Rap Machine

De Huo dhuo@ucsc.edu
Ke Wang kwang82@ucsc.edu
Nursultan Kabylkas nkabylka@ucsc.edu
Ramesh Jayaraman rkjayara@ucsc.edu

Yuan Yang yyang175@ucsc.edu

Abstract

We present Rap Machine, an image based lyrics generator. We use a CNN model extract features from images, caption the image using a RNN model, and generate lyrics in the style of an artist using a sequence to sequence model. We train and evaluate our image captioning model using the Flickr 8k dataset, and use lyrics from several artists across different genres and writing styles to train and generate lyrics based on the caption we obtained from the image captioning model.

1 Introduction

We are proposing to combine image captioning implemented with Convolution Neural Nets and poem generation implemented with Recurrent Neural Nets. The idea is to train CNN to caption images with sentences, and also train RNN with the song lyrics of different famous artists, such as: 2Pac, 50 cent, Eminem, Coldplay, Tool, Iron Maiden etc.

Once the two Neural Net models are trained, we will use a test set of images to generate image caption, using the CNN model, and feed the generated text as an input of the RNN model, to generate the Poem/Lyrics. The purpose of this project is to evaluate the performance of RNN model, and compare generated lyrics among different artists, so as to compare their ”style” of writing.

We will first provide details about our implementation of image captioning in section 2, followed by our implementation of lyrics generation in section 3. We will then present our evaluation, combining the models together in section 4, and conclude our work in section 5.

2 Image Captioning

Image captioning is the process of generating textual descriptions of images. It mainly consists of two parts, image features extraction parts using CNNs and textual descriptions generating parts using RNNs. Training a CNN network costs huge amount of time and requires a super large data pool. So based on our limitations, we implemented a well trained CNN model for our image extraction, it is InceptionV3. After we got the image features, we merged the image's feature and the word's feature together and feed it into a bidirectional RNN to caption our image.

2.1 Image Feature Extraction

Convolutional Neural Networks is the most commonly used network to process images. After the great breakthrough made in 2012, “AlexNet”, a few well performed CNN models came, such as “GoogleLeNet” and “InceptionV2”. They were all focusing on improving the functional performance of the single convolutional module. InceptionV3 is a more advanced CNN model to process the image. It factorizes the two dimensional convolutional layer to two one dimensional convolutional

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

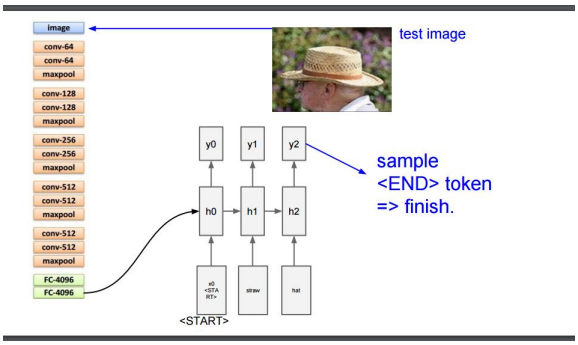

Figure 1: Architecture of the proposed implementation

vectors for accelerating the calculating speed and increasing the networks' depth **??** Inception V3 model can classify images into 1000 classes with high accuracy and high computational efficiency. Therefore, we used pre-trained inception V3 model to obtain features of images.

2.2 Textual Descriptions Generation

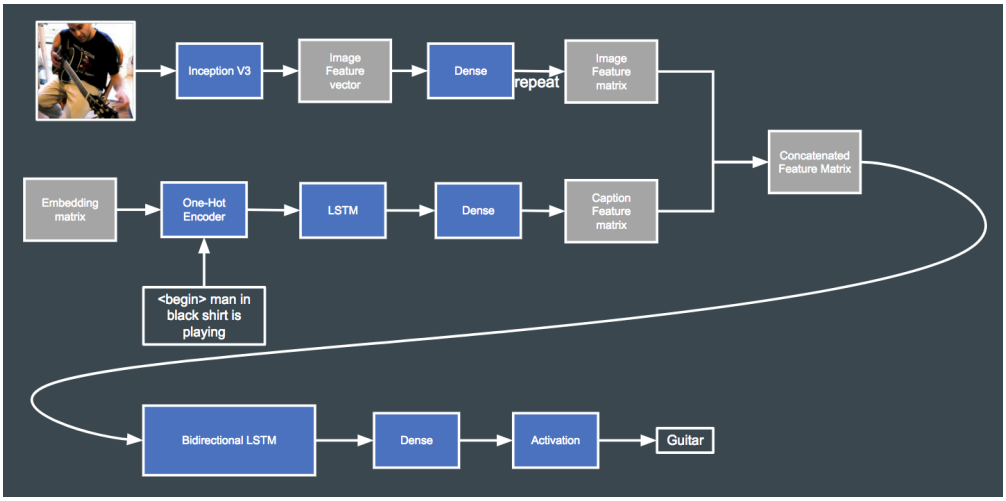

Figure 2: Architecture of our image captioning model

2.2.1 Architecture

We feed one picture into InceptionV3 model, then the CNN will give us a feature vector of that picture after processing it. The dimension of the image feature vector is 2048. We don't want to train too many parameters in RNN model, because it costs too much time. So we compact the dimension to 256, and let it repeat 40 times, which is the max length of our text sentences. And now we get a single image matrix (256 x 40).

We let the model train the word embedding matrix with dimension 256 , and we use the One-Hot matrix which generates from each description sentences for a picture to look up a word feature matrix (256 x 40). Then we sent the matrix to a LSTM RNN for training the embedding matrix. Another purpose of this LSTM is to get the sentence's feature for each word in this sentence. The dimension of this matrix is also 256 x 40, and we call it Caption feature matrix.

We concatenate Image feature matrix and Caption feature matrix into a concatenates feature matrix with size 600 x 40. Let it be the input of Bidirectional LSTM and output a feature vector of the generated word (1 x 512), using a dense layer to transfer the size to 1 x 8000. At last, using softmax activation layer to obtain the vector that represents the generating word.

When training, the input is the image and the front part of the sentence which includes the words that have already generated before. The output is the next word of the input uncompleted sentence. Training of one sentence halt once output is "<end>".

2.2.2 Dataset

In general, datasets for image captioning include images and some captions that describe these images. There are several datasets for image captioning, such as Flickr 8k, Flickr 30k, Microsoft COCO, UIUC Pascal Sentence and so on. Because of limitations of time and training environments, we chose Flickr 8k as our dataset since it has relatively small size comparing to Flickr 30k and Microsoft COCO, and more images and sentences comparing to Microsoft COCO. Flickr 8k dataset contains 6000 training images, 1000 validation images and 1000 test image with 5 textual descriptions of each image. 21% images have static verbs like sit, stand, wear, look or no verbs.

2.2.3 Hyperparameters

The optimizer of our model is the Adam optimizer. The parameter of Adam optimizer comes from its original paper[2]. Convergence of Adam is quite fast compares to other methods. The Loss function we chose is the softmax cross entropy loss. The training batch size we chose is 256.

2.3 Results

After 100 epochs training, our loss reached 2.016 and we got a training accuracy of 0.501. One of test results is shown in Figure 3.

Out[321]:

In [322]:
```
print ('Normal Max search:', predict_captions(try_image))
print ('Beam Search, k=3:', beam_search_predictions(try_image, beam_index=3))
print ('Beam Search, k=5:', beam_search_predictions(try_image, beam_index=5))
print ('Beam Search, k=7:', beam_search_predictions(try_image, beam_index=7))
```

```
Normal Max search: A child climbing on a rope ropes .
Beam Search, k=3: A little girl in a red shirt and green shirt climbs on a playground .
Beam Search, k=5: A child in a red shirt and red shirt climbs up ropes .
Beam Search, k=7: A child in a red shirt and green climbs on a playground .
```

Figure 3: Architecture of the proposed implementation

3 Lyrics Generation

3.1 Architecture

3.2 Dataset

As we wanted to train the model using different writing styles that different artists follow (a genre/artist specific model), we used a script and scraped our lyrics from AZLyrics [cite]. We then removed repeating lines, abbreviations, and other content that we felt would interfere with the training of our model. We also used the Kaggle 55000+ song dataset [cite] to train and test our model for generalized

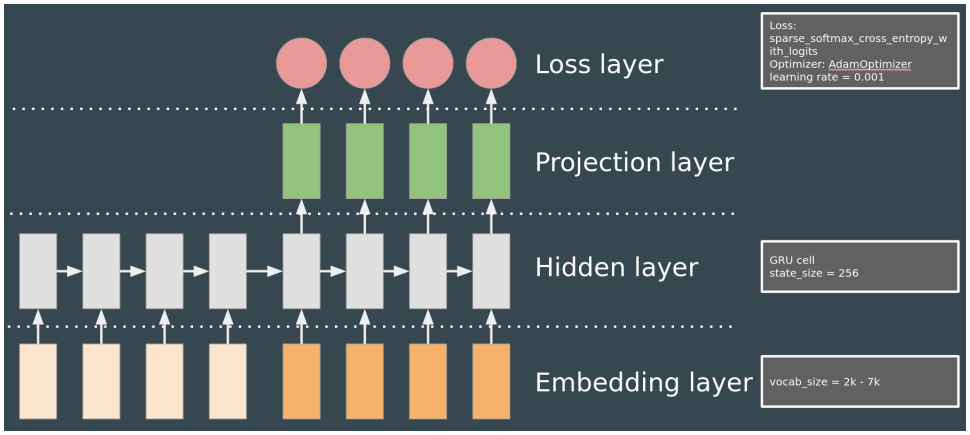

Figure 4: Architecture of our image captioning model

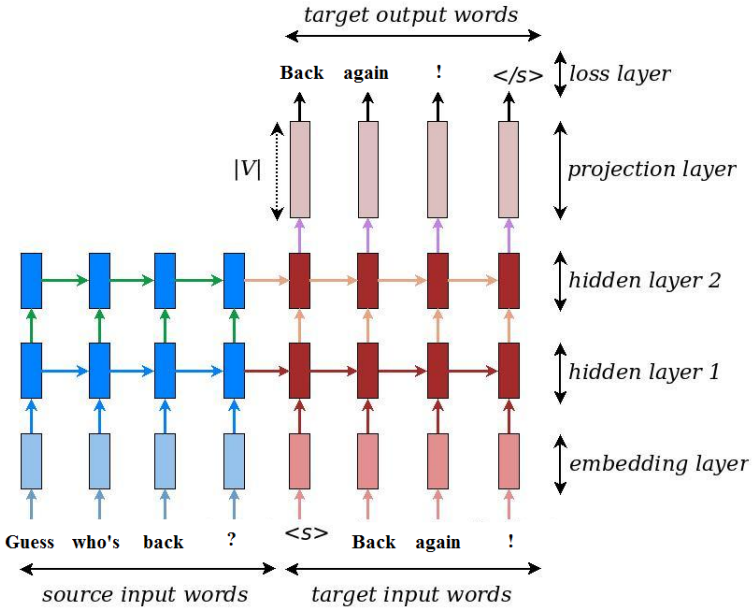

Figure 5: Architecture of our image captioning model

lyrics generation (as a generic lyrics generation model). We also wanted to use the million song dataset [cite], but we were unable to train and test our model using this dataset due to limited time and computational resources despite using GPUs.

3.3 Hyperparameters

3.4 Results

4 Evaluation

In this section we provide our evaluation by combining the image captioning and lyrics generation models, and generating lyrics based on the caption generated by our model.

5 Conclusion

We implemented the proposed image captioning based lyrics generation model, and evaluated it using the Flickr 8K dataset and lyrics from several artists. We achieved accurate captioning and lyrics generation for different artists and across artists using the Kaggle 55000+ song dataset.