Video-Description-CNN-RNN

Ruth Okoilu*
Email: rookoilu@ncsu.edu

Rohit Naik*
Email: rtnaik@ncsu.edu

Nirav Jain*
Email: najain@ncsu.edu

Udit Deshmukh*
Email: udeshmu@ncsu.edu

*North Carolina State University Raleigh, USA

I. Background

CNN

II. Motivation

III. Related Work

Video description generation has been investigated and studied in other work, such as [**? ? ? ?**]. Most of these examples have, however, constrained the domain of videos as well as the activities and objects embedded in the video clips. Furthermore, they tend to rely on hand-crafted visual representations of the video, to which template-based or shallow statistical machine translation approaches were applied.

IV. Literature Review

Convolutional neural networks is a class of deep learning architecture inspired by the natural visual perception mechanism of the living creatures. It involves feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. A CNN consists of an input and an output layer, as well as multiple hidden layers . [1]. CNN are very effective in areas such as image classification and recognition.

The encoder-decoder framework consists of two neural networks; the encoder and the decoder. Multiple frames are fed into the encoder to generate representations which are then fed into a decoder to generate the desired output. In terms of video to text, video clips serve as input to the encoder network, an intermediate representation is the output and input to the decoder whose output is a text - description of the events in the video. Encoder-decoder framework has been found to be useful in applications such as language translation and video to text frameworks. For the later, the encoder network is usually an image- trained CNN used on each frame separately. However, in the case of machine translation, it is natural to use a recurrent neural network (RNN) for the encoder as the input is a variable-length sequence of symbols[3]. While RNN, Gated Recurrent networks or variants to LSTM can be used as decoder in both cases since the output is a natural language sentence.

CNN in video recognition has been implemented by a two-stream ConvNet architecture that incorporates spatial and temporal networks. In [2] ConvNet were trained on multi-frame dense optical flow is able to achieve very good performance in spite of limited training data. A similar architecture is based on two separate recognition streams (spatial and temporal), which they combined by late fusion. The spatial stream performs action recognition from still video frames(images), while the temporal stream is trained to recognise action from motion in the form of dense optical flow. Both streams are implemented as ConvNets. This architecture is related to the two-streams hypothesis of the human visual cortex which has two pathways: the ventral stream (which performs object recognition) and the dorsal stream (which recognises motion). The input to the model is formed by stacking optical flow displacement fields between several consecutive frames.

The performance of such architecture has been evaluated on UCF-101 and HMDB-51 benchmarks as well as evaluation metrics such as BLEU, ME- TEOR and CIDEr.

V. Plan of Work

Data collection and exploration We explored a preprocessed Youtube2Text dataset - checked the structure, contents and what those contents represent. The entire dataset has 1970 video files. Each file has varying number of frames within it (depending on the length of the video clip). Each frame is represented using a 1024 dimensional frame-wise feature vector. This feature vector is obtained by selecting 26 equally-spaced frames out of the first 240 from each video and feeding them into the CNN.

Resolution of dependencies and local set up The project requires a couple of modules - Jobman and coco-caption. Jobman is used to initialize the model configuration parameters such as patience, number of epochs, batch size, optimizer, etc. coco-caption on the other hand, is a caption generation module which we are leveraging for our project. Further, we need to add both these modules to our local PYTHONPATH variable so that we can use them as modules in our project. We have written a script for installing all such dependencies and setting up the environment.

Explored Google Collab for leveraging GPU capability We explored Google's Cloud Service for AI called Google Colab because it provides free GPU computation capability. It also supports shared development by allowing multiple people to work on the same ipython notebook, and supports multiple python versions as well. We used the script created in step 2, and resolved dependencies on Google Colab workspace as well. However, the GPU provided by Google Colab is limited. First of all it is not always available, and secondly executing commands on Google Drive's workspace took a lot of time. A simple "rm -rf" took more than 5 minutes. Thus, we decided

to use the ARC cluster provided by the university for our GPU requirements.

Plan for exploration Visualize output of each convolutional layer: We plan to visualize the image obtained after each of the three convolutional layers. This will help us to understand visually which features are extracted by each convolutional layer. Change parameters of config.py and understand importance of each: We plan to study what effect do different parameters have on the training. Some of these parameters include learning rate, optimizer, number of epochs, patience. We plan to compare the performance of the model on BLEU and METEOR metrics for different model configurations. Intermediate results of the description: In addition to the above mentioned metrics, we also plan to qualitatively compare the model performance. For this, we will visualize the output, that is the text description generated for the video clip, at intermediate steps during model training and also for different model parameter settings. Modifying frame length parameter: The current model considers 2 time frames while building the features. We plan to study the effect of using multiple time frames in this stage.

ANTLR

VI. Conclusion

In the end, we achieved a