh, Sungbok Lee, Ulrich Neumann, and Shrikanth Narayanan. 2004. Analysis of emotion recognition using facial expressions, speech and multimodal information. In *Proceedings of the 6th international conference on Multimodal interfaces*. ACM, 205–211.

[4] Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2422–2431.

[5] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. 2014. Deep convolutional filter banks for texture recognition and segmentation. *arXiv preprint arXiv:1411.6836* (2014).

[6] Gunnar Farnebäck. 2003. Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*. Springer, 363–370.

[7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587.

[8] Matthew Guzdial, Shukan Shah, and Mark Riedl. 2018. Towards Automated Let's Play Commentary. *arXiv preprint arXiv:1809.09424* (2018).

[9] Wei-Ning Hsu, Yu Zhang, Ron J Weiss, Heiga Zen, Yonghui Wu, Yuxuan Wang, Yuan Cao, Ye Jia, Zhifeng Chen, Jonathan Shen, et al. 2018. Hierarchical Generative Modeling for Controllable Speech Synthesis. *arXiv preprint arXiv:1810.07217* (2018).

[10] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[11] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[12] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. 2015. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th python in science conference*. 18–25.

[13] Pingbo Pan, Zhongwen Xu, Yi Yang, Fei Wu, and Yueting Zhuang. 2016. Hierarchical Recurrent Neural Encoder for Video Representation With Application to Captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[14] Rakshith Shetty and Jorma Laaksonen. 2016. Frame-and segment-level features and candidate pool evaluation for video caption generation. In *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 1073–1076.

[15] Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 568–576. http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.pdf

[16] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[17] RJ Skerry-Ryan, Eric Battenberg, Ying Xiao, Yuxuan Wang, Daisy Stanton, Joel Shor, Ron J Weiss, Rob Clark, and Rif A Saurous. 2018. Towards End-to-End Prosody Transfer for Expressive Speech Synthesis with Tacotron. *arXiv preprint arXiv:1803.09047* (2018).

[18] Stanley Smith Stevens, John Volkmann, and Edwin B Newman. 1937. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America* 8, 3 (1937), 185–190.

[19] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[20] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. 2015. Sequence to Sequence - Video to Text. In *The IEEE International Conference on Computer Vision (ICCV)*.

[21] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2014. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729* (2014).

[22] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3156–3164.

[23] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. 2048–2057.

[24] Fei Yan, Krystian Mikolajczyk, and Josef Kittler. 2016. Generating commentaries for tennis videos. In *Pattern Recognition (ICPR), 2016 23rd International Conference on*. IEEE, 2658–2663.

[25] Li Yao, Atousa Torabi, Kyunghyun Cho, Nicolas Ballas, Christopher Pal, Hugo Larochelle, and Aaron Courville. 2015. Describing videos by exploiting temporal structure. In *Proceedings of the IEEE international conference on computer vision*. 4507–4515.

[26] Quanzeng You, Hailin Jin, Zhaowen Wang, Chen Fang, and Jiebo Luo. 2016. Image captioning with semantic attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4651–4659.